Private Cloud Management

Transforming VMware SDDC into Your IaaS



PART 3 OF 4

July 2026

5 Edition

Iwan "e1" Rahabok
VMware Cloud Foundation | Broadcom
e1@broadcom.com

Back of cover page. Delete if you do not plan to print.
Thank you Mai Mai for the design. Daddy proud of you!

PART 3

Counters

Table of Contents

Chapters:

Introduction	
CPU	CPU metrics. It focuses on VM and ESXi, with comparison with Guest OS.
Memory	As above, but for memory.
Storage	Storage metrics
Network	Network metrics
Consumer	Consumer means VM. It covers complex metrics such as performance and capacity. These metrics span beyond the 4 infrastructure elements.
Provider	Provider means the shared infrastructure, such as ESXi and vSAN.
MS Windows	Microsoft Windows
Linux	[future chapter]

The World of Metrics

Part 3 Chapter 1

VMware Cloud Foundation ships with many metrics and properties. If we take object by object, and document metrics by metrics, it will be both dry and theoretical. You will be disappointed as it does not explain how your real-world problems are solved. You're not in the business of collecting metric.

Metrics Nuances

Our Inspiration	Our Endeavour
Space, the final frontier These are the voyages of the Starship Enterprise Its five-year mission To explore strange new worlds To seek out new life And new civilizations To boldly go where no man has gone before	Metrics, the final frontier These are the voyages of The Book Its 10-year mission To explore strange new systems To seek out new observability And new proactive operations To boldly go where no IT Architect has gone before

Metric is essentially an accounting of system in operations. To understand the counter properly hence requires a knowledge of how the entire system works. Without internalizing the mechanics, you will have to rely on memorizing. In my case, memorizing is only good for getting a certificate.

Take time to truly understand the reasons behind the metrics. You will appreciate the threshold better when you know how it is calculated.

Complexity

It is useful to know the subtle differences in the behaviour of metrics. By knowing their differences, we can pick the correct metrics for the tasks at hand.

Naming Complexity

Same object, same metric, different name	OpenTelemetry uses "Usage" for GB, but "Utilization" for %, instead of simply "Usage (%)". For example, for Kubernetes Pod memory, the metrics are called Memory Usage (GB) and Memory Utilization (%).
Same name, same object, different formula	The metrics have the same name, belong to the same object, yet they have a different formula depending on where in the object you measure it. Example: VM CPU Used in vCPU level does not include System time but at the VM level it does. The reason is that System time does not exist at vCPU level since the accounting is charged at the VM level.
Same name, same purpose, different formula	Metrics with the same name do not always have the same formula in different vSphere objects.
	Memory Usage: in VM this is mapped to Active, while in ESXi Host this is mapped to Consumed. In Cluster, this is Consumed + Overhead. Technically speaking, mapping usage to active for VM and consumed for ESXi makes sense, due to the 2-level memory hierarchy in virtualization. At the VM level, we use active as it shows what the VM is physically consuming

	(related to performance). At the host and cluster levels, we use consumed because it is related to what the VM claimed (related to capacity management). This confusion has resulted in customers buying more RAM than what they need. VCF Operations uses Guest OS data for Usage, and falls back to Active if it's not available.
	Memory Consumed: in ESXi this includes memory consumed by ESXi, while in Cluster it only includes memory consumed by VM. In VM this does not include overhead, while in Cluster it does.
	VM Used includes Hyper Threading but penalty is 37.5%. ESXi Used is also aware of HT but the penalty is 50%.
	Virtual Disk: in VM includes RDM, but in Datastore it does not. Technically, this makes sense as they have different vantage points.
	Steal Time in Linux only includes CPU Ready, while stolen time in VM (CPU Latency) include many other factors including CPU frequency.
Same name, different meaning	Metrics with identical name, yet different meaning. Be careful of misinterpretation.
	VM CPU Usage (%) shows 62.5% when ESXi CPU Usage (%) shows 100%. This happens due to different perspectives. When both threads are running, ESXi CPU Usage does not.
	Disk Latency and Memory Latency indicate a performance problem. They are in fact the primary counter for how well the VM is being served by the underlying IaaS. But CPU Latency does not *always* indicate a performance problem. Its value is affected by CPU Frequency, which can go up or down. Sure, the VM is running at a higher or lower CPU speed, but it is not waiting to be served. It's the equivalent on running on older CPU.
Same name, different behaviour	VM CPU Usage (MHz) is not capped at 100%, while ESXi CPU Usage (MHz) is. The later won't exceed the total capacity.
	Memory Reservation and CPU Reservation have different behaviors from monitoring viewpoint.
	In Microsoft Windows, the CPU queue includes only counts the queue size, while the disk queue excludes the IO commands being processed.
Same purpose, different name	You would expect if the purposes are identical then the labels or names will be identical.
	Kubernetes uses the term "Request", while vSphere uses the term "Reservation". The excuse from Google Borg is the persona is different. The developer is requesting a reservation.
	Swapped Memory in VM is called Swapped, while in ESXi is called Swap Used. Static frequency CPU utilization in VM is called Run, while ESXi calls it Utilization. Different names make sense as they reflect different vantage points.
	What vCenter calls Logical Processor (in the client UI) is what ESXi calls Physical CPU (in esxtop panel). I think the word logical is confusing. Easier to use virtual vs physical.

	vCenter uses Consumed (%) and Usage (%) for the same ESXi CPU utilization.
	Microsoft Windows has 3 counters to measure CPU consumption. • Performance Monitor uses the name Processor Time. • Task Manager uses the name CPU Utilization. • Resource Monitor uses the name CPU Usage
Confusing name	The name of the counter may not be clear.
	OpenTelemetry uses the following definition: **"usage** - an instrument that measures an amount used out of a known total (limit) amount should be called entity.usage." **"utilization** - an instrument that measures the fraction of usage out of its limit should be called entity.utilization."
	VM CPU Wait counter includes idle time. Since many VMs do not run at 100%, you will see CPU Wait counter to be high. You may think it's waiting for something (e.g. Disk or Memory) but it's just idle. If we see from the viewpoint of kernel schedule, that vCPU is waiting to be used. So the name is technically correct
	The term virtual disk includes RDM. It's not just VMDK. The reason is RDM appears as virtual disk when you browse the directory in the datastore, even though the RDM file is just a pointer to an external LUN.
Incorrect name	Task Manager in Windows is not correct as the kernel does not have such concept. The terminology that Windows has is called Job. A job is a group of processes that can be managed as one. Do you want it to be called Job Manager? 😊 I think Process Manager is better as that's what running on top of the kernel.
Mixing terminology	Allocation and reservation are 2 different concepts. When you allocate something to someone, it does not mean it's guaranteed. If you want a guarantee, then do reservation. Allocation is a maximum (you can't go beyond it), while Reservation is a minimum. The actual utilization can be below reservation but can't exceed allocation. You cannot overcommit reservation as it's a guarantee. You can overcommit allocation as it is not a guarantee. Avoid using metric names like these: • Allocation Reservation. This makes no sense. • Maximum Reservation. Simply use Allocation instead. • Minimum Allocation. Simply use Reservation instead.
	The word device, disk and LUN are used interchangebly. They are actually not identical. Device is actually generic. It does not have to be storage device. It can be network device or security device. Disk tends to be phsyical, something you can hold in your hands. It is typically part of a disk group called RAID. Advanced storage array takes the concept further. It can create a logical volume spanning multiple RAID groups. The lazy name LUN was chosen. I prefer to call it Logical Volume, but will stick to LUN in this book. LUN is what

	ESXi host mounts and then format it with VMFS filesystem.

Architecture Complexity

The 4 basic elements of infrastructure have been around since Adam built the first data center for Eve. Humankind has not invented the 5th Element yet. However, each of these 4 has their own unique nature. They have "speed vs space" dimensions. This in turn creates complexity in observability.

The following table shows how the 4 elements of infrastructure relate to these 2 dimensions.

	Speed	Space	Analysis
CPU	Yes	N/A	Speed is measured in Hertz. Space is measured in vCPU or physical thread.
Memory	Limited	Yes	Speed is practically not applicable as the latency is in nanoseconds. It only matters in application that needs low latency such as GenAI. Memory read/write is speed, but since the latency negligible, it is generally not used in performance management.
Storage	Yes	Yes	2 metrics for speed. Bytes per second and IO per second. Space is measured in bytes.
Network	Depends on the nature		Unlike server and storage, network takes >1 shape. It's both a line and a dot. The network cables are the lines, the switch is the dot.

Let's elaborate.

CPU	The speed of CPU is far higher than the speed of the observability code. This results in values that is not 100% correct. For example, if you count the number of threads per thread state in Windows or Linux, you may get 5% variation. Your code operates in miliseconds while CPU runs in nanoseconds.
	The primary speed metric (GHz) is not comparable across different hardware generation or architecture. 1 GHz in today's CPU is more efficient than 1 GHz in older CPU. You get more work done.
	Different variants of CPU models in the same generation from the same vendor.
Memory	Its function is caching, so its counters tend to be near 100%, and that is what you expect as cache is typically much smaller than the actual source. Anything less than 100% is not maximizing your money. CPU and memory metrics have different nature. 95% utilization for memory could be low, while 85% for CPU could be high already.
	It's a form storage, so its metrics are mostly about space and not speed. There is VM CPU Demand, but not VM Memory Demand. Demand does not

	apply to memory as it's a form of storage, just as there is no such thing as a Demand metric for your laptop disk space.
Storage	It has 2 sides (speed and space), but both have consumption metrics. The speed has 2 components for consumption: IOPS and Throughput. The throughput depends on the block size. It is also measured in bytes/second, not bit/second. The reason is you're measurign the amount of disk space being read or written from the storage. Unlike network throughput, which focuses on the cable or wire, the disk throughput focuses on the node.
	There is limit for disk IOPS, but not for disk throughput. The latter is not practical to implement due to varying block size.
Network	While server and storage are nodes, network is interconnect. This makes it more challenging. We will cover this in-depth in the System Architecture section

Their complexity results in difference in type of metrics applicable:

	Utilization	Reservation	Allocation
CPU	Yes, but 3 different units for ESXi (thread, Hertz, vCPU)		
Memory	Yes	Yes	Yes
Disk	Yes	Not Applicable	Yes
Network	Yes	Yes	Yes

And lastly, beyond metrics there are also further complications such as:

VM vs ESXi	The CPU metrics from a VM viewpoint differs to the CPU metrics from ESXi viewpoint. A VM is a consumer. Multiple VMs can share the same physical core, albeit at the price of performance. So metrics such as Ready does not apply to ESXi. The core and the thread are always ready.
ESXi vs vCenter	While ESXi is the source of metrics, vCenter may add its own metrics and the formula don't always match 100% in all scenarios, such as Used vs Usage.
	ESXi provides Run (ms), Used (ms), Demand (MHz) for VM CPU. vCenter adds Usage (MHz) and Usage (%).
	ESXi shows Used (%), while vCenter shows Used (ms). The first one affected by CPU frequency and can go beyond 100%.
ESXi ≠ VMs + Kernel	The metrics at ESXi is more complex than the sum of its VM + the kernel.
M:N relationship	A VM with multiple virtual disks can span across multiple datastores, and even RDMs. On the other hand, a datastore typically hosts many VMs. An ESXi may mount multiple LUNs and a LUN is typically presented into multiple ESXi or even multiple clusters. These many to many relationships make the metrics across VM, datastore, ESXi, Cluster, Data Center inconsistent when viewed overall. Each of them is correct as each has to look from their own vantage point.

Other Nuances

Time	Some metrics reflect what happened in the past. That's why the name is in past tense.
	Memory ballooned does not mean there is an active ballooning happening. The process could have happened days ago, and the page was simply idling, never been asked by Guest OS.
Roll up	vCenter measures every 20000 ms, but the maximum value for a completely idle thread is 10000. The reason is 20000 is the value set at the core level. Since a core has 2 threads when HT enabled, each was allocated 10000.
Unit	VM CPU Ready can be above 100%. If you look at esxtop, many VM level metrics are >100%. This is due to summation as opposed to average maths.
	Why are CPU metrics expressed in milliseconds instead of percentage or GHz? How can a time counter (milliseconds in this case) account for CPU Frequency?
	For CPU, 1 GHz = 1000 MHz. For memory and disk space, 1 GiB = 1024 KiB and 1 GB = 1000 KB.
	Esxtop and vSphere Client use different units for the same metric. For example, esxtop use megabit while vCenter UI use kilobyte for networking counter.
Formula	ESXi CPU Core Utilization reports 100% when 1 thread is running. But it also reports 100% when both threads are running. To know which one is the case, you need to check ESXi CPU Thread Utilization.
	ESXi CPU Idle (ms) includes low activities. It also considers CPU frequency, and not simply whether CPU is running or now.
Sampling	Counters are sampled, so you should expect small rounding/timing drift, not a perfect mathematical 100.000% on every sample.
	Sampling can also drift in a busy system. For example, you want to sample every 20 second. If the sample takes a few seconds because you're sampling massive amount of data and your system is busy, your final numbers will drift over time against wall clock. To overcome this, your collection logic must consider the time taken to collect and process.

In addition to all the above nuances, there are complexity created by choices and scale.

Choices	When you have 2 watches showing different times, you become unsure which watch is the correct one.
	There are 5 metrics for VM CPU "consumption": Run, Used, Usage, Usage in MHz, and Demand.
	There are 7 metrics for ESXi CPU "consumption": Core Utilization, Utilization, Used, Usage, Usage in MHz, Consumed, and Demand.
Volumes of Metric	The sheer number of metrics make analysis difficult.

	Take for example, vCenter has 17 CPU metrics available at the VM level, and 12 of them are available at a vCPU level too. In addition, each VM comes with 28 memory metrics. That means a VM with 4 vCPUs will have 93 metrics (17 + 4 x 12 + 28). A vSphere environment with 1,000 VMs with 4 vCPUs as the average VM size will have process 93K metrics each time it collects. If you do that every minute, you will collect almost 134 million metrics per day. Since many customers like to keep for at least 6 months, that's 24+ billion metrics.
	With so many metrics, the amount of business value received becomes a valid concern. At the end of the day, you are not in the business of collecting metrics.

Interpretation Challenge

At the end of the day, you care about the interpretation of the value. What is considered good? As you can guess, the answer is "it depends". Different applications have different tolerances. Even the same software has variable tolerance, depending on the use case. Even on identical user case, the end users have different tolerances.

The above makes it challenging to set up out of the box thresholds, which are universal for all cases. Set too low, and you see red too often when there is no actual complaint, and set too high, and you miss the actual problem.

As an example, let's take disk latency. I take Microsoft SQL Server as that's a popular application and it's used in many production environments. The following is a guidance from Microsoft:

If these waits exceed 10-15 milliseconds consistently, I/O is considered a bottleneck.

> **Note**
>
> To provide context and perspective, in the world of troubleshooting SQL Server, Microsoft CSS has observed cases where an I/O request took over one second and as high as 15 seconds per transfer-such I/O systems need optimization. Conversely, Microsoft CSS has seen systems where the throughput is below one millisecond/transfer. With today's SSD/NVMe technology, advertised throughput rates range in tens of microseconds per transfer. Therefore, the 10-15 millisecond/transfer figure is a very approximate threshold we selected based on collective experience between Windows and SQL Server engineers over the years. Usually, when numbers go beyond this approximate threshold, SQL Server users start seeing latency in their workloads and report them. Ultimately, the expected throughput of an I/O subsystem is defined by the manufacturer, model, configuration, workload, and potentially multiple other factors.

It says 10 – 15 milliseconds. But that's at MS SQL layer. That means at MS Windows layer it needs to be lower, to account for processing latency and queue in the stack. This also means at vSphere VM layer, it needs to be even lower. So what should we set the value at VM level? How about read versus write? It is indeed a challenge to set.

It says "consistently". Since IO measure per second, does it mean 300 seconds is far too long? That means 300x, which is definitely consistent. From what I've seen, if you set at 5 minutes, you are likely to get a lot of alerts but little complaint.

You can read more at Troubleshoot slow SQL Server performance caused by I/O issues.

Documentation Challenge

To master metrics, you need to be able to see them from 3 different perspectives:

Technology	Computer architecture has not fundamentally changed since the first day of the mainframe. You have CPU, memory, disk and network. Documenting via this route is useful for infrastructure.
	There are variants such as GPU and APU, and components become distributed and virtual. You need to be able to see how the metrics behaviour change.
Pillars of Operations	Each pillar brings their own set of metrics. For examples: • Capacity creates new metrics such as Time Remaining and Recommended Size • Cost creates new metrics such as Month-to-Date Cost • Performance creates new metric such as KPI (%) • Compliance creates new metric such as PCI-DSS Compliance Score (%)
Product	Product such as vSphere, NSX, and Kubernetes bring their own set of objects. Each type of object (e.g. VM, K8 Pod, AWS S3, Oracle DB) has their own set of metrics. Since objects typically have relationship with other objects and are grouped under a parent (e.g. a K8 Cluster has multiple nodes, which in turn can have multiple pods), you end up with multiple parallel hierarchies or overlapping hierarchies.

As a book has a simple and fixed structure, it's impossible to document in 3 different ways. That's why this book blends the approach.

- It starts with the raw metrics, followed by derived metrics. Raw metrics tend to be simple and narrow in scope. Derived metrics combine multiple raw metrics. It tends to cover multiple areas.
- For raw metrics, it follows the technology route. For each of the 4 infrastructure elements, it covers contention first, then consumption.
- For derived metrics, it follows the product or object approach. Within each object, it follows the pillar of operations.

Ideally, we use an interactive website where you can browse from different perspectives. Or maybe we simply use Generative AI, assuming we can augment it with logic instead of 100% relying on English and Maths.

Virtualization Impact

vSphere counters are more complex than physical machine counters because there are many components as well as inconsistencies that are caused by virtualization. When virtualized, the 4 elements of infrastructure (CPU, RAM, Disk, Network) behave *differently*.

Virtualization splits the IT stack into Provider and Consumer. The providers have 2 types of counters:

- Provider metric. This metric does not exist at consumer. For example, the physical network in ESXi.
- Sum of Consumer metric. This metric aggregates the metric of consumers. For example, the sum of VM network in the ESXi.

The confusion comes because the metric name does not tell what type of metric it is. For example, is ESXi CPU Demand a consumer or provider metric?

Not all VMware-specific characteristics are well understood by management tools that are not purpose-built for it. Partial understanding can lead to misunderstanding as wrong interpretation of metrics can result in wrong action taken.

The complexity is created by a new layer because it impacts the adjacent layers below and above it. So the net effect is you need to learn all 3 layers (Guest OS layer, virtualization layer and physical layer). That's why from a monitoring and troubleshooting viewpoint, Kubernetes and container technology require an even deeper knowledge as the boundary is even less strict. Think of all the problems you have with vSphere Resource Pool performance troubleshooting, and now make it granular at process level. You're having a good time mastering K8 right? 😉

Since VMkernel is technically an operating system, it has kernel space and user space. Some counters include both, while others don't.

What, exactly, is a VM?

One reason why VM gets mixed up with Guest OS is the mental model. When you think "machine", you think of a complete entity. Your mental model is other machines in real life, like washing machine, ATM machine, coffee machine. A VM is not complete. It's nothing but a flat piece of circuit board with electronics mounted on.

Think virtual motherboard, not virtual machine.

From observability viewpoint, a VM is not what most of us think it is. It changes the fundamental of operations management. It introduces a whole set of metrics and properties, and relegates many known concepts as irrelevant.

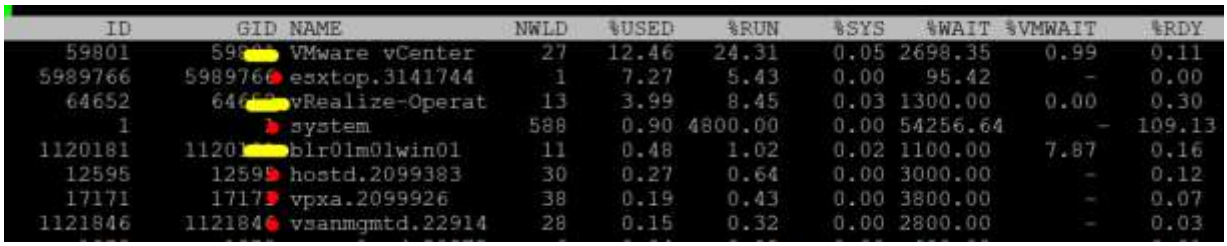
For example, you generally talk about these types of system-level metrics in Windows or Linux

- Processes
- Threads
- System Calls/sec

But when it comes to VM, you don't. The reason these OS-level metrics are not relevant is because a VM is not an OS.

To master vSphere metrics, you need to know ESXi kernel. The kernel is a different type of OS as its purpose is to run multiple virtualized motherboard (I personally prefer to call VM as virtual motherboard). As a result, its metrics are different to typical OS such as Windows and Linux.

From the kernel's vantage point, a VM is just a collection of process that needs to be run together. Each process is called World. So there is a world for each vCPU of a VM, as each can be scheduled independently. The following screenshot shows both VM and non VM worlds running side by side. I've marked the kernel modules with red dot. You can spot familiar process like vpxa and hostd running alongside VM (marked with the yellow line).



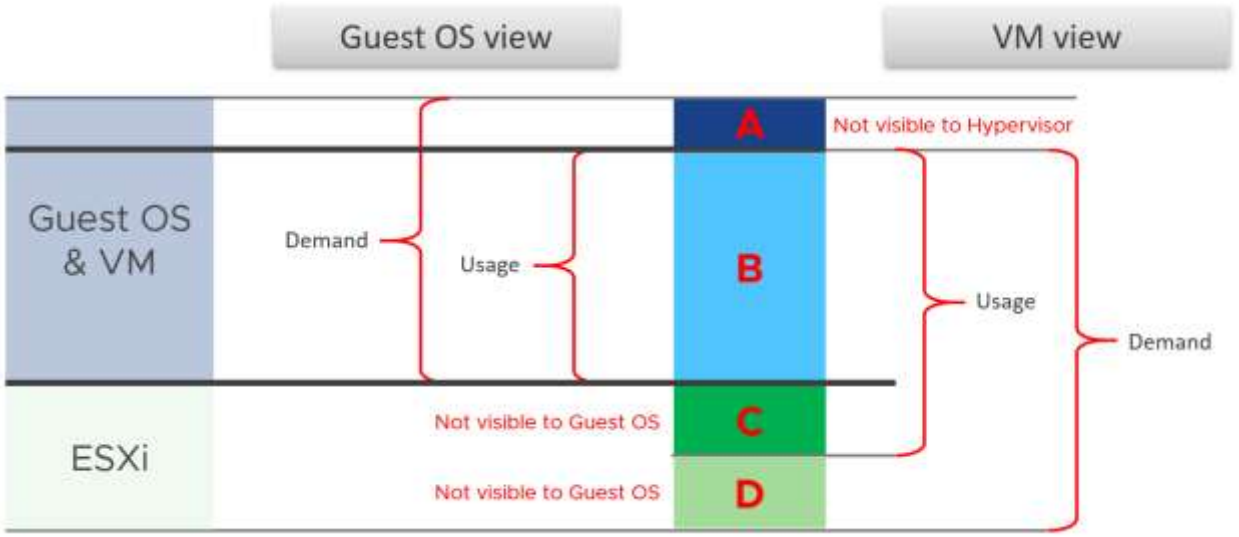
ID	GID	NAME	NWLD	%USED	%RUN	%SYS	%WAIT	%VMWAIT	%RDY
59801	59[redacted]	VMware vCenter	27	12.46	24.31	0.05	2698.35	0.99	0.11
5989766	5989766	esxtop.3141744	1	7.27	5.43	0.00	95.42	-	0.00
64652	64[redacted]	vRealize-Operat	13	3.99	8.45	0.03	1300.00	0.00	0.30
1	1	system	588	0.90	4800.00	0.00	54256.64	-	109.13
1120181	1120[redacted]	blr01m01win01	11	0.48	1.02	0.02	1100.00	7.87	0.16
12595	12595	hostd.2099383	30	0.27	0.64	0.00	3000.00	-	0.12
17171	17171	vpxa.2099926	38	0.19	0.43	0.00	3800.00	-	0.07
1121846	1121846	vsanmgmtd.22914	28	0.15	0.32	0.00	2800.00	-	0.03

Visibility

Guest OS and VM are 2 closely related due to their 1:1 relationship. They are adjacent layers in SDDC stacks. However, the two layers are distinct, each provide unique visibility that the other layer may not be able to give. Resource consumed by Guest OS is not the same as resource consumed by the underlying VM. Other factors such as power management and CPU SMT also contribute to the differences.

The different vantage points result in different metrics. This creates complexity as you size based on what happens inside the VM, but reclaim based on what happens outside the VM (specifically, the footprint on the ESXi). In other words, you size the Guest OS and you reclaim the VM.

The following diagram uses the *English* words demand and usage to explain the concept, where demand consists of usage and unmet demand. It does not mean the demand and usage metrics in vSphere and VCF Operations, meaning don't assume these metrics actually mean this. They were created for a different purpose.



I tried adding application into the above diagram, but that complicated the whole picture that I removed it. So just take note that some applications such as Java VM and database manage their own resources. Another virtualization layer such as Container certainly takes the complexity to another level.

We can see from the above that area A is not visible to the hypervisor.

Layer A	Queue inside the Guest OS (CPU Run Queue, RAM Page File, Disk Queue Length, Driver Queue, network card ring buffer). These queues are not visible to the underlying hypervisor as they have not been sent down to the kernel. For example, if Oracle sends IO requests to Windows, and Windows storage subsystem is full, it won't send this IO to the hypervisor. As a result, the disk IOPS counter at VM level will under report as it has not received this IO request yet.
Layer B	What the Guest uses. This is visible to the hypervisor as a VM is basically a multi-process application. The Guest OS CPU utilization somehow translates into VM CPU Run. I added the word "somehow" as the two metrics are calculated independently of each other, and likely taken at different sampling time and use different roll up technique.
Layer C	Hypervisor overhead (CPU System, CPU MKS, CPU VMX, RAM Overhead, Disk Snapshot). This overhead is obviously not visible to the Guest OS. You can get some visibility by installing Tools, as it will add new metrics into Windows/Linux. Tools do not modify existing Windows/Linux metrics, meaning they are still unaware of virtualization. From the kernel viewpoint, a VM is group of processes or user worlds that run in the kernel. There are 3 main types of groups: • VM Executable (VMX) process is responsible for handling I/O to devices that are not critical to performance. The VMX is also responsible for communicating with user interfaces, snapshot managers, and remote console. • VM Monitor (VMM) process is responsible for virtualizing the guest OS instructions, and managing memory mapping. The VMM passes storage and network I/O requests to the kernel, and passes all other requests to the VMX process. There is a VMM for each virtual CPU assigned to a VM. • Mouse Keyboard Screen (MKS) and SVGA processes are responsible for rendering the guest video and handling guest OS user input. When you console into the VM via vCenter client, the work done is charged to this process. This in turn is charged to the VM, and not specific vCPU. If you want to see example of errors in the above process, review this KB article.
Layer D	Unmet Demand (CPU Ready, CPU Co-Stop, CPU Overlap, CPU VM Wait, RAM Contention, VM Outstanding IO). The Guest OS experiences a frozen time or slowness. It's unaware what it is, meaning it can't account for it.

I've covered the difference in simple terms, and do not do justice to the full difference. If you want to read a scientific paper, I recommend this paper by Benjamin Serebrin and Daniel Hecht.

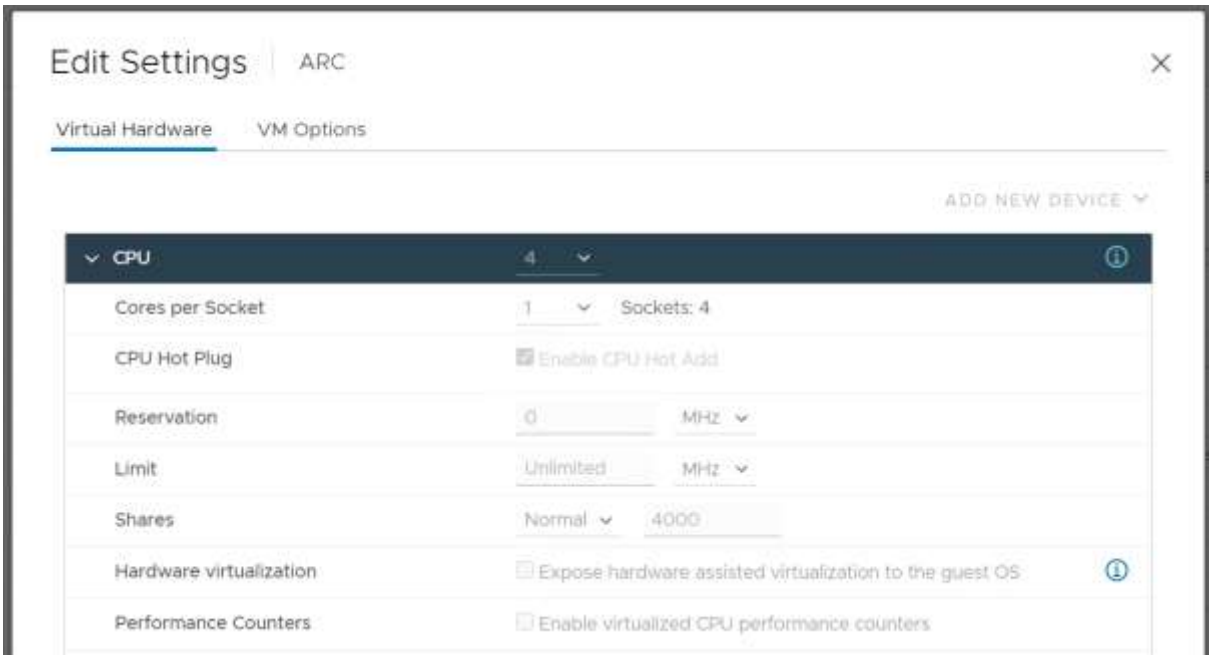
Resource Management

vSphere uses the following to manage the shared resources:

- Reservation
- Limit
- Share
- Entitlement

Reservation and Limit are absolute. Share is relative to the value of other VMs on the same cluster.

Unlike a physical server, you can configure a Limit and a Reservation on a VM. This is done outside the Guest OS, so Windows or Linux does not know. You should minimize the use of Limit and Reservation as it makes SDDC operations more complex.



Reservation represents a guarantee. It impacts the Provider (e.g. ESXi) as that's where the reservation takes place. However, it works differently on CPU vs RAM.

CPU	CPU Reservation is on demand. If the VM does not use the resource, then it does not come into play as far as the VM is concerned. The reservation is basically not applied. Accounting wise, it does not impact CPU utilization metrics. Run, Used, Demand, Usage do not include it. Their value will be 0 or near 0 if the Guest OS is not running.
RAM	Memory Reservation is permanent, hence impacts memory utilization metric. The Memory Consumed counter includes it even though the page is not actually consumed yet. If you power on a 16 GB RAM VM into a BIOS state, and it has 10 GB Memory Reservation, the VM Consumed memory counter will jump to 10 GB. It has not actually consumed the 10 GB, but since ESXi has reserved the space, it is not available to other VMs. If it's not yet used, then it does not take effect. Meaning ESXi Host does not allocate any physical RAM to the VM. However, once a VM asks for memory and it is served, the physical RAM is reserved. From then on, ESXi continues reserving the physical RAM even though the VM is no longer using it. In a sense, the page is locked despite the VM become idle for days.

Limit should not be used as it's not visible to the Guest OS. The result is unpredictable and could create a worse performance problem than reducing the VM configuration. For CPU, it impacts the CPU Ready counter. For RAM, in the VMX file, this is sched.mem.max.

Entitlement

Entitlement means what the VM is entitled to. It's a dynamic value determined by the hypervisor. It varies every second, determined by Limit, Shares and Reservation of the VM itself and any shared allocation with other VMs running on the same host. For Shares, it certainly must consider shares of other VMs running on the same host. A VM can't use more than what ESXi entitles it.

Obviously, a VM can only use what it is entitled to at any given point of time, so the Usage counter cannot go higher than the Entitlement counter.

In a healthy environment, the ESXi host has enough resources to meet the demands of all the VMs on it with sufficient overhead. In this case, you will see that the Entitlement and Usage metrics will be similar to one another when the VM is highly utilized.

The numerical value may not be identical because of reporting technique. vCenter reports Usage in percentage, and it is an *average* value of the sample period. vCenter reports Entitlement in MHz and it takes the *latest* value in the sample period. This also explains why you may see Usage a bit higher than Entitlement in highly-utilized vCPU. If the VM has low utilization, you will see the Entitlement counter is much higher than Usage.

Overhead vs Not Overhead

Be careful not to lump every additional load as overhead.

- Overhead means it's mandatory (cannot be avoided) and has negative impact (such as slower performance or more resource required).
- Not Overhead means it's optional. You do not have to use the feature. Typically, it brings new capabilities that is often not possible to achieve without virtualization.

Let's list some examples:

Overhead	ESXi kernel. While it delivers value, it's not optional and it's not negligible. It impacts your usable capacity too.
	IO processing by hypervisor. There is an additional processing done by the kernel, which could result in IO blender effect. However, this does not reduce the resource allocated to Guest OS.
	VM CPU and Memory overhead for the VM Monitor layer. This is a small amount and operationally negligible.
	ESXi memory consumed and CPU used by vSAN processes.
	VM log files. VM is a layer on its own and the log provides necessary observability.
Not Overhead	VM snapshot. Snapshot is optional and it delivers new functionalities not available in Guest OS.
	VM memory snapshot. This does not have the same purpose with hibernation file

	inside Windows or Linux. This feature enables memory overcommit at ESXi level.
	vSphere HA. The extra ESXi hosts provides availability protection.
	vSAN Failures-to-Tolerate policy. They provide availability protection since vSAN does not use hardware-level redundancy. For workloads where the VM is transient and you have the master template, you can set this to 0 (no protection).

In addition, there is buffer. There are at least 3 types of buffers:

- Capacity buffer. This reduces the Usable Capacity.
- Cost buffer. This reduces the Sellable Capacity. This is typically on top of the Usable Capacity.
- Disaster Recovery buffer
- Security Attack buffer. This is sometimes call the clean room.

vMotion

Is it an overhead? It is not an overhead to the VM, but there can be impact to the VM metrics.

For example, unshared swap page in the source ESXi may become a regular memory page on the destination. This will increase the Consumed metric of the VM on the destination.

Similarly, sometimes the version of ESXi running on source and destination is different and may use different compression algorithms for memory compression. In that case, a compressed page may become a regular page in the destination increasing consumption.

Metric Mastery

There are thousands of metrics across a diverse architecture. How do you master them?

One way is to see the commonality. I studied The USE Method by Brendan Gregg, The RED Method by Weave Works, and the Golden Signal by Google. Based on their strengths and weaknesses, I came up with **The Triple See Method**, which is designed specifically for Broadcom VCF.

Triple See Method

There are 3 types of metrics you need to see and verify as 1 integrated set, before you make your final conclusion. These **Triple See** metric-type are peers, and they have their own purpose. They map well with the Pillar of Operations.

Type of Metrics	Primary Application
Contention	Performance Management
Consumption	Capacity Management Cost Management Sustainability
Context	Configuration Management Compliance Management Inventory Management

Let's explain further:

Contention	They measure something bad. They can be further divided by their impact • Performance. Metrics such as contention, latency, and queue, impact the performance of the system. The system is not down, meaning nothing is dropped, but the overall throughput is reduced. • Availability. Metrics such as errors and dropped impact the availability. Most system can recover from soft errors by re-attempting the operations (e.g. retransmit the dropped packet, recalculate the cache, resend the SCSI command). There is lack of observability for such automatic-recovery, and the even the logs may not reveal.
Consumption	They measure something good. A high number is good for the business, if the load is useful (e.g. not a DDOS attack). Hence, make sure both the patterns and values match your expectation. There are 3 types of consumption. Deploy all 3 techniques above accordingly. **Utilization** — Measures the actual consumption. There is on standard naming convention in the industry, so after a while you have a wide variety of names. Examples are IOPS, throughput, active, usage and workload. It is the main input for the capacity family of metrics (Capacity Remaining, Time Remaining, Recommended Size). **Allocation** — Used in capacity when utilization is not useful as value is too low. **Reservation** — Used in capacity to prevent contention from happening. Think of it as seat reservation in a restaurant. It can overlap with utilization because an object can have lower or higher reservation relative to its utilization at different times
Context	They provide answer to the "it depends" type of answer, by accounting for something (e.g. inventory, configuration) and provide context. The context is obviously only useful to someone who can derive insight from that context, else there is no meaning to it. For example, a high number of vMotion depends on your expectation. If you're doing cluster live migration after office hours, you expect the number to match the theoretical limit.

While contention is what you care, consumption gets the limelight as it's easier to monitor and simpler to explain. Also, many systems do not scale well. Their "performance" drops when reaching certain level of utilization. Take a parallel database. As you add more nodes, the overall throughput drops as the nodes spend more time maintaining overall integrity among them. The CPU utilization of each node gets higher, only to be spent on overhead activities. In this case, what you should do is measures the overhead and the metric you refer to as "performance". Do not use the CPU utilization to represent all these metrics.

There is a tendency to monitor utilization as if it is a pillar of operations. Just like contention, utilization is not something you manage. Yes, you monitor utilization, but you monitor it for a reason. By itself, utilization has no meaning. The meaning depends on the purpose.

Primary | Secondary Metrics

Metrics can be grouped into 2.

Primary	Secondary
The "What" It defines the situation.	The "Why" It explains. It covers the possible causes behind the value of the primary metric.
Typically only 1 metric per use-case	Typically many metrics to explain that single primary metric.
Typically can be color coded	Some can be color coded, some cannot as it's contextual
The unit is normally percentage, where 0% is bad and 100% is good	Unit varies. Examples are GB, MHz, packets/seconds, and milliseconds.
Used in Monitoring	Used in Troubleshooting
Example: Capacity Remaining (%)	Supporting secondary metrics: • CPU Utilisation • Memory Usable Capacity • CPU Allocation
Example: VM Performance (%)	Supporting secondary metrics: • VM peak vCPU Ready among all the VM vCPU • VM peak Read/Write latency among all the VM virtual disks • VM CPU Context Switch

Collection Interval

When you collect a metric, you have a choice on what to collect:

1. Collect the data at that point in time.
   Example: collect the value at 00:00:00.

2. Collect the average of all the data within the collection cycle.
   Example: collect the average of all values from 00:00:00 to 00:05:00.
3. Collect the maximum (or minimum) of all the data within the collection cycle.

The 1st choice is the least ideal, as you will miss majority of the metric. For example, if you collect every 5 minutes, that means you collect the data of the 300th second, and miss 299 seconds worth of data points. On the other hand, it's good when the latest data is more valuable, such as disk space and free memory.

The 2nd choice gives you the complete picture, as no data is missing. The limitation is your collection interval can't be too long for the use case you're interested in.

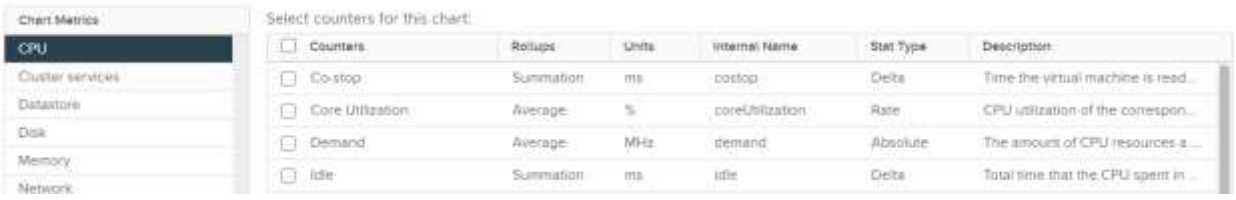
Comparing the 2 choices, the 1st choice will result in wider fluctuations. You will have a higher maximum and lower minimum over time. Telegraf chooses the 2nd choice, while Tools choose the 1st choice. You can see below the result. Overall, their pattern will be similar, especially for something relatively stable such as memory consumption and disk space consumption.



The 3rd choice complements the 2nd choice by picking the worst. That means you need 2 number per metrics for certain use case.

As you collect regularly, you also need to decide if you reset to 0, or you continue from previous cycle. Most metrics reset to 0 as accumulation is less useful in operations.

Let's take a look at what you see at vCenter UI, when you open the performance dialog box. What do the columns Rollups and Stat Type mean?



Stat Type explains the nature of the metrics. There are 3 types:

Delta	The value is derived from a running counter that perpetually accumulates over time.

	What you see is difference between 2 points in time. As a result, all the units in milliseconds are of delta type.
Rate	The value measures the rate of change, such as throughput per second. Rate is always the average across the 20 second period. Note: there are metrics with percentage as unit and rate as stat type. I'm puzzled why.
Absolute	The value is a standalone number, not relative to other numbers. Absolute can be latest value at 20th second or the average value across the 20 second period.

Some common units are milliseconds, MHz, percent, KBps, and KB.

Metrics in MHz is more complex as you need to compare with the ESXi physical CPU *static* frequency. In large environments, this can be operationally difficult as you have different ESXi hosts from different generations or sport a different GHz. This is one of the reasons why I see vSphere cluster as the smallest logical building block. If your cluster has ESXi hosts with different frequencies, these MHz-based metrics can be difficult to use, as the VMs get vMotion-ed by DRS.

Why Milliseconds as Unit?

vSphere uses 3 types of units for CPU: millisecond, MHz and %.

Of the 3, the millisecond is the source. Time is the raw unit, meaning both the percentage unit and the MHz unit are derived from it, because they are expressed as the average/minimum/maximum over time. When we see the CPU demand is 2 GHz at 9:00:00 am what vSphere likely means is it the average from previous collection. It is not a point in time.

Time as a unit to measure CPU utilization does not seem logical. Where does it come from and why?

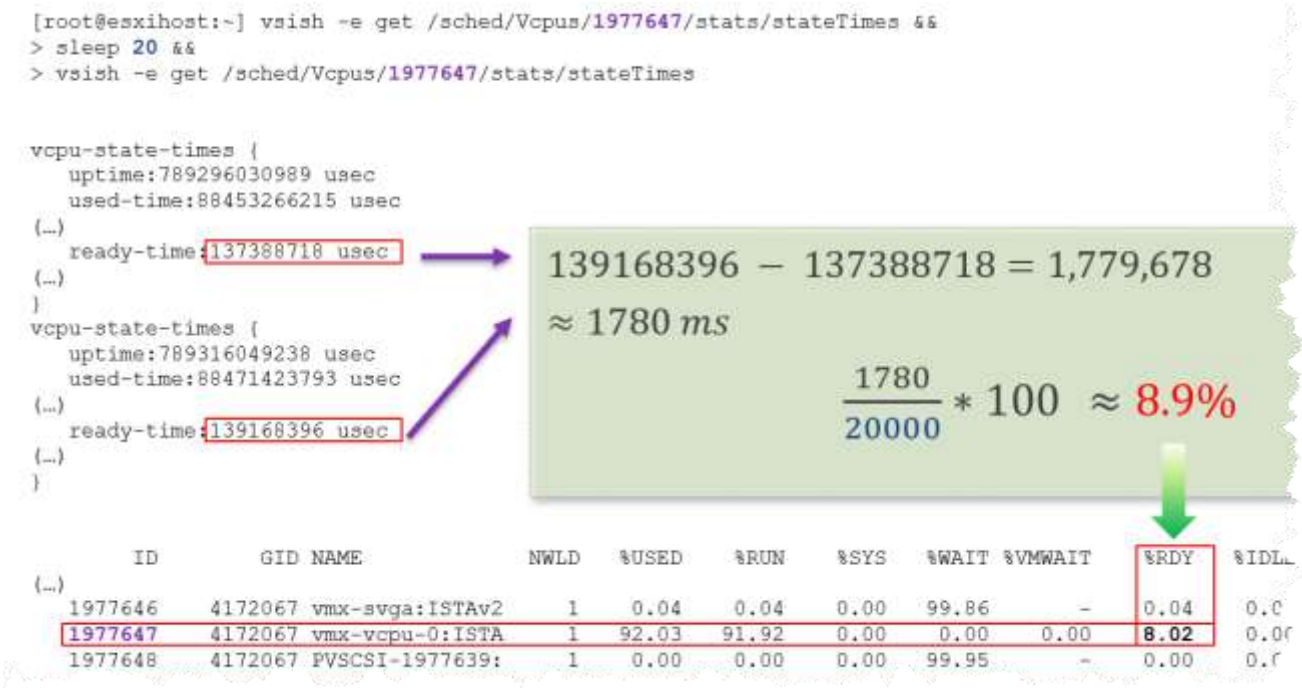
Hint: the stat type is Delta.

To answer that, we need to see from the ESXi kernel scheduler point of view. Think in terms of the passage of time and the amount of CPU cycles that get completed during that time. A CPU core running at 2 GHz will get 2x CPU cycles completed compared with a core running at 1 GHz. The same goes with Hyper Threading. You get less cycles completed when there is a peer thread competing at the same time.

What you think as utilization or usage or demand or used, it will be easier if you see them as cycles, once you make that small paradigm shift.

Let's take VM CPU Ready. The following is taken from ESXi vsish[1] command. It shows that the original, raw counter is actually a running number. To calculate the CPU ready of a given time period, we need to subtract the last number from the first number. To convert to percentage, we divide over the collection, which is 20000 ms in the screenshot.

[1] VMkernel System Information Shell. It is a hidden, interactive command-line utility used to view and manipulate the internal state of the ESXi VMkernel. Think of it like the /proc or /sys filesystem in Linux; it exposes a hierarchical tree of nodes representing hardware, drivers, networks, and virtual machine.



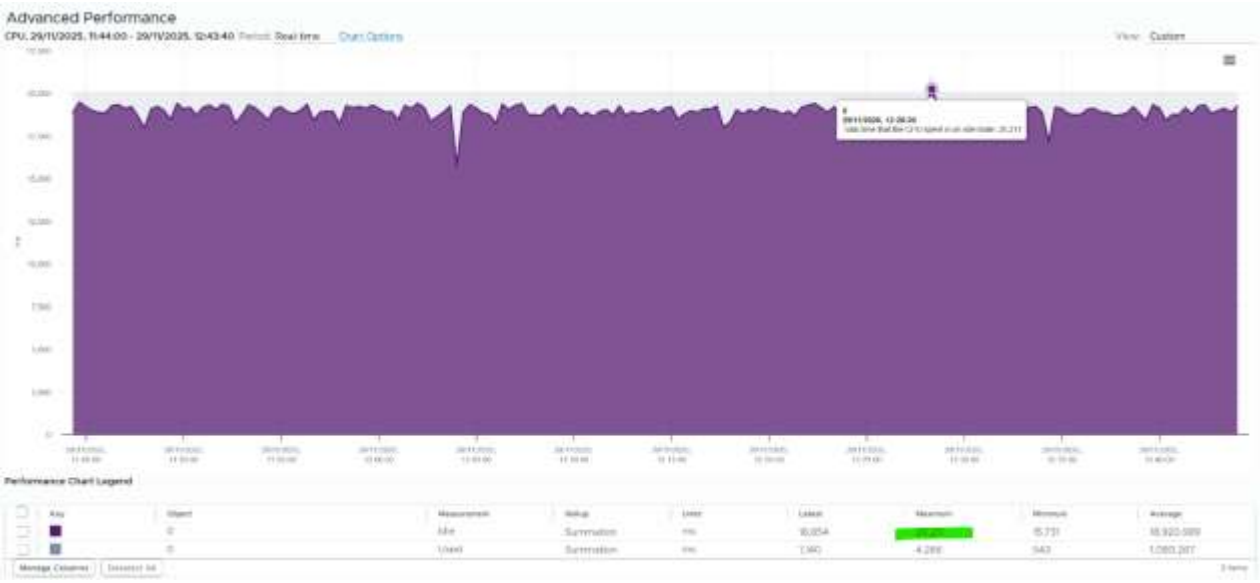
In the above, the slightly different values are due to different time in sample interval start and end.

I'll take another example, to show that the original unit is time (microsecond, not millisecond).

```
/sched/groups/169890525/stats/cpuStatsDir/> cat /sched/groups/169890525/stats/cpuStatsDir/cpuStats
group CpuStats {
   number of vsmps:7
   size:19
   used-time:905379300543 usec
   latency-stats:latency-stats {
      cpu-latency:798578245914 usec
      memory-latency:memory-latency {
         swap-fault-time:0 usec
         swap-fault-count:0
         compress-fault-time:0 usec
         compress-fault-count:0
         mem-fault-time:17939139 usec
         mem-fault-count:3834600
      }
      network-latency:0 usec
      storage-latency:0 usec
```

In vSphere UI and API, the counter for CPU Latency is percentage. But in the above, you can see that it's true unit is microseconds.

Because of the above, when collection is running late or the source is busy, the sum may exceed 20000 ms by a bit. In that case, you may see something like this.



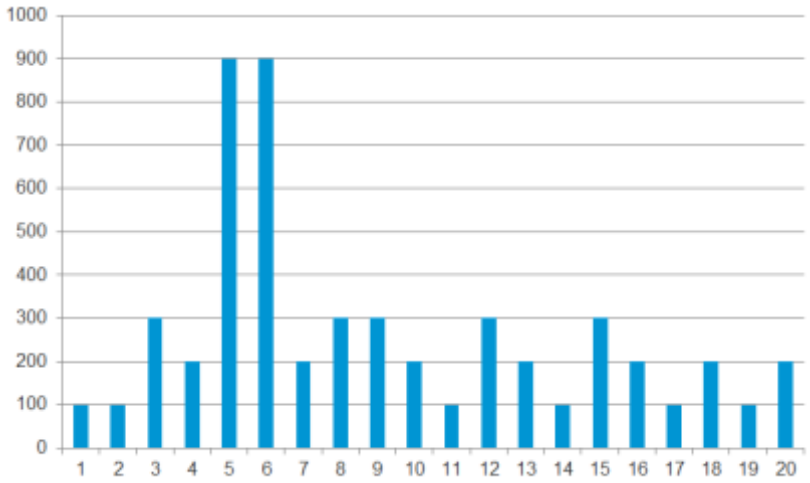
Rollup Type

The **Rollups** column tells you how the data is rolled up to longer time period. The types of roll up are:

Average	This means taking the average of the collected values. vCenter may average 20 values and report it every 20 seconds as a single value.
Maximum	This means taking the highest value. Useful when you want to catch the outlier.
Minimum	The opposite of above. Useful in situations such as free memory.
Summation	It is actually average for those metrics where accumulation makes more sense. For example, CPU Ready Time gets accumulated over the sampling period. vCenter reports metrics every 20 seconds, which is 20000 milliseconds. This is why the number keeps going up as you roll up to bigger object.
Latest	This is the most recent value. This is useful in cases such as disk space, where you care more about the present value.

VCF Operations adds percentile, which is useful when you want to ignore outlier.

The following table shows a VM has different CPU Ready Time on each second. It has 900 ms CPU Ready on the 5th and 6th second, but has lower number on the remaining 18 seconds.



Over a period of 20 seconds, a VM may accumulate different CPU Ready Time for each second. vCenter sums all these numbers, then divides it by 20,000. This is actually an average, as you lose the peak within the period.

Latest, on the other hand, is different. It takes the last value of the sampling period. For example, in the 20-second sampling, it takes the value between 19th and 20th seconds. This value can be lower or higher than the average of the entire 20 seconds period. Latest is less popular compared with average as you miss 95% of the data.

Rolling up from 20 seconds to 5 minutes or higher results in further averaging, regardless of the rollup techniques (summation or average). This is the reason why it is better to use VCF Operations than vCenter for data older than 1 day, as vCenter averages the data further, into a 0.5 hour average.

Because the source data is based on 20-second, and VCF Operations by default averages these data, the "100%" of any millisecond data is 20,000 ms, not 300,000 ms. When you see CPU Ready of 3000 ms, that's actually 15% and not 1%.

By default, VCF Operations takes data every 5 minutes. This means it is *not* suitable to troubleshoot performance that does not last for 5 minutes. In fact, if the performance issue only lasts for 5 minutes, you may not get any alert, because the collection could happen exactly in the middle of those 5 minutes. For example, let's assume the CPU is idle from 08:00:00 to 08:02:30, spikes from 08:02:30 to 08:07:30, and then again is idle from 08:07:30 to 08:10:00. If VCF Operations is collecting at exactly 08:00, 08:05, and 08:10, you will not see the spike as it is spread over two data points. This means, for VCF Operations to pick up the spike in its *entirety* without any idle data, the spike *may* have to last for 10 minutes.

VCF Operations is capable of storing the individual 20-seconds data. But that would result in 15x more data. In most cases, what you want is the peak among the 15 data points.

The Collection Level in vCenter, shown in the following table, does *not* apply to VCF Operations. Changing the collection level does not impact what metrics get collected by VCF Operations. It collects majority of metrics from vCenter using its own filter, which you can customize via policy.

Statistics Levels	Metrics
Level 1	Cluster Services (VMware Distributed Resource Scheduler) – all metrics CPU –entitlement, total MHz, usage (average), usage MHz Disk – capacity, max Total Latency, provisioned, unshared, usage (average), used

	Memory – consumed, mem entitlement, overhead, swap in Rate, swap out Rate, swap used, total MB, usage (average), balloon, total bandwidth (DRAM or PMem) Network – usage (average), IPv6 System – heartbeat, uptime VM Operations – num Change datastore, num Change Host, num Change Host datastore
Level 2	Level 1 metrics, plus the following: CPU – idle, reserved Capacity Disk – All metrics, excluding number Read and number Write. Memory – All metrics, excluding Used, maximum and minimum rollup values, read or write latency (DRAM or PMem). VM Operations – All metrics
Level 3	Level 2 metrics, plus the following: Metrics for all metrics, excluding minimum and maximum rollup values. Device metrics
Level 4	All metrics, including minimum and maximum rollup values.

Take note: vSAN API gives you the last data, not the average or peak of the entire period. Since VCF Operations collect every 5 minutes, you get the data for the 300th second.

Real Time Collection

Do we really need real-time collection and analysis for every single metrics, on every single objects, 24 x 7?

We collect the metrics for a reason, such as performance and capacity. The reasons dictate the frequency for each type of metrics.

Take note that how frequent you collect is not the same with how granular the data points. For example, VCF Operations collect every 5 minutes by default from vCenter, but it grabs 15 data points in 1 collection cycle. For majority of the data, it averages these 15 data points and store as 1 number.

Use Case	Collection Point	Collection Frequency
Performance: Profiling	1 – 20 seconds for all counters	1 – 20 seconds
Performance: Troubleshooting	1 - 20 seconds for raw contention, 5 minutes for everything else. More explanation after this table.	5 minutes for both
Performance: SLA	5 minutes. Why SLA differs to troubleshooting and why 5-minute is the sweet spot is covered in the book VMware Operations Transformation, 4th edition.	5 minutes
Capacity	15 minutes for all. Value is the average over 15 minutes, not the peak.	
Cost	Functionally, you do not need 15-minute granularity. Operationally, it's safer to	

Compliance	do 15 minutes. If there is collection failure, either due to collector or target system, you only lose 15 minutes' worth of data.
Sustainability	
Inventory	

Performance Troubleshooting

For troubleshooting, you want per-second data. Who does not want sharper visibility? However, there are potential problems:

1. It may not be possible. The system you're monitoring may not be able to produce the data, or it comes with capacity or performance penalty.
2. It's expensive. Your monitoring system might grow to be as large as the systems being monitored. You could be better off spending the money on buying more hardware, preventing the problem to begin with.
3. You get diminishing return. The first data point is the most valuable. Subsequent data points are less valuable if they are not providing new information.
4. The remediation action is likely the same as there are only a handful of things you can do to fix the problem. The number of problems outweigh the actual solution.

So what can you do instead?

Begin with the end in mind. Look at the solution (e.g. add hardware, change some settings) and ask what metrics are required. For each required metrics, ask what granularity is required.

I find that 1 – 20 second is only required for the contention-type of metrics. For utilization-type and contextual-type, I think 5 minute is enough. You need higher resolution when the contention-type metrics do not exist. For example, there is no metric for network latency and packet retransmit at VM level. All you have is packet dropped. To address the missing metrics, use utilization metric such as packet per second and network throughput.

Units

No thanks to lack of consistent implementation among vendors, there is a common confusion on units.

1000 vs 1024

There is confusion between 1024 and 1000.

- Is 1 gigabyte = 1024 megabyte or 1000 megabyte?
- Is 1 gigabit = 1000 mb or 1024 mbit?

The answer from Google is 1000 for both.

Google does not distinguish network throughput and storage throughput.



As expected, Google uses 1 byte = 8 bits

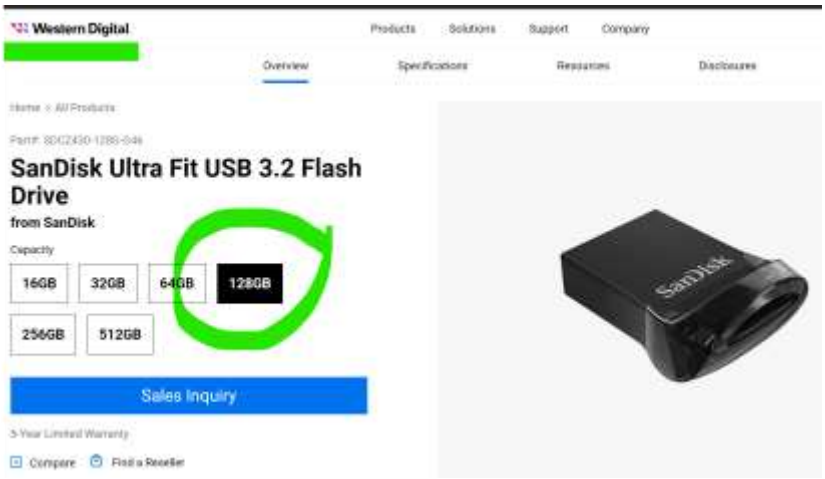


Storage Vendors

However, many products from many vendors use the binary conversion instead of decimal. This is one of those issues, where what's popular in practice is different to what it should be in theory.

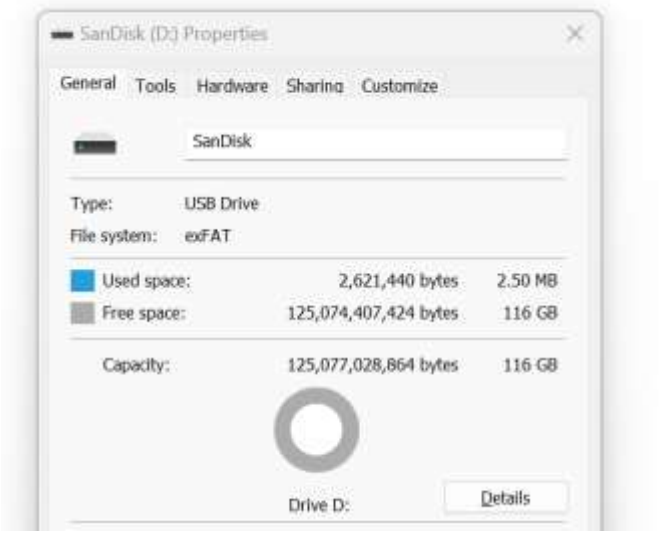
To add further confusion, there is consistency among storage and network vendors. For example, you get shortchanged when you buy storage hardware.

Guess how many GB do you actually get from this 128 GB? This screenshot is from the vendor official website. I think many hardware vendors do the same.



The answer is 116 GB.

The following is the above actual disk in Windows 11.



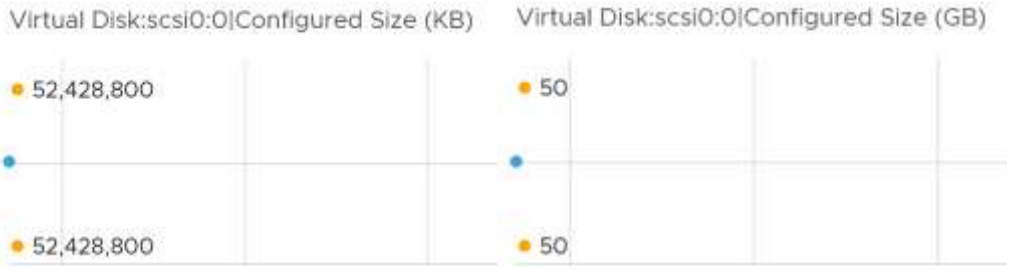
You lost 9%!

Microsoft Windows use 1024 for storage.

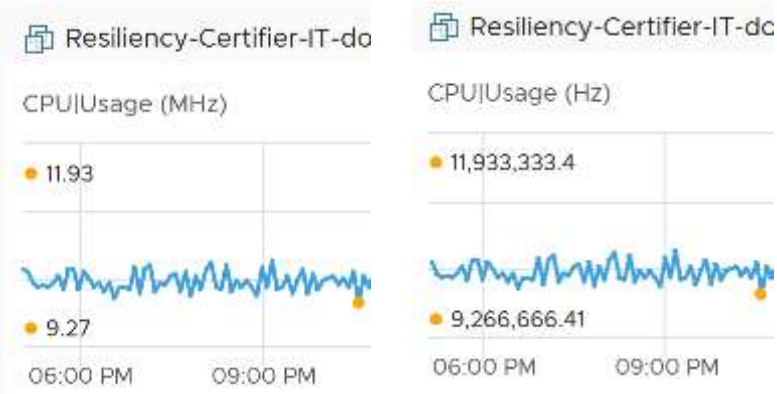
VCF Operations

VCF Operations use 1024 for storage, 1024 for network, 1024 for memory, but 1000 for CPU.

Here is the proof for storage. Yes, I validated network and memory too.



VCF Operations use 1000 for CPU



Kilo vs Kibi

To address the confusion, the committee at International System of Quantities came up with a new set of names for the binary units. Instead of kilo, mega, giga, they use kibi, mebi, gibi, tebi, and pebi.

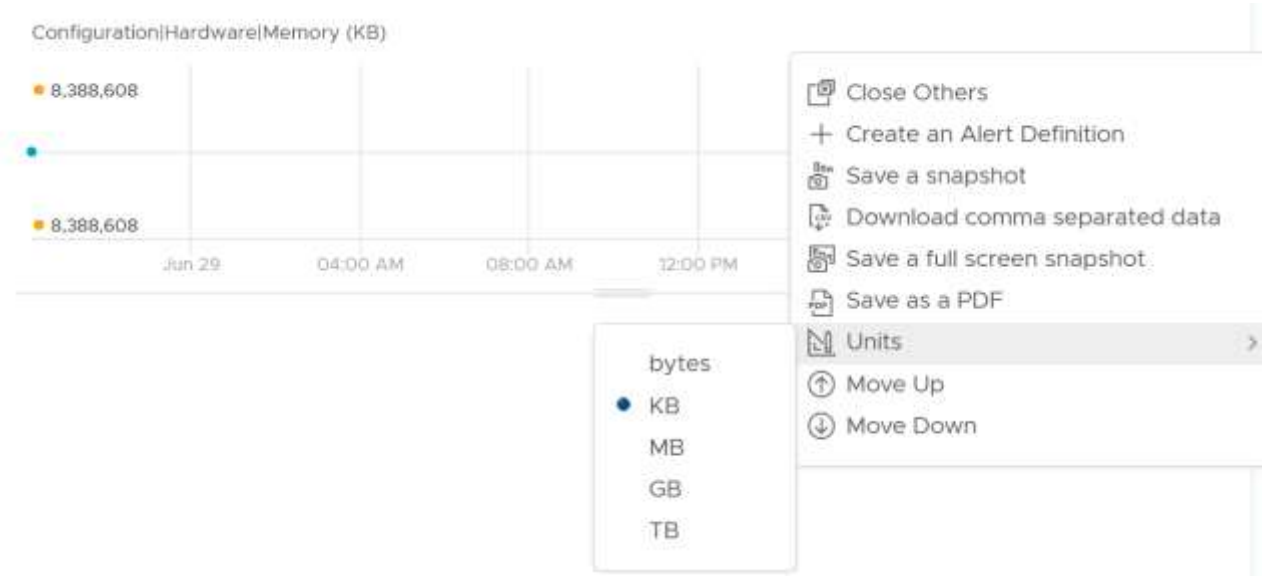
It is confusing to drop familiar terms like kilo, mega and giga. I prefer kilobi instead of kibi as it shows the relationship to the commonly known units. Or if you want to emphasize the binary nature, perhaps kilo2byte, mega2byte, giga2byte as the name.

Let's take an example

- 1 Kibibyte = 1024 bytes. That means 1 Kibibyte = 1.024 KB.
- 1 Gibibyte = 1024 Mebibytes = 1,073,741,824 bytes

The abbreviation is also changed from K, M, G to Ki, Mi, Gi, where the letter ***i*** is small case. This confuses the population further as Ki can mean Kilo or Kibi. I'd have chosen another letter instead of ***i***.

VCF Operations still uses the traditional (KB, MB, GB) as shown in the following screenshot.



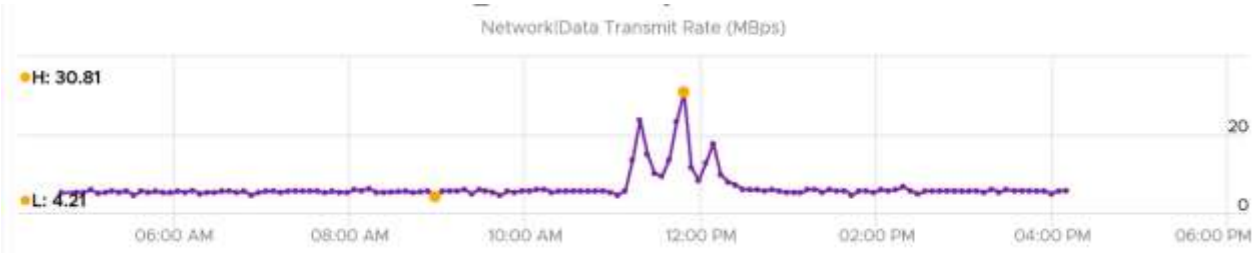
Note the conversion from byte to bit remains. 1 byte = 8 bit.

Bit vs Byte

Do you use Byte/second or bit/second?

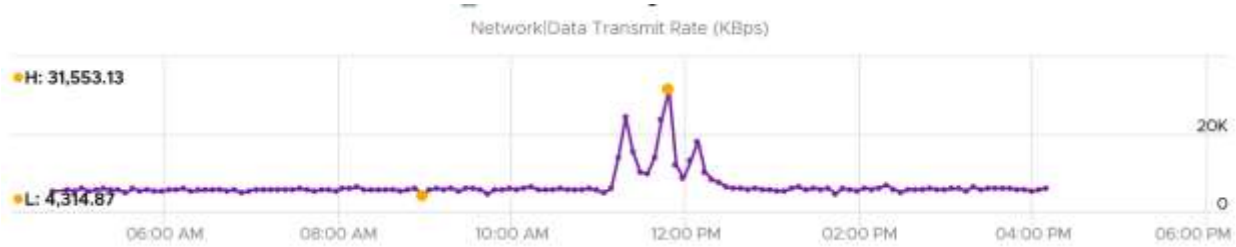
To me, it depends on the context. If you talk about disk space, you should use byte. You measure the amount of disk space read or written per second. If you talk about network line, you should use bit. You measure the amount of SCSI blocks travelling inside ethernet or FC cable. Pearson uses 1024 for disk space, and 1000 for transmission speed, in their certification. There are other references, such as gbmb.org, NIST, and Lyberty. In short, there is really no standard.

The following is network transmit. It's showing 30.81 MBps. So this is a rate, showing bandwidth consumption or network speed.



What would it show if you convert into KBps?

30810, if it uses 1000.



Since vRealize treats 1 Mega = 1024 Kilo, the above is what you get.

Since it's network, let's convert into bit.

What do you expect you get in Mbps?



You get 31 x 553.13 x 8 bits / 1024 = 246 / 51.

Aggregation

We discussed in earlier part of the book about data collection. After collecting lots of data across objects and across time, how do you summarize so you get meaningful insight?

Aggregating to a higher-level object is complex as there is no lossless solution. You are representing a wide range of values by picking up 1 value among them, so you lose some details. The choices of techniques are mean, median, maximum, minimum, percentile, sum and count of.

The Problem with Average

The default technique used by many observability tools is the average() function as that represents every value to be aggregated. Average is great across time, but not across members of a group.

The problem is it will mask out the outlier unless they are widespread. By the time the average performance of 1000 VMs is bad, you likely have a hundred VMs in bad shape.

Let's take an example. The following table shows ESXi hosts. The first host has CPU Ready of 149,116.33 ms. Is that a bad number?

Name	Cores	Co-stop	Ready	VM
10.73...	36 Cores	7,627.33 ms	149,116.33 ms	67
10.73...	36 Cores	4,832.73 ms	128,698.73 ms	45
evn-l...	12 Cores	26.67 ms	18.2 ms	3
10.26...	8 Cores	2.33 ms	238.53 ms	7

It is hard to conclude. It depends on the number of running vCPU, not the number of physical cores.

That host has 67 running VMs, and each of those VMs can have multiple vCPU. In total there are 195 vCPU. Each vCPU could potentially experience CPU Ready of 20,000 ms (which is the worst possible scenario).

If you sum the CPU Ready of the 67 VM, what number would you get?

Running VMs in the ESXi Host

Name	Ready	Co-Stop	vCPU
APP...	2,108.6 ms	6.47 ms	2 vCPUs
vRO...	1,955.6 ms	1.8 ms	2 vCPUs
jjuch...	1,915.4 ms	0 ms	4 vCPUs
DND...	1,899 ms	2.13 ms	8 vCPUs
WEB...	1,797.87 ms	0.87 ms	2 vCPUs
Shar...	1,759.87 ms	3.8 ms	2 vCPUs
NSX...	1,614.27 ms	5.13 ms	4 vCPUs
DND...	1,556.07 ms	0.93 ms	4 vCPUs
Sum	149,116.33 ms	7,627.33 ms	195 vCPUs

1 - 20 of 50 items < 1 2 3 >

You're right, you get the same number reported by the ESXi host.

This means the ESXi CPU Ready = Sum (VM CPU Ready), and the VM CPU Ready = Sum (VM vCPU Ready).

Because it's a summation of the VMs, to convert into % requires you to divide with the number of running VM vCPU.

```
ESXi CPU Ready (%) = ESXi CPU Ready (ms) / Sum (vCPU of running VMs)
```

Are the CPU Ready values equally distributed among the VMs? What do you think?

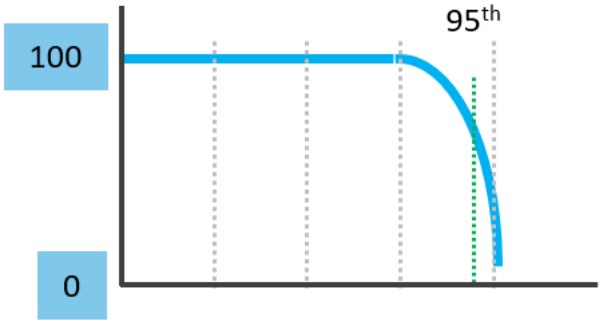
It depends on many settings, so there is a good chance you get something like the following. This heat map shows the 67 VMs on the above host, colored by CPU Ready and sized by VM CPU configuration. You can see that the larger VMs tend to have higher CPU ready, as they have more vCPU.



Lagging vs Leading Indicators

When it comes to performance management, you need to see the problem while it's still early, when only a small percentage of users or applications are affected. For that, you need a leading indicator, not a lagging indicator (after the fact). Leading indicators complement the lagging indicator by giving the early warning, so you have more time to react.

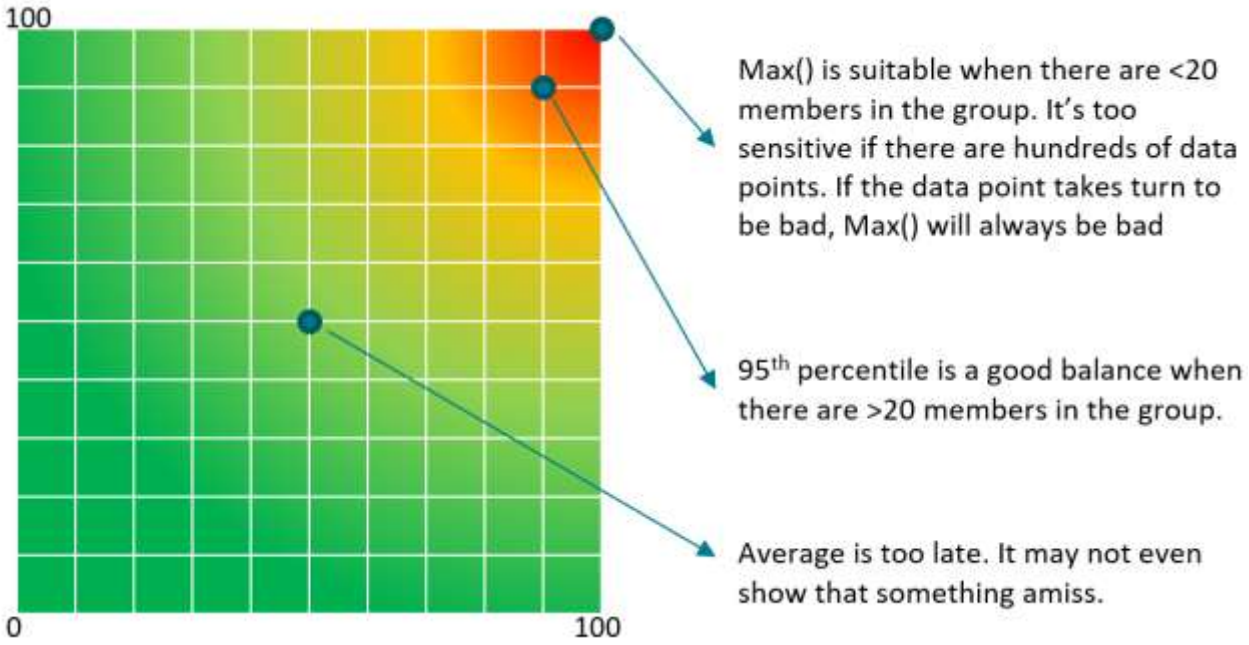
Performance use the contention metric as its primary input. The problem with contention metrics is it drops (or spike, depending on how you see it) suddenly, typically at the point of overcommit. It differs to consumption metrics which goes up towards 100%.



As a result, average is not suitable for rolling up performance metrics to higher level parents. For example, a VDI system that was designed for 1000 users should serve the first 1000 well, and it should struggle after the capacity is exceeded.

Average is a lagging indicator. The average of a large group tends to be low, so you need to complement it with the peak. On the other hand, the absolute peak can be too extreme, containing outliers.

The following chart shows where Maximum() picks up the extreme (outlier) while average fails to detect the problem. This is where the worst 5th percentile or the worst 1st percentile makes more sense.



These are the techniques to complement average(). Depending on the situation, you apply the appropriate technique.

<table>
<tr><td>Worst()</td><td>This returns the worst value of a group.
It's suitable when the number of members is low, such as ESXi hosts in a cluster or containers in a Kubernetes pod. It's also suitable for hourly value, as there are only 12 data points.
If you want to ignore outlier, then use Percentile function.</td></tr>
<tr><td>Percentile()</td><td>It is similar to the Worst() function, but it returns the number after eliminating a percentage of the worst. See this handy calculator to learn the percentile function. I've summarized the most common scenarios, showing the worst 5th percentile works well the number of members is less than 100. If the number of members is >250, I'd take 97th percentile (3 standard deviations).
<table>
<tr><th>Count</th><th>Worst 5th Percentile</th><th>Exclusion</th></tr>
<tr><td>10</td><td>9.50</td><td>0.50</td></tr>
<tr><td>20</td><td>19.00</td><td>1.00</td></tr>
<tr><td>40</td><td>38.00</td><td>2.00</td></tr>
<tr><td>50</td><td>47.50</td><td>2.50</td></tr>
<tr><td>100</td><td>95.00</td><td>5.00</td></tr>
<tr><td>200</td><td>190.00</td><td>10.00</td></tr>
<tr><td>500</td><td>475.00</td><td>25.00</td></tr>
</table>
The problem with percentile is it picks a single member. It cannot tell if there is a population problem.</td></tr>
<tr><td>Average of Worsts</td><td>This solves the percentile by averaging all the numbers above the percentile. If you take the average from 95th percentile to 100th percentile, you represent all these numbers. This results in a number that is more conservative than 95th percentile.</td></tr>
</table>

	This is superior to percentile as the band between 95 – 100 may vary. By not hardcoding at a single point, you pick a better representation.
	The limitation of this technique is when you have outlier. It can skew the number. If you suspect that, choose a lower percentile, such as 90th or 92.5th percentile.
Count()	This is different to the Worst() or Percentile(), as you need to define the threshold first. For example, if you do Count of VM that suffers from bad performance, you need to define what bad is. That's why Count() requires you to define the band for red, orange, yellow and green. You can then track the number of objects in the red band, as you expect this number to be 0 at all times. Waiting until an object reaches the red band can be too late in some cases, so consider complimenting it with a count of the members in orange band.
	Count() works better than average() when the number of members is very large. For example, in a VDI environment with 100K users, 5 users affected is 0.005%. It's easier to monitor using count as you can see how it translates into real life.
Sum()	Sum works well when the threshold for green is 0. Even better, the threshold for yellow is 0. With this, you just need to watch when the total is above 0. The main limitation of Sum() is setting the threshold. You need to adjust the threshold based on the number of members. If there are many members, it's possible that they are all in the green, but the parent is in the red.
Disparity()	When members are uniformed and meant to share the load equally, you can also track the disparity among them. This reveals when part of the group is suffering when the average is still good.

In some situations, you may need multiple metrics for more complete visibility. For example, you may need Worst for the depth and Percentile for the breadth. In this case, pick one of them as the primary and the rest as secondary metrics. You can also use worst and worst 5th percentile numbers together for better insight. If the numbers are far apart, that indicates the Worst number is likely an outlier. If the numbers are similar, you have a problem

"Peak" Utilization

One common requirement is the need to monitor for peak. Be careful in defining peak, as by default, averages get in the way.

How do you define peak utilization or contention without being overly conservative or aggressive?

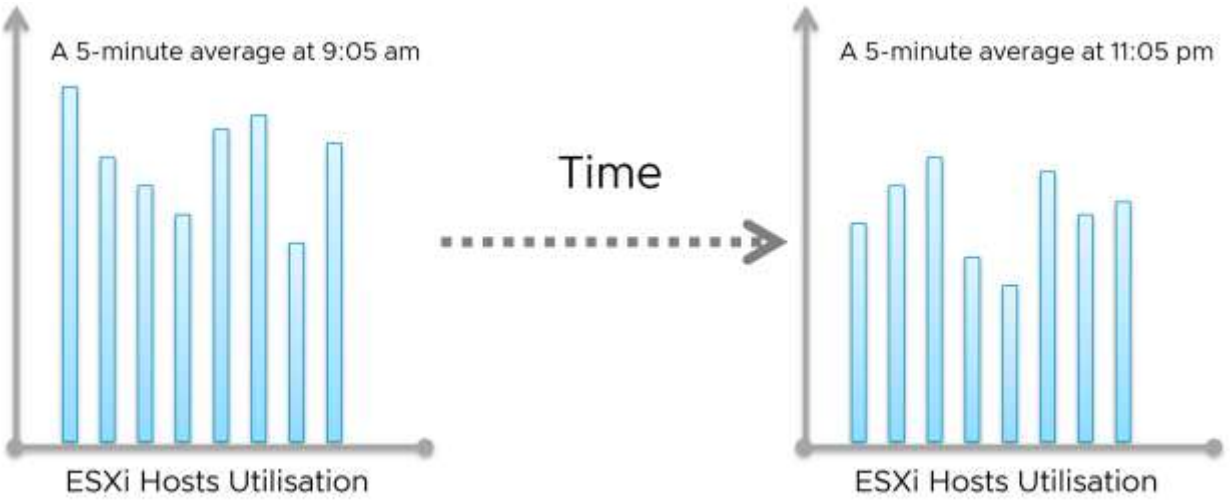
There are two dimensions of peaks.

1. Peak across time of the same metric.
2. Peak among members of the group at the same time

Let's take a cluster with 8 ESXi hosts as an example. The following chart shows the 8 ESXi utilizations.

What's the cluster peak utilization on that day?

The problem with this question is there are 1440 minutes in a day, so each ESXi Host has at least 288 metrics (based on the 5-minute reporting period). So this cluster has 288 x 8 = 2304 metrics on that day. A true peak has to be the highest metric among these 2304 metrics.



To get this true peak, you need to measure across members of the group. For each sample data, take the utilization from the host with the highest utilization. In our cluster example, at 9:05 am, host number 1 has the highest utilization among all hosts. Let's say it hit 99%. We then take it that the cluster peak utilization at 9:05 am is also 99%.

You repeat this process for each sample period (e.g. 9:10 am, 9:15 am). You may get different hosts at different times. You will not know which host provides the peak value as that varies from time to time.

What's the problem of this true peak?

Yup, it might be too sensitive. All it takes is 1 number out of 2304 metrics. If you want to ignore the outlier, you need to use percentile. For example, if you do 99th percentile, it will remove the highest ~23 datapoints.

Take note that the most common approach is to take the average utilization among all the 8 ESXi hosts in the cluster. So you lose the true peak, as each data point becomes an average. For the cluster to hit 80% average utilization, at least 1 ESXi host must have hit over 80%. That means you can't rule out the possibility that one host might hit near 100%.

The same logic applies to a VM. If a VM with 64 vCPUs hits 90% utilization, some cores probably hit 100%. This method results in under-reporting as it takes an average of the "members" at any given moment, then take the peak across time (e.g. last 24 hours).

This "averaging issue" exists basically everywhere in monitoring, as it's the default technique when rolling up. For a more in-depth reading, look at this analysis by Tyler Treat.

There is another nuance of peak.

Depth vs Breadth

There are 2 dimensions of problem:

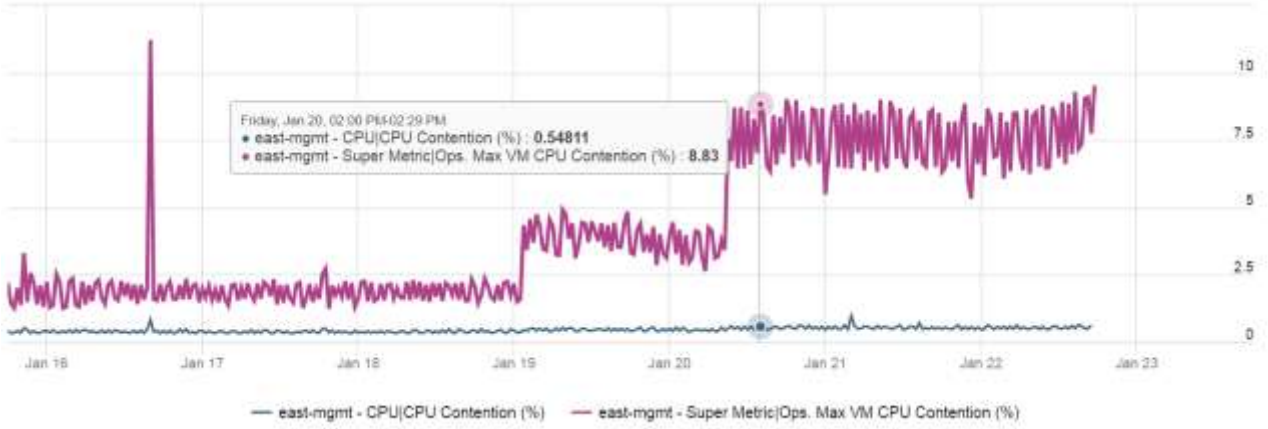
- How deep or acute is the problem?
- How widespread is the problem?

Both measure the severity of the problem.

A deep problem is suitable for a single entity. For example, if a VM has a very high CPU ready, it's worth looking into.

A broad problem is suitable for the population. For example, if many VMs experience a moderate CPU ready, it's worth looking into. You do not want to wait until the CPU ready becomes higher. As a result, use a lower threshold.

What do you notice from the following screenshot? There are 2 metrics, the maroon line shows the worst among all the VMs in the cluster, the pale blue shows the cluster wise average.



Notice the Maximum is >10x higher than the average. The average is also very stable relative to the maximum. It did not move even though the maximum became worse. Once the Cluster is unable to cope, you'd see a pattern like this. Almost all VMs can be served, but 1-2 were not served well. The maximum is high because there is always one VM that wasn't served.

Be careful when you look at metrics at parent object such as cluster and datastore, as average is the default counter used in aggregation. Here is another example. This shows a cluster wise average. What do you think of the value?



That's right. No performance issue at all in the last 7 days. The cluster is doing well. This cluster run hundreds of VMs. What you see above is the average experience of all these VMs, aggregated at cluster level. If there is only a few VMs having a problem, but the majority are not, the above fails to show it.

Now look at the pattern. You can see there are changes in that 1 week period.

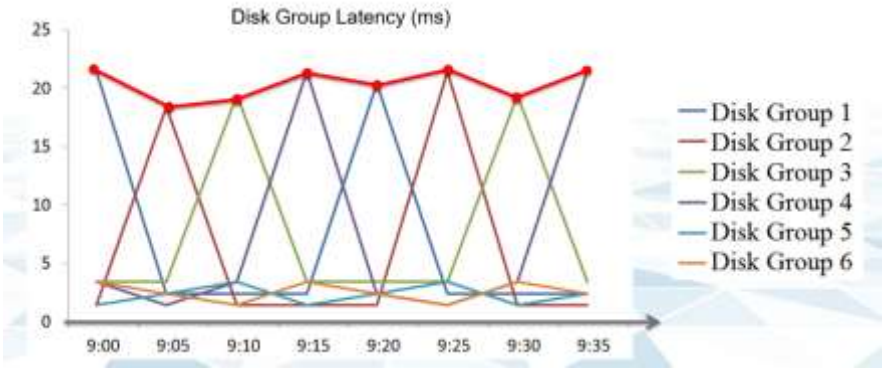
What do you expect when you take the worst of any VM? Would you get the same pattern?



Answer is possible (not always!), if every VM is given the same treatment. They will take turn to be hit.

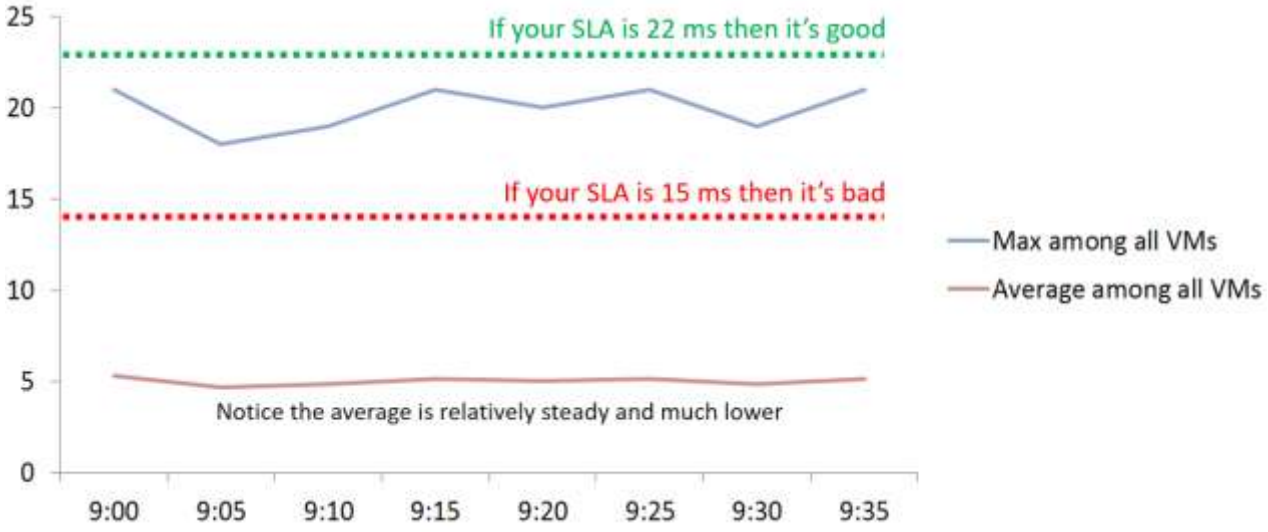
Notice the scale. It's 60x worse.

The following diagram explains how such thing can happen.



The above charts show 6 objects that have varying disk latency. The thick red line shows that the worst latency among the 6 objects varies over time.

Plotting the maximum among all the 6 objects, and taking the average, give us two different results as shown below:



If you don't have the SLA Line, how do you know how much **buffer** you have?

Proactive monitoring requires insights from more than one angle. When you hear that a VM is hit by a performance problem, your next questions are naturally:

- How bad is it? You want to gauge the depth of the problem. The severity also may provide a clue to the root cause.
- How long did the problem last? Is there any pattern?
- How many VMs are affected? Who else are affected? You want to gauge the breadth of the problem.

Notice you did not ask "What's the average performance?". Obviously, average is too late in this case.

The answer to the 3rd question impacts the course of troubleshooting. Is the incident isolated or widespread? If it's isolated, then you will look at the affected object more closely. If it's a widespread problem then you'll look at common areas (e.g. cluster, datastore, resource pool, host) that are shared among the affected VMs.

How do you calculate the breadth of a problem?

There are 2 methods:

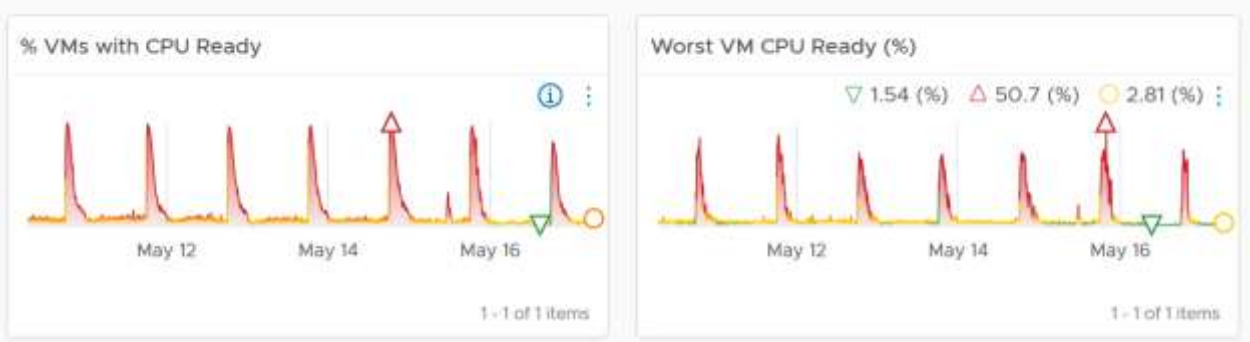
- Threshold based. You determine the percentage of the population above a certain threshold. The limitation is defining the threshold is hard as it depend on the metric.
- Percentile based. You determine the number at certain percentile. I recommend 90th percentile as average is too late and you want a leading indicator. The limitation is you don't know the percentage of the population.

I recommend the percentile-based as it can be consistently applied to any metric.

The following table uses the threshold-based.

	Example Metrics	Green	Yellow	Orange	Red
How Broad?	% VM with CPU Ready >1%	0 – 2.5%	2.5 – 5	5 – 10	10 - 15
	% VM with RAM Contention >1%	0 – 2.5%	2.5 – 5	5 – 10	10 – 15
	% VM with Disk Latency >10 ms	0 – 2.5%	2.5 – 5	5 – 10	10 – 15
How Deep?	Max VM CPU Ready	0 – 2%	2 – 4	4 – 8	8 – 16
	Max VM RAM Contention	0 – 0.5%	0.5% - 1%	1% - 2%	2% - 4%
	Max VM Disk Latency	0 – 10 ms	10 – 20	20 – 40	40 – 80

The following shows an example where both breadth and dept confirm you have a CPU Ready problem.



Usage Disparity

Imbalance among utilization can reveal a problem, as there are many examples where you expect balance utilization:

- Usage among VM vCPU. If a VM has 32 vCPU, you don't want the first 8 are heavily used while the last 16 are not used.
- Usage among ESXi in a cluster
- Usage among RDS Hosts in a farm
- Usage among Horizon Connection Server in a pod
- Usage among disk in a vSAN disk group
- Usage among web server in a farm

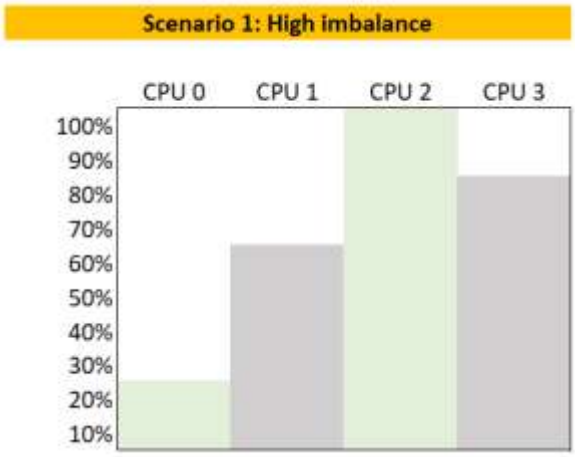
We define imbalance as highest – lowest.

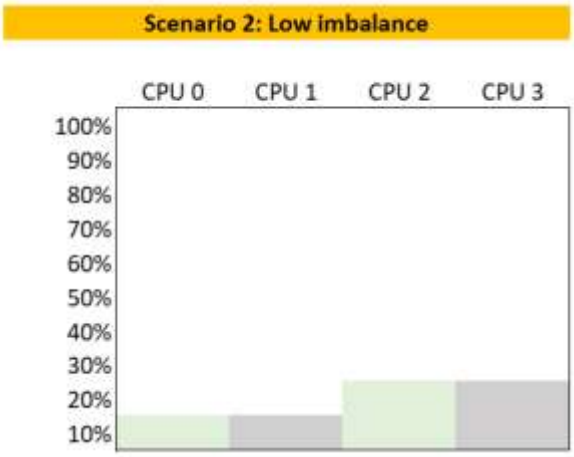
It is expressed in percentage, meaning we need to divide over something. There are 2 options:

- Divide over total. This is a fixed number, as the total is a constant number.
- Divide over max (highest). This is a dynamic number, as the max is fluctuating. The imbalance is relative, as it depends on the value of the Max metric.

Both use cases have their purpose. We are taking the first use for these reasons:

1. That's the most common one. The second use case is used in low level application profiling or tuning, not general IaaS operations.
2. It's also easier to understand.
3. It does not result in high number when imbalance is low in absolute terms. See the charts below





The following calculation shows that using the relatively imbalance results in a high number, which can be misleading as the actual imbalance is only 10%

VM	Utilization
CPU 1	10%
CPU 2	1%
CPU 3	1%
CPU 4	1%
CPU 5	1%
CPU 6	1%
CPU 7	1%
CPU 8	1%
CPU 9	1%
CPU 10	1%
VM Average	**2%**

Highest among VM vCPU	10%
Lowest among VM vCPU	1%
Imbalance	9%
Imbalance over Highest	**90%**
Imbalance over Total	9%

Modelling

There are 3 main reasons why we need to model performance of a system or object into a simple, higher-level set of metrics.

Persona	Monitoring is first performed by Level 1 operators. As first line of defence, they need to cover wide not deep. It's not practical to track many metrics for thousands of objects. In addition, some of the raw metrics require specialist knowledge.
Scale	A single metric enables monitoring at scale. For example, if you have a single metric representing a VM, you can do aggregation on these VMs.
Reporting	Senior IT management prefers to see a higher-level metric as it's easier to relate to the business. They understand VM Performance (%) but may not appreciate CPU Throttled (ms).
Pattern	Combining into a single value unlocks trend analysis as pattern may emerge that mimic business cycles or security attack.

Quantifying something complex with many components is difficult. It's like trying to figure out the inflation rate of a country. It's impossible to have the Consumer Price Index that properly represents the economy as different individuals have different baskets of goods. Even if we could develop the basket for each individual, that basket changes each year, rendering comparison with previous year invalid.

On the other hand, a country needs the number in order to manage its economy. The approximation is certainly much better than nothing.

Let's take another example. This time, we go down to a much smaller scope. A person, you or me. We take annual an health check, performing all sorts of tests, and the results will be a series of metrics (e.g. your bad cholesterol level). Are they 100% accurate for you from young until you are old? Are the guidelines 100% accurate for everyone in your country?

Beside the absolute value, the relative movement of the value over time also provides insight. The patterns and trends over time are useful for management.

Now that we've seen real life examples, can we apply them to IT systems? For example, how do we define the performance of a large system such as vSphere, NSX, Horizon or Kubernetes?

Actual | Potential

One obvious challenge is there are many components that makes up high level metric.

Since the goal is to be proactive, we need 2 metrics:

- 1 measures reality. It quantifies the actual. A low value is bad.
- 1 measures risk. It quantifies the potential. A low value is good.

	Performance (%)	Risk (%)
Purpose	Quantify actual slowness. It is essential to know reality.	Quantify potential slowness. It is essential to know what may come your way.
Metric Type	Contention • Actual • Saturation (zipped, swapped)	Consumption • Allocation • Utilization and imbalance • Reservation
	Since the goal is proactive, this includes saturation signs.	Count of Monster VM, as they increase risk.
Assumption	Configuration (settings) is already optimized, and all availability issues have been dealt with separately. This enables the metrics to focus on contention and consumption.	

Calculation

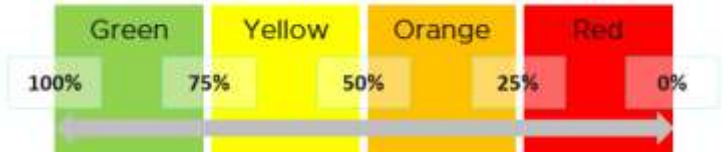
Performance is defined as 0 – 100%.

100%	It means best possible performance, not absolute theoretical figure. This means a perfect score of 100.00% is not possible as certain contention such as disk latency cannot be 0.
0%	It means it's at your worst expectation, not the absolute slowest.
	For example, if you expect 40 ms as the least you can tolerate, then the value will turn to 0% when disk latency hits 40 ms. The value 400 ms will not make it negative.
	This means a value of 50% is likely acceptable cost is far more important

Do not confuse performance with SLA. In SLA, you're aiming to be 100%. In performance, the number 75% is actually pretty good. The reason is we use 4 colors.

We use 4 colors, so we can divide 100% into 4 equal parts. So Green is simply 75% - 100% and Red is simply 0% - 25%. This is more natural than dividing into 3, where you end up with odd numbers such as 33.33% and 66.67%.

The other advantage of having 4 colors is it gives you leading indicator (shown as yellow).



Why don't we make green 95% - 100%? 75% for green sounds rather bad or low.

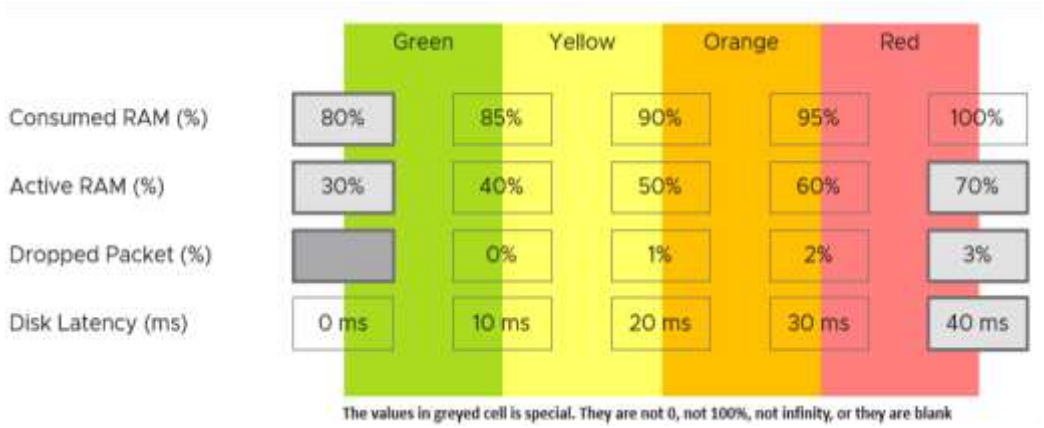
My answer is if you create an unequal distribution, some bands will have to be narrower than others. With uneven bands, you also need to be extra careful when defining the threshold for each metric that make up the KPI. I made the 4 bands equal, so the thresholds are easier to set.

The threshold maps to a timeline.

Red	Take a look as soon as you can. Within a few hours.
Orange	Take a look at today. Anytime, but not tomorrow.
Yellow	Take a look at tomorrow.
Green	All good

Making the threshold easy to set is critical. As you design your KPI, you will vrealize that the threshold is the hardest part. In fact, I exclude metric when I do not feel comfortable with its threshold.

The following KPI uses 4 metrics as its input. Each metric has a set of thresholds for green, yellow, orange and red.



Now that we have the threshold for each metric, we can convert each metric into green – red. The model is also able to handle when the entire range is defined by a single number. This is useful when you want to define green = 0. That means a single packet loss will put the metric into the yellow range already.

What if anything above 0 is red?

You simply set 0 for green, yellow and orange. Within the red zone, you can set 0 – 1, or 0 to something.

Translation

Since the metrics have different units, we need translate to a common unit so we can aggregate.

How do we translate a row?

Let's use an example. Take the Disk Latency (%) metric. It has range from 0 to 40 ms, which maps into the 0 – 100% using the following mapping table.

VM Disk Latency (ms)

	Start	End	Range
Green	0	10	75 - 100%
Yellow	10	20	50 - 75%
Orange	20	30	25 - 50%
Red	30	40	0 - 25%

Actual value	9 ms
relative value	77.5 %
Actual value	11 ms
relative value	72.5 %
Actual value	21 ms
relative value	47.5 %
Actual value	42 ms
relative value	0 %

With the above mapping, we can be precise in assigning the value. For examples:

- 9 millisecond disk latency translates into KPI value of 77.5%, which is green. The reason is green ranges from 75% to 100%, where 0 ms equals to 100% and 10 ms equals to 75%. So each millisecond is around 2.5%.
- 42 millisecond disk latency translates into 0%. It is above the upper threshold of 40 millisecond. Since we do not show negatives, anything above the limit is shown as 0%

Threshold

The threshold is designed to support proactive, not alert-based operations. Hence, the red range does not mean emergency and you must drop everything. It means you need to take a look within the next few hours. This also gives you time to evaluate how many times it falls into the red zone and the overall trend.

The threshold could be argued from 2 ways:

- Scientifically
- “Practically”

Scientifically, a VM does not care what’s stopping it. Whether it’s Ready or Co-Stop or Overlap, the Guest OS does not know. Using this logic, you should set all the threshold the same way. On the other hand, you can follow what happens in production, in healthy environment. These metrics do not follow the same scale.

I take the lowest of the two, as the requirement is proactive monitoring.

Progressive Weightage

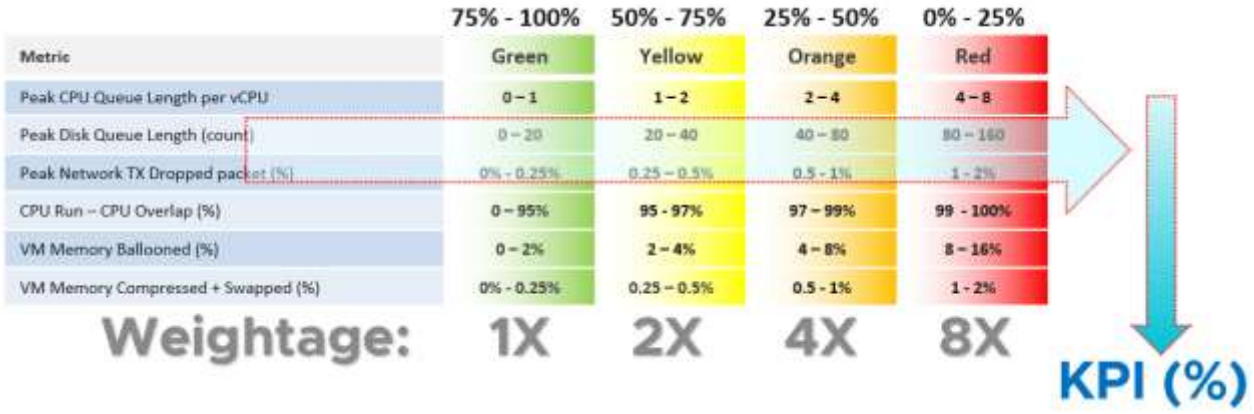
If you have many metrics that make up the KPI, and one of them is red but the remaining is all green, the overall KPI value may not reveal that there is problem. That single red does not have enough weight to bring down the rest.

We assign weightage so that yellow is 2x green, orange is 2x yellow and red is 2x orange. Mathematically, a single red has equal weightage with 8 green. The following table shows that 1 perfect red and 8 perfect green will result in the score of 50.

	Score	Weightage	Adjusted Score
1	100	1	100
2	100	1	100
3	100	1	100
4	100	1	100
5	100	1	100
6	100	1	100
7	100	1	100
8	100	1	100
9	0	8	0
	Total	**16**	**50.00**

That also means that if you have 1 perfect red, and your green are not perfect, you can expect your value to be in the orange category.

This relative weightage plays a key role in determining the threshold. Try to match the actual value so they also correspond to 1x 2x 4x 8x. For example, set the VM disk latency so it goes up from 20 ms → 40 ms → 80 ms → 160 ms. Notice they always double.



Note that this method does not replace assigning different weightage to each metric. You can still do that.

Optional Features

The above model is capable of handling many cases. To handle more complex requirements, apply any of these optional techniques. They can also be used together or individually.

Important Metrics

What if some metrics are relatively more important than others?

Give it a higher weightage. If metric A is 2x more important than metric B, then ensure metric A weightage is 2x of metric B.

If both are equally important, but the chance of metric B hitting bad is 2x higher, then give it 2x the weightage if you want an earlier warning.

The sum of all weightages is 100%. In the following example, the CPU is given highest weightage. Notice they sum to exactly 100%

Metric	Weightage
CPU Usage	40%
Memory Usage	20%
Disk Usage	20%
Network Usage	20%

Multi-table KPI

What if you have a long list of metrics and they are equally important? For example, if you have 21 metrics, and one of them is red while others are green, that single red will not be able to pull the overall KPI down low. If you want the KPI to be low because the issue is severe, you need to split the large table into smaller table. Ensure the grouping is logical.

For example, a Kubernetes cluster can be affected by either infrastructure problem or application (workload) problem. If we combine all the metrics, each metric has a relatively low weightage, as their sum is 100%. We solve this by combining them into 2. We then take the lowest of the 2 sub-KPI metrics.

Creating a 2nd table is also useful for override.

How do you deal with a serious problem that *rarely* happens? What weightage do you give it? Take for example, error packet in Kubernetes. Say you give it 40%, because it is a serious hit on performance. But since it rarely happens, then this 40% is basically a given. If you want to avoid taking 40% from the total weightage, you take this metric from the row, and place it on a separate table. This 2nd table overrides the first table with its value if it produces a lower value.

For example, if there is any error packet, you want to set the KPI value to red to reflect that there is a severe problem happening.

Validation

Once you design the KPI for a specific object, do a validation. This helps you validate if the thresholds, weightage, metrics actually deliver the score that matches your expectation. Write down the common scenarios along with the expected value.

I got a surprise on the result, that I thought there was a bug in the formula. Remember that 1 red has the weight of 8 green? So when I see 3 reds and 9 greens, I expect the value to be in the red, which is below 25. But I got a low orange.

So let's do some validation. I find testing the corner case useful. So let's see what value we get when we have 9 perfect green and 3 worst red. What value do you expect?

A simple, non-weighted average will give a value of 75. This is right in the border of green or yellow.

What color does the weightage score give us?

	Score	Weightage	Adjusted Score	
1	100	1	100	
2	100	1	100	
3	100	1	100	
4	100	1	100	
5	100	1	100	
6	100	1	100	
7	100	1	100	
8	100	1	100	
9	100	1	100	
10	0	8	0	
11	0	8	0	
12	0	8	0	
Summary		33	900	27.27

It gives us a low orange. It is not red, but close enough to be red. This is why the score is important too, not just the color.

What if your red is not the worst, but barely red? How many borderline red (near 25%) required before a perfect green (100%) is showing red?

The following table shows 1 perfect green score and 11 barely-red scores. What color do you get at the end?

	Score	Weightage	Adjusted Score	
1	100	1	100	
2	24.9	8	199.2	
3	24.9	8	199.2	
4	24.9	8	199.2	
5	24.9	8	199.2	
6	24.9	8	199.2	
7	24.9	8	199.2	
8	24.9	8	199.2	
9	24.9	8	199.2	
10	24.9	8	199.2	
11	24.9	8	199.2	
12	24.9	8	199.2	
Summary	31.16	89	2291.2	25.74

Yup, you get orange, not red. It takes many red scores, which makes it practically impossible to get a red if each red is barely there. That's why your red threshold needs to be 2x your orange threshold. If you make it too big, you will get barely-red in most cases.

In actual environment, you certainly do not want to see red, even in development environment. Each VM will have their own score, but overall you want to see majority green. Use heatmap to show, as it will automatically order them by the value.

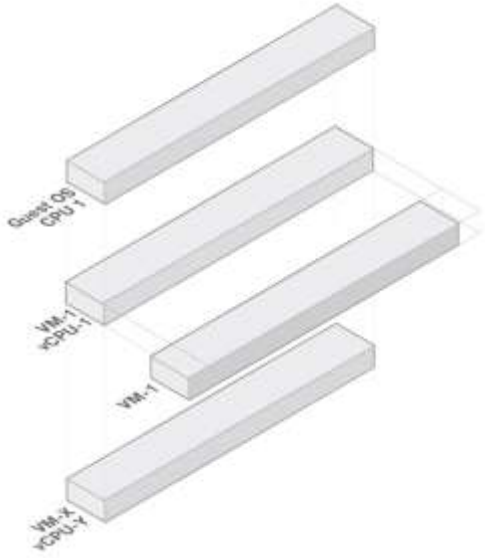


Part 3 Chapter 2

Architecture

What used to be Windows or Linux running on a server has transformed into Guest OS → VM → ESXi. There 3 distinct layers resulted in complexity documented in Part 2 Chapter 1. This is not as complex as memory, where you have 4 layers as process running inside a Guest OS represents another layer.

The following infographic shows how the nature of CPU metrics changes as a result of virtualization.



When mastering CPU metrics, it helps to consider the big picture.

First, there are Guest OS and VM. While one is on top the other, they are independently calculated by 2 different OS (Guest OS and VMkernel)

Second, there is VM and VM vCPU. Some CPU cycles in MKS, VMX and VMM world are not charged to each vCPU.

Third, there could be other VM competing on the same ESXi physical thread.

[2]

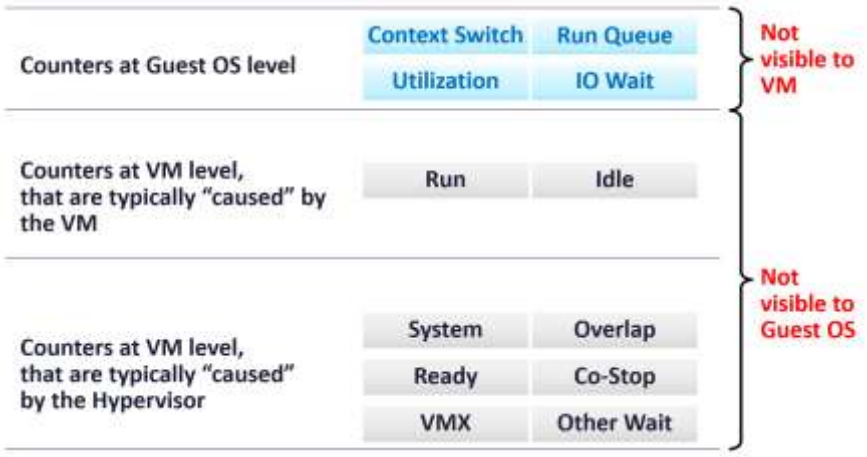
Specifically for CPU, we need to be aware of dynamic metric. This means their values fluctuates depending on CPU clock speed and HT effect. As a result, the values are harder to figure out due to lack of observability on the fluctuation. This is not an issue if the range is negligible. It is not. For example, HT can increase the value of CPU Latency anywhere from 0% to 37.5%.

[2] If you suspect that I can't create professional graphic like this, you are right! That's done by Abhishek Chouksey

Guest OS vs VM

CPU metrics for a VM differ greatly from those in the Guest OS. For example, vCenter provides 5 metrics to account for the utilization of VM CPU, yet none directly maps to Windows/Linux CPU utilization.

The following diagram shows some of the differences.



When the kernel de-schedules a VM to process something else (e.g. other VM, kernel interrupt) on the same physical thread or core, the Guest OS does not know why it is interrupted. In fact, it experiences frozen time for that particular vCPU running on the physical core. Time jumps when it's scheduled again. See this paper for the explanation.

Microsoft Windows and Linux handle this freeze differently when running on vSphere.

- For Windows, the frozen part is not accounted, resulting in higher value being reported. Let's take an example
  - 0 – 1 second: CPU runs 100%
  - 1 – 2 second: CPU halted. Windows unaware.
  - If Windows reports every 2 seconds, then it reports 100% while vSphere reports 50%.
- Linux accounts via the metric Steal Time. The value is accurate so long you use paravirtualized system clock.

On the other hand, ESXi cannot see how the Guest OS schedules its processes. ESXi can only see what's being sent out by the Guest.

Because of this, it's important to use the correct metrics at the correct layers.

Difference in the CPU Utilization metrics:

Guest OS	Depending on the version of Guest OS, some may not be aware of CPU scaling. The real time frequency has to be obtained from the BIOS or VMM layer. I'm unsure how frequent this gets updated by ESXi to the VM, as it varies per core and over time.
	It's unaware of the hyper-threading, because the information is not being supplied by ESXi VMM module.
VM	When the vCPU shares a physical core, the CPU Usage will drop by 37.5%.
	Includes IO processing done on behalf of the hypervisor. In heavy databases or NSX Edge, this is significant. This increases VM CPU Usage.

Both layers need to be monitored, as each measure different performance problems. Hence it's imperative to install VMware Tools. It reports the statistics about Guest OS to the ESXi host every 20 seconds by default.

The following example summarizes that mapping between Guest and VM is not possible.

Type	MS Windows Metric	VM Metric
Contention	Run Queue	None. All these are internal operations of Windows or Linux.
	DPC Time	
	Context Switch	
	C1 Time C2 Time C3 Time	None. ESXi does not break down per VM as it focuses on the physical core.
	None	CPU Ready CPU Co-Stop CPU System
Utilization	Usage	Run – Overlap – VMX – System. In Linux, the usage counters shown by the commands top, htop, or vmstat measures the time spent. It does not include the speed during the run.

I put Linux separately as Windows and Linux have different approaches. The metrics for Linux are:

User	This means non kernel. Applications typically run here.
System	This is the kernel or privileged ring, where core operating systems function run.
Idle	Not running. This should correlate with the CPU run queue.
Wait	The time spent waiting for IO (disk or network)
Steal	I think this is only applicable in virtual machine, not physical. This is the time it's frozen because underlying hypervisor does not have physical thread to run the VM vCPU.

The following diagram shows the difference between Guest OS and VM.

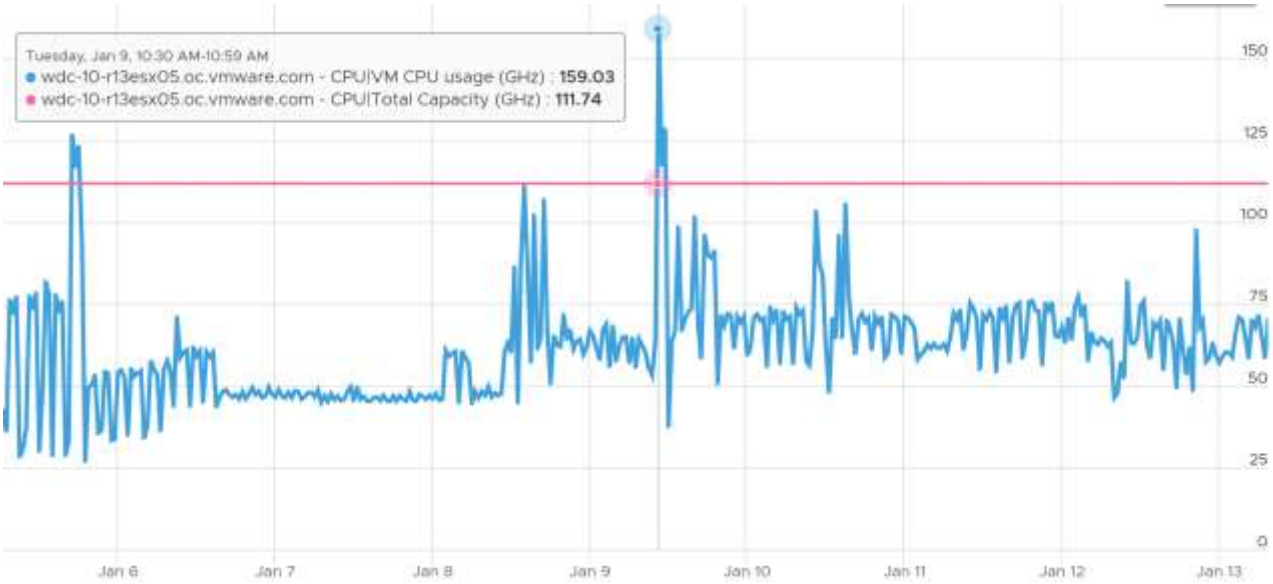
<table>
<tr><td>CPU[0]</td><td colspan="6">Guest OS: Run</td><td>Guest OS not run</td></tr>
<tr><td>Guest OS</td><td>Run
(productive)</td><td>Run
(Context X)</td><td colspan="2">Run
(wants to run)</td><td colspan="2">Wait</td><td>Idle</td></tr>
<tr><td>VM</td><td colspan="2">Run</td><td>Ready</td><td>Co-Stop</td><td>Run
(Guest OS IO)</td><td>Wait
(VM IO)</td><td>Idle</td></tr>
<tr><td>Remarks</td><td colspan="2">vCPU running</td><td colspan="2">vCPU paused</td><td colspan="2">Double IO Processing</td><td></td></tr>
</table>

We will explain the above diagram after we cover the counters.

ESXi ≠ VM + Kernel

Let's use an example to drive the point.

Review the diagram. What do you notice?

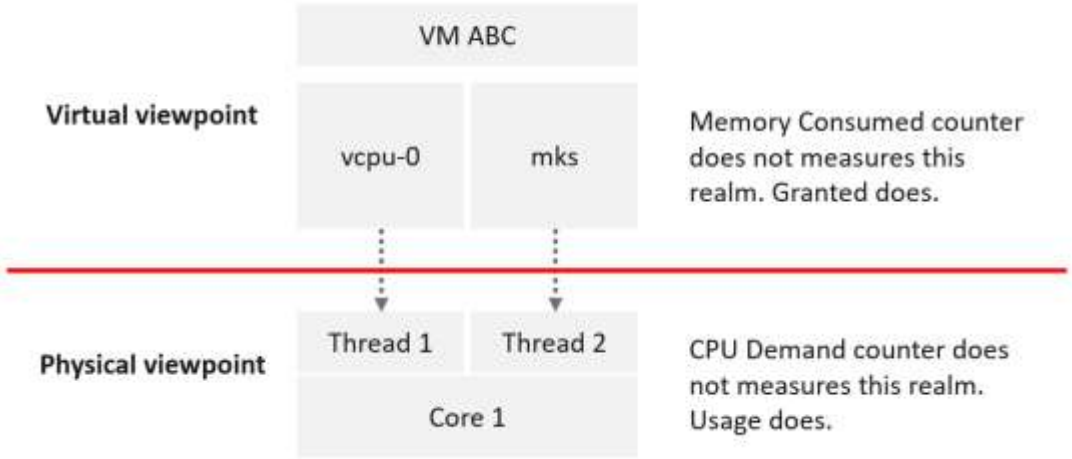


Yes, the sum of VM CPU Usage is greater than the ESXi total capacity! How could it be?

The answer is different perspective.

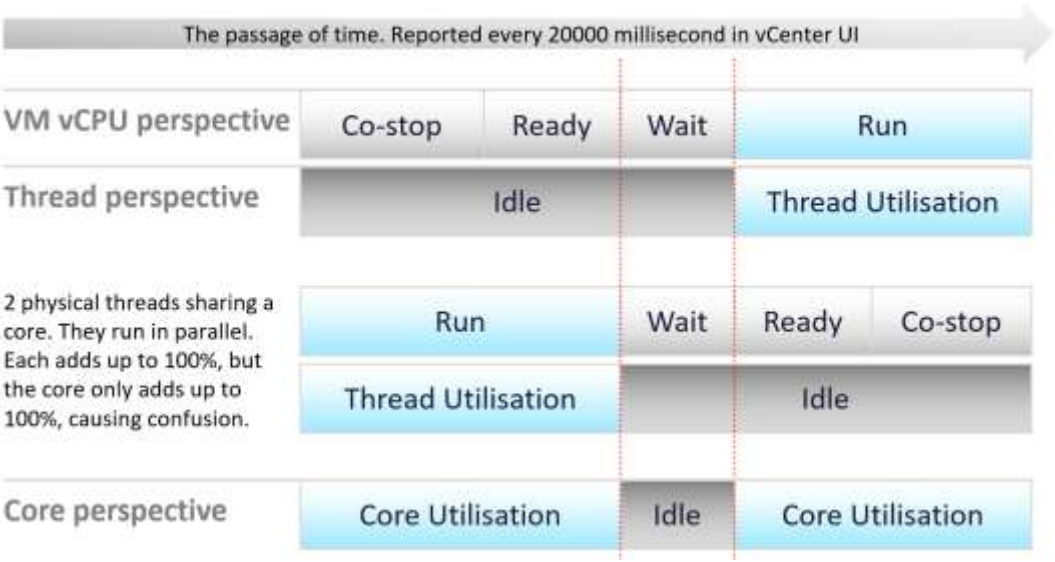
Just like Guest OS and VM have different vantage points, the same complexity happens between VM and ESXi.

The complexity comes from the different vantage points. The metrics at ESXi and VM level measure different things. As a result, the counter at ESXi level cannot be the sum of its VMs + kernel.



VM takes the consumer view, meaning it sees the virtual layer. It sees 2 virtual CPUs, unaware of HT. A VM may compete with other VMs, but they are always unaware of one another.

ESXi takes the provider view, meaning it sees the physical layer. It sees 1 core with 2 threads, fully aware of HT. Concept such as Ready and Wait are not applicable as a core is either runs or idle. The kernel practically does not experience contention as it has the highest priority.



The preceeding diagram shows that there are actualy 3 levels of measurements:

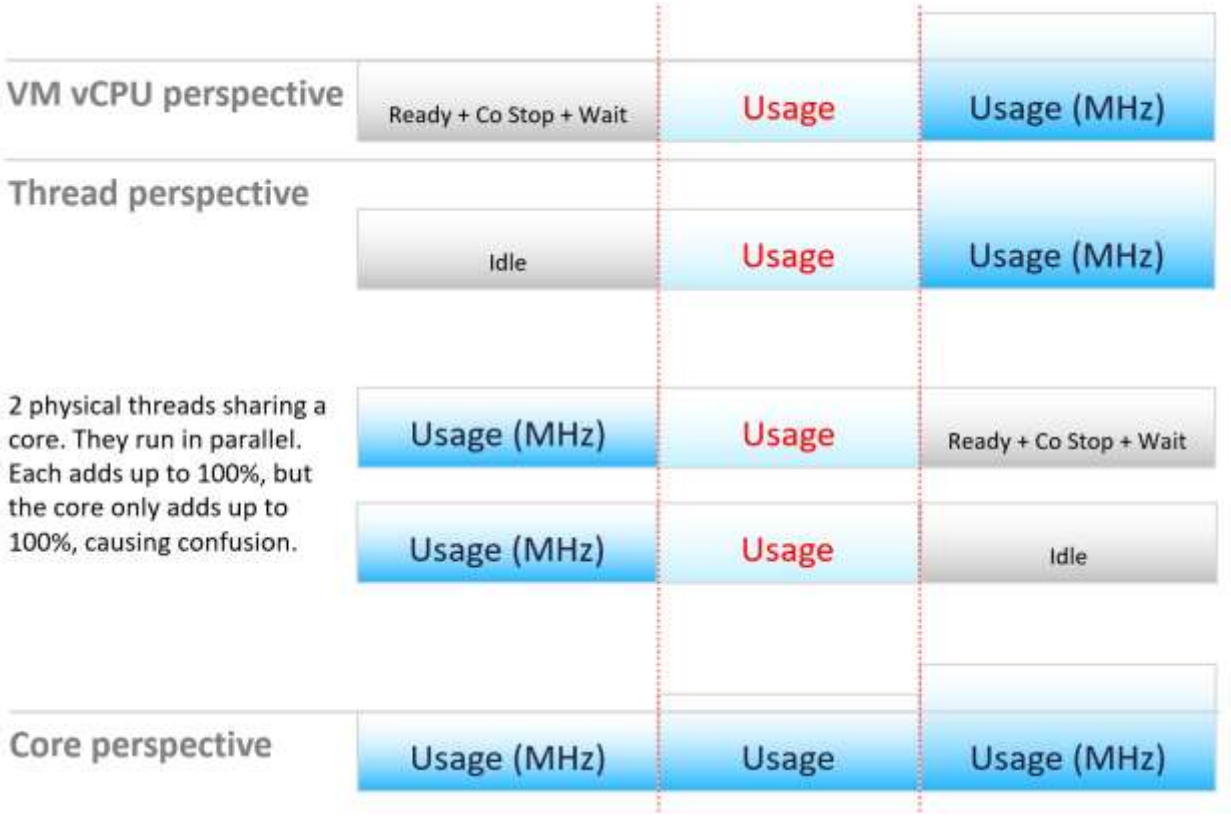
VM vCPU	Since a vCPU is a consumer, when it does not get what it wants, it needs to account for it. This is why you see counters such as ready and co stop.
ESXi Thread	Since a thread is consumer, it never experiences contention. It either runs or not.
	Since each thread runs in parallel, each gets their own 100%. Now, do not total them and conclude the core can run at 200%. Yes, I know esxtop shows that. It is not wrong, but there is a simpler way. Simply look at it from core perspective.
ESXi Core	The core is the one doing the actual work, not the thread. So the one that is running is actually the core. That's why it's correct that a core registers 100% and not 50% when it only runs one of its threads.

Now, there is a complication.

When a core is running, it can run at different frequency and efficiency.

This is where the Usage (MHz) counter comes in.

If the word Usage is confusing, think of the word throughput or cycles completed.

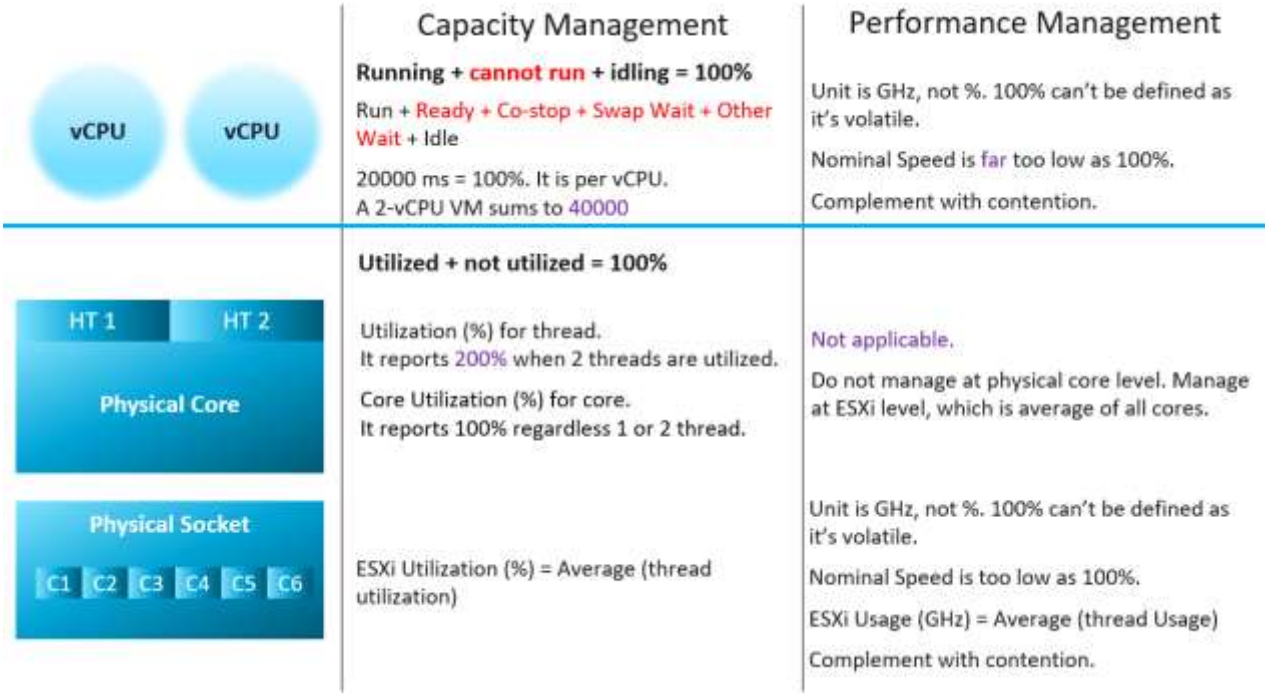


Let's apply the above into the 2 pillars of operations management:

- performance
- capacity

Review the following infographic. Go through it vertically, then horizontally.

What's your take on the metrics?



Capacity is always based on a static counter. That's why it can use percentage as the unit.

- For VM, it's about consuming the given vCPU.
- For ESXi, it's about the utilization of the threads.

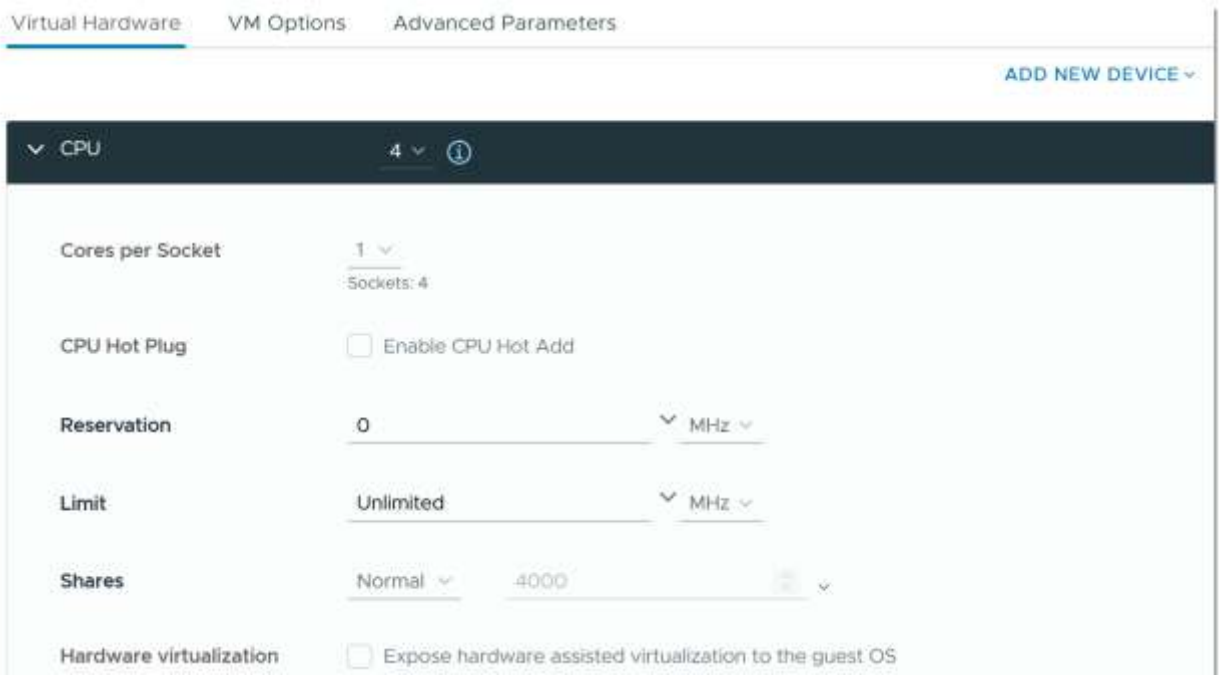
The CPU frequency is about performance, not capacity. That's why metrics such as Usage and Demand are not used in capacity.

Performance is about speed, not space. That's why CPU frequency, a key indicator of speed, has to be included.

- The capability of today's CPU means the CPU speed varies over time and varies among its cores. As such, there is no 100% as the upper limit cannot be determined. Even if it is possible, there is no API to access this information.
- This means we can't use percentage as unit.
- Since the upper limit is not known, we need to complement with contention counters (e.g. ready and overlap) so we can proactively manage.

CPU vs vCPU

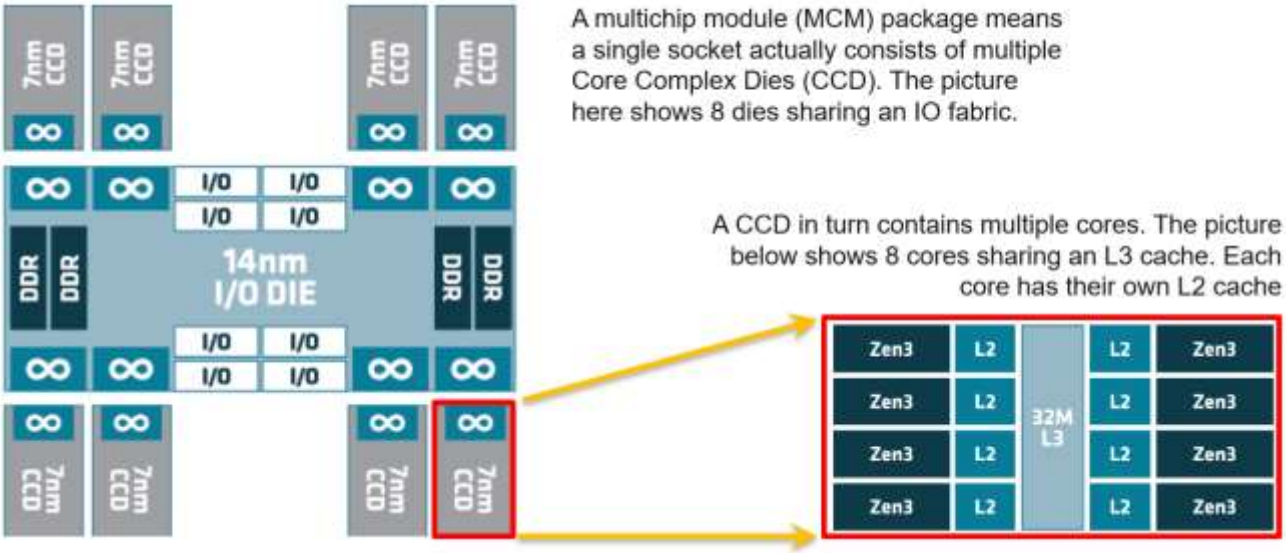
One major difference is there is no hyperthreading at vCPU level. vSphere only expose vSocket and vCore. Within a vCore, there is no hyperthreading.



System on a Chip

As CPU architecture moves towards System on a Chip design, it's important not to assume that a CPU socket is a simple and linear collection of cores. Take a 64-core AMD EPYC for example. It's actually made of 8 Core Complex Dies. Each CCD has their own L3 cache. Within a CCD, there are 8 Zen 3 cores, each having their own L2 cache.

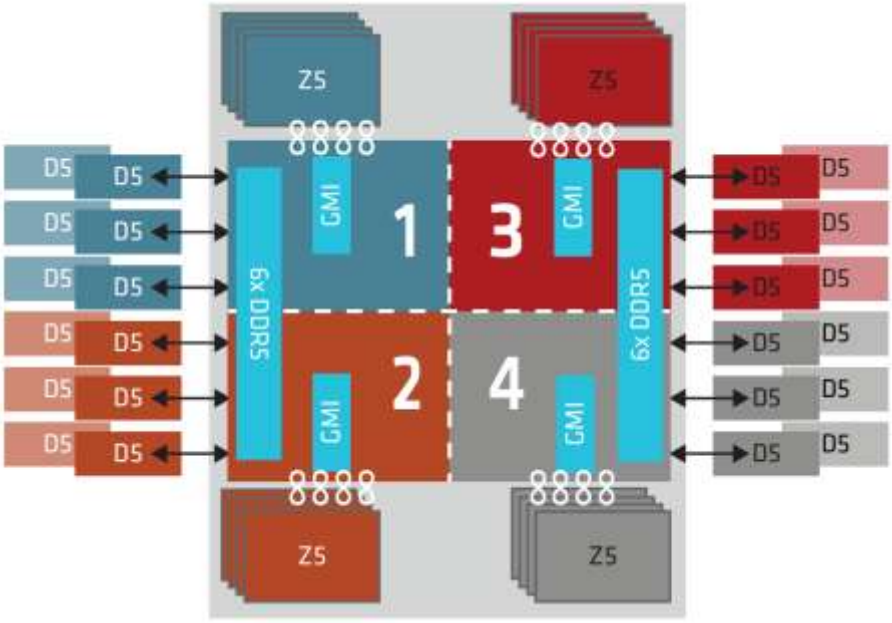
The following diagram (taken from page 5 on the AMD link above) shows there its locality effect within a single socket. A thread is closer to another thread on the same CCD. You can see an example of the performance impact on this blog here by Todd Muirhead.



NUMA

NUMA Node = Socket / Package, as 1 socket can have >1 package (if you enable Cluster-on-Die feature of Intel Xeon).

The whitepaper from AMD on EPYC 9005 Series processors shows that the socket is best divided into 4 NUMA domains. This setting is called NPS4. The following diagram shows NPS4, where each domain has 6 memory DIMMS and 3 memory controllers.



Review the NUMA effect in this KB.

From workload perspective, the situation is similar in Intel CPU. The paper from SAP below shows that SAP recommends Intel Sub-NUMA Cluster (SNC) to be enabled:

The support for SNC-2 in SAP HANA takes into account the increasing density of processors, memory controllers, processor interconnects, and supporting infrastructure within a single chip, as the size of the CPU decreases. Increasing the number of CPU cores and, consequently, CPU performance is advantageous for users. However, this increased density is associated with a longer data transfer time between different parts of the CPU chip.

The process of accessing the memory within a CPU is facilitated by a uniform memory access (UMA) domain, which offers a unified and contiguous address space that is interleaved among all the memory controllers. UMA lacks a mechanism for optimizing the flow of data from the closest available resources. Processor affinity is typically employed to specify the processor(s) that a particular software thread utilizes. Consequently, in a Uniform Memory Access (UMA) system, all cores possess equal access to both the last level cache and memory. In this scenario, there is a possibility of a processor accessing a memory controller or a portion of the last level cache located on the opposite side of the CPU chip compared to the nearest.

However, when running several SAP HANA VMs on a CPU socket, there are more fluctuations. Because of these fluctuations, half-socket vSphere VM deployments for Sapphire Rapids CPUs requires to leverage Intel SNC.

In terms of latency, this site showed the test result of Xeon 6 cache. The latency went up from 33 ns for local L3 cache to 80 ms to an L3 that is further away.

State of a VM vCPU

ESXi Scheduler keeps in mind the following goals:

- To balance load across physical cores.
- To preserve cache state, minimize migration cost.
- To avoid contention from hardware (hyperthreading, low level cache, etc.) and sibling vCPUs (from the same VM).

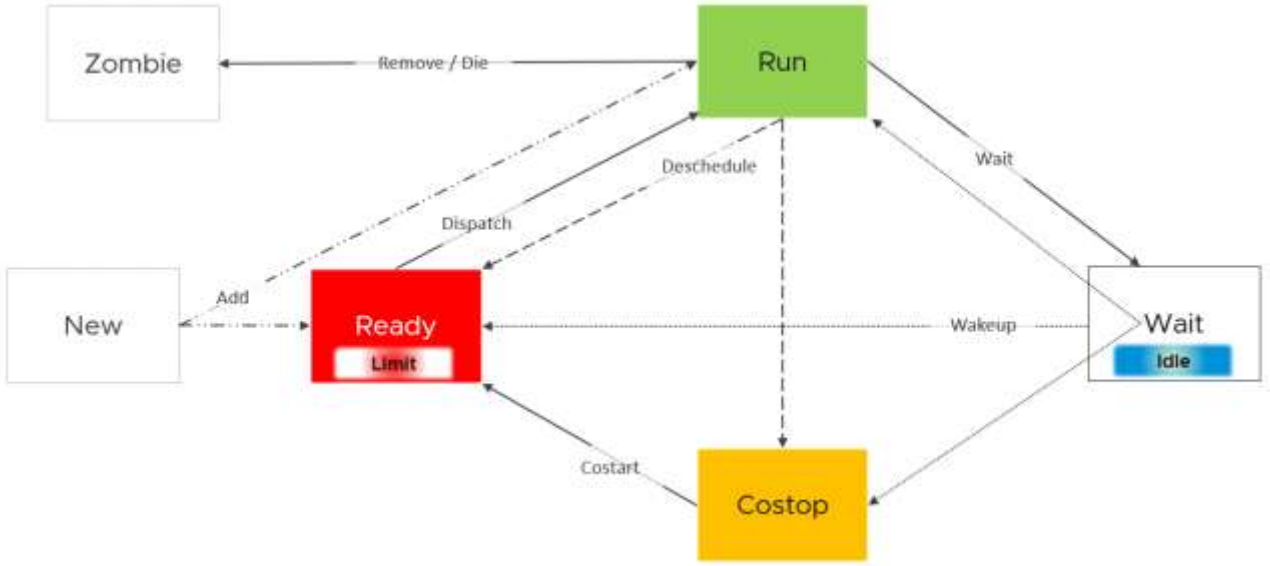
- To keep VMs or threads that have frequent communications close to each other.

With the above understanding, now look at the life of a single vCPU of a VM.

At the most basic level, a VM CPU is either being utilized or not being utilized by the Guest OS. At any given moment, it either runs or it does not, there is no "walk" state.

Being used	The hypervisor must schedule the vCPU. A multi vCPU VM has multiple schedules, 1 for each vCPU. For each vCPU: • If the kernel has the physical CPUs to run it, then the vCPU gets to **Run**. The Run counter is increased to track this. • If the kernel has no physical CPUs to run it, then the vCPU is placed into **Ready** State. The VM is ready, but the hypervisor is not. The Ready counter tracks this. • If the kernel has resource, but other vCPU of the VM is far behind, then the vCPUis placed into **Co Stop** state.
Not being used	There are 2 possible reasons why it's used: • The CPU is truly idle. It's not doing any work. The **Idle Wait** counter accounts for it. • The CPU is waiting for IO. CPU, being faster than RAM, waits for IO to be brought in. There are 3 sub cases here (Co-stop, VM Wait and memory wait), and they will be covered later.

With the above understanding, we're ready to examine the following state diagram[3]. The diagram shows a single schedule (1 vCPU, not the whole VM). It's showing the view from hypervisor (not from inside the Guest OS):



ESXi places each vCPU of the VM in one of the 4 above states. A vCPU cannot be in 2 states *at the same time*. This is fundamental in understanding the formula behind CPU metrics.

- Run does not check how fast it runs (frequency) or how efficient it runs (hyperthreading). Run measures how long it runs, hence the counter is in milliseconds, not GHz.

[3] Thanks, Valentin Bondzio, for permission to modify the original diagram. Taken from one of his many talks!

- Ready and Co-stop.
  They are mutually exclusive as they have opposite reason.
  Co-stop is "caused by" the VM, while Ready is caused by the ESXi. You can reduce Co-stop by reducing the VM size, especially relative to the NUMA boundary. You can't do the same on Ready.
- Wait handles both Idle and Wait. The reason is the hypervisor cannot tell whether the Guest OS is waiting for IO or idle. As far as the hypervisor concern, it's not doing anything. This also measures the state where the wait is due to hypervisor IO.

Regardless of the cause of the pause, a freeze is a freeze. The impact to Windows or Linux is the same.

Back to the kernel 4 possible states, you can conclude that:

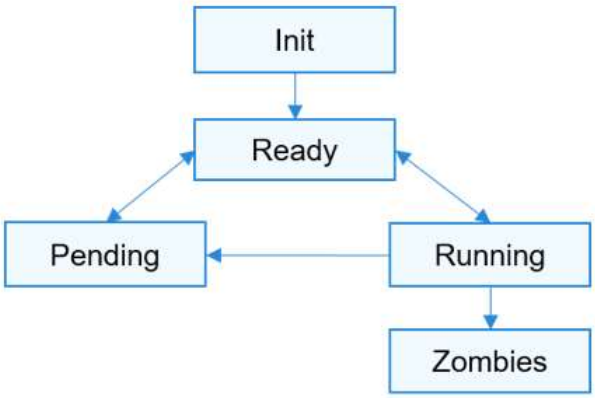
- Run + Ready + Co-stop + Wait = 100%
- VM 2 can run when VM 1 is on Co-stop state, Ready state, or Wait state. This is because the physical thread becomes available.

Ready, Co-stop, Wait are unaware of contention due to hyperthreading. The vCPU is not placed in ready state because both threads can execute at the same time. The contention for shared resources happens at low level hardware and essentially transparent to ESXi scheduler. If you are concerned about this certain degradation in throughput when two worlds execute at the same time on the same core, what counter should you use?

You're right. It's CPU Latency. Different purpose, different counter.

OS Comparison

Those of you familiar with Operating Systems[4] kernel will notice that the diagram is similar with a physical OS scheduler state diagram. I'm taking Huawei Harmony OS as an example as it's a new OS and it's designed for a range of device[5].



Init	The process is being created. Maps to New in the kernel
Ready	The process is in the ready list and waits for being scheduled by the CPU. Maps to Ready in the kernel

[4] Understanding how an OS works is paramount and well worth it. Here is a 3.5 hour lecture by Mike Murphy.
[5] Designing an OS for multiple hardware classes is hard. Notice Apple MacOS, iPhone OS, and iPad OS. Google has Android and ChromeOS.

Running	Maps to Run in the kernel
Pending	The process is blocked and suspended. When all threads in a process are blocked, the process is blocked and suspended. Maps to Wait in the kernel. Notice they also include Idle here in their Wait state.
Zombies	Maps to Zombies in the kernel
“none”	Our Co-stop is unique as VM is a multi-process scheduled entity

Fair Share Scheduler

The preceding state diagram is only for 1 vCPU. ESXi schedules multiple VMs, and each can have multiple vCPU. How do you ensure fairness since cores in a single socket can run at different speed? How do you account for hyperthreading? How do you account for VM doing heavy IO processing?

Enter the Fair Share Scheduler.

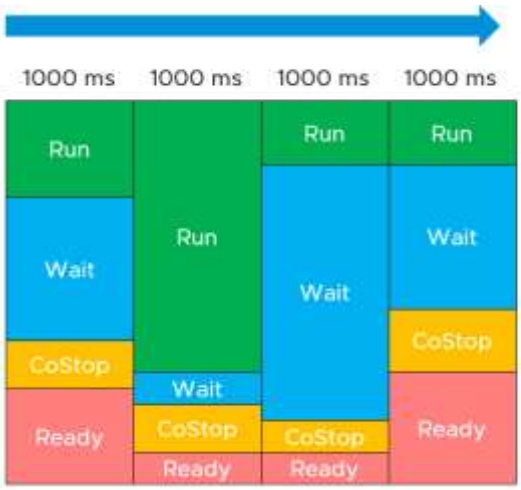
It gives each vCPU a slice of time. Since the quality can vary, ESXi extends the time for a particular vCPU to maintain relative fairness. It uses a CPU counter called Used (ms), which adjusts the simple runtime (CPU Run) by:

- Increasing the run time if the vCPU runs on a shared physical core.
- Adjusting the run time to account for CPU frequency.
- Decreasing the run time to account for IO work performed by the hypervisor on behalf of the vCPU.

BTW, if you need a comparison, this is a good reading on the latest Linux EEVDF Scheduler. This Earliest Eligible Virtual Deadline First has completey replaced the Completely Fair Scheduler (CFS) in Linux 6.12. The main limitation of CFS is how it handles short burst tasks that need to be scheduled immediately.

State Across Time

The above is at any given moment. To measure over time and report it (say every 20 seconds), we need to add a time dimension. The following example shows the above state diagram repeated over time, where each block is 1 second. In reality, the scheduler checks every 2- 30 milliseconds.



How millisecond is converted into %

In the 4 second reporting period:

Total value = 4000 ms

This equals 100%.

Run accumulates 787 ms in the 4 second window.

In percentage, that means 787 / 4000 = 19.7%.

The above is an example. vCenter uses 20 seconds.

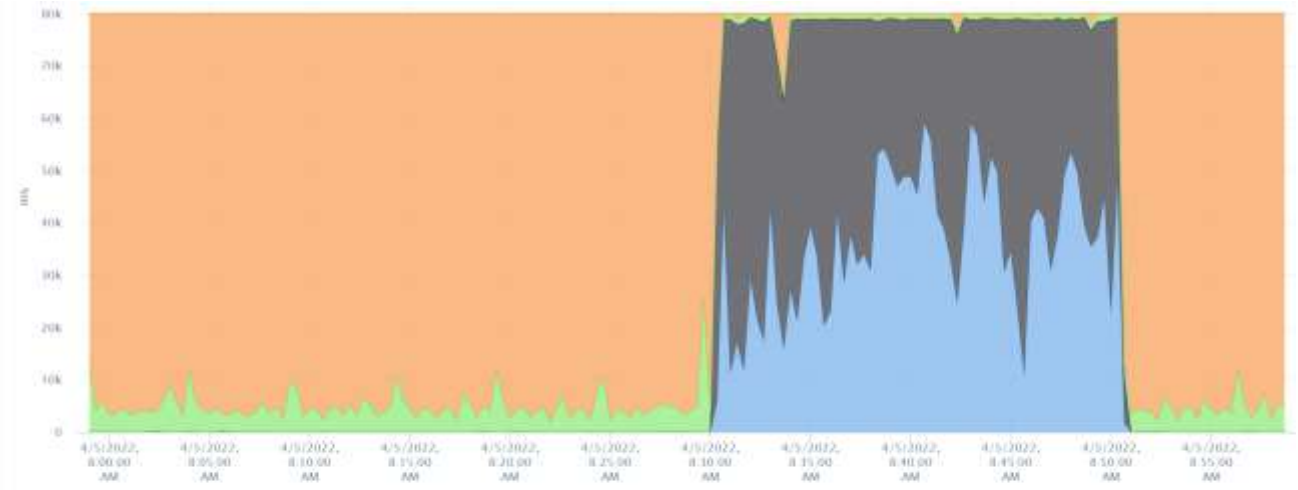
I don't have the energy to draw 20 blocks.

vCenter happens to use 20000 milliseconds as the reporting cycle, hence 20000 milliseconds = 100%.

The above visually shows why Ready (%) + Co-stop (%) needs to be seen in context of Run. Ready at 5% is low when Run is at 95%. Ready at 2% is very high when Run is only 10%, because 20% of the time when the VM wanted to run it couldn't.

The above is *per vCPU*. A VM with 24 vCPU will have 480,000 as the total. It matters not if the VM is configured with 1 vCPU 24 vCores or 24 vCPU with 1 vCore each.

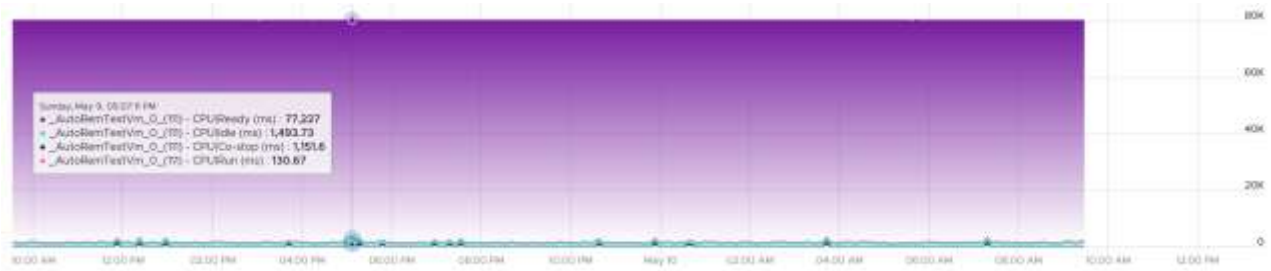
You can prove the above by stacking up the 4 metrics over time. In this VM, the total is exactly 80000 ms as it has 4 vCPU. If you wonder why CPU Ready is so high, it's a test VM where we artificially placed a limit.



Performance Chart Legend

Key	Object	Measurement	Rollup	Units	Latest	Maximum	Minimum	Average
	VROPS-86	Co-stop	Summation	ms	0	59,243	0	12,368.417
	VROPS-86	Ready	Summation	ms	1	67,873	0	14,136.461
	VROPS-86	Run	Summation	ms	5,203	25,414	577	3,565.561
	VROPS-86	Wait	Summation	ms	74,801	77,948	1	49,929.57

The formula for the millisecond metrics in VCF Operations are also not normalized by the number of vCPU. The following shows the total adds up to 80000 as the VM has 4 vCPU.

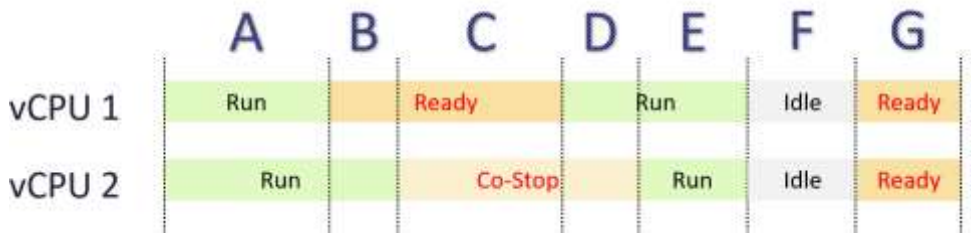


This is why you should avoid using the millisecond counter. Use the % version instead. They have been normalized.

State Across Multiple vCPU

Ready and Co Stop can impact one another.

The following shows a 2 vCPU VM.



A	Run & Run	Both vCPU are running.
B	Ready & Run	CPU 1 hit Ready (for whatever reason). CPU 2 still run as ESXi has available CPU thread to run it.
C	Ready & Co-Stop	CPU 1 still on Ready as ESXi does not have physical thread to run it. CPU 2 must be placed in Co-Stop as it's too far ahead (system clock not in-synced).
D	Run & Co-Stop	CPU 1 run as ESXi has CPU. CPU 2 is still on Co-Stop as its time is still too far ahead.
E	Run & Run	CPU 2 run as its clock is now the same. So, both run at the same time
F	Idle & Idle	Both are idle or waiting for IO. So, they are either in Wait or Idle state.
G	Ready & Ready	Both want to run, but no available physical CPU Thread to run either.

2 Metrics Not 1

When you order a taxi, you expect 2 numbers on your mobile phone.

- The first number shows time. It tells you how long you have to wait.
- The second number show distance. It tells you how far the car is.

For example, the application tells you the car will reach in 3:50 minutes and it's 2.8 kilometres away.

What metric you don't need, hence it's not provided?

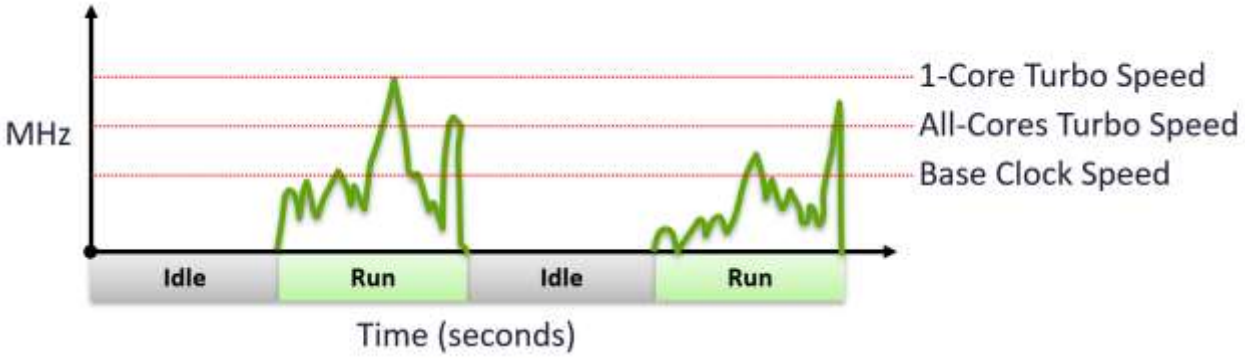
The progress is 78%. A relative metric like this has no purpose as the 100% is undeterministic. The car may experience traffic jam, diversion, or simply makes a wrong turn.

The same thing in CPU, although it's not obvious to us.

The following table provides the comparison.

A car is travelling...	A CPU is running...	
In the last 10 minutes, it drove for 9 minutes and stopped for 1 minute. So it was moving for 90% of the time.	In the last 10 minutes, it ran for 9 minutes and idle for 1 minute. So it was running for 90% of the time.	CPU Run (%)
When it was moving, its speed average 60 km/hour. Speed varies due to traffic.	When it was running, its speed average 3 GHz. Speed varies due to power management.	CPU Throughput (MHz)

Review the following diagram. What do you observe?



There are 2 metrics:

Run Time	A simple metric as it's binary. Counting whether CPU run or not running. It measures duration or time.
	How much the CPU runs in a time period. E.g. if it runs 60% of the time in the last 100 seconds, that means it runs for 60 seconds accumulatively in that period. That's why you see many metrics in millisecond. They track the consumption over time.
	Use it for capacity management.
Run Quality	A complex metric. Yes, the amount of CPU cycles completed varies. It's only activated when Metric 1 is not 0. It measures intensity of work. This is the equivalent of distance travelled.
	Use it for performance management.

Frequency | Efficiency | Efficacy

There are 3 factors that get mixed into a single counter that measure the amount of work completed. Unfortunately, this accounting simplification creates confusion as the factors are technically different.

Frequency	impacted by clock speed	All else being equal, a 2 GHz CPU does 2x more cycles than 1 GHz CPU. Throughput impacts utilization. The faster it can complete a task, the shorter it has to work. That's why you see some metrics in MHz.
Efficiency	impacted by Hyperthreading	All else being equal, a core running 2 threads completes 1.25x more cycles at the expense of each thread gets only 0.625x. This efficiency is then translated into MHz. Note these numbers are hardcoded for ease of reporting.
Efficacy	Feature of core architecture	It is not part of the accounting. See below.

Efficacy is one reason why upgrading to a new generation CPU with lower clock speed result in faster performance.

Unlike memory, storage and network, there is another capability of CPU that makes the same application get more done despite running on the same frequency. Yes, this means that 1 GHz on a brand-new Intel Xeon is probably 30% faster than 1 GHz on a 5-year-old Intel Xeon. Same company, same CPU famiy, same clock speed.

Why your application can get 30% more work done is due to design change. The CPU has both more L2 cache and more L3 cache per core, so even a single thread process gets more work done despite running at the same speed. Other design improvements such as superscalar, better branch prediction, better pipeline, and higher pipes all contribute to it.

CPU Frequency

What is your CPU Speed?

There are 3 main numbers that defines the speed:

Base	This is what vSphere uses as the total capacity. It is the nominal frequency and most commonly shown. The CPU rarely runs at this speed. It runs above it, giving you extra performance for free.
Single-Core Turbo	This is what marketing will show you. Benchmarking of single thread app is done with this frequency. This is typically much higher. Using AMD EPYC 9000 series family, it ranges from 7% - 64% additional speed gain. Using Intel Xeon Platinum 8593Q, the gain is 77%.
All Cores Turbo	This is what you should pay attention to.

	AMD shares the All Core Boost speed, while Intel no longer does that. Since Intel does not provide the All Core Boost via API, we can't use it as part of capacity. Using AMD EPYC 9000 series family, the gain ranges from 1% - 48%.

Intel has a page called What is CPU Clock Speed. Below is an extract, with relevant highlight from me:

> Intel Turbo Boost Technology enhances clock speed dynamically to deal with heavy workloads. It works without requiring any installation or configuration by the user. The technology judges the amount of heat the system can tolerate, as well as the number of cores in use, and then boosts clock speed to the maximum safe level.
>
> **Base Processor Frequency** and **Max Turbo Frequency** are two core performance metrics that refer to different usage scenarios. For high-intensity gaming, the turbo frequency is the more important metric. Given adequate cooling, this is the speed your CPU will operate at when dealing with heavy gaming workloads such as traveling through a highly detailed environment, or calculating AI behavior on an enemy turn in a strategy game in the most CPU-intensive titles.
>
> Recent features like the Intel Thread Director allow the latest gen Intel processors to intelligently distribute workloads to multiple cores. That's one reason why newer processors often outperform older ones on benchmark tests even when they have similar clock speeds.

Each core can have its own frequency. This makes rolling up the number to ESXi level more complex. You can't derive one throughput counter from the other. Each has to be calculated independently at core level.

Total Capacity

The above brings an interesting question: what is your CPU Total Capacity?

The higher the frequency (GHz), the faster the CPU run. Ceteris paribus, a CPU that run at 1 GHz is 50% slower than when it runs at 2 GHz. On the other hand, Turbo Mode can kick in and the CPU clock speed becomes higher than stated frequency. Turbo Boost normally happens together with power saving on the same CPU socket. Some cores are put to sleep mode, and the power saving is used to turbo mode other cores. The overall power envelope within the socket remains the same.

How high is the gap?

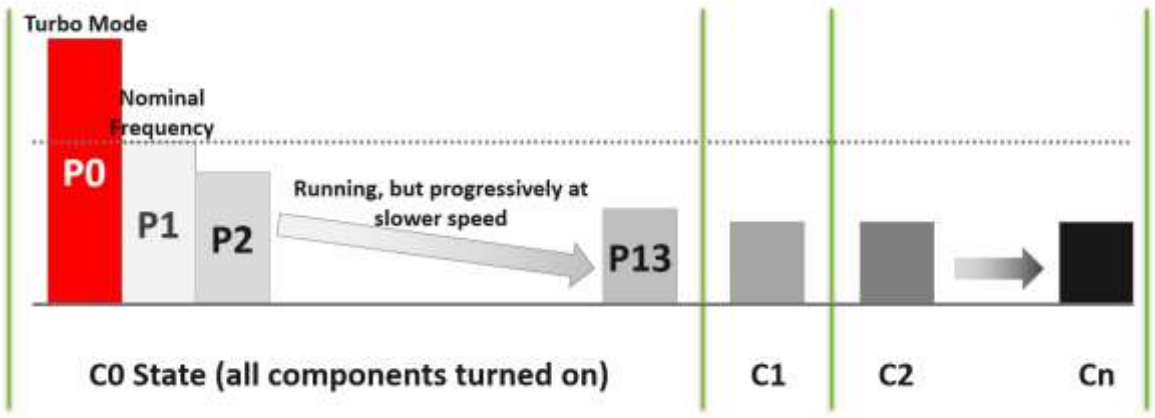
The gap between the base clock speed and the highest turbo is around 2x. The gap started to appear in the year 2008, when Intel Nehalem was introduced. The early Turbo versions could turn cores off or on, they generally forced all active cores to run at the same frequency. In 2014, Intel introduced Per-Core P-States (PCPS) with the Haswell-EP (Xeon E5-2600 v3). This allowed each individual core to reside in its own power state and frequency independently. AMD achieved a similar, even more granular level of control with the first Zen (EPYC/Ryzen) processors in 2017, using a network of hundreds of sensors to adjust clock speeds in 25MHz increments per core.

	2006	2026
Examples	Intel Xeon 5160 / AMD Opteron 2220	Intel Xeon 6 / AMD EPYC 9005
Typical Base Clock	3.0 GHz	2.0 – 2.3 GHz
Max Turbo Clock	N/A (Locked at Base)	3.9 – 5.0 GHz
Core Count	2 Cores	128 – 192 Cores

Turbo/Base Ratio	1.0x (No variance)	~1.7x to 2.2x

Power States

There are 2 types of power states as defined by ACPI standard.



C-State	C here means CPU. When a core is idle, ESXi applies deep halt states, also known as C-states. The deeper the C-state, the less power the CPU uses, but the longer it takes for the CPU to start running again. ESXi predicts the idle state duration and chooses an appropriate C-state to enter. There are 3 possible sub-states in C-state:

C0	Fully running. Within this C0 state, there is a further dimension called P-State. An idle CPU can be in C0.
C1	A shallow state where the clock is gated (switched off) and the main internal clock is stopped. However, all the modules remain active, and the processor can go back to the active C0 state instantaneously. In power management policies that do not use deep C-states, ESXi stops at C1.
C2	Stops the external bus clock. Saves more power than C1.
C3	The CPU flushes its L1 and L2 caches and stops all internal clocks.
C4 - Cn	Varying degrees of CPU sections turned off. The higher the C state, the deeper the sleep.

P-State	P here means Performance. There are 14 grades of CPU performance, measured by its frequency. You can see all the frequencies in esxtop if your hardware supports it. P0 state where Turbo Boost happens. P1 is where it runs at Nominal Frequency (NF).

	P13 is the lowest CPU frequency.

Where is the above implemented?

It's done in 2 levels. It needs to be enabled in both the BIOS and the OS (ESXi in this case).

BIOS Level

Since it's the lower layer, it can override the upper layer implementation.

BIOS can also set the power management to BIOS controlled (not OS controlled) in which case ESXi has no say.

To see the setting, use the following vsish command.

```
/power/> cat /power/platformFeatures
Platform Power Management Features
{
   Deep C-states Available:1
   Deep P-states Available:1
   CPPC Capability Available:1
   CPPC Enabled by BIOS:0
   Turbo Boost Enabled:1
}
```

ESXi Level

Just because BIOS passes the control to ESXi, it does not mean that it's enabled. There are 2 settings

In the vSphere Client UI, go to the ESXi Advanced Settings. You will see these entries:



Use C States = Use deep ACPI C-states (C2 or below) when the processor is idle. Frequency will go lower than base clock speed. Disabling does not mean disabling Turbo. The firmware's thermal management decide on the individual frequency of each core.

Use P States = Use ACPI P-states or CPPC to save power when the processor is busy. When deactivated, the ESX configures to use the highest available frequency.

Collaborative Processor Performance Control (CPCC) is part of the ACPI 5.0+ specification. As the name implies, the OS and the CPU hardware collaborate. Instead of frequencies, the OS hands down a set of abstract performance capabilities and targets to the CPU's on-die microcontroller.

C State	P State	ESXi CPU Frequency Control
1	1	Full Dynamic Range: Cores can drop below Base Nominal to save power, go to 0 Hz (sleep) when idle, and scale up to All Core or Single Core Turbo when heavy workloads demand it.
	0	Rigid Base / Sleep Only: While executing tasks, cores are pinned strictly to Base Nominal (no Turbo expansion, no power-saving down-throttling). Cores only drop power when completely idle via C-state sleep.
0	1	Throttled Range: Cores can scale *down* below Base Nominal to save power, but they can never go higher into Turbo because the lack of core sleep states

	destroys the thermal budget.
0	Flatline: The CPU runs flat-out at its Base Nominal frequency 24/7 across all cores. No power saving, no sleep, and zero Turbo capabilities.

For details on P-State and C-State, see Valentin Bondzio and Mark Achtemichuk, VMworld 2017, Extreme Performance Series. Also read this page of VCF official manual.

Turbo

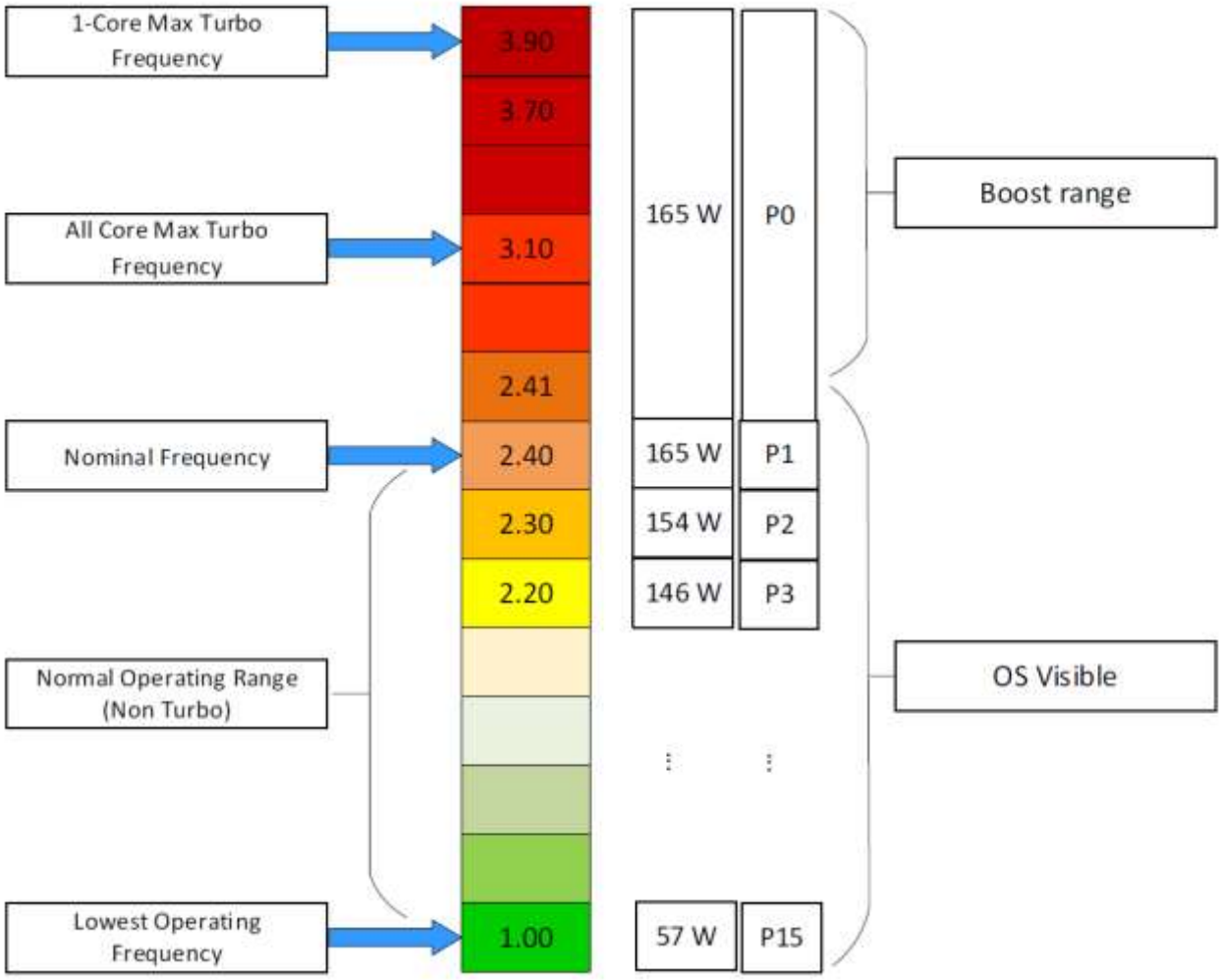
How high can Turbo/Boost go?

It turns out that it is high enough that both performance and capacity need to account for it.

Because speed is the primary attribute for performance, and an important contributor to capacity, the upper limit of CPU performance and capacity become harder to manage.

- Within a core, it impossible to predict the core frequency. It all depends on the instructions and what other cores are doing. Likely, there is no pattern at all.
- Within a socket, it's impossible to predict which cores will run faster at the expense of others. A socket with 20 cores will run at >1 frequency in real world.

The following diagram is taken from page 12 of "Host Power Management in VMware vSphere 7.0" whitepaper by Ranjan Hebbar and Praveen Yedlapalli. It shows that Intel Xeon Platinum 8260 can increase its speed by 1.29x (from 2.4 GHz to 3.1 GHz). If it only needs to increase 1 core, that single core can go up by 1.62x. This will be noticeable by application that is CPU intensive. Consider this benefit before you decide to disable power management. The high performance is static, it runs at the same frequency throughout.



At 2.4 GHz (Base): The CPU uses less than 165W for standard math, but might hit 165W for heavy math.

At 3.1 GHz (All-Core Turbo): The CPU uses its full 165W power budget to maintain that higher speed across all cores. If the workload becomes too intense (like AVX-512), the CPU cannot maintain 3.1 GHz within 165W, so it will automatically drop the frequency to stay under the limit.

Let's take a more recent example. The following is Intel Xeon 6520P. It can run at 3.4 GHz, which is 42% higher than the base speed.

CPU Specifications

Total Cores	24
Total Threads	48
Max Turbo Frequency	4 GHz
All Core Turbo Frequency	3.4 GHz
Processor Base Frequency	2.4 GHz

BTW, there is no setting in ESXi to disable or enable Turbo Boost directly. Yes, I'm aware Windows has it. In the case of ESXi, it's enabled by default and all you have to do is set the power management.

Thanks to Sushil Kavi, I learned Turbo can still kick in despite the BIOS setting to static high power management. We verified that the value of %A/MPERF in esxtop showing all cores running at 129%.

All Cores Turbo

Review this article by IOFLOOD.

Done?

Good. I've highlighted in green a key point stating your CPU is actually operating at a higher speed.

> In routine use, unless you make unusual changes to your BIOS settings, the all-cores-turbo is the real speed that the CPU will function at, whenever all of the CPU cores are actively processing data. This speed is no longer actively advertised by Intel or AMD, to the point that you have to jump through hoops to learn what the All Cores Turbo speeds are for any given processor. Because this is the true speed of the CPU, and therefore very important to know when comparing CPUs, this lack of documentation is a big problem that we hope to resolve in this post.

While it's a good news that you have more capacity, the issue is this number is not always exposed via the CPU API.

Impacts on ESXi

The following table shows the difference in value if All Core Turbo is 20%.

Core Utilization	Usage (%)	My Recommendation	Analysis
25%	30%	20%	
50%	60%	40%	No performance impact to VM
67%	80%	54%	as no HT pair running. Gap is
75%	90%	60%	due to turbo boost.
100%	120%	80%	
125%	150%	100%	performance impact from HT

Usage (%) is what vSphere Client UI shows.

High Performance vs Balance

Should you always set power management to maximum?

No. ESXi uses power management to save power without impacting performance. A VM running on lower clock speed does not mean it gets less done. You only set it to high performance on latency sensitive applications, where sub-seconds performance matters. VDI, VoIP, video calling, Telco NFV are some examples that are best experienced with low latency.

Review the vSphere 9 performance best practices paper. I've copied the part you need to focus below:

Power Policy Options in ESX

ESX offers the following power policy options:

- **High Performance**
  This power policy uses no power saving features, thus maximizing performance for a system where all or most cores are fully loaded.

  This setting is designed to provide the highest performance when a system is heavily loaded. However, if a significant percentage of the cores are unused or lightly used, this setting might not result in the best performance. Although this has always been the case, it's more pronounced with some recent CPUs.

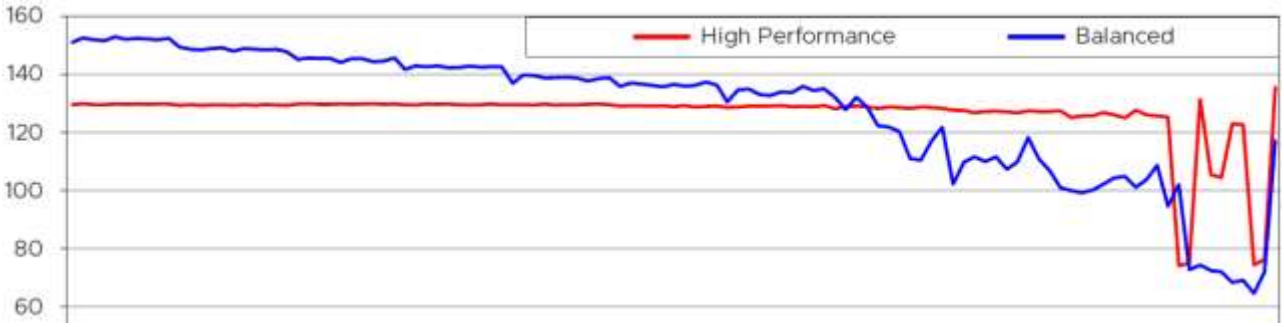
- **Balanced**
  This power policy (the default) is designed to reduce host power consumption while maintaining power usage flexibility, thus typically incurring little or no reduction in performance. On a system where some CPUs are idle part of the time, this policy might actually increase performance by allowing active (non-idle) CPUs to enter faster Turbo Boost states than are available in the High Performance policy.

In mission critical cluster, the overcommit ratio is lower and the VMs tend to be over-provisioned. Or you have 2 hosts for HA as you worry about availability. The end result is your ESXi actual core utilization is not high.

Impact on Oversized VM

One downside of an oversized VM is higher risk of some vCPU becoming idle. When the VM becomes idle, the power enters C1 State, but does not go deeper to C2. This enables the VM to quickly spike, which is evident on the spikes at the end.

The following diagram is taken from page 24 of "Host Power Management in VMware vSphere 7.0" whitepaper by Ranjan Hebbar and Praveen Yedlapalli.



I cut out the chart so we can explain how balanced power management delivers higher performance than high performance setting.

The vertical axis is the CPU Frequency, where the 100 is the nominal frequency. Yes, at the start the CPU was running well above the nominal frequency. Balanced was running higher than High Performance.

The red line is the VM CPU frequency when ESXi power management was set to High Performance. The blue line is Balanced.

The horizontal axis is not time. It's how busy the core is. It starts with 100% busy and steadily goes down to 0, meaning the VM was idle. The VM was not powered off at the end.

The red line is fairly constant until the VM becomes idle. This makes sense as the entire CPU socket is kept on high, so all the cores are equal. As the result, the CPU boost only goes to 130% mark.

The blue line starts at much higher throughput. This makes sense as the CPU has flexibility. It can boost the running cores to 151% as other cores are idle. This is why balanced performance can deliver higher performance on low to medium load ESXi.

As the core gets less busy, the CPU reduces its clock speed. Notice it is still higher than 100 until it became idle. When the VM is idle, the CPU entered the deep C2 State. Notice there was no spike and the frequency dropped deeper.

CPU Efficiency

CPU SMT (Hyper Threading as Intel calls it) impacts CPU accounting as it delivers higher *overall throughput*. It increases the overall CPU cycles completed of the core, but at the expense of individual thread throughput. The increase varies depending on the load.

Accounting wise, it is expensive to measure as the core speed is not affected. It is still running at the same frequency. ESXi records this overall boost at 1.25x regardless of the actual increase, which may be less or more than 1.25x. Your application may experience 2.x or 1.x, the counter will simply report 1.25x.

If both threads are running at the same time, the core records 1.25x overall throughput but each thread only gets 62.5% of the shared physical core. This is a significant drop from the perspective of each VM. From the perspective of each VM, it is better that the second thread is not being used, because the VM could then get 100% performance instead of 62.5%. Because the drop could be significant, enabling the latency sensitivity setting will result in a full core reservation. The CPU scheduler will not run any task on the second HT.

The following diagram shows 2 VMs sharing a single physical core. Each run on a thread of the shared core. There are 4 possible combinations of Run and Idle that can happen:



Each VM runs for half the time. The CPU Run counter = 50%, because it's not aware of HT. But is that really what each VM gets, since they have to fight for the same core?

The answer is obviously no. Hence the need for another counter that accounts for this. The diagram below shows what VM A actually gets. The allocation is fixed.



The CPU Used counter takes this into account. In the first part, VM A only gets 62.5% as VM B is also running. In the second part, VM A gets the full 100%. The total for the entire duration is **40.625**%. CPU Used will report this number, while CPU Run will report **50%**.

If both threads are running all the time, guest what CPU Used and CPU Run will report?

62.5% and 100% respectively.

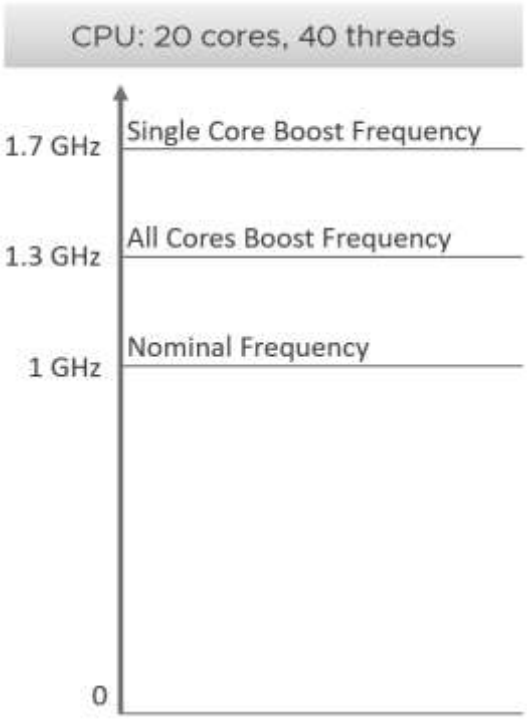
CPU Throughput

The CPU efficiency and frequency have the same final effect. Both impacts the number of CPU productive cycles processed attributed to a particular thread of vCPU. This is why I prefer to call the metric CPU Throughput instead of CPU Usage and CPU Used. The word usage and used are confusing as it's a general English word.

A CPU is always running. A CPU could be running many cycles while waiting for data from memory or a disk, and those cycles are effectively "wasted" from the perspective of completing the task. I hope the English word throughput conveys that it refers to the useful cycles, and not idle loop.

Impacts on VM

Let's take an example to quantify the impact of both CPU frequency and CPU efficiency on a VM vCPU.



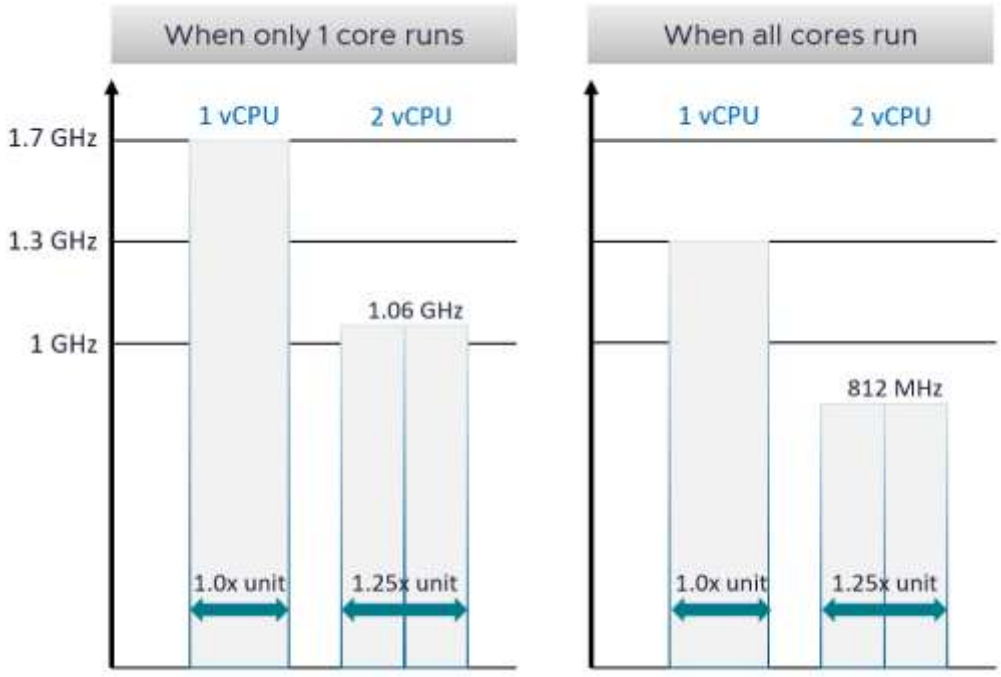
The CPU can run all its 20 cores at 30% higher speed. When only a single core is used, it runs up to 70% faster. An example of CPU capable of such performance boost is AMD EPYC 9634.

For ease of calculation, this sample CPU has a nominal clock speed of 1 GHz. This is what you see in vSphere Client UI, and what you consider as 100%. The number is shown in red as using this number as total capacity leaves money on the table.

Now let's look at what the experience that Windows or Linux receive as a result.

You will see that a VM can run 100% flat out for hours yet the amount of work completed varies wildly by the second. This is why VM performance feels unpredictable.

We consider 2 extreme scenarios. In typical data center, you're fluctuating between these 2 edges.

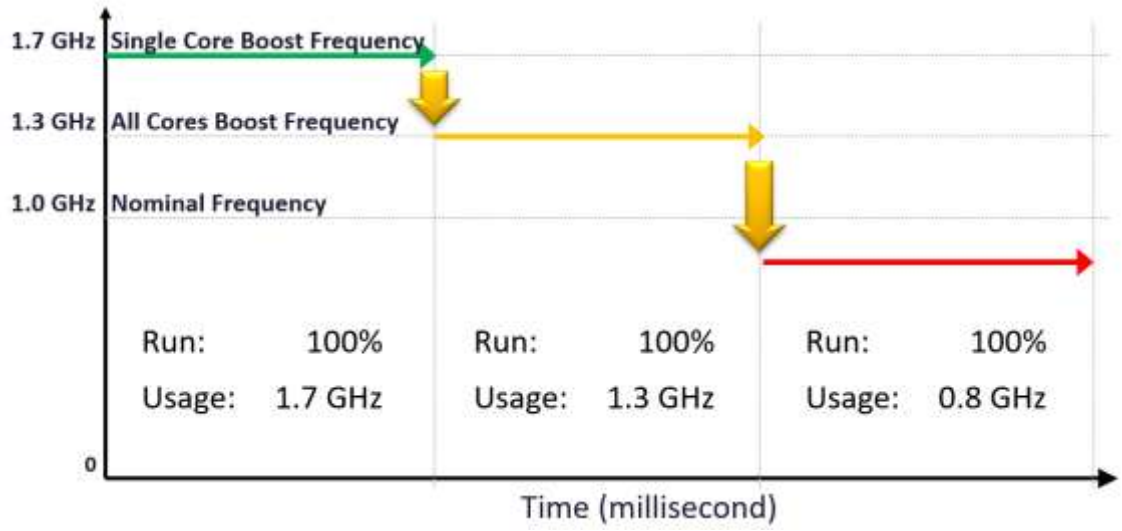


One 1 extreme, the entire CPU only runs 1 core out of 20 cores. If a vCPU runs at 100% and has 0% contention, it will get 1.7 GHz cycles. This is 70% higher than the nominal speed. VM CPU counter such as Demand will report 170%. If you're not careful, you will think you need more vCPU. In this case, adding more vCPU will make the situation worse.

Let's say you add a second vCPU. For simplicity, let's assume they run on the same core. What happens is hyperthreading penalty kicks in. Each only gets 62.5%. So now both vCPU gets 1.06 GHz. So instead of getting 3.4 GHz total, you get 2.12 GHz.

Now let's look at the other extreme. All the cores are running at the same time. A VM with 1 vCPU will run at 1.3 GHz. Again, since VM CPU Demand is based on 1 GHz, it will report 130%. You then add another vCPU. Now, since all the cores are busy, there is a real chance that the VM 2 vCPU will end up on the same core. In this case, each will run at 812 GHz. You will now be confused, as both runs at 81% yet it seems to have maxed out.

The flow over time in the preceding example is 1.7 GHz → 1.3 GHz → 1.06 GHz → 0.81 GHz, as ESXi CPU scheduler will maximize throughput. The problem is CPU Usage (%) metric mask out this degradation. Couple with the fact that it happens over time as VMs get added into the clusters, it becomes complex to figure out.



VM

Take note that some metrics are for the kernel internal consumption, and not for vSphere administrators. Just because they are available in the UI and have names that sound useful do not mean it's for your operations. Their name is written from CPU scheduler viewpoint.

I will use the vSphere Client as the source of metrics in the following screenshots.

vSphere provides 6 metrics to track contention.

Counters	Rollups	Units	Stat Type	Description
Latency	Average	%	Rate	Percent of time the virtual machine is unable to run because it is contending for access to the physical CPU(s)
Max limited	Summation	ms	Delta	Time the virtual machine is ready to run, but is not run due to maxing out its CPU limit setting
Overlap	Summation	ms	Delta	Time the virtual machine was interrupted to perform system services on behalf of itself or other virtual machines
Readiness	Average	%	Rate	Percentage of time that the virtual machine was ready, but could not get scheduled to run on the physical CPU
Ready	Summation	ms	Delta	Time that the virtual machine was ready, but could not get scheduled to run on the physical CPU during last measurement interv...
Co-stop	Summation	ms	Delta	Time the virtual machine is ready to run, but is unable to run due to co-scheduling constraints

You get 9 metrics for consumption.

Counters	Rollups	Units	Stat Type	Description
Demand	Average	MHz	Absolute	The amount of CPU resources a virtual machine would use if there were no CPU contention or CPU limit
Demand-to-entitlement ratio	Latest	%	Absolute	CPU resource entitlement to CPU demand ratio (in percents)
Entitlement	Latest	MHz	Absolute	CPU resources devoted by the ESX scheduler
System	Summation	ms	Delta	Amount of time spent on system processes on each virtual CPU in the virtual machine
Usage	Average	%	Rate	CPU usage as a percentage during the interval
Usage in MHz	Average	MHz	Rate	CPU usage in megahertz during the interval
Used	Summation	ms	Delta	Total CPU usage
VCPU Usage	Average	%	Rate	Virtual CPU usage as a percentage during the interval
Run	Summation	ms	Delta	Time the virtual machine is scheduled to run

I group Wait metrics separately as it mixes both contention and consumption.

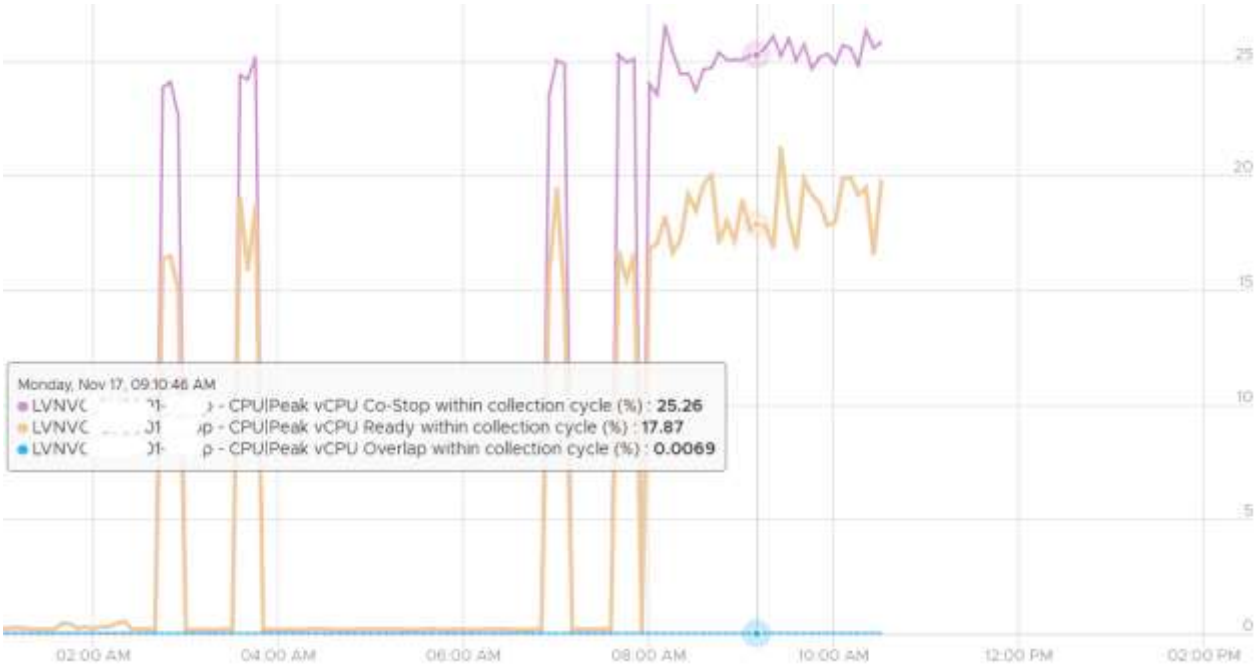
Counters	Rollups	Units	Stat Type	Description
Swap wait	Summation	ms	Delta	CPU time spent waiting for swap-in
Idle	Summation	ms	Delta	Total time that the CPU spent in an idle state
Wait	Summation	ms	Delta	Total CPU time spent in wait state

Contention Metrics

Let's dive into each counter. As usual, we start with contention type of metrics, then utilization.

Contention has to be judged within the context of consumption. A 2% CPU Ready relative to a 10% CPU Run is "not good", while 2% Ready on 98% Run is "good". What I meant is the application may not feel the later, as 98% Run means the application gets a lot of work done. On the other hand, if you have an oversized VM, and it only needs 10% CPU power because business is bad, a 2% CPU Ready means 20% of that business transactions were delayed.

The main 2 counters are Ready and Co-stop. The other 2 counters (Overlap and Other Wait) tend to show much lower value, hence less important operationally. The following shows a typical observation, where both are very high, yet CPU Overlap is near 0.



The preceding showed the 20-second peak metric, taken at vCPU level. Taking at the VM level (it has 8 vCPU) shows a similar pattern.

Guest OS is not aware of both Co-stop and Ready. The vCPU freezes. "What happens to you when time is frozen?"[6] is a great way to put it. As far as the Guest OS is concerned, time is frozen when it is not scheduled. Time jumps when it's scheduled again.

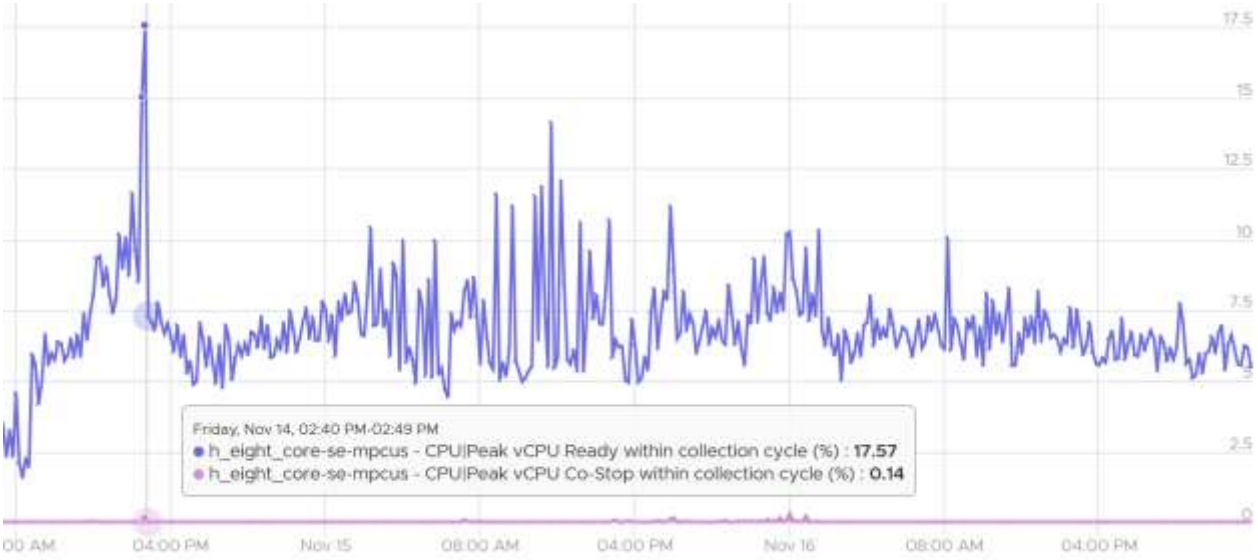
[6] Asked to me by Valentin Bondzio in one of the VMworld events where we got to meet. Those were the days!

The time it spends under Co-stop or Ready should be included in the Guest OS CPU sizing formula as the vCPU wants to run actually.

If VM utilization is not high, reduce its vCPU while following NUMA best practice.

High CPU Ready can happen at low CPU Co-stop. This means everytime the VM vCPU wants to run, there is enough physical CPU threads to run all the vCPU at the same time. So it's an all or nothing situation.

The following shows an 8 vCPU VM with ready hit 7.5% multiple times, while Co-stop remains near 0% for days.



Ready

Ready tracks the time when a VM vCPU wants to run, but ESXi does not have a physical thread (not core) to run it. This could be due to the VM itself (e.g. it has insufficient shares relative to other VMs, it was vMotion) or the ESXi (e.g. it is highly utilized. A sign of ESXi struggling is other VMs are affected too).

When the above happens, ESXi CPU Scheduler places the VM vCPU into Ready state.

Ready also accounts when Limit is applied, as the impact to the vCPU is the same (albeit for a different reason altogether). When a VM is unable to run due to Limit, it accumulates limbo time when sitting in the limbo queue. Be careful when using a Resource Pool, as it can unintentionally cause limits.

Take a look at the high spikes on CPU Ready value. It hits 40%!

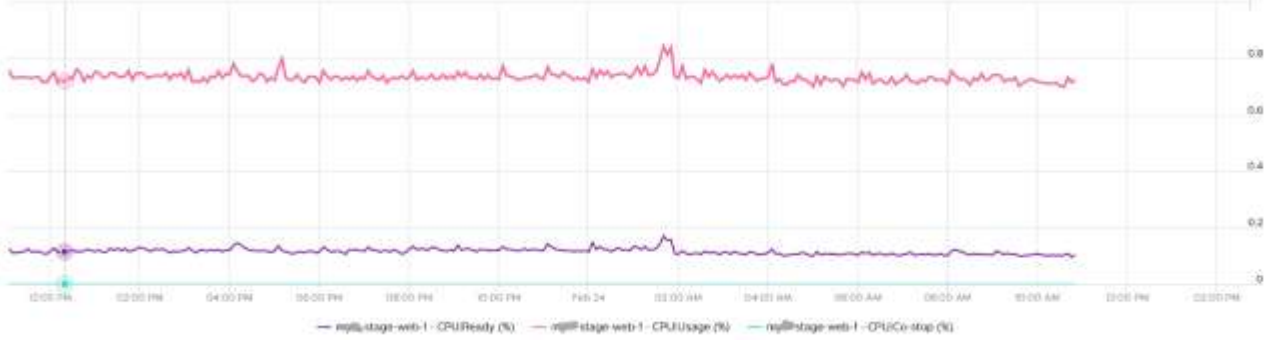


Notice the overall pattern of the line chart correlates very well with CPU Usage and CPU Demand. The CPU Usage hit 3.95 GHz but the Demand shot to 6.6 GHz. This is a 4 vCPU VM running on a 2.7 GHz CPU, so its total capacity is 10.77 GHz. Why did Usage stop at 3.95 GHz?

What's causing it?

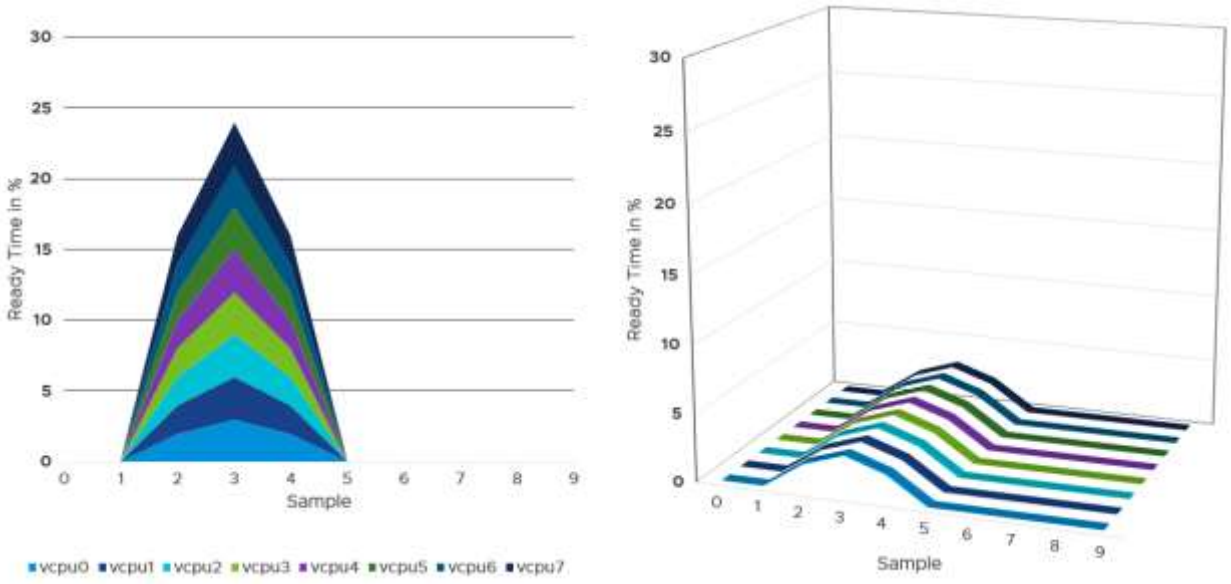
If your guess is Limit you are right. This VM had a limit set at 4 GHz.

Ready also includes the CPU scheduling cost (normally completed in microseconds), hence the value is not a flat 0 on idle VM. You will notice a very small number. Ready goes down when Guest OS is continually busy, versus when a process keeps waking up and going to sleep, causing the total scheduling overhead to be higher. The following shows Ready is below 0.2% on an idle VM (running at only 0.8%). Notice Co-stop is basically flat 0.



CPU Ready tends to be higher, ceteris paribus, in larger VMs. The reason is the chance of running more vCPU at the same time is lower in a busy ESXi host.

Instead of thinking of CPU ready in 2D (as shown in the first chart below), think in 3D where each vCPU moves across time. Each needs to be scheduled, and ideally they run together. The 2nd chart below shows how the 8 vCPUs move across time better[7].

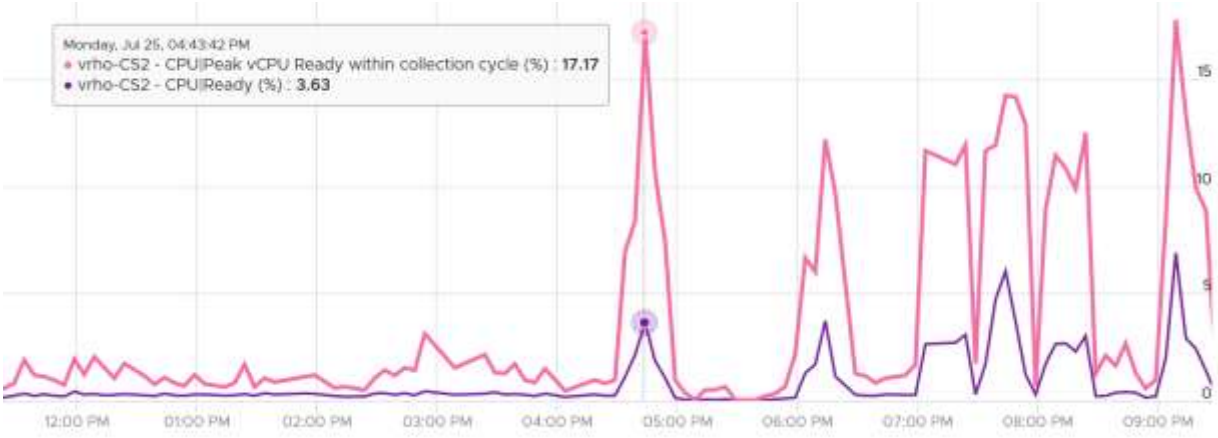


Best Practice

I sample 3937 VMs from production environment. For each of them, I took the 20-second peak and not the 5-minute peak.

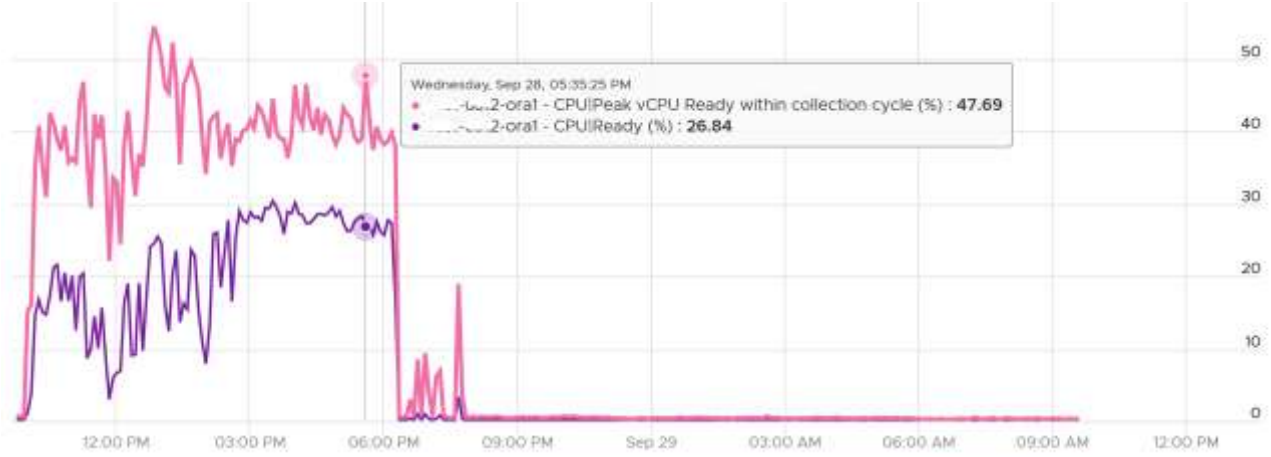
Why do I take the 20-second?

Unless the performance issue is chronic, CPU Ready tends to last in seconds instead of minutes. The following is one such example.



The following shows a different behaviour. Notice initially both metrics are bad, indicating severe CPU ready. However, the gap is not even 2x. I think partly because the value is already very high. Going beyond 50% CPU Ready when CPU Usage is high will result in poor performance. This VM has 16 vCPU.

[7] Source is one of many Valentin Bondzio presentations



Subsequently, the performance improved, and both values became very similar and remained in a healthy range.

I collected 4 months' worth of data, so it's around 35K metrics per VM.

The following screenshot was my result. What do you expect to get in your environment?

Name	20-second: Worst	20-second: 99th Percentile	vCPU
HSS-P...	40.24 %	0.06 %	4
RPA-P...	33.32 %	4.41 %	4
GPRT-...	29.75 %	0.21 %	4
RPA-P...	29.08 %	2.21 %	4
RPA-P...	25.96 %	0.46 %	2
RPA-P...	24.66 %	0.55 %	2
RPA-P...	24.26 %	2.32 %	2
cp-mic...	22.8 %	14.33 %	4
RPA-P...	22.32 %	0.77 %	2
control...	17.44 %	8.81 %	8
Average	0.97 %	0.37 %	5.37

1 - 10 of 3937 items ‹ 1 2 3 4 5 ... 394 ›

The first column takes the highest value from ~35K data points. The table is sorted by this column, so you can see the absolute worst from 35040 x 3937 = 137 million data points. Unsurprisingly, the number is bad. Going down the table, it's also not surprising as the worst 10 are bad.

But notice the average of these "worst metrics". It's just 0.97%, which is a great number.

The 2nd column complements the first one. I eliminate the worst 1% of the data, then took the highest. So I took out 350 datapoints. Since VCF Operations collects every 5 minutes, that eliminates the worst 29 hours in 4 months. As you can expect, for most VMs the values improve dramatically. The 2nd column is mostly green.

Ready | Readiness

There are 2 metrics provided: Ready (ms) and Readiness (%).

I plotted both of them. They show identical pattern. This is a 4 vCPU VM, hence the total is 80000 ms.



The Readiness (%) has been normalized, taking into account the number of vCPU. Notice 80000 ms matches with 100%. If it is not normalized, you will see 80000 as 400%.

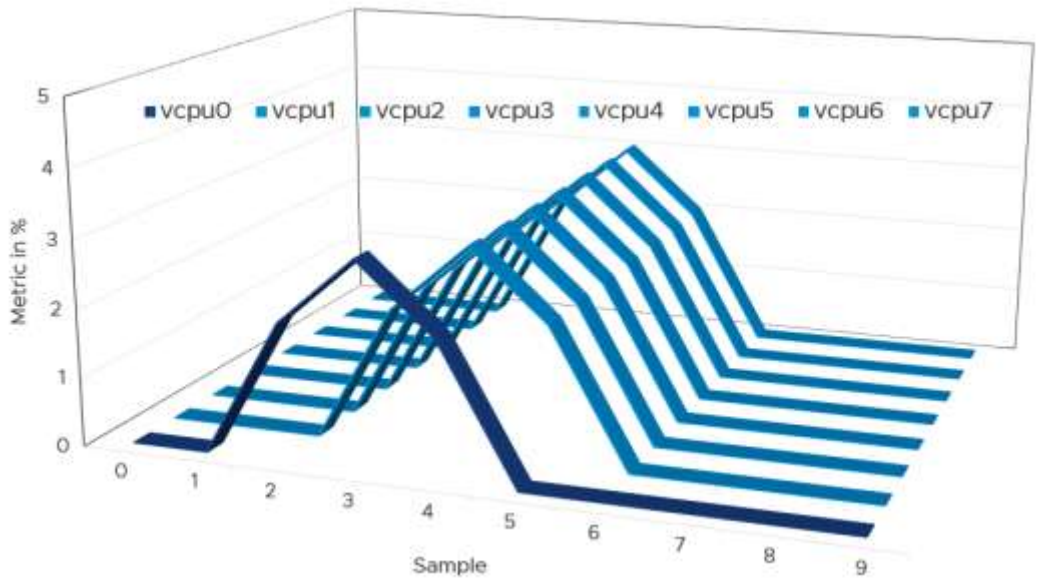
Co-stop

Co-stop is a different state than Ready because the cause is different. The effect to the VM is the same. A pause is a pause. The Guest OS is unaware of the cause and experience the same contention.

Co-stop only happens on Simultaneous Multi Processor (SMP) VMs. SMP means that the OS kernel executes parallel threads. This means Co-stop does not apply to 1 vCPU VMs, as there is only 1 active process at any given time. It is always 0 on single vCPU VM.

In a VM with multiple vCPUs, ESXi kernel is intelligent enough to run some of the VM vCPUs when it does not have all physical threads to satisfy all the vCPU. At some point, it needs to stop the running vCPU, as it's too far ahead of its sibling vCPU (which it cannot serve, meaning they were in ready state). This prevents the Guest OS from crashing. The Co-stop metrics track the time when the vCPU is paused due to this reason. This explains why Co-stop tends to be higher on a VM with more vCPUs.

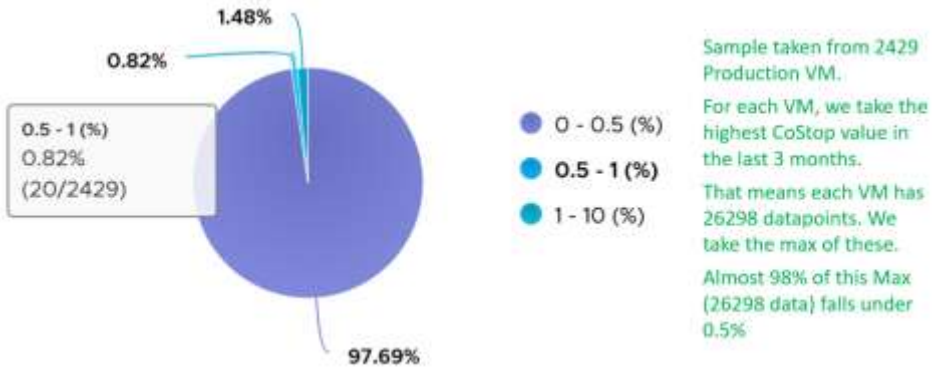
Say one vCPU is in ready state. The remaining vCPU will eventually be co-stopped, until all the vCPUs are co-started. The following diagram show vCPU 0 hit a ready state first. Subsequently, the remaining 7 vCPU hit a co-stop, even though there were actually physical thread to run them.



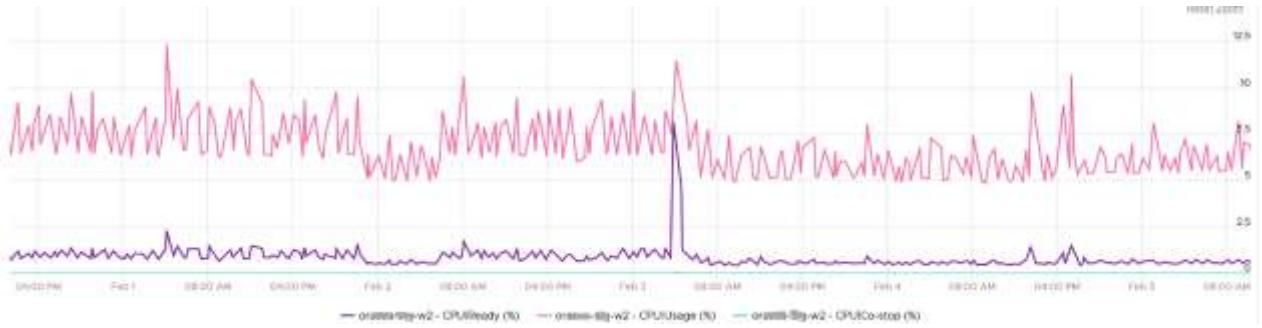
One reason for Co-stop is snapshot. Refer to this KB article for details.

Best Practice

The value of Co-stop should be <1% in high performing environment. This is based on 64 million datapoints, as shown on the following pie chart.



The value of Co-stop tends to be smaller than Ready, as shown below. Ready and Co-stop may or may not corelate with Usage. In the following chart you can see *both* the correlation and lack of correlation.



Overlap

When ESXi is running a VM, this activity might get interrupted with IO processing (e.g. incoming network packets). If there is no other available cores in ESXi, the kernel has to schedule the work on a busy core. If that core happens to be running VM, the work on that VM is interrupted. The counter Overlap accounts for this, hence it's useful metric just like Ready and Co-stop counter.

The interrupt is to run a system service, and it could be on behalf of the interrupted VM itself or other VM.

Notice the word *system services*, a process that is part of the kernel. This means it is not for non-system services, such as vCPU world. That's why the value in general is lower than CPU Ready or even Co-Stop. The value is generally affected by disk or network IO.

Some documentation in VMware may refer to Overlap as Stolen. Linux Guest OS tracks this as Stolen time.

When a vCPU in a VM was interrupted, the vCPU Run counter is unaware of this and continues tracking. To the Guest OS, it experiences freeze. Time stops for this vCPU, as everything is paused. The clock on motherboard does not tick for this vCPU. Used and Demand do account for this interruption, making them useful in accounting the actual demand on the hypervisor. When the VM runs again, the Guest OS experiences a time jump.

Review the following charts. It shows CPU Usage, CPU Overlap and CPU Run. See the green highlights and yellow highlights. What do you notice?



The above prove that Run is not aware of overlap. Notice when overlap went up, Run did not go lower. CPU Usage however, did go down as it's aware of overlap.

The correlation is not perfect as Usage is also aware of hyperthreading and CPU frequency.

The Overlap counter is useful to troubleshoot performance problem, complementing Ready, Co-stop, Other Wait and Swap Wait. Ready does not include Overlap as the VM remains on the Run State (see the CPU State Diagram).

The unit is millisecond, and it's the summation of the entire 20 seconds. VCF Operations averages over 300 seconds. So the amount at 300 seconds is max 20000 (this is 100%), and must be multiplied by 15 if we want to see the actual average in the 300 second period.

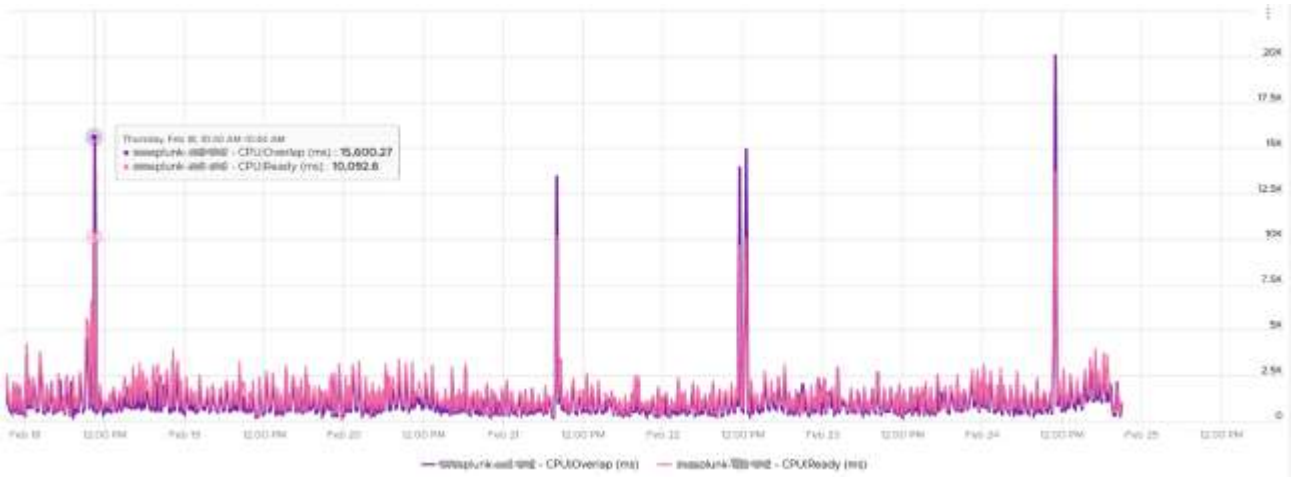
The amount is the sum of all vCPU, so you need to divide by the number of running vCPU if you are converting into a percentage. Divide over 20000 ms x 100%. When I did that, and plot the highest 5 among ~3K production VMs, I get this.

Overlap (ms)	vCPU	Overlap (%)
6,169	30	1.03%
284	2	0.71%
509	4	0.64%

484	4	0.61%
237	2	0.59%

The above indicates the VMs only experienced minimal interruption by the kernel.

Let's dive into a single VM. The following is a 68 vCPU VM running Splunk. In the last 7 days, it experienced a low but sizeable CPU overlap. 10K is relatively low for a 68 vCPU VM, but it still represents half a vCPU worth of interruption.



Overlap should be included in **Guest OS sizing** as the Guest OS wants to run actually. The effect is the same with an unmet Demand.

A high overlap indicates the ESXi host is doing heavy IO (Storage or Network). Look at your NSX Edge clusters, and you will see the host has relatively higher Overlap value versus non IO-intensive VM.

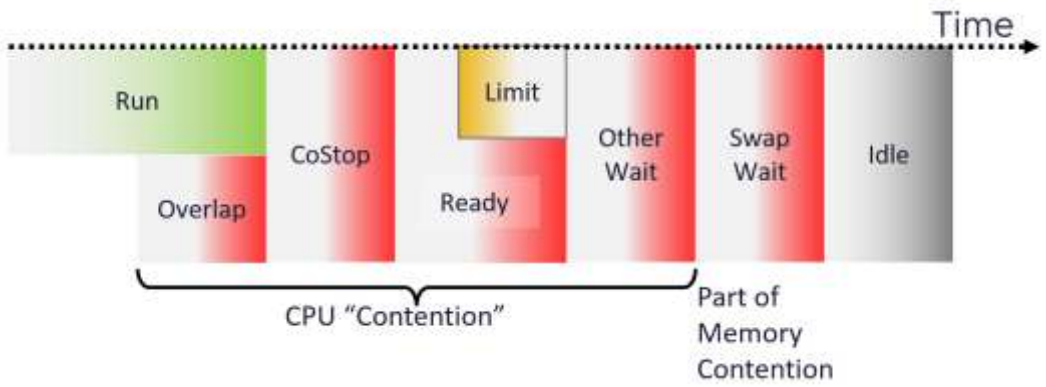
Contention

This is a value add by Aria Operations in July 2024. Before that, it's mapped to the CPU Latency metric.

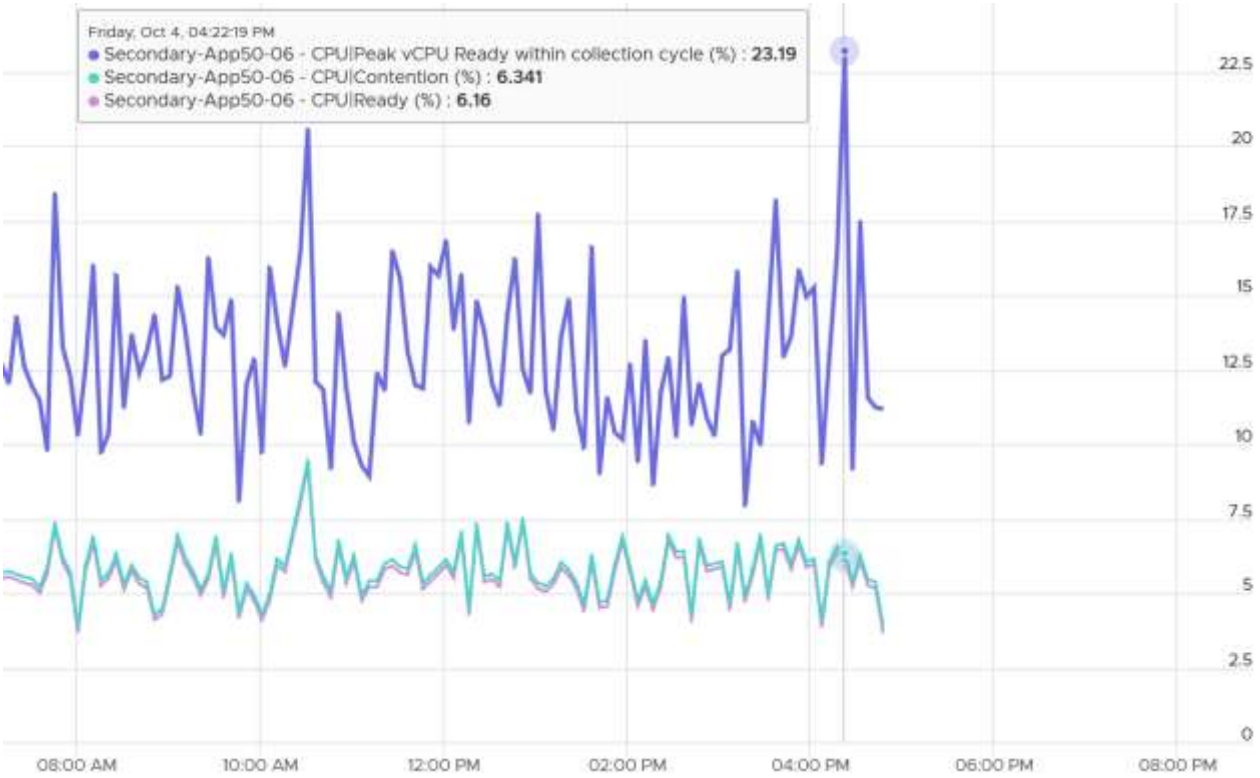
The formula is.

```
Contention (%) = Overlap + CoStop + Ready + Other Wait (if less than 1%)
```

Contention means the vCPU is not getting the CPU time. Using this logic, the CPU frequency is not even applicable as the vCPU is not even running.



In most situations, the value will be mostly coming from CPU Ready. The following screenshot shows the CPU Contention is basically identical to CPU Ready



Why does the counter include Ready by Limit?

Because we need to separate the what and the why. The fact is the VM faces contention. Whether that is because of VM Limit, insufficient VM Shares, or high ESXi utilization is a different issue.

What is the limitation for including Ready by Limit?

You cannot use the counter to define ESXi or cluster performance. The limit has nothing to do with the cluster.

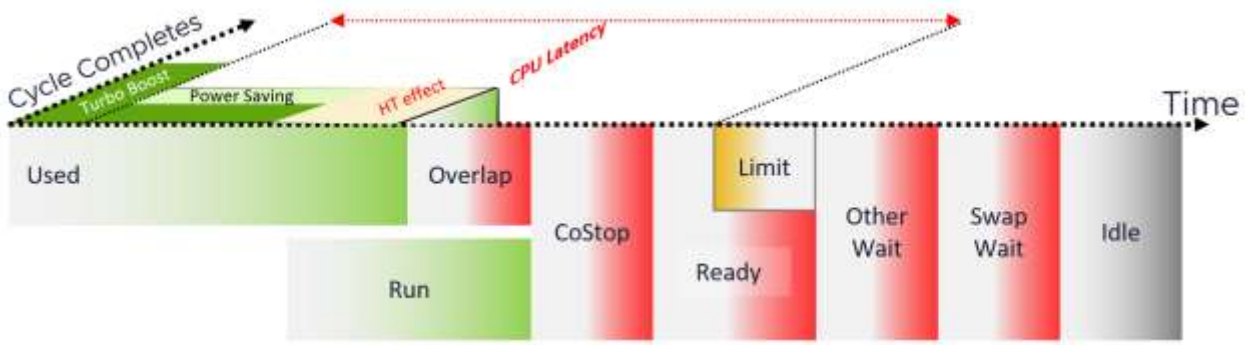
Latency

CPU Latency tracks the "stolen time", which measures the CPU cycle that could have been given to the VM in ideal scenario. It maps to %LAT_C counter in esxtop.

The diagram[8] shows what it includes. Latency excludes Max Limited in Ready, but it includes Co-stop even if the Co-stop was the result of Limit. It also excludes Other Wait.

Notice that HT and CPU Frequency are effect and not metrics. You can see the impact of CPU Frequency in esxtop %A/MPERF counter.

[8] Modified from one of many Valentin Bondzio diagrams



Latency also includes 37.5% impact from Hyper Threading. In ESXi CPU accounting, Hyper Threading is recorded as giving 1.25x throughput, regardless of actual outcome. That means when both threads are running, each thread is recorded as only getting 62.5%. This will increase the CPU Latency value to 37.5%. All else being equal, VM CPU Contention will be 37.5% when the other HT is running. This is done so Used + Latency = 100%, as Used will report 62.5% when the vCPU has a competing thread running.

In the above scenario, what's the value of CPU Ready?

Yup, it's 0%.

CPU Latency also accounts for power management. When the clock speed falls below nominal frequency, CPU Latency goes up accordingly.

Because of these 2 factors, its value is more volatile, making it less suitable as a formal Performance SLA. Use CPU Ready for Performance SLA, and CPU Contention for performance troubleshooting.

The following table only shows 5 VM out of 2500 that I analyzed. These 2 metrics do not have good correlation, as they are created for different purpose.

	Max Contention	Ready (at CPU|Contention (%) maximum)	Max Ready	Contention (at CPU|Ready (%) maximum)
	33.12 %	2.32 %	2.32 %	33.12 %
)T49	29.75 %	0.54 %	1.2 %	9.72 %
-a2	27.92 %	0.91 %	1.47 %	11.12 %
4	27.56 %	0.78 %	0.97 %	23.76 %
-a1	27.16 %	0.93 %	1.18 %	20.42 %

In many cases, the impact of both threads running is not felt by the application running on each thread. If you use CPU Latency as formal SLA, you may be spending time troubleshooting when the business does not even notice the performance degradation.

The following screenshot shows CPU Latency went down when both Ready and Co-stop went up.



How about another scenario, where Latency is near 0% but Ready is very high? Take a look at this web server. Both CPU Demand and CPU Usage are similar identical. At around 1:40 am mark, both Demand and Usage showed 72.55%, Contention at 0.29%, but Ready at above 15%. What's causing it?



The answer is Limit. Unlike CPU Ready, it does not account for Limit (Max Limited) because that's an intentional constraint placed upon the VM. The VM is not contending with other VMs. VMware Cloud Director sets limit on VM so this counter will not be appropriate if you aim to track VM performance using Contention (%) metric.

Here is a clearer example showing latency consistently lower than Ready due to limit.



A better and more stable metric to track the contention that a VM experience is Ready + Co-stop + Overlap + VM Wait + Swap Wait. Note that the raw metric for all these are millisecond, not GHz.

Where do you use CPU Contention then?

Performance troubleshooting for CPU-sensitive VM.

If the value is low, then you don't need to check CPU Ready, Co-stop, Power Management and CPU overcommit. The reason is they are all accounted for in CPU Contention.

If the value is high (> 37.5%), then follow these steps:

- Check CPU Run Queue, CPU Context Switch, "Guest OS CPU Usage", CPU Ready and CPU Co-stop. Ensure all the CPU metrics are good. If they are all low, then it's Frequency Scaling and HT. If they are not low, check VM CPU Limit and CPU Share.
- Check ESXi power management. If they are set to Maximum correctly, then Frequency Scaling is out (you are left with HT as the factor), else HT could be at play. A simple solution for applications who are sensitive to frequency scaling is to set power management to max.
- Check CPU Overcommit at the time of issue. If there is more vCPU than pCore on that ESXi, then HT could be impacting, else HT not impacting. IMHO, it is rare that an application does not tolerate HT as it's transparent to it. Simplistically speaking, while HT reduces the CPU time by 37.5%, a CPU that is 37.5% faster will logically make up for it.

There is a corner case accounting issue in %LAT_C that was resolved in ESXi 6.7[9]. VMs with Latency Sensitive = High on ESXi 6.5 or older, will show any "guest idle" time of vCPUs as LAT_C, for those VMs the counter should not be relied on. This is a corner case because majority of VM should not be set with this, as it impacts performance of other VMs.

Wait Metrics

I put the wait metrics separately as it includes both the contention and consumption type of metrics.

[9] Both 6.5 and 6.7 have End of General Support on 15 October 2022 and End of Technical Guidance on 15 November 2023

CPU is the fastest component among infrastructure resources, so there are times it must wait for data. The data comes from memory, disk or network.

There are also times when there is nothing to do, so the CPU is idle. The hypervisor did schedule the VM vCPU to run, but there was nothing to run. Whether the upper layer (Guest OS vCPU) is truly idle or blocked by pending IO, the kernel does *not* have the visibility. It can only see that Windows or Linux is not doing any work.

There are 3 sub-metrics that make up Wait.

- Idle. Waiting for work.
- Swap Wait. Waiting for memory.
- Other Wait. Waiting for other things.

CPU|Wait (%)	CPU|Idle (%)	CPU|Other Wait (%) ↓	CPU|Swap wait (%)
99.99 %	93.78 %	6.21 %	0 %
99.99 %	93.79 %	6.2 %	0 %
99.93 %	95.88 %	4.05 %	0 %
99.85 %	96.68 %	3.16 %	0 %
99.79 %	97.21 %	2.58 %	0 %
99.83 %	97.66 %	2.17 %	0 %
99.85 %	97.71 %	2.14 %	0 %
98.88 %	96.75 %	2.13 %	0 %
98.59 %	96.79 %	1.81 %	0 %
98.56 %	96.77 %	1.79 %	0 %

Guest OS isn't aware of both Other Wait and Swap Wait. Just like other type of contention, it experiences freeze. The time it spends under Other Wait and Swap Wait should be included in the Guest OS CPU sizing formula as the VM wants to run actually.

Idle	It tracks when VM is not running. Regardless of the reason in the upper-layer, VM Idle should not be included in both VM sizing, and definitely not in Guest OS sizing. The reason is the vCPU is not running and you can't predict what the usage would be. You should address the IO and memory bottleneck in Guest OS level, using Windows and Linus metrics.
Swap Wait	It tracks the time CPU is waiting for Memory page to come in from ESXi swap. Adding more RAM or faster RAM may result in lower CPU Wait, hence higher CPU Run. This metric was superseded by Memory Contention metric.

Other Wait

It tracks the time CPU is being blocked by things other than memory. This can be disk IO, network IO, or vMotion. For example, the VMM layer is trying to do something and it's blocked. The number of reasons vary and it's hard to pinpoint exactly which one, as you need low level debug logs such as stats vmx, schedtraces, and custom vprobes. You're better off removing the common reasons. Snapshot is a common reason here[10], that it was mistakenly named as IO Wait.

The formula is:

```
Other Wait = Wait – Idle – Swap Wait
```

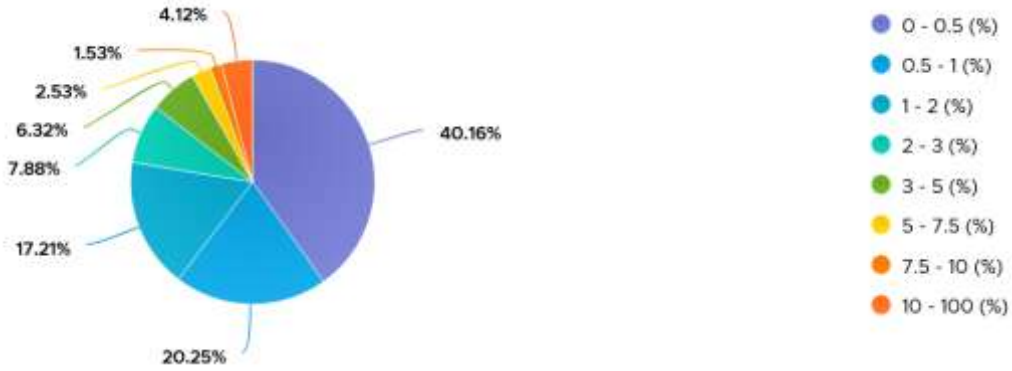
The above is a formula at VCF Operations.

Actions you can do to reduce Other Wait:

- vMotion the VM.
- Remove Snapshot
- Update to the latest build of ESXi (incl. physical device drivers), virtual HW and VMware Tools (virtual device drivers).

If this happens to multiple VMs, find commonality. If the above is not helping in your case, file a Support Request with VMware GSS and tag me. Please mention that you get it from here, so I have a context.

I plotted Other Wait for 4000 production VMs. Surprisingly, the value is not low.



I was curious if the value corelates with CPU Ready or Co-stop. From around 4000 production VM in the last 1 month, the answer is a no.

[10] Based on this KB article, snapshot increases the read operations as every snapshot has to read to ensure you're fetching the correct data. Write is not impacted as you simply write a new block and not updating existing one.

Name	Worst Ready	Worst Co-stop	Worst VM Wait	99p VM Wait
RP...	27.83 %	29.26 %	6.63 %	2.33 %
RP...	24.04 %	16.79 %	1.48 %	0.12 %
RP...	23.84 %	21.04 %	10.4 %	1.03 %
HSS...	20.85 %	17.87 %	0.22 %	0.0058 %
GP...	20.72 %	21.88 %	6.7 %	0.15 %
RP...	19.4 %	15.1 %	1.07 %	0.04 %
RP...	17.08 %	11.62 %	1.69 %	0.07 %
mo...	10.14 %	10.05 %	4.11 %	1.45 %
hdp...	9.4 %	4.32 %	1.3 %	0.06 %
mo...	9.07 %	7.72 %	2.21 %	1.51 %

Since snapshot is another potential culprit, let's compare with disk latency and outstanding IO.

What do you expect?

Name	99p VM Wait	Worst VM Wait	Worst Outstanding IO	Worst Read Latency	Worst Write Latency
ep...	20.02 %	21.13 %	0.4 OIOs	0.76 ms	6.67 ms
scr...	19.59 %	23.45 %	0.11 OIOs	8.87 ms	32.87 ms
har...	17.83 %	19.06 %	0.87 OIOs	0.14 ms	18.53 ms
ATL...	16.59 %	17.91 %	0.48 OIOs	1.67 ms	15.47 ms
mig...	16.56 %	16.62 %	0 OIOs	0 ms	0 ms
edd...	16.16 %	20.17 %	1.55 OIOs	2.07 ms	15 ms
cm...	15.91 %	16.89 %	0.02 OIOs	0.25 ms	6.33 ms
mig...	15.4 %	15.41 %	0 OIOs	0 ms	0 ms
darr...	14.69 %	18.08 %	0.0005 OIOs	0 ms	4.73 ms
cm...	12.75 %	15.71 %	0.01 OIOs	0.24 ms	6.36 ms

Again, negative corelation. None of the VMs with high VM Wait is experiencing latency. Notice I put a 99th percentile, as I wanted to rule out a one-time outlier. I'm plotting the first VM as the value at 99th is very near to the max, indicating sustained problem.



It turned out to be true. It has sustained VM Wait value around 15% (above is zoomed into 1 week so you can see the pattern).

I'm curious why it's so high. First thing is to plot utilization. I checked Run, Usage and Demand. They are all low.



Using VCF Operations correlation feature, I checked if it correlates with any other metric. The only metric it founds is Idle, which is logical they basically add up to 100% when Run is low.

Take note of a known accounting issue that wrongly inflates the value of Other Wait and esxtop %VMWait.

Consumption Metrics

Consumption metric covers only utilization. Reservation and Allocation are properties, not metrics.

The following table shows the 5 VM utilization metrics.

Counter	Unit	Source	Observability
Run	Millisecond	ESXi	vCPU level
Used	Millisecond	ESXi	vCPU level VM level
Usage	MHz	vCenter	vCPU level VM level
Usage	%	vCenter	VM level
Demand	MHz	ESXi	VM level

Note: CPU Used and Usage at the VM level includes the system time and other charges that is not associated with a particular vCPU. If you sum up the vCPU amounts, you will notice a small gap to the amount at VM level. You can verify by comparing the values show on the vCenter performance chart legend.



Run | Net Run

Run is when the Guest OS gets to run and process instruction. It is the most basic counter among the CPU consumption metrics. It's the *only* counter not affected by CPU frequency scaling and hyper threading. It does not check how fast it runs (frequency) or how efficient it runs (SMT).

```
Run at VM level = Sum of Run at vCPU levels
```

Since the unit is millisecond, this means the value of CPU Run at VM level can exceed 20000 ms in vCenter.

The following screenshot shows CPU Run higher than CPU Used. We can't tell if the difference is caused by power management or hyperthreading, or mix of both.



CPU Run does not account for the following:

- Interrupt
- System time. IO performed by hypervisor has to be charged back to the VM.
- Power Management or CPU Frequency Scaling
- Simultaneous Multithreading (Hyper Threading as Intel calls it)

Because CPU Run does not take into account this external work, and not aware of CPU speed and HT, it is *suitable* as input to size the Guest OS and not the VM. The portion that needs to be removed is Overlap. VCF Operations added a new counter in July 2024 called Net Run.

The formula is:

```
Net Run = Run – Overlap
```

As you can guess, Net Run cannot hit 100%.

Used | Usage

The scope of CPU Run metric means it can't answer 2 important questions. When a vCPU is running,

How fast is the "run"?	All else being equal, a 5 GHz CPU is 5x faster than a 1 GHz CPU. The faster it can complete a task, the shorter it has to work. That's why you see some metrics in MHz, because they account for this speed.
How efficient is the "run"?	If there is competing thread running in the same core, the 2 threads have to share the core resource. Both threads do not drop their CPU frequency, but the cycles that each thread receives is 37.5% less. This is where it's better to think in terms of cycles and not frequency

This is where ESXi introduces the CPU Used (ms) counter.

vCenter then adds Usage (MHz) metrics, so you can see the total footprint of the VM in terms of CPU cycles.

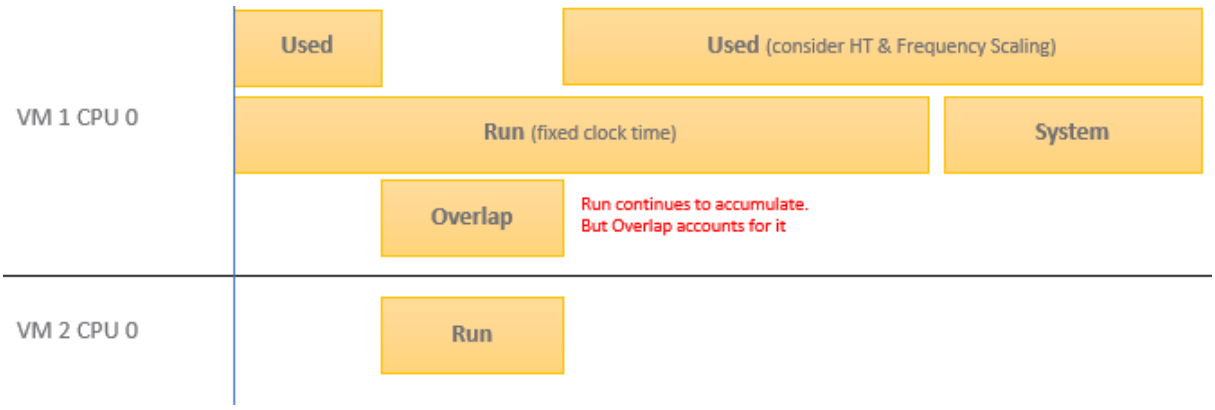
BTW, if the English word usage confuses you, think of CPU Cycles Completed or CPU Throughput instead.

By covering the above 3, CPU Used covers uses cases that CPU Run does not.

- Amount of work done.
- VM Migration. Moving VM to another ESXi requires that you know the actual footprint of the VM, because that's what the destination ESXi needs to deal with.
- VM Chargeback. You should charge the full cost of the VM, and not just what's consumed inside the VM. In fairness, you should also charge the actual utilization, and not rated clock speed.

Used

Here is how Used differs to Run:



Based on the above, you can work out the formula for VM level Used, which is:

```
VM level Used      = Run - Overlap +/- E + System + VMX + MKS + Helper
VM vCPU level Used = Run - Overlap +/- E
```

Where **E** is the combination of

- efficiency gained from CPU Turbo Boost. This is significant, could be 50% more when other physical cores in the socket are idle.

- efficiency loss from power savings. This is also significant. If the frequency is dropped to 40% of the nominal frequency, we consider 60% of the CPU time was stolen.
- 37.5% efficiency loss from CPU SMT. ESXi accounting records HT as 1.25x overall gain, hence each thread drops to 62.5% only. This is a significant drop that should be accounted.

MKS is typically negligible. This thread is dedicated strictly to processing the video frame buffer, mouse coordination, and console graphic rendering. It accounts for CPU cycles spent while you're consoling to the VM, where you use mouse and keyboard on the virtual screen.

VMX handles VM power states, configuration alterations, and high-level virtual hardware emulation outside the execution engine. In esxtop, System time is charged to the VM VMX world.

Helper covers the asynchronous I/of O helper worlds, virtual storage controllers, and background kernel threads specific to that VM instance. This metric is negligible.

Because Used accounts for the actual frequency, you may expect it to be measured in GHz and not millisecond. The "conversion" from GHz to millisecond was based on static frequency. I know it requires a bit of mental mathematics 😊

Quiz:

- Why does the formula state VM level, and not individual vCPU level. What's the reason?
  Answer: CPU Used has a different formula at VM level and vCPU level. At vCPU level, it does not include System Time. At VM level, it includes the work done by the kernel that is charged at VM level, such as System and other worlds.
- How will Used compare with Run in general? Do you expect it to be higher or lower? If it's higher, what can cause it?

For example, a physical chip comes with 2 GHz as its standard speed. ESXi may increase or decrease this speed dynamically, resulting in turbo boost or power saving. If ESXi increases the clock speed to 3 GHz, Used counter will be 50% higher than the Run counter. Older Guest OS (e.g. Windows or Linux) version was not aware of this performance boost. It reported a value based on the standard clock speed, just like Run does. On the other hand, if ESXi decreases the clock speed to 1.5 GHz, then Used will report a value that is 25% lower than what Run reports.

Usage

Usage is Used + work done by kernel on behalf of the VM. This is why you see a gap on VM that processes a lot of network packets or storage commands.

There are two metrics:

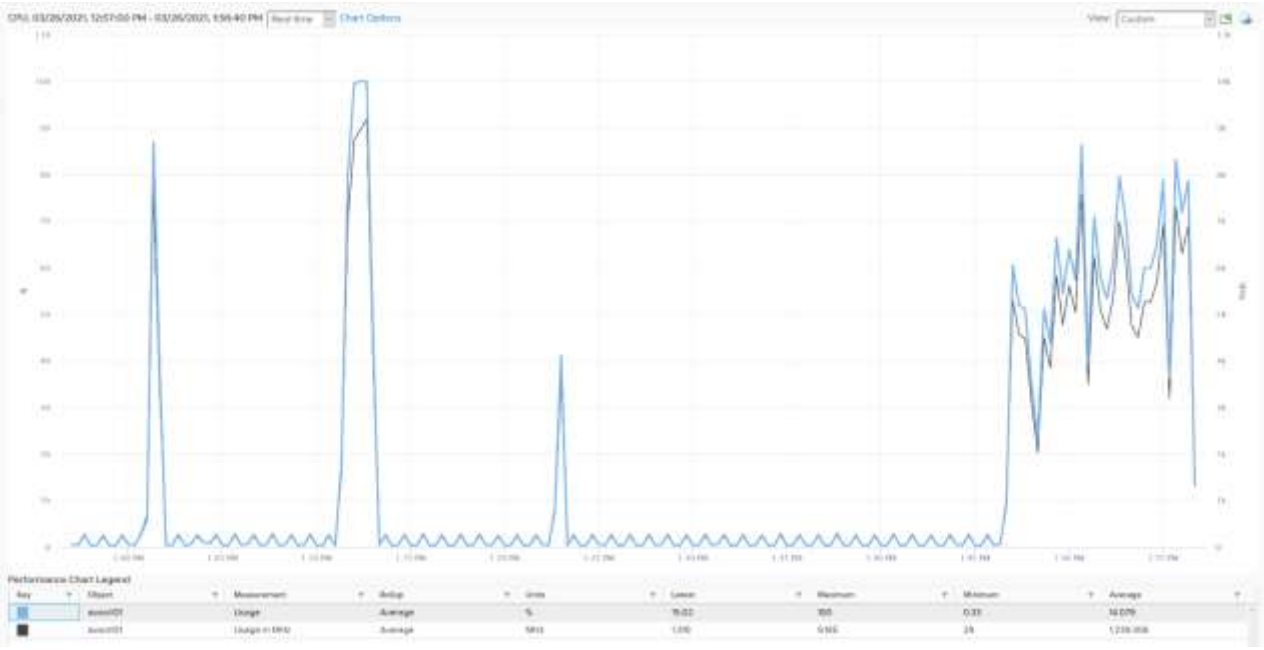
- Usage (MHz)
- Usage (%)

Usage (%) is only available at VM level. Usage (MHz) is available at both vCPU and VM level.

```
Usage (%) = ( Usage MHz for each vCPU + VM level load ) / VM Static CPU Speed
```

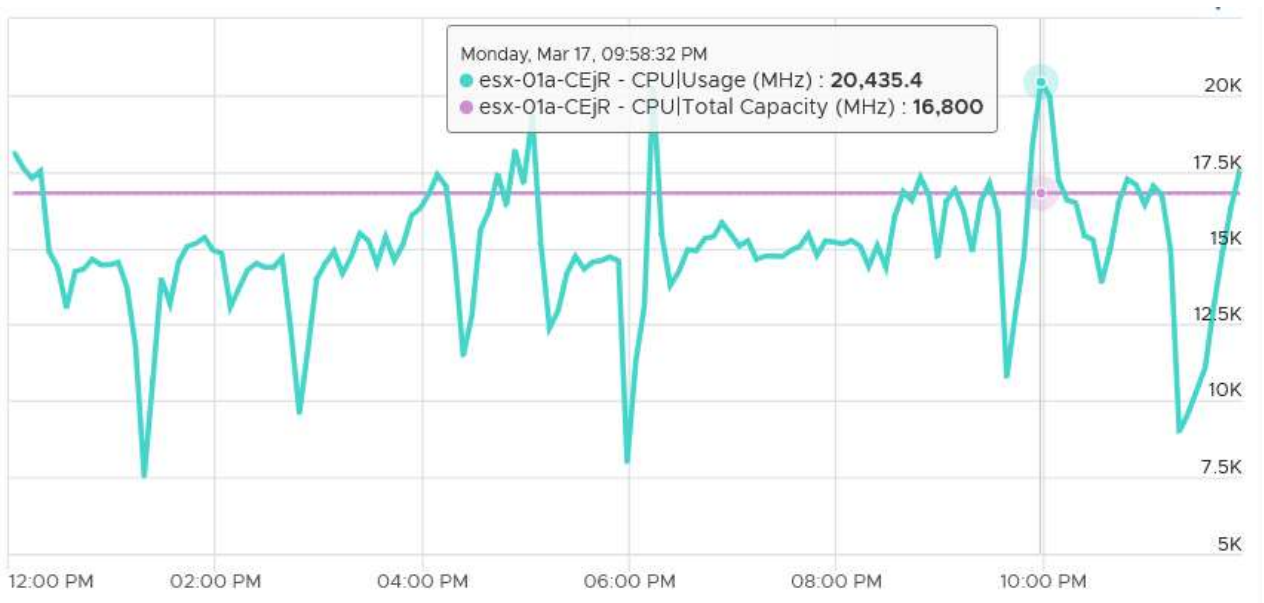
These 2 metrics do not exist in ESXi, meaning they only exist in vCenter.

The reason is Usage (%) is not available on a per vCPU basis, while Usage MHz is. The 2 charts are also *very* similar but not 100% identical. Notice in the following screenshot there is time where they are 100% identical, and there is time they are not. My guess is Usage (%) contains VMX load as it's not available on a per vCPU basis.



Usage (MHz)

Turbo Boost's impact on Usage is real. In the following, you can see the value exceeds the total capacity by a sizable amount.



Let's do a calculation so you can quantify the impact of power management.

Review the following example. This is a single VM occupying an entire ESXi.

The ESXi has 12 cores with nominal frequency of 2.4 GHz. The number of sockets does not matter in this case.

Since HT is enabled, the biggest VM you can run is a 24 vCPU. The 24 vCPU will certainly have to share 12 cores, but that's not what we're interested here.

What do you expect the VM CPU Usage (GHz) when you run the VM at basically 100%?



Well, that depends on the all-core turbo boost of the CPU.

The result above is slightly less than 39 GHz.

If there is no turbo boost, the answer will be 36 GHz.

Why not 57.6 GHz, because it's 24 vCPU x 2.4 GHz?

Because HT does not yield 2x. It yields 1.25x only. At the end of the day, the one that does the computation is the core not the thread.

12 cores x 2.4 GHz 1.25 HT = 36 GHz total capacity with hyperthreading enabled.

In the preceding example, power management was enabled. Naturally Turbo Boost kicked in, albeit not so dramatic as all the physical cores were turbo at the same time.

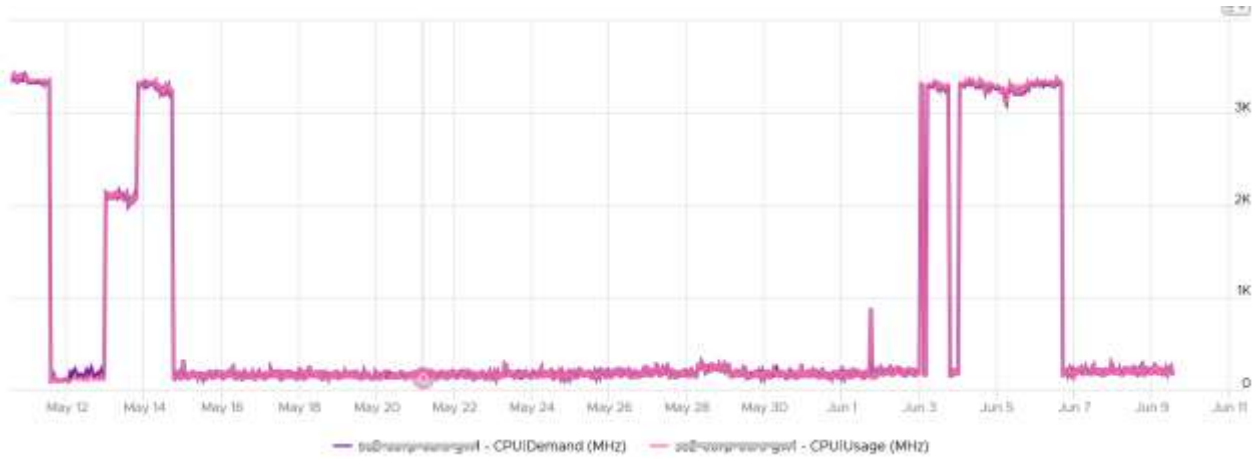
You got around 39 GHz, a small increase over 36 GHz. Formula is 2.4 GHz x 12 cores x 1.25 HT x 1.08x Turbo Boost.

What happens when we disable turbo boost? That's what we did at point 1 in the diagram above.

CPU Usage drops to slightly below 36 GHz.

Usage (%)

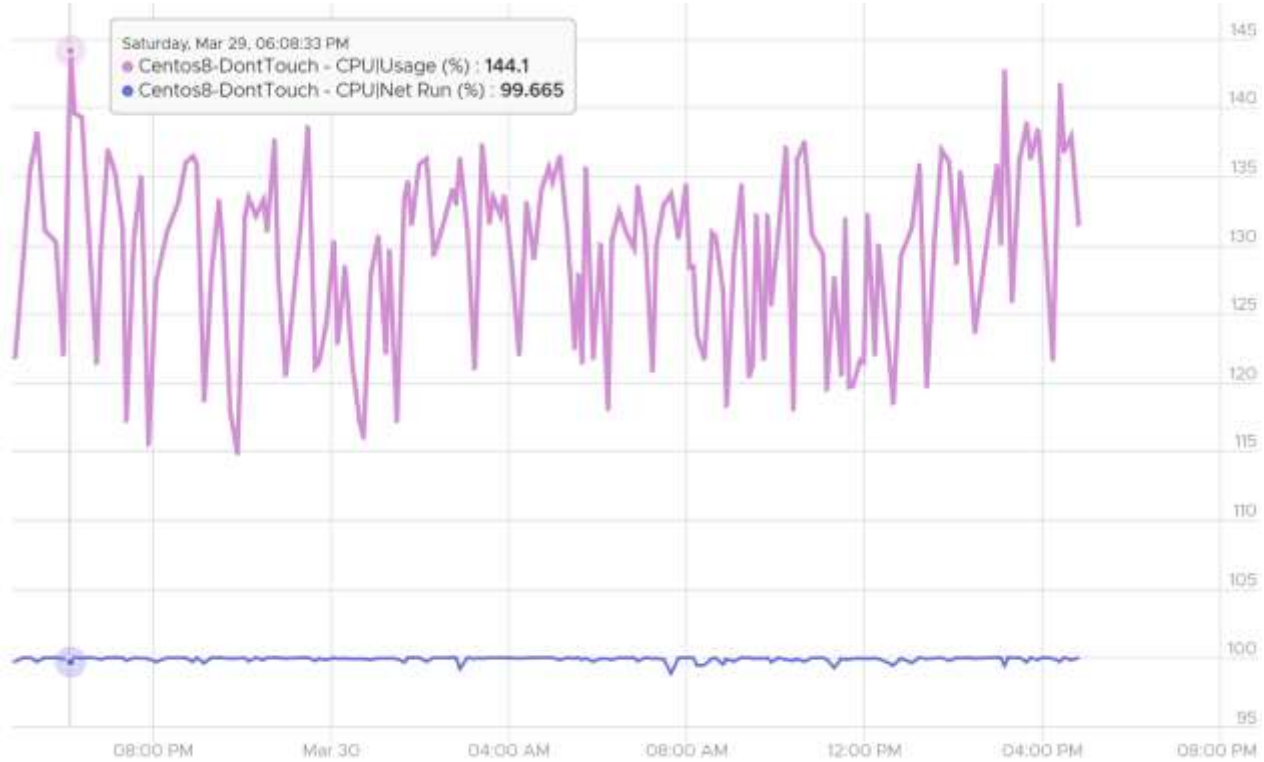
The following is a single vCPU production Windows Server. Both CPU Usage (MHz) and Demand jump to over 100%. Their values are identical for almost 30 days. The VM had near 0% Contention (not shown in chart), hence the 2 values are identical.



Around 12 May, the VM experienced some contention. That's why Demand was higher than Usage.

Usage vs Net Run

What do you notice on the following chat?



Yes, Net Run was basically flat out at 100% for 24 hours. The value did not exceed 100%. Usage on the other hand, was wildly fluctuating. By now you should know why 😊

What do you notice on the following chart?



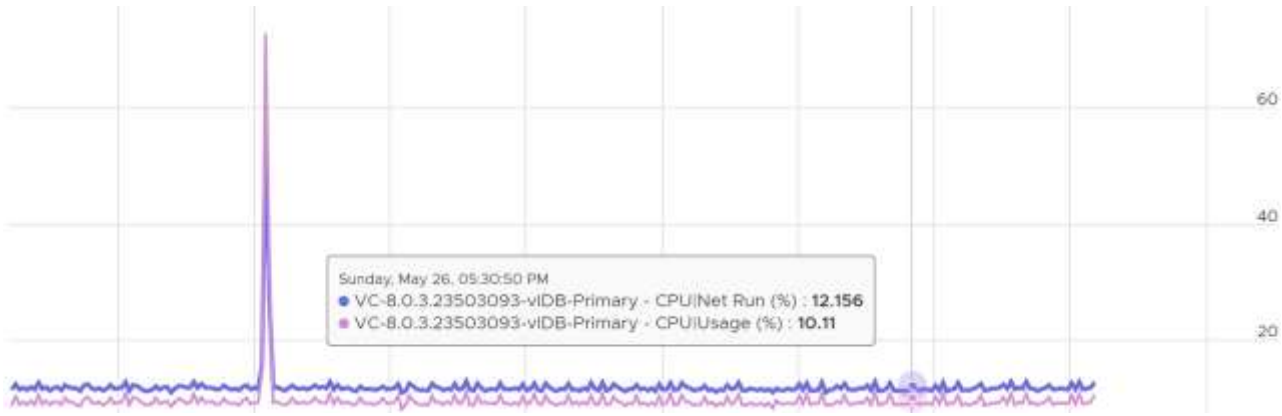
CPU Usage is both higher and lower than CPU Net Run. Why is that so?

Usage is greatly affected by CPU speed and power management. As a result, it's more volatile than CPU Run.

Generally speaking, during high utilization, Usage (%) will over-report due to CPU turbo boost.

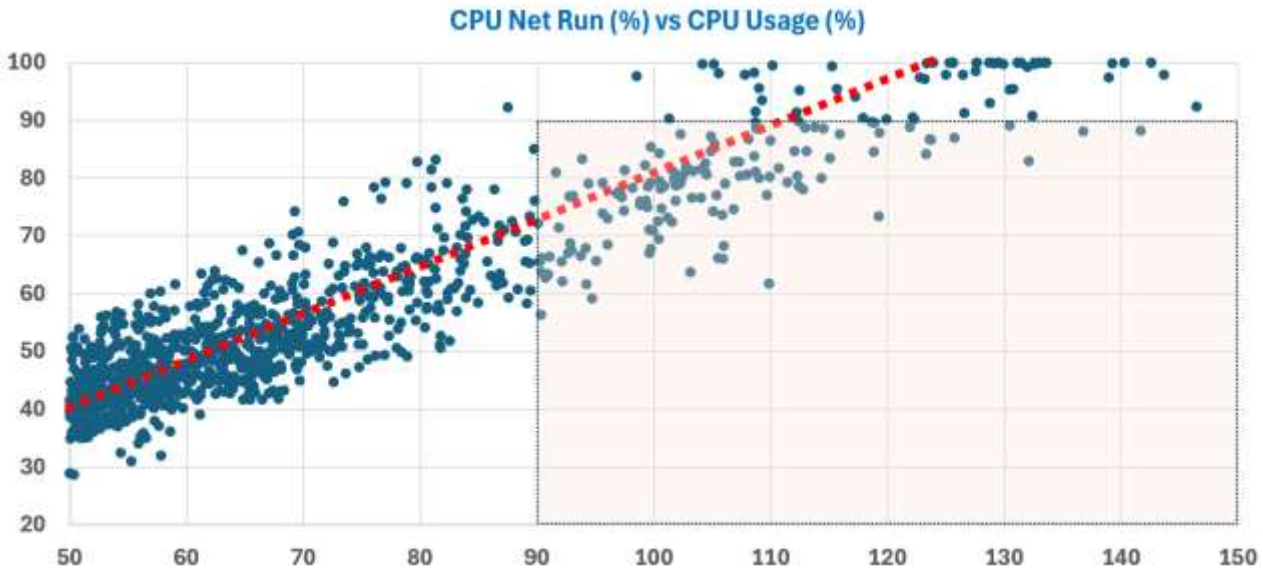


During low utilization, Usage (%) will under report due to power savings



Overall

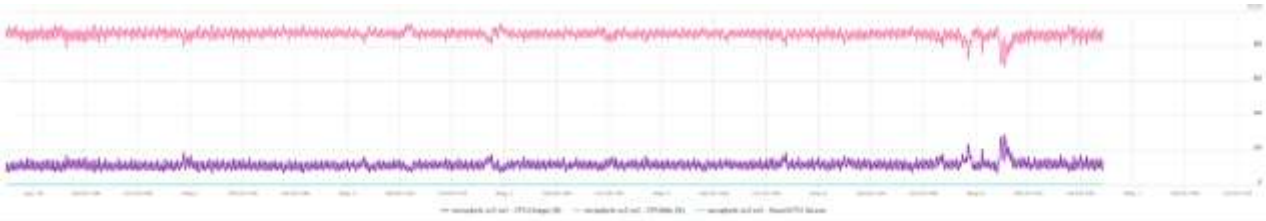
I plotted 1102 VM with Usage (GHz) > 2 and Usage (%) > 50%.
The rectangle is VMs that likely get upsized when there is no need. Usage > 90% but Run is < 90%.
The red line shows that Usage tends to be 25% higher.



CPU Usage Disparity

This metric is required to convince the owners of the VM to downsize their *large* VMs. It's very common for owners to refuse sizing it down even though utilization is low, because they have already paid for it or cost is not an issue.

Let's an example. This VM has 104 vCPU. In the last 90 days, it's utilization is consistently low. The Usage (%) counter never touch 40%. Demand is only marginally higher. Idle (%) is consistently ~20%.



All the key performance metrics such as Guest OS CPU Run Queue are low.

Obviously the VM does not need 104 vCPU. How to convince the owner if he is not interested in refund? The only angle left is performance. But then we're faced with the following:

1. CPU Run Queue inside the Guest OS is low. Decreasing CPU will in fact increase it, which is worse for performance.
2. CPU Context Switch is high from time to time.
3. CPU Co-Stop is very low (max of 0.006% in the last 90 days). Decreasing CPU may or may not make it lower. Regardless, it's irrelevant. Same goes with VM Wait and Swap Wait.
4. CPU Ready is very low (max of 0.14% in the last 90 days).

The only hope we have here to convince VM owner is to give insight on how the 104 vCPU are used. There are 2 ends of the spectrum:

At one end, all 104 are balanced	All are running at that low 20%. This triggers an interesting discussion on why the application is unable to even consume a single vCPU. Is this inefficiency the reason why the application vendor is asking for so many vCPU? Commercially,

	it's wasting a lot of software license
Imbalance	Some are saturated, while others are not. • The Peak among vCPU metric will capture if any of them is saturated. This is good insight. • The Min among vCPU is not useful as there is bound to be 1 vCPU among 104 that is running near 0%. • The delta between Max and Min will provide insight on the degree of the usage disparity. Does it fluctuate over time? This type of analysis helps the application team. Without it they have to plot 104 vCPU one by one.

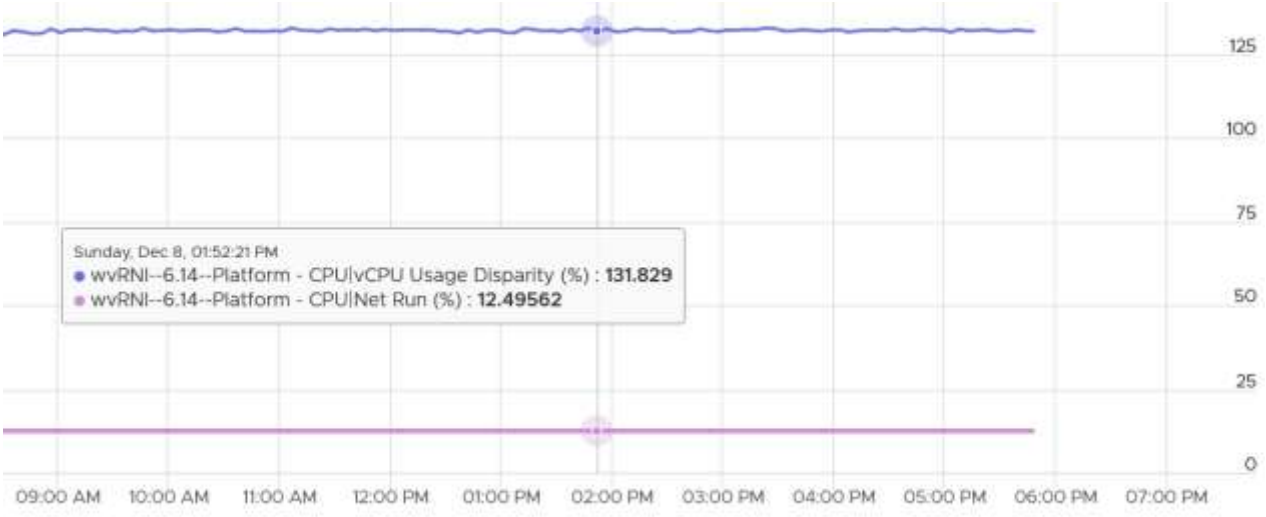
In reality, there could be many combinations in between the 2 extremes. Other insights into the behaviour of the 104 vCPU are:

1. Jumping process. Each vCPU takes turn to be extreme high and low, as if they are taking turn to run. This could indicate process ping pong, where processes either do frequent start/stop, or they jump around from one vCPU to another. Each jump will certainly create context switch, like the cache needs to be warm up. If the target CPU is busy, then the running process was interrupted.
2. CPU affinity. For example, the first 10 vCPU is always much busier than the last 10 vCPU. This makes you think why, as it's not normal.

Naming wise, vCPU Usage Disparity is a better name than Imbalance vCPU Usage. Imbalance implies that they should be balanced, which is not the case. It's not an indication that there is a problem in the guest OS because VCF Operations lacks the necessary visibility inside the guest OS

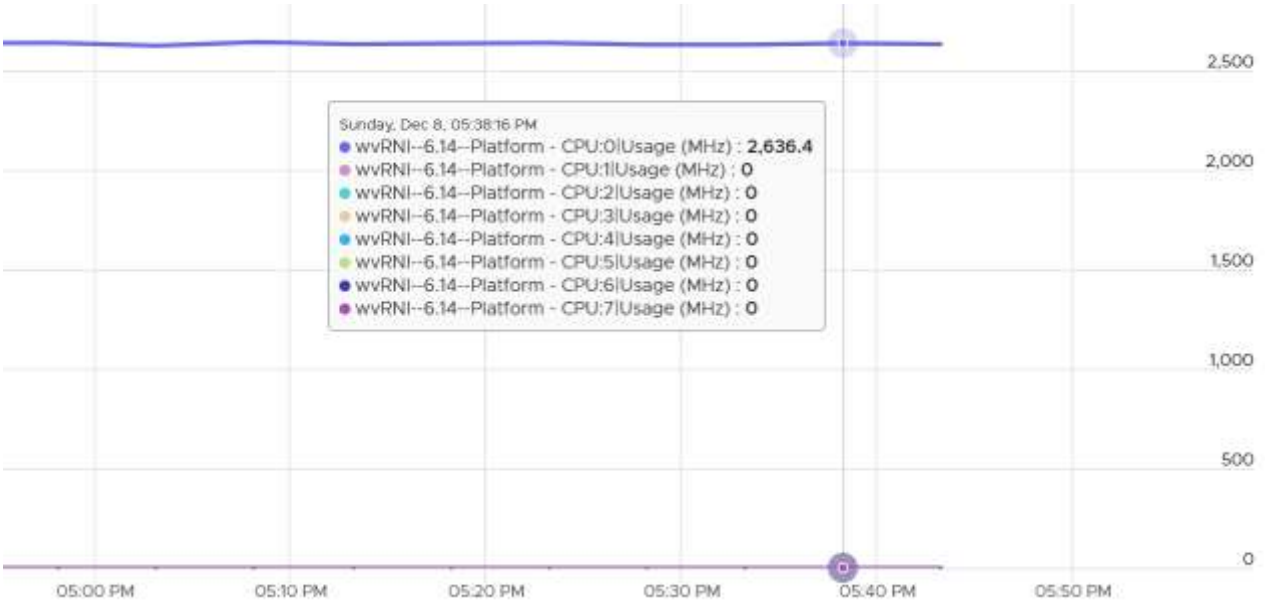
Quiz!

Review the following. How can the average utilization is only 12.% yet the Usage Disparity is >100%



To answer, we need to plot each vCPU one by one.

Below is what we get.



Why only the first vCPU?

My take is uniprocessor kernel. It was compiled without SMP support.

Why did it go above 100%?

That's CPU Frequency. Turbo has kicked in.

Demand

Demand is similar to Usage. It differs to Usage as it assumes the VM does not share the physical core. It's what the VM utilization would be had it not experienced any issue, including hyper-threading.

Demand is not what Usage would be had the VM got everything it demanded. This metric is for the kernel resource scheduler. Ignore it.

Just to assure you, get a VM with high CPU Ready and Co-Stop that was intentionally caused. Set a limit. I did that, and here is what I got:

```
Co-Stop: 8.78%
Ready:   15.91%
Wait:    13.16%
Run:     62.15%
```

Total of the 4 = 100%.

This is expected. So far so good

But then here is what I got for Demand (%) and Usage (%):



Demand is identical to Usage, despite Contention (%) being near 0%.

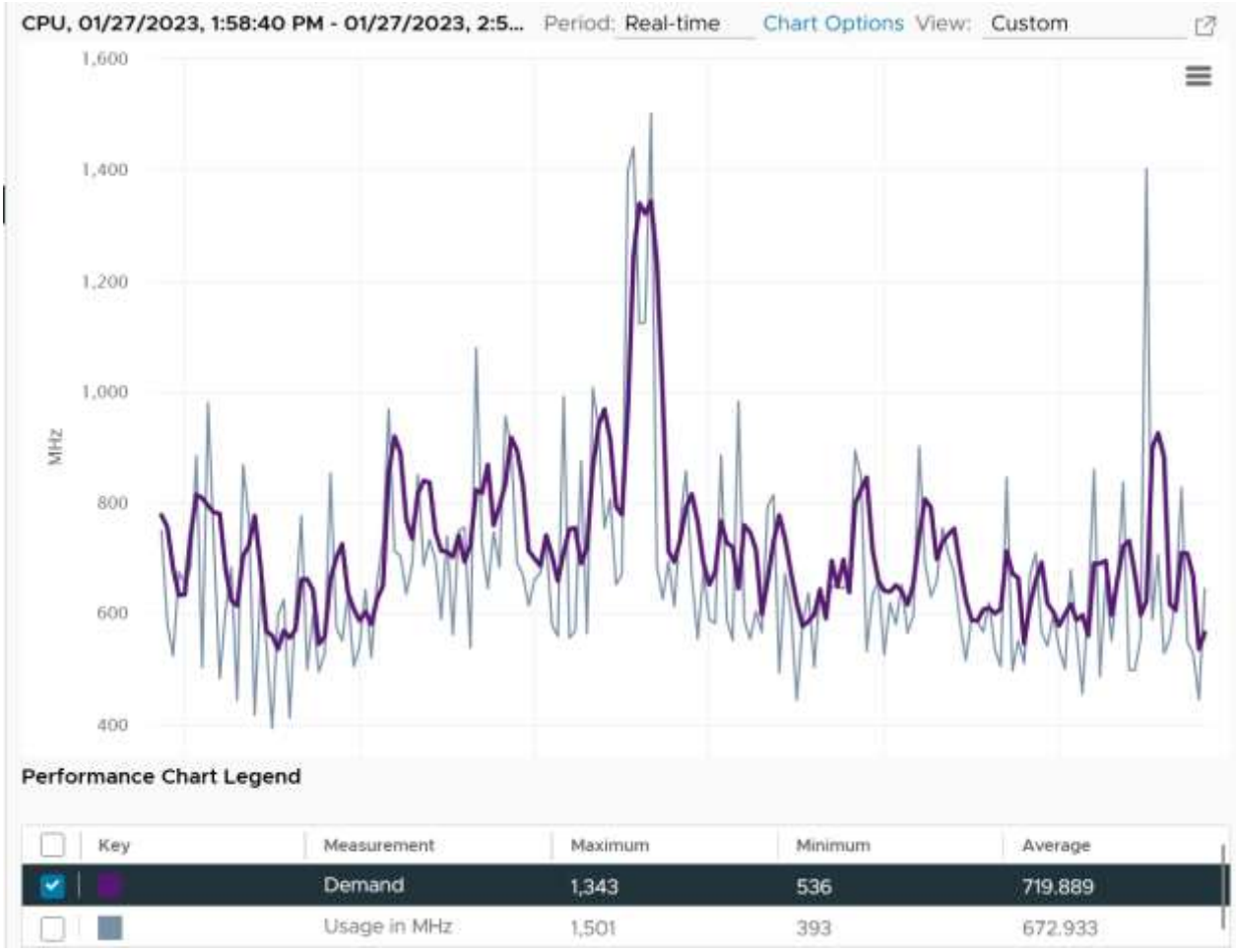
The reason is Demand is a kernel internal metric. From the kernel perspective, the VM did not demand as limit is intentionally set.

In the event the VM vCPU is running on a core where both threads are busy, the value of Usage will be 37.5% lower, reflecting the fact that the VM only gets 62.5% of the core. This makes sense as the HT throughput benefit is fixed at 1.25x.

If there is no competition for resource, Demand and Usage will be similar.

Take a look at the following screenshot from vCenter. It's comparing Demand (thick line) and Usage (grey line)

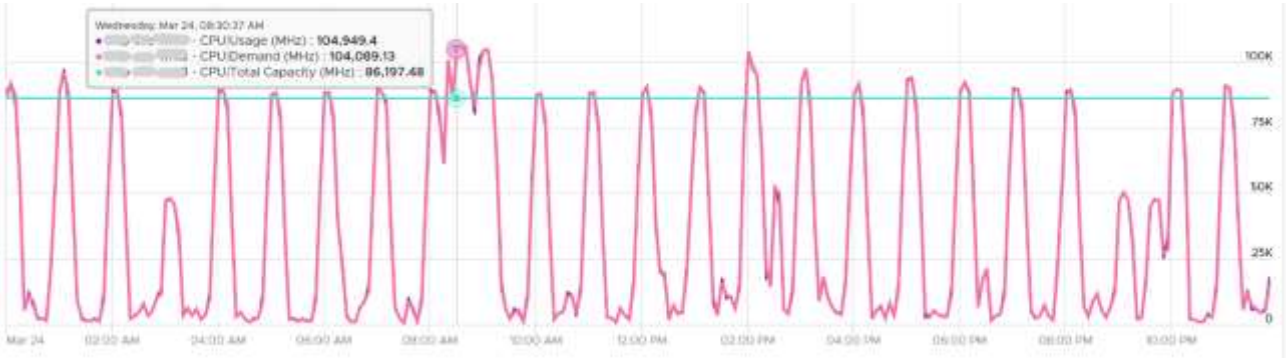
What do you notice?



How can Usage be higher than Demand at some of the point?

The reason is Demand is averaged over a longer time, giving it a steadier value. That's why the peak is shorter but wider. Notice the average over 1 hour is higher for Demand.

Due to Turbo Boost, Demand (MHz) and Usage (MHz) can exceed 100%. The following is a 32-vCPU Hadoop worker node. Notice it exceeds the total capacity multiple times, as total capacity is based on base clock speed. Demand and Usage are identical as it's the only VM running and the host has more than 32 cores, hence there is 0 issue.



Okay, now that you have some knowledge, let's test it 😊

Quiz Time! Looking at the chart below, what could be causing it?



Why did Demand jump while Usage dropped? VM CPU Contention (%) jumped even more. What is going on?

And why is that Contention is much more than Demand – Usage?

The reason why Demand metric jumps is the VM wanted more vCPU. That's the only explanation.

Usage drops because the additional demand could not be met.

The VM experiences contention, which includes hyperthreading sharing. I did not include CPU Ready, Co-stop, Overlap, VM Wait and Swap Wait, as they do not matter in this case.

From the chart you can see that the formula for VM CPU Contention > Demand – Usage. Contention (%) is around 20% when Demand is 25% and Usage is 15%. The reason is Contention accounts for both CPU frequency and hyper threading, while the difference between Demand and Usage is only hyper-threading.

VCF Operations provides a percentage metric for Demand. The formula is

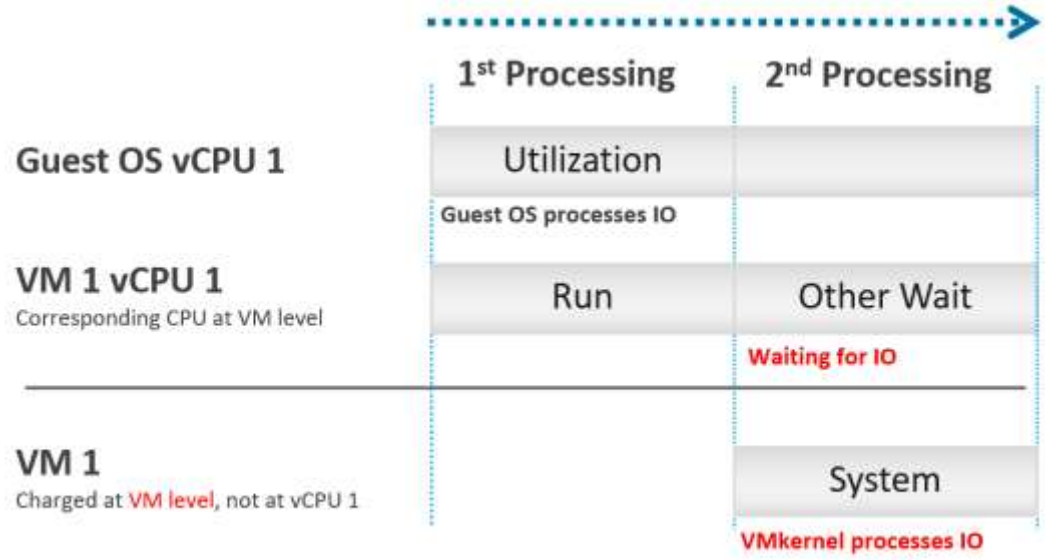
```
Demand (%) = Demand (MHz) / Total Capacity (MHz) x 100
```

Source wise, the metric in VCF Operations simply maps to vCenter counter cpu.demand.average.

System

A VM may execute a privilege instruction, or issue IO commands. These 2 activities are performed by the hypervisor, on behalf of the VM.

IO processing differs to non-IO processing as it has to be executed twice. It's first processed inside the Guest OS, and then in the hypervisor storage subsystems, because each OS has their own storage subsystem. For ESXi, its network stack also has to do processing if it's a IP-based storage.



IO is processed 2x. First by the Guest OS, then by the Hypervisor

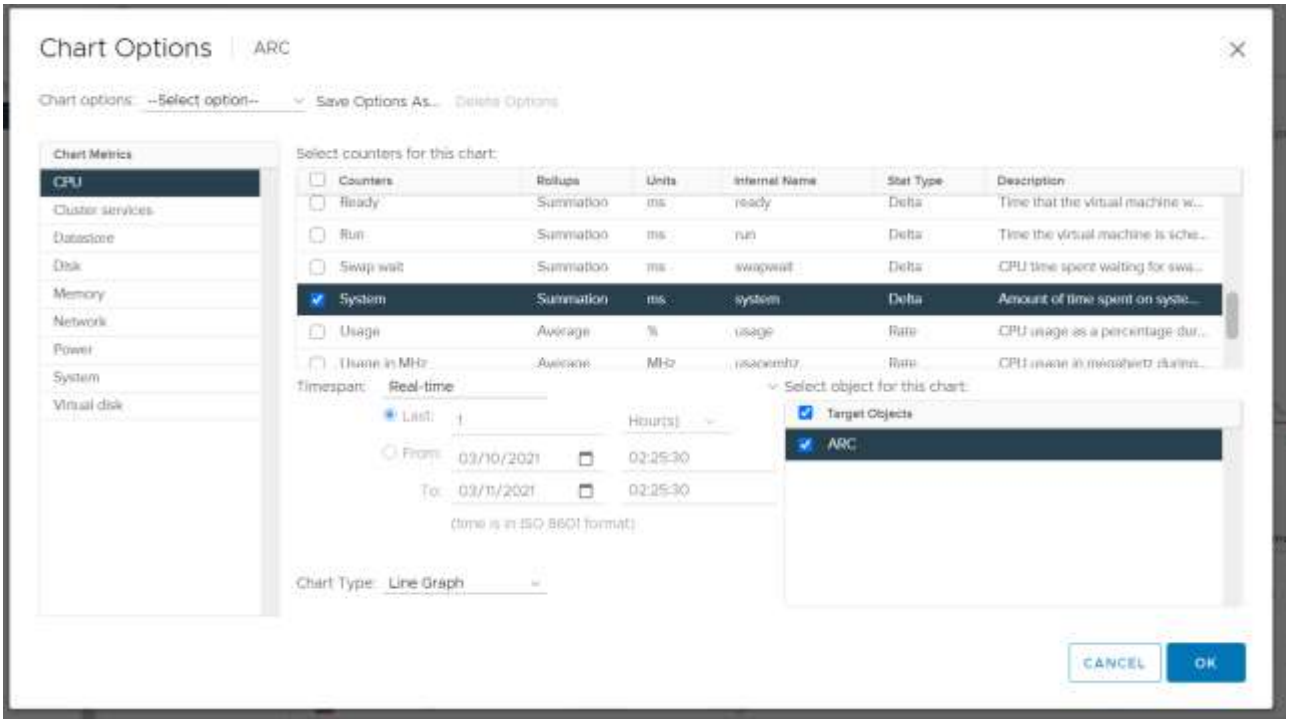
ESXi typically uses another core for this work instead of the VM vCPU, and put that that VM vCPU in wait state. This work has to be accounted for and then charged back to the associated VM. The System counter tracks this. System counter is part of VMX world of the VM.

Guest OS isn't aware of the 2nd processing. It thinks the disk is slower as it has to wait longer.

If there is snapshot, then the kernel has to do even more work as it has to traverse the snapshot.

The work has to be charged back to the VM since CPU Run does not account for it. Since this work is not performed by any of the VM CPU, this is charged to the VM CPU 0. The system services are accounted to CPU 0. You may see higher Used on CPU 0 than others, although the CPU Run are balanced for all the VCPUs. So this is not a problem for CPU scheduling. It's just the way the kernel does the CPU accounting.

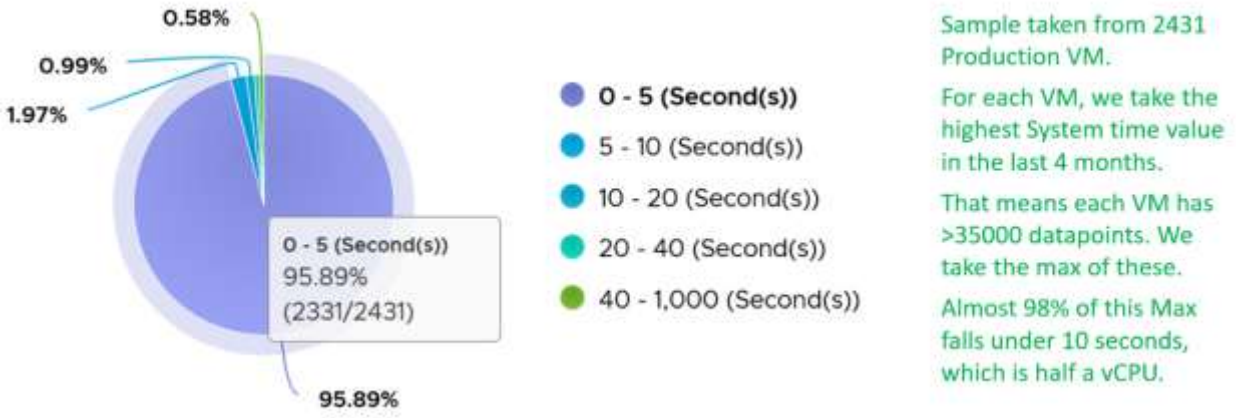
The System counter is not available per vCPU. Reason is the underlying physical core that does the IO work on behalf of the VM may be doing it for more than 1 vCPU. There is no way to break it down for each vCPU. The following vCenter screenshot shows the individual vCPU is not shown when System metric is selected.



ESXi is also performing IOs on behalf of all VMs that are issuing IOs on that same time, not just VM 1. The kernel may serialize multiple random IO into sequential for higher efficiency.

Note that I wrote to CPU accounting, not Storage accounting. For example, vSphere 6.5 no longer charges the Storage vMotion effort to the VM being vMotion-ed.

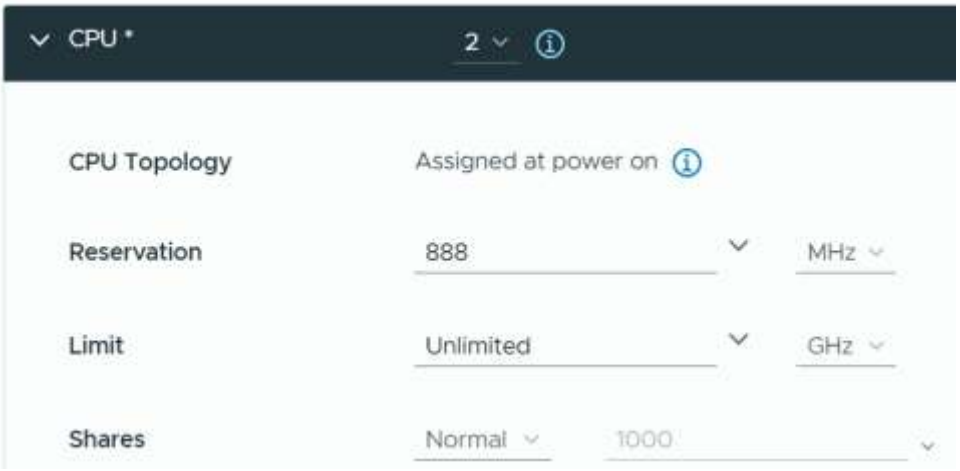
Majority of VMs will have System value less than 0.5 vCPU most of the time. The following is the result from 2431 VMs.



On IO intensive VM like NSX Edge, the System time will be noticeable, as reported by this KB article. In this case, adding more vCPU will make performance worse. The counter inside Linux will differ to the counter in vSphere. The following table shows high system time.

Provisioned	CPU|System (ms) ↓
32 vCPUs	74,124.27 ms
32 vCPUs	68,004.8 ms
4 vCPUs	39,645.13 ms
4 vCPUs	35,389.27 ms
4 vCPUs	33,230.67 ms
2 vCPUs	27,066.8 ms

Reservation



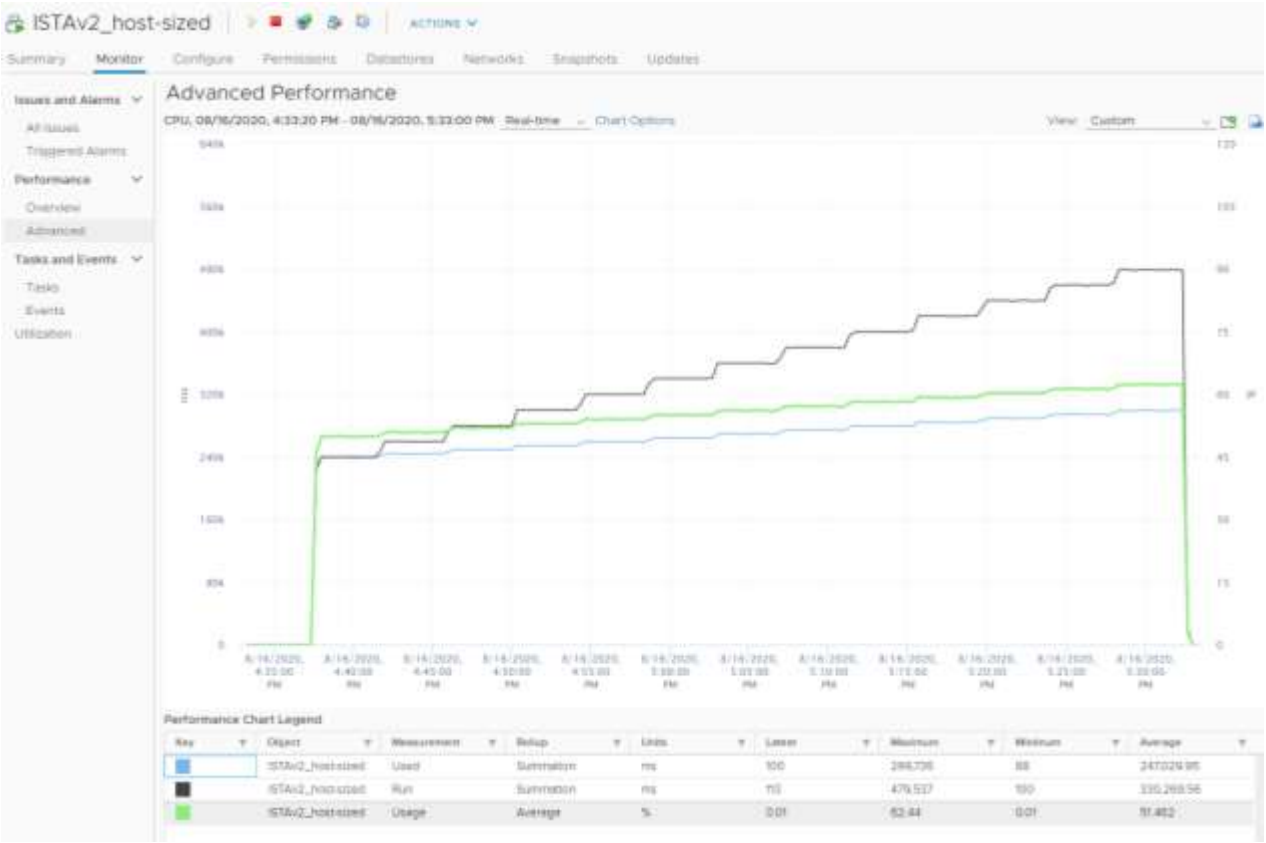
The number is only available in MHz or GHz, not in vCPU. That means when you move the VM to an ESXi of different frequency, you need to adjust the number manually.

Quiz!

By now I hope you vrealize that the various "utilization" metrics in the 4 key objects (Guest OS, VM, ESXi and Cluster) varies. Each has their own unique behaviour. Because of this, you are right to assume that they do not map nicely across the stack. Let's test your knowledge 😊

VM vs ESXi

Review the following chart carefully. Zoom in if necessary.



The vCenter chart[11] above shows a VM utilization metrics from a single VM. The VM is a large VM with 24 vCPUs running controlled CPU test. The power management is fixed so it runs at nominal clock speed. This eliminates CPU frequency scaling factor.

The ESXi only has 12 cores. Hyper threading is enabled, so it has 24 threads.

The VM starts at 50% "utilization", with each vCPU pinned to a different physical core. It then slowly ramps up over time until it reaches 100%.

Why do the three metrics moved up differently? What do they measure?

Now let's look at the impact on the parent ESXi. It only has a single VM, but the VM vCPU matches the ESXi physical cores. The ESXi starts at 50% "consumption", then slowly ramp up over time until it reached 100%.

[11] Provided by Valentin Bondzio



Why does CPU Usage moves in tandem with Core Utilization, but Utilization moved up differently? What do they measure?

Let's break it down…

At the start of the test

The VM runs 12 vCPU, but each vCPU was pinned to each ESXi core. So all cores are 100% utilized, but each running 1 thread.

VM CPU Run (ms) is 240K milliseconds, which is 20K milliseconds x 12 (half of its 24 vCPU).

VM CPU Used (ms) is also at 240K milliseconds. There is no loss from overlap, the VM does not do much IO, and no efficiency loss/gain due to HT.

VM CPU Usage is 50%.

So at this point, all 3 metrics of VM CPU are 50%.

The counter at ESXi tells a different story. The ESXi Core Utilization (%) immediately went up to 100% while Utilization went up to only 50%. The reason is Core Utilization measures whether the core is used or not. It takes a different perspective.

Usage (%) is identical to Core Utilization in this case, because CPU frequency is static.

On the other hand, ESXi Utilization (%) looks at if each thread HT is running or not. It does not care about the fact that the 2 threads share a core, and simply roll up to ESXi level directly from thread level. This is why it's showing 50% as it only cares whether a thread is running or not, at any point in time.

During Ramp Up period

VM is being ramped up steadily. You can see all 3 metrics went up in steps.

VM CPU Run (ms) ramps up from 240K to 480K. All 24 vCPU has 20K ms value, which equals to 100%.

VM CPU Used (ms) barely moved. From 240K to 300K. That's 1.25x, demonstrating that Used understands HT only delivers 1.25x throughput.

VM CPU Usage (%) ramp up from 50% to 62.5%, also demonstrating awareness of contention due to HT.

Used (ms) = Usage (%)

ESXi CPU Usage (%) counter stayed flat at 100%. The reason is all 12 cores were already busy and the metric is capped at 100%. That means VM CPU Usage (%) is aware of HT, but ESXi CPU Usage (%) is not.

ESXi CPU Core Utilization (%) matches VM Run. Both went 2x.

Towards the end of the run

VM CPU Run is at 480K ms. This counter is suitable for VM Capacity sizing, as it correctly accounts that each vCPU is used by Guest OS.

VM CPU Used is at 300K milliseconds, which is 62.5%.

VM CPU Usage (%) is at 62.5%. On average, each of the VM vCPU only gets 62.5%. If you use this for your VM capacity, you will get the wrong conclusion as it's already running 100%

ESXi CPU Usage (%) is at 100%. This makes it suitable from Capacity viewpoint, albeit too conservative. It is not suitable from Performance, as you cannot tell if there is still room.

ESXi CPU Utilization (%) is at 100%. Because it tracks the ramp correctly, it can be used from Performance. You can use it for capacity, but take note that 100% means you get performance hit from. In fact, at 50% the HT effect will kick in.

When Run is lower than Used

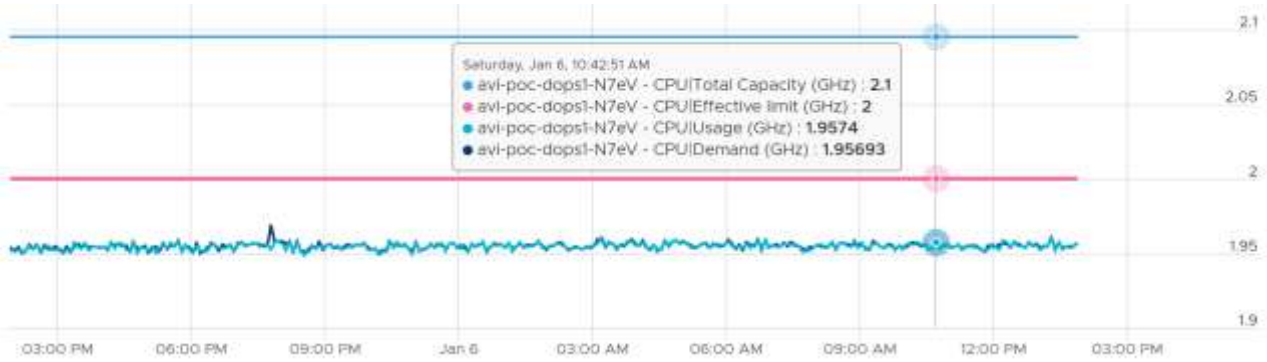
Take a look at this VM. It's a single vCPU, so co-stop is not applicable.

The ESXi clock speed is 2.095 GHz, so the VM has a capacity of 2.095 GHz. A limit is imposed at 2 GHz.

What's your conclusion on the Demand and Usage counters?

They are basically identical, hovering around 1.96 GHz. If you use the 2.095 GHz as the 100%, basically both counters are flat around 94%.

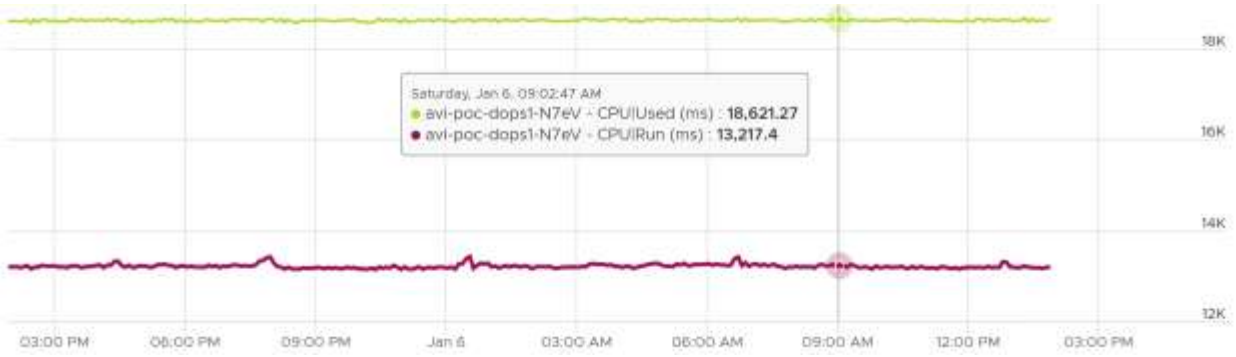
Why are they similar? Shouldn't Demand be higher?



Now let's see the Used (ms) metric.

It's also flat, around 18600 ms. Using the 20000 as the 100%, it's around 93%. This is fairly expected, as I think Usage (MHz) contains non vCPU worlds (e.g. MKS)

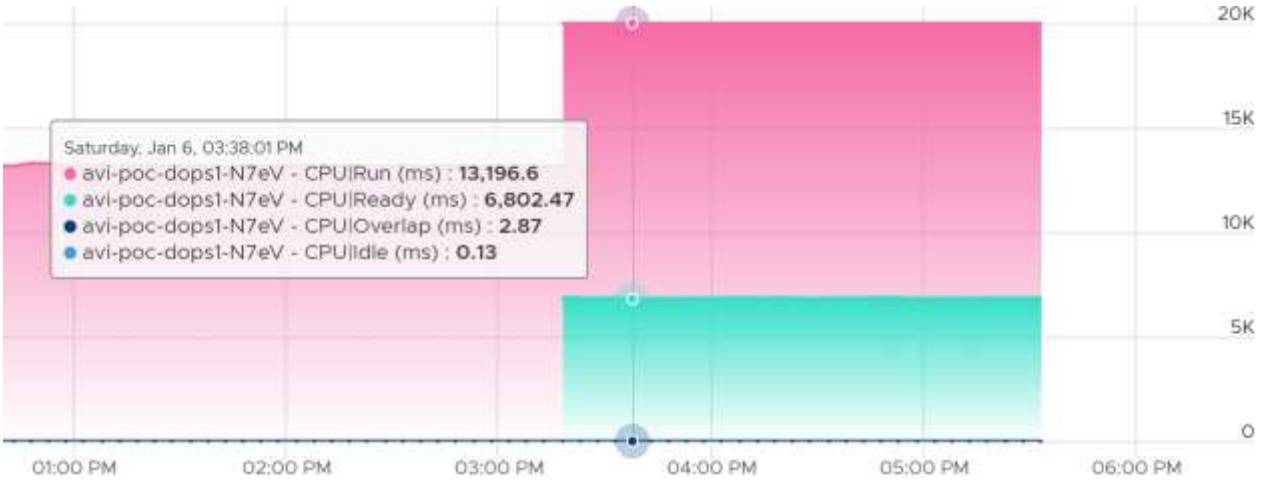
Now let's plot Run (ms) alongside. Why is it much lower?



Why is Used consistently 41% higher than Run?

Answer is CPU Ready.

If you sum up Run + Ready + Overlap + Idle, it will be 20000.



When Run is higher than Used

Now let's look at the opposite scenario.

This VM is a 64-bit Ubuntu running 4 vCPU. Used (ms) is around 44% of Run (ms). The VM had minimal System Time (ms) and Overlap (ms), so Used is basically lowered by both power savings and CPU SMT.



In this example, if Run is far from 100% and the application team want faster performance, your answer is not to add vCPU. You should check the power management and CPU SMT, assuming the contention metrics are low.

ESXi

Throughout this book, I always cover the contention metrics first, then consumption. Why is it that I swap the order for ESXi Host?

Because in the provider layer there is no contention. The one that faces contention is the consumer (VM).

Consumption Metrics

vSphere Client UI provides 6 counters to track ESXi CPU "consumption" and 1 to track reservation.

Counters	Rollups	Units	Stat Type	Description
Core Utilization	Average	%	Rate	CPU utilization of the corresponding core (if hyper-threading is enable
Demand	Average	MHz	Absolute	The amount of CPU resources a virtual machine would use if there we
Usage	Average	%	Rate	CPU usage as a percentage during the interval
Usage in MHz	Average	MHz	Rate	CPU usage in megahertz during the interval
Used	Summation	ms	Delta	Total CPU usage
Utilization	Average	%	Rate	CPU utilization as a percentage during the interval (CPU usage and CI

Why 6?

Let's dive into the utilization metrics with a quiz.

Quiz: 50% or 75% or 100%?

Hope you like the tour of VM CPU accounting. Now apply that knowledge into ESXi and explain the following:



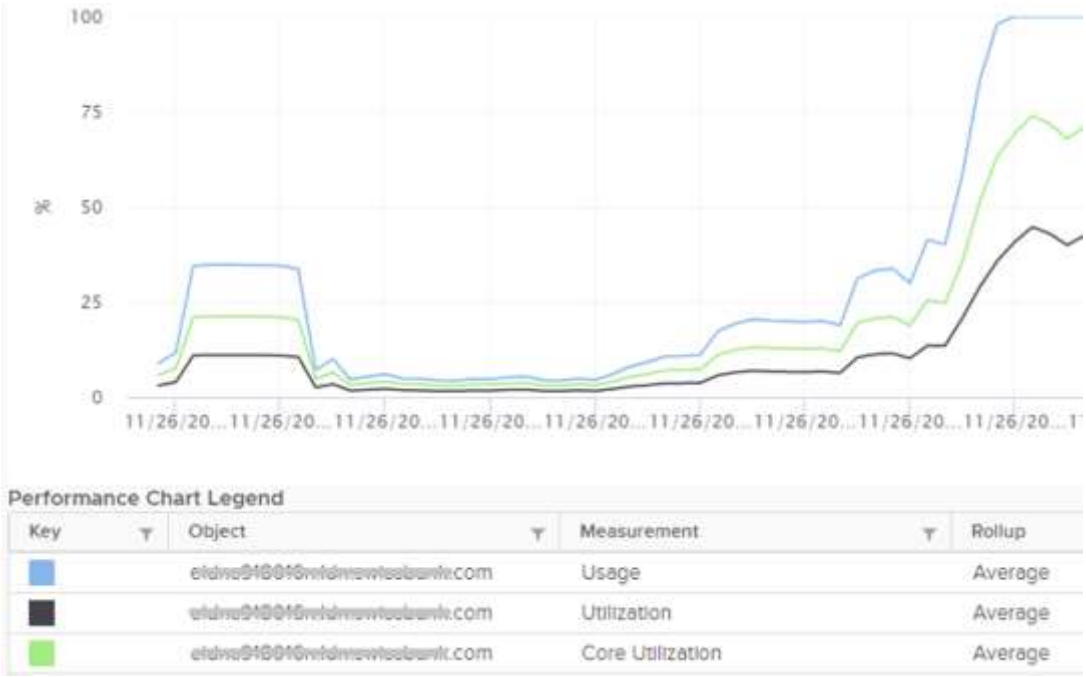
The above is an ESXi host, showing 3 types of utilization metrics.

- One shows ~50%, indicating you have capacity.
- The second one shows 100%, indicating you do not have capacity. BTW, this was the old version, where the metric was capped at 100%.
- The 3rd shows ~70%.

Which metrics do you take for the ESXi CPU "consumption" then?

They also do not move in tandem. Towards the end, both the black line and green line fluctuate, but the blue one was flat at 100%. Why?

Since the graph is a bit small, let's zoom in:



Notice they have similar pattern, but their *sensitivity* differs.

- Why is Usage (%) = 100% when Utilization (%) is around 47%? The gap is more than double. What could be causing it?
- Why is Utilization (%) fluctuating yet Usage (%) remains constant? Notice both Utilization varies between 45% and 55% while Usage remains flat at 100%
- Why is Core Utilization (%) in the "middle"? What does it actually measure then?

High Load Example

It's easier to think the metric Utilization (%) as Thread Utilization (%), so I'm going to refer to it as such.

You're welcome.

The preceding example shows Thread Utilization (%) mostly below 50%. Let's pick the opposite example, where at least half the threads are utilized.

What do you expect the value of Usage (%) and Core Utilization (%)?



Notice the Utilization (%) metric went up from ~80% to 90%.

At around 80%, not all cores are running all the time. This means a few cores were idle, despite most cores running both threads. Yes, that indicates imbalance.

As Utilization (%) reach 90%, the Core Utilization got closer to 100%, which is the absolute limit.

Usage (%) was constantly above 100%, as it's compared with the base frequency. This indicates the CPU was constantly running above the nomimal speed. Notice the pattern of Usage is not as volatile as Utilization (%). The reason is hyperthreading only gains 1.25x.

Core vs Thread

If Core Utilization (%) is < 2x of Thread Utilization (%), you have imbalanced. Some cores are running 2 threads, while others are idle.

HT and power management is done at core level, not thread. This creates complexity in rolling up the counters at thread level to core level.

Let's start. The basic unit is time, expressed in 20000 ms.

20000 ms = 100% at the core level. This means the value remains 20000 if you disable HT.

Since there are 2 threads, what happens when both are idle?

You assigned Idle = 10000 ms for each.

When one of the threads is running hot, the highest the Used value of the thread can go is 20000. This means the CPU frequency doubles. What happens to the paired thread then? Does it show 0 ms? I am yet to test here.

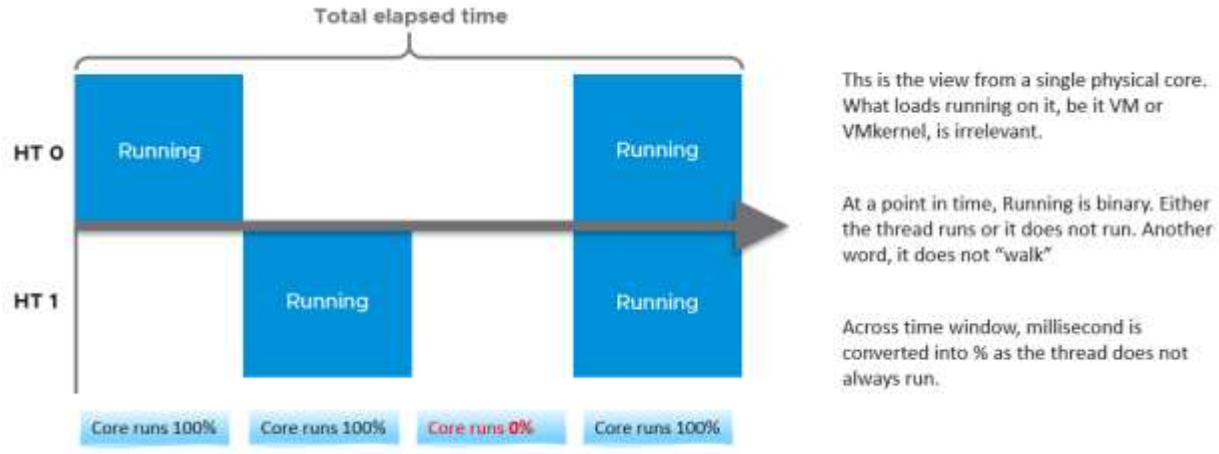
At thread level, 100% Used = 20000 ms, while 100% Idle = 10000 ms. This creates confusion when you work at thread level.

Thread Utilization

At a thread level	This is the most basic counter. It's the ESXi equivalent of VM CPU Run. It tracks at a single physical thread level. At any given moment, a thread is either running (unhalted) or not (halted). So it's binary (0% or 100%). Using a human analogy, think of it as a person who is either running or standing, and never walking. It's not considering CPU Frequency nor HT. Over a time period, the value is averaged. So when you see the number as 50%, it does not mean it's running 100% at half the "speed". It means it's running half the time, which is why the original counter is in millisecond and not percentage. This metric is only relevant when hyper-threading is enabled.
At a core level	Core Utilization, as the name implies, rolls up at the core level. It is a simple metric. If one of the threads is running, then this metrics reports 100%. This is logical as the core is indeed running. If both are running, this metrics also reports 100%. I agree that reporting the same number on 2 different scenarios will certainly cause confusion.

Let's apply the above into an example. We start with a single core of a physical socket. The socket may have many cores; we are just interested on 1 core only. The core has 2 threads as it supports CPU SMT.

In a time period of say 20 seconds[12], this core had the following consumption:



[12] I use 20 second as it's a familiar number. That's what you see in the real time chart in vCenter client, and 20000 ms is often used as the 100% when converting millisecond unit to percentage.

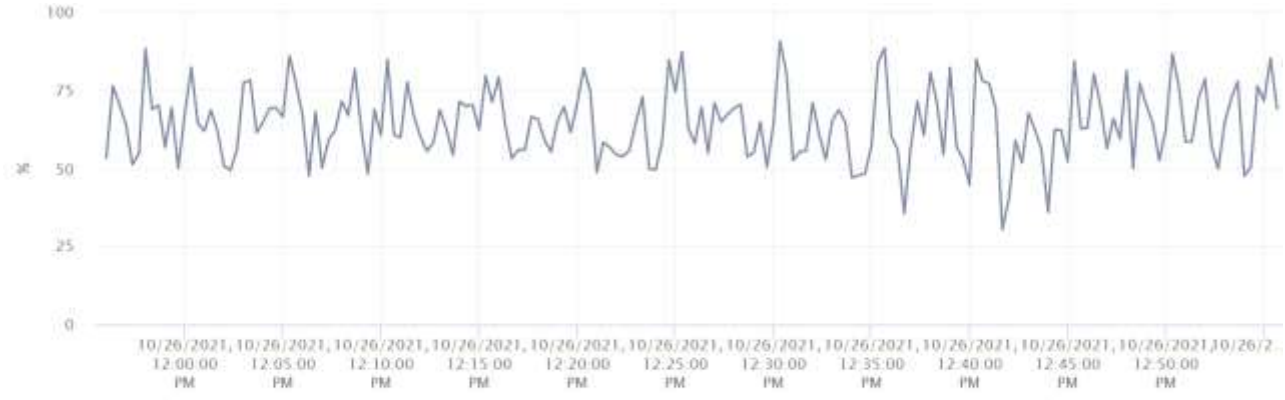
Going back to our example, here are metrics reported:

- Thread Utilization (%) for HT 0 = 10 seconds / 20 seconds = 50%
- Thread Utilization (%) for HT 1 = 10 seconds / 20 seconds = 50%
- Core Utilization (%) for entire core = 15 seconds / 20 seconds = 75%

BTW, in vSphere Client performance tab, you can't select a core if you enable HT. You can only choose PCPU, which is a thread. So what happens on the Core Utilization counter at thread level?

Does it get split into half?

As you can see below, no. The value is duplicated.

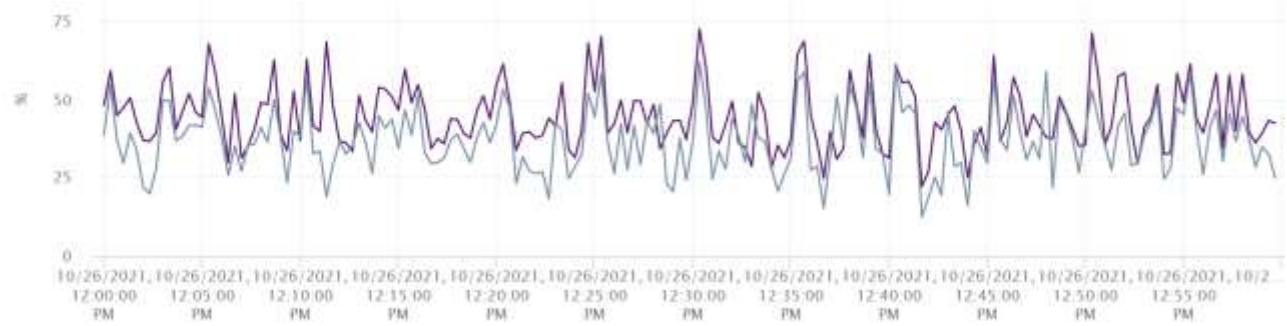


erformance Chart Legend

Key	Object	Measurement	Rollup	Units	Latest	Maximum	Minimum	Average
	0	Core Utilization	Average	%	68.96	90.67	30.47	64.279
	1	Core Utilization	Average	%	68.96	90.67	30.47	64.279

Notice in the above chart, the 2 have identical value.

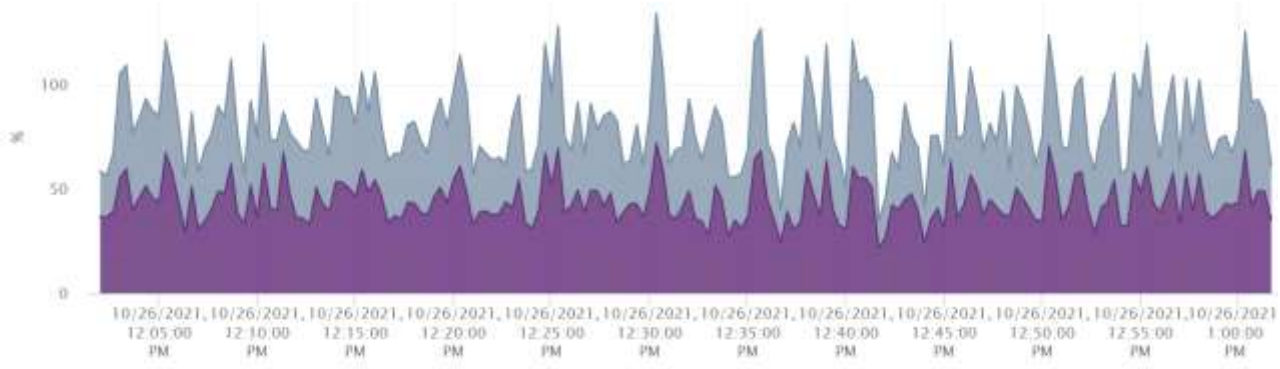
Thread Utilization (%), on the other hand, will be different. Each thread has different value.



erformance Chart Legend

Key	Object	Measurement	Rollup	Units	Latest	Maximum	Minimum	Average
	0	Utilization	Average	%	42.34	72.48	22.15	44.762
	1	Utilization	Average	%	24.93	61.97	12.4	36.857

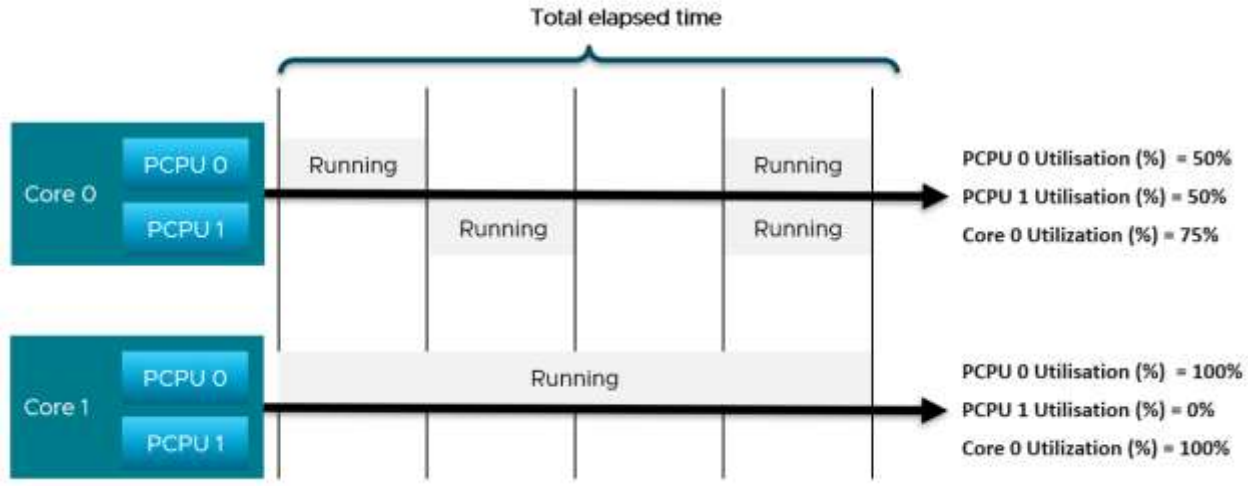
If you simply sum them up, you get more than 100%, so don't! Their context is a single thread.



erformance Chart Legend

Key	Object	Measurement	Rollup	Units	Latest	Maximum	Minimum	Average
	0	Utilization	Average	%	35.77	72.48	22.15	44.749
	1	Utilization	Average	%	25.46	61.97	12.4	36.937

Now let's roll this up to the ESXi level. The following shows a tiny ESXi with 2 cores, where each core has 2 threads.

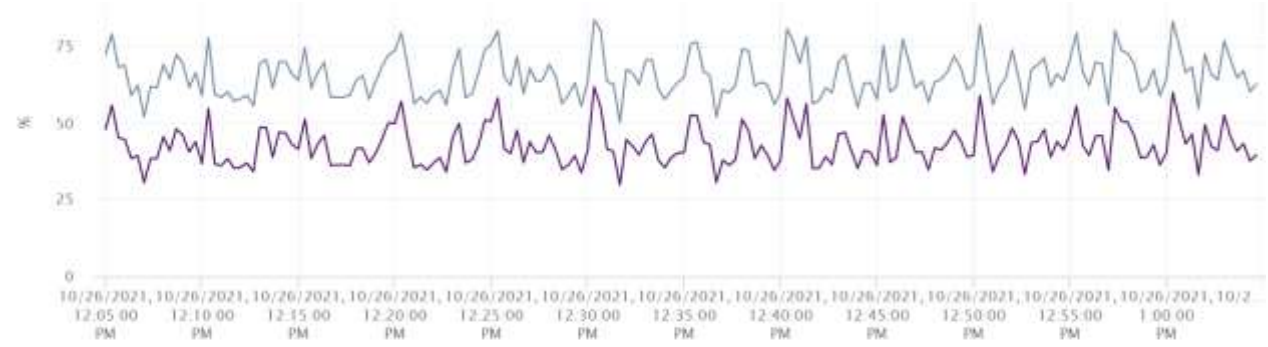


The metrics at ESXi level is

- CPU Thread Utilization (%) = (50% + 50% + 100% + 0%) / 4 = 50%.
- CPU Core Utilization (%) = (75% + 100%) / 2 = 87.5%

Thread Utilization = 50% because each thread is counted independently. There are 4 threads in the preceding ESXi, each runs 50%, so the average at ESXi level is 50%. This counter basically disregards that HT does not deliver 2x the throughput.

This is why the Core Utilization (%) will tend to be consistently higher than Thread Utilization (%). The following chart demonstrate that.



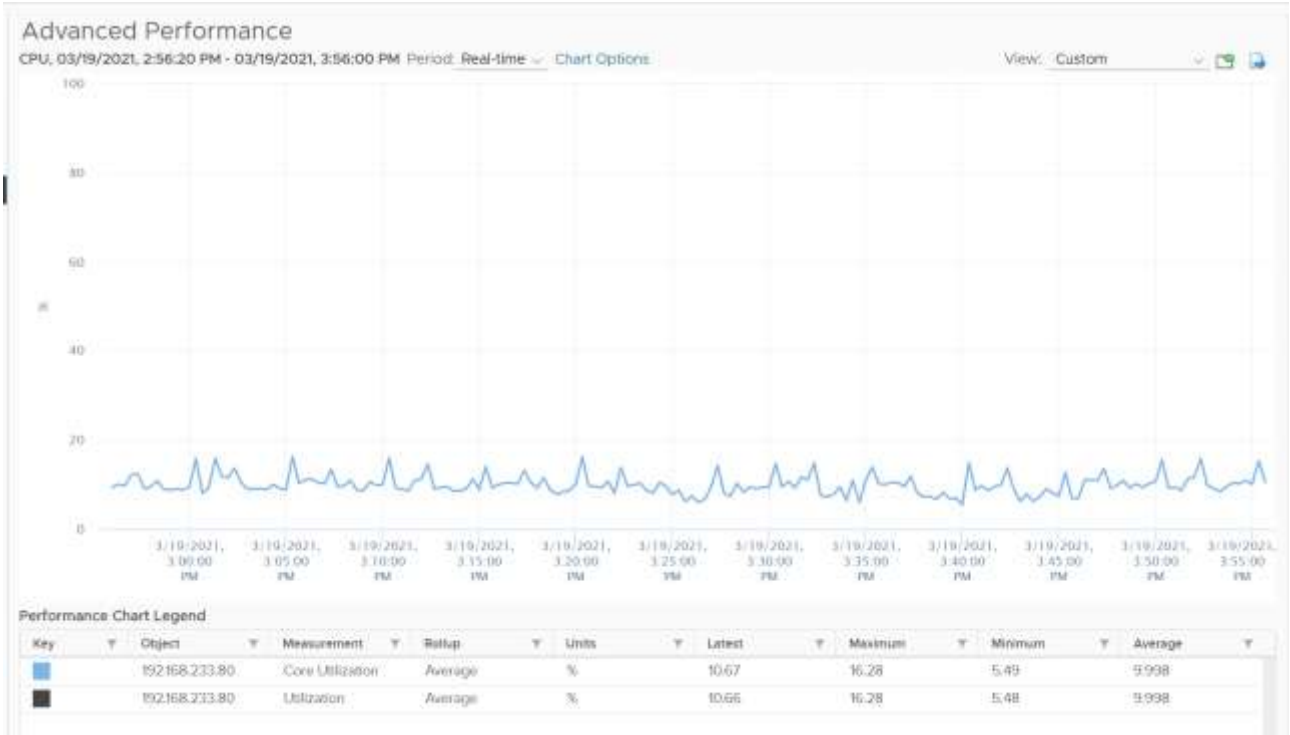
Performance Chart Legend

Key	Object	Measurement	Rollup	Units	Latest	Maximum	Minimum	Average
	192.168.233.149	Utilization	Average	%	39.43	61.61	29.74	42.525
	192.168.233.149	Core Utilization	Average	%	62.57	83.43	50.06	65.37

Now let's go back to the chart shown earlier. Can you now explain Thread Utilization (%) and Core Utilization (%)?

Great! Let's move to the next one.

In the following example, this ESXi has no hyper-threading. What do you notice?



Yup, the Core Utilization is identical with Thread Utilization.

Peak Core CPU Usage

An ESXi with 72 CPU cores will have 144 threads. You will not be able to see when a single core peak at ESXi Host level as it's the average of 144 metrics. If you are concerned that any of them is running hot, you need to track the peak among them.

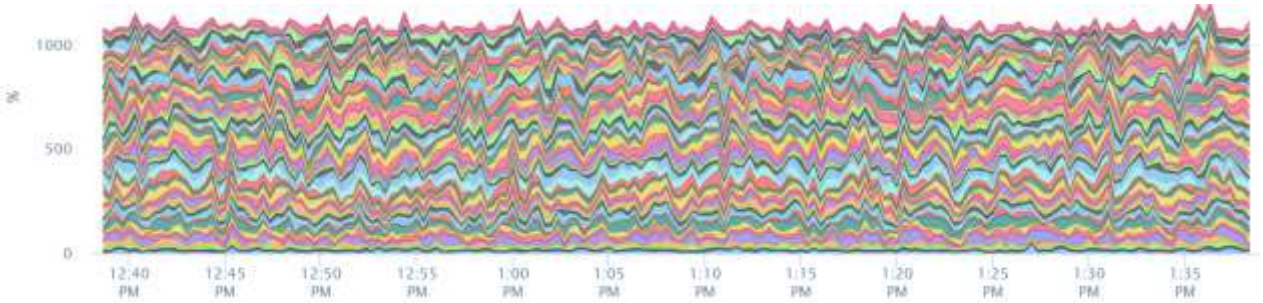
Imbalance among the cores happen because when a VM runs, it runs on as few core as possible, not spread out to all ESXi cores. It's more efficient to schedule that way, as will requires less context switches.

The following shows an ESXi where 3 of the threads hit >90%

Performance Chart Legend

Key	Object	Measurement	Rollup	Units	Latest	Maximum	Minimum	Average
	38	Utilization	Average	%	7.02	99.96	0.44	22.468
	43	Utilization	Average	%	31.06	97.11	0.14	19.26
	50	Utilization	Average	%	10.51	91.03	0.14	9.57
	37	Utilization	Average	%	29.48	82.31	0.66	27.999
	31	Utilization	Average	%	14.43	77.1	1.15	29.518
	46	Utilization	Average	%	55.53	76.98	0.33	21.858

The average, however, is low. The total usage can hit 5600% as there are 56 threads, hence the total is only hovering 1100%, which translates into 20%.



Peak CPU Core Usage (%) tracks the highest CPU Usage among the CPU cores. A constantly high number indicates that one or more of the physical cores has high utilization. So long the highest among any cores at any given time is low, it does not matter which one at a specific point in time. They can take turn to be hot, it does not change the conclusion of troubleshooting. Max() is used instead of 95thpercentile as both result in the same remediation action, and Max() can give better early warning.

The imbalance value among the cores is not needed because it is expected when utilization is not high.

Used | Usage

You are now ready to tackle the next metrics, which are Used (ms), Used (%), Usage (%) and Usage (MHz). Used (%) is used in esxtop, while vSphere Client UI uses the other 3 metrics.

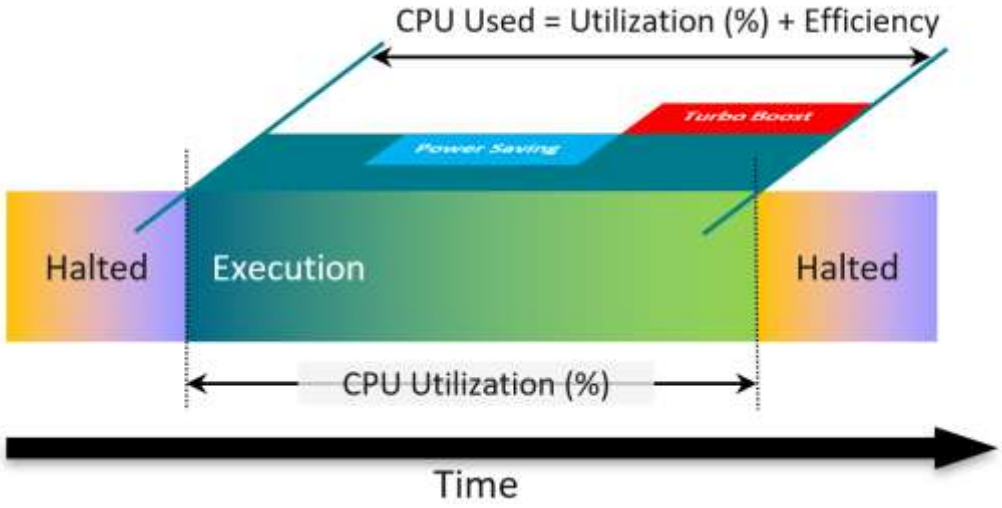
vSphere Client uses the name Used instead of Usage. But in the metrics chart page, it uses Usage. As a result, I'm going to assume that Used (MHz) = Usage (MHz) as vSphere Client UI uses them interchangeably. While I've found differences in rare cases, it is safe to take as Usage = Used.

I only see the use case for Usage (MHz). The other 2 units (second and percentage) are not needed. Using millisecond is hard to account for "how fast you run" and "how efficient you run". With MHz, we gain an additional dimension, which is speed. We can plot the speed across time.

If you have a specific need on Used (ms), Used (%) and Usage (%), reach out to me with the reason and happy to provide the documentation on these 3 metrics.

Usage (MHz) relates to Utilization in a similar way that VM Usage metric relates to VM Run metric. The difference is a physical thread does not experience overlap, and system is not applicable as it's run in the kernel.

Here is how Utilization and Usage are related at PCPU level:



Used is calculated based on a hardware counter called Non-Halted Core Cycle (NHCC). Logically, the higher the CPU clock speed, the more cycles you complete. That's why the value gets higher in turbo boost.

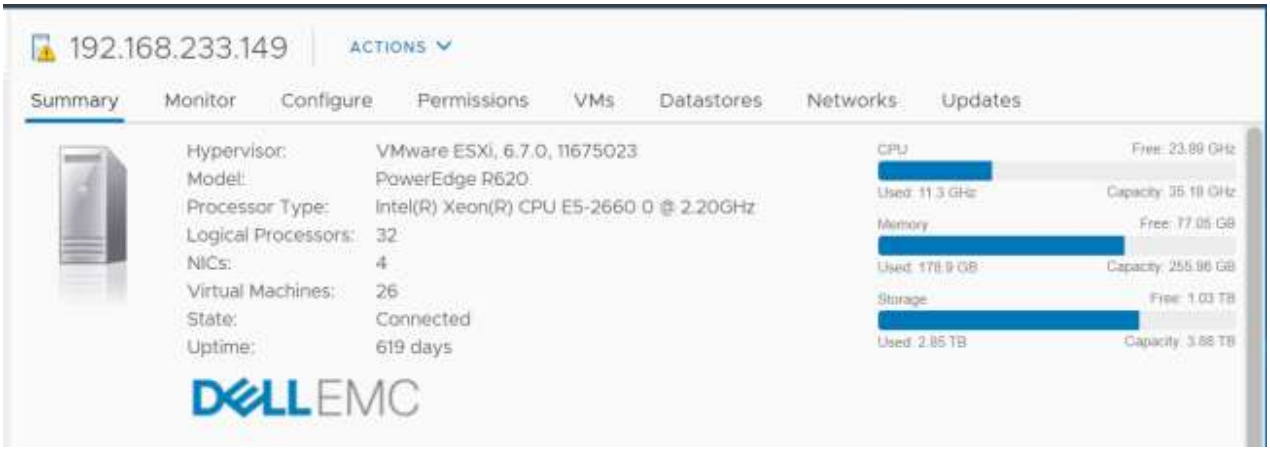
From the diagram above, you can see that Usage accounts for 2 factors that Utilization does not:

- A physical thread is either executing (running) or halted (idle). Its execution will be less efficient if its paired thread is also running at the same time.
- While it's running, it can run at lower/higher CPU clock speed due to power management.

Usage

vCenter adds this counter, meaning it does not exist at ESXi level.

You see both the Capacity of 35.18 GHz and Used of 11.3 GHz. There is no concept of Usable Capacity in vSphere, so the Free amount is basically Capacity – Used.



The Used CPU is summary.quickStats.overallCpuUsage.

The value above is likely some average of say 5 minutes as it remains static for a while, and it does not exactly match the number below as the roll up period is not the same.

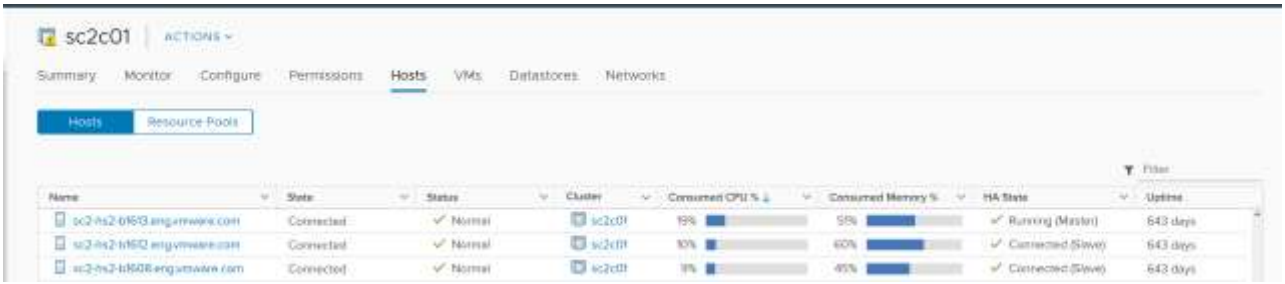
Usage and 100%

Usage no longer capped at 100% of total capacity at the nominal frequency. This was fixed in Aria Operations 8.18. In a highly utilized host, you will see Usage exceed it. This is indeed a desirable situation as you bought the host to be used as much as possible.



Consumed

When vSphere UI lists ESXi Hosts, it typically includes the present utilization. It lists the metrics as Consumed CPU (%) and Consumed Memory (%).

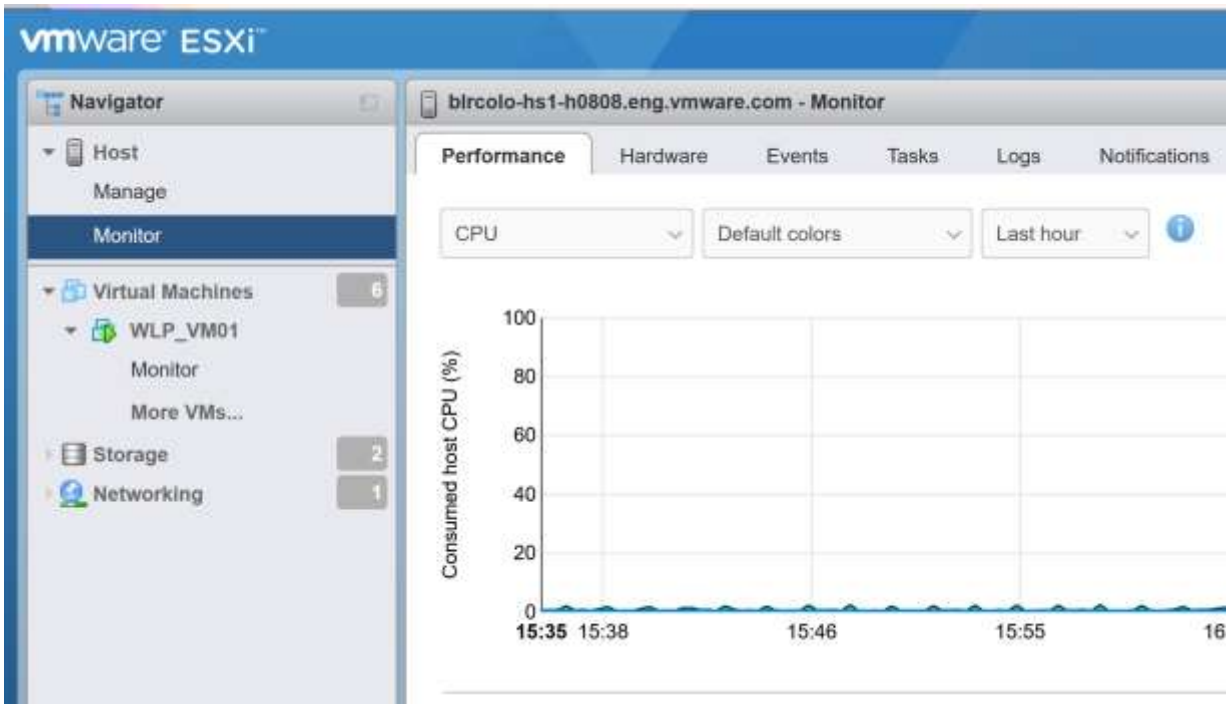


Consumed CPU maps to CPU Usage (%). Consumed Memory (%) maps to Memory Consumed (KB).

To confirm it, simply plot CPU Usage value. The last value is what you see at the table.

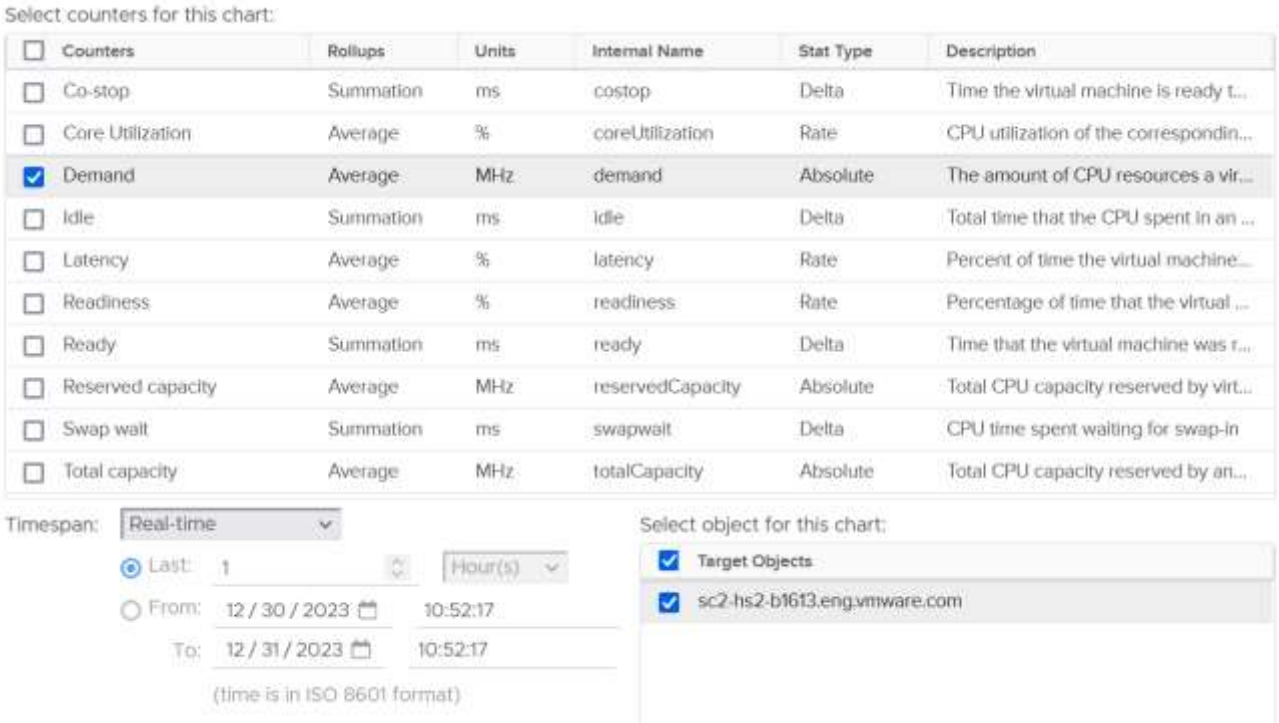


You also see them in vSphere Host Client.



As a bonus, you get the breakdown by CPU package.

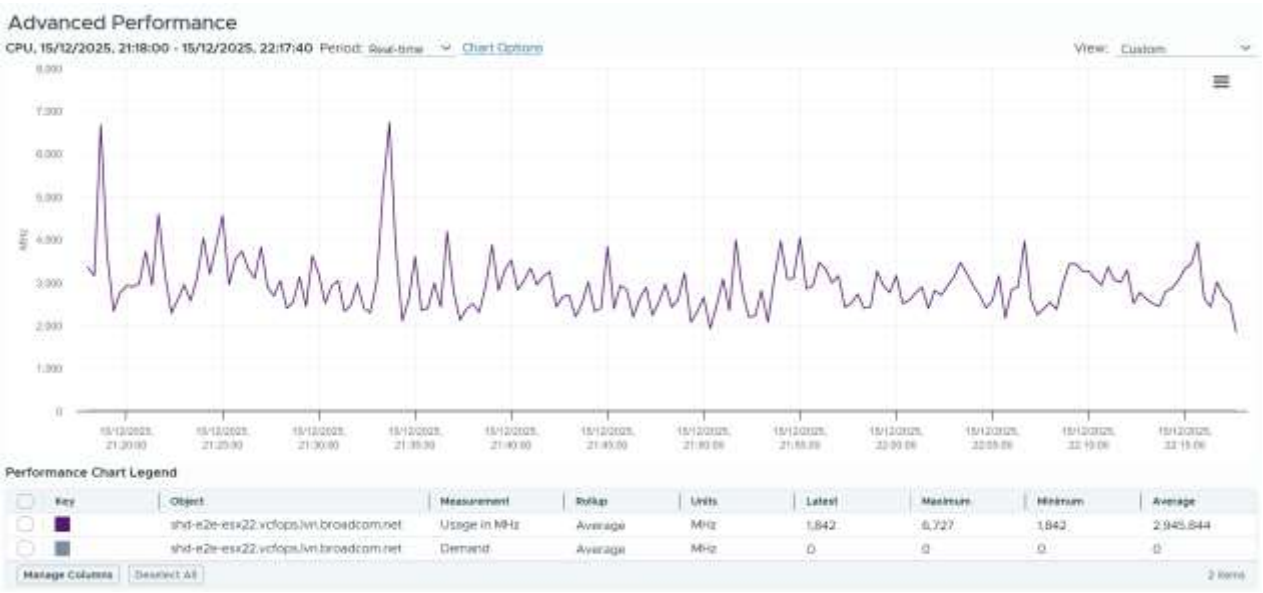
Demand



This is an internal counter. It's for the kernel CPU scheduler to optimize the running of VM as the kernel is aware that hyper-threading has performance impact. As a result, demand looks at different *context* than Utilization/Used. The value you see at ESXi is the summation of all the VMs, not physical threads.

Demand is consumer-view, while Usage is provider view. Now you know why Demand is not available on a per-core or thread basis.

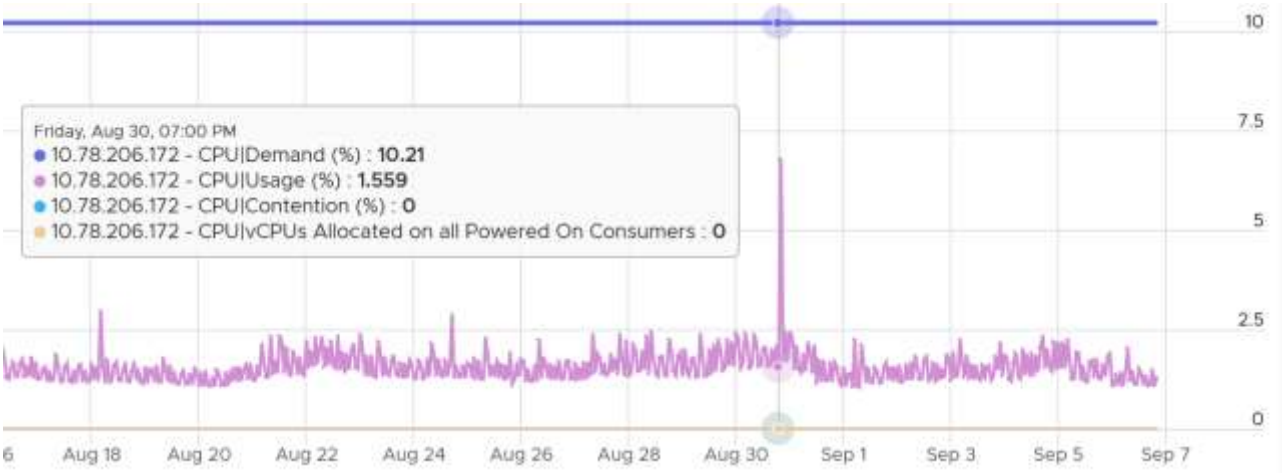
If a host has no VM, Demand will likely show flat 0. Usage will not show 0 as it includes the kernel. The following ESXi has no VM, but run kernel such as vSAN and NSX.



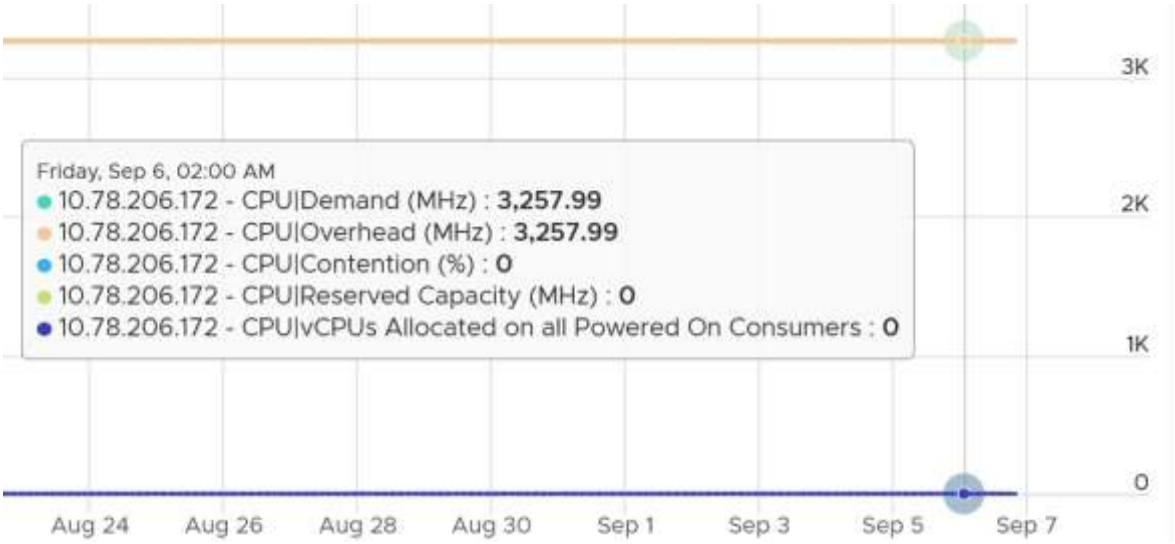
Demand includes the reservation by running VM or kernel.

Usage includes the actual utilization of running VM and kernel.

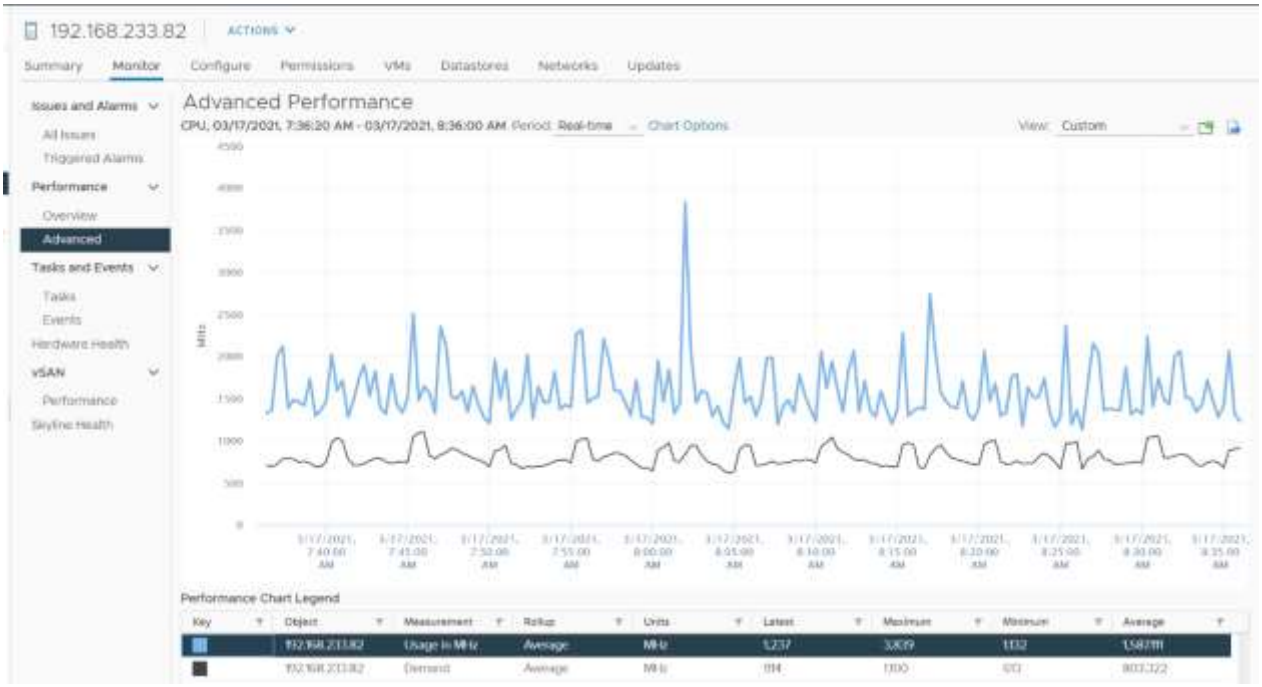
The following shows Demand remains perfectly flat at 10.21% for weeks. The ESXi has 0 running VM. The only explanation is the kernel reservation.



The following chart shows the same ESXi, where the Demand (MHz) = Overhead (MHz). Overhead tracks the kernel reservation.



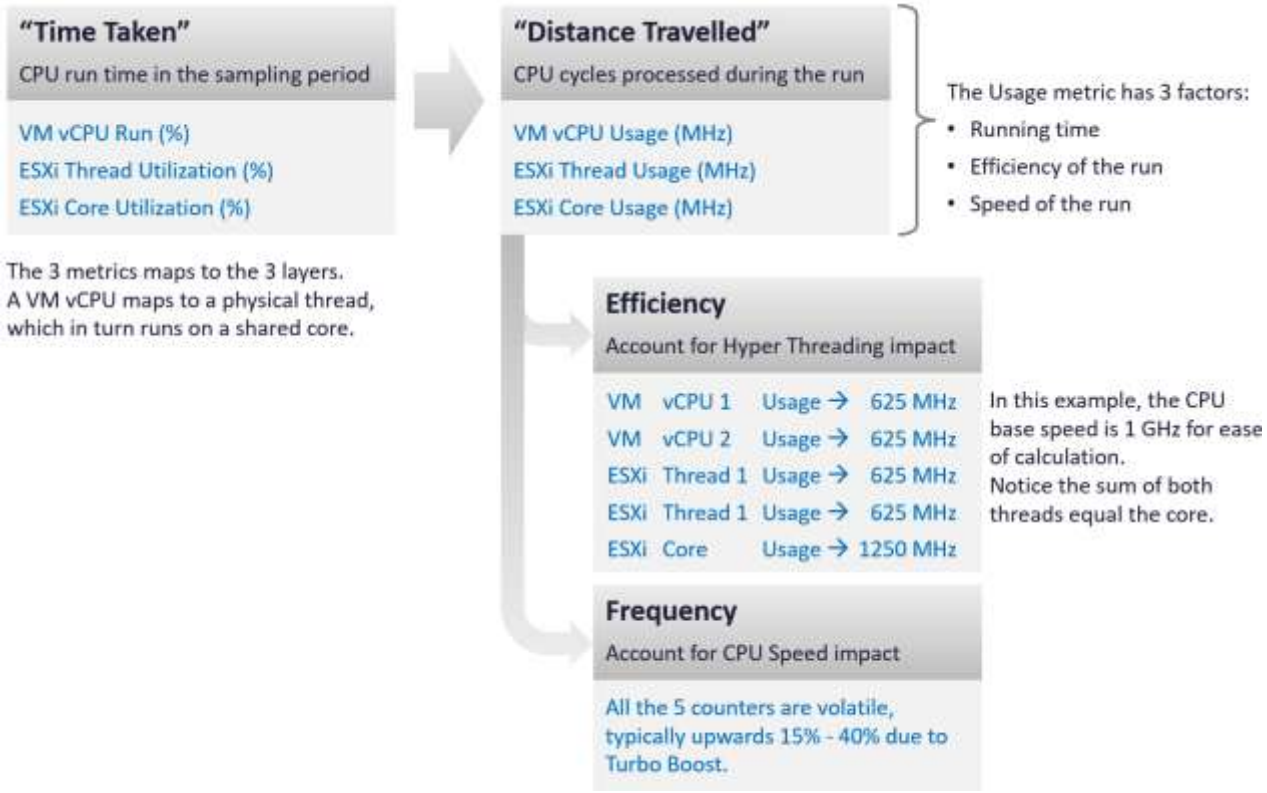
If the kernel reservation is lower than the kernel utilization, then Demand could be lower than Usage. The following example shows that.



Once there is enough utilization, defined as some VMs are sharing the same cores, Demand will be higher than Usage. The following screenshot shows Demand being consistently higher.



Summary



Comparison

Let's evaluate all the possible scenarios so you can compare the values returned by the metrics. We will use a simple ESXi with 2 cores. Each core has 2 threads. In each of the scenario, a thread is either running or not running. There is no partial run within a thread as that's mathematically covered in our scenarios.

I will also use 20000 ms as that's more familiar. The following table shows an ESXi with 2 cores. There are 6 possible permutations in their utilization.

Scenario	Utilization								Used							
	Core 1		Core 2		Core 1		Core 2		Core 1		Core 2		Core 1		Core 2	
	HT 1	HT 2	HT 1	HT 2	HT 1	HT 2	HT 1	HT 2	HT 1	HT 2	HT 1	HT 2	HT 1	HT 2	HT 1	HT 2
1	20000	20000	20000	20000	100%	100%	100%	100%	10000	10000	10000	10000	50%	50%	50%	50%
2	20000	20000	20000	0	100%	100%	100%	0%	10000	10000	20000	0	50%	50%	100%	0%
3	20000	0	20000	0	100%	0%	100%	0%	20000	0	20000	0	100%	0%	100%	0%
4	20000	20000	0	0	100%	100%	0%	0%	10000	10000	0	0	50%	50%	0%	0%
5	20000	0	0	0	100%	0%	0%	0%	20000	0	0	0	100%	0%	0%	0%
6	0	0	0	0	0%	0%	0%	0%	0	0	0	0	0%	0%	0%	0%

The table shows clearly that Used splits the Utilization into 2 when both threads are running.

Look at scenario 1. While Utilization charges 20000 ms to each thread, Used charges 10000. This is not intuitive as ESXi considers HT to deliver 1.25x. Personally I find 12500 easier to understand. The good news is this number is normalized back when it is rolled up to the ESXi host level.

How will those scenarios roll up at the ESXi level?

The following table shows the 4 metrics (Utilization, Used, Core Utilization, Usage). I have expressed each in % so it's easier to compare.

There are 6 different scenarios, so logically there should be 6 different values. But they are not, so I added my personal take on what I like them to show. I'm keen to hear your thought.

Scenario	Utilization	Core Utilization	Usage	My Recommendation	
1	100%	100%	100%	= (125%+125) / 2	125.0%
2	75%	100%	100%	= (125%+100) / 2	112.5%
3	50%	100%	100%	= (100%+100) / 2	100.0%
4	50%	50%	50%	= (125%+0) / 2	62.5%
5	25%	50%	50%	= (100+0) / 2	50.0%
6	0%	0%	0%	= (0%+0%) / 2	0.0%

What's causing the difference?

Yup. Hyper Threading.

Why do I choose 125% instead of 100%?

To me, the 1.25x bonus factor has to be shown. Without HT, it's 100%. HT is a bonus. While it provides 1.25x overall throughput, each thread pays an expensive price, as each suffers 37.5% penalty.

Scenario	Analysis
1	Do you notice something strange with the value of Usage (%)? Yes, it's no longer 50%. It's 100%. The average of 50% is 100%. The reason is the accounting does not count each thread as 20000. Each core has 20000 and not 40000. If you say that is similar behaviour to Core Utilization, you're right.
3	Utilization is only showing 50% when both cores are in fact already utilized. I prefer this to show 80% as HT only delivers 1.25x, not 2x.
5	Utilization is again showing too low a value.

Now let's add CPU clock speed. What happens when there is power management?

I'd focus on just Used and Usage to highlight the difference.

What do you notice from the table below?

Frequency	Usage	E1
0.6x	60%	75%
0.6x	60%	68%
0.6x	60%	60%
0.6x	30%	38%
0.6x	30%	30%
0.6x	0%	0%

Frequency	Usage	E1
1.3x	100%	163%
1.3x	100%	146%
1.3x	100%	130%
1.3x	65%	81%
1.3x	65%	65%
1.3x	0%	0%

Usage is capped at 100%. I prefer this not to be capped, so you know its actual value. The good part is Demand metric is not.

For comparison, I put forth what I think the value should be.

Recommendation

The answer depends on the purpose: performance (VM-centric) or capacity (ESXi-centric)? Performance is about giving VM the highest quality resource (no HT), but capacity is about using all the resources (including HT).

Here is how the counters stack up in terms of sensitivity:

Metric	Analysis
Utilization	The value shows 50% when it should report 80%
Core Utilization	The value goes up to 100% when it should report 80%
Usage	These 3 metrics are essentially the same.
Used	The value is affected by CPU frequency. During low utilization it under-reports due to power savings, while during high utilization it over-reports due to turbo-boost.
Consumed	
Demand	The value is even higher than Usage.

I'd use Core Utilization (%) and Thread Utilization (%) as the primary counters. Yes, 2 counters.

I'd complement with Usage (MHz) for performance.

If Core Utilization is not yet 100% or Thread Utilization is not yet 50% then there is still physical cores available. You can go ahead deploy new VMs.

If Core Utilization = 100% (meaning Thread Utilization is at least 50%) then review Thread Utilization and ensure it's not passing your threshold. I'd keep it around 80% - 90% per ESXi, meaning the level at cluster level will be lower as we have HA host. Also, check contention.

Regardless of what counters you use, As Mark Achtemichuk said it best: "drive by contention".

Real World Samples

How do these values compare in production environment? The following table shows 297 ESXi hosts. It shows the various possibilities covered earlier.

Name	CPU|Demand (%)	CPU|Usage (%)	CPU|Core Utilization (%)	CPU|Utilization (%)	CPU|Used (Second(s))	Configured (C
wdc-08-r0...	51.88 %	54.81 %	42.25 %	24.43 %	438.5 Second(s)	40 Cores
wdc-10-r13...	39.23 %	22.5 %	41.13 %	24.4 %	252.03 Second(s)	56 Cores
wdc-08-r0...	65.77 %	60.63 %	46.04 %	24.31 %	485.08 Second(s)	40 Cores
wdc-08-r0...	57.96 %	55.69 %	42.77 %	24.3 %	445.6 Second(s)	40 Cores
wdc-08-r0...	53.27 %	53.68 %	41.66 %	24 %	429.51 Second(s)	40 Cores
wdc-08-r0...	62.14 %	56.68 %	43.25 %	23.91 %	453.48 Second(s)	40 Cores
wdc-10-r13...	39.22 %	23.42 %	40.61 %	23.84 %	262.32 Second(s)	56 Cores
wdc-10-r13...	41.45 %	24.93 %	40.25 %	23.7 %	279.22 Second(s)	56 Cores
Average	**33.39 %**	**28.64 %**	**23.96 %**	**13.54 %**	**260.39 Second(s)**	**47.57 Cores**

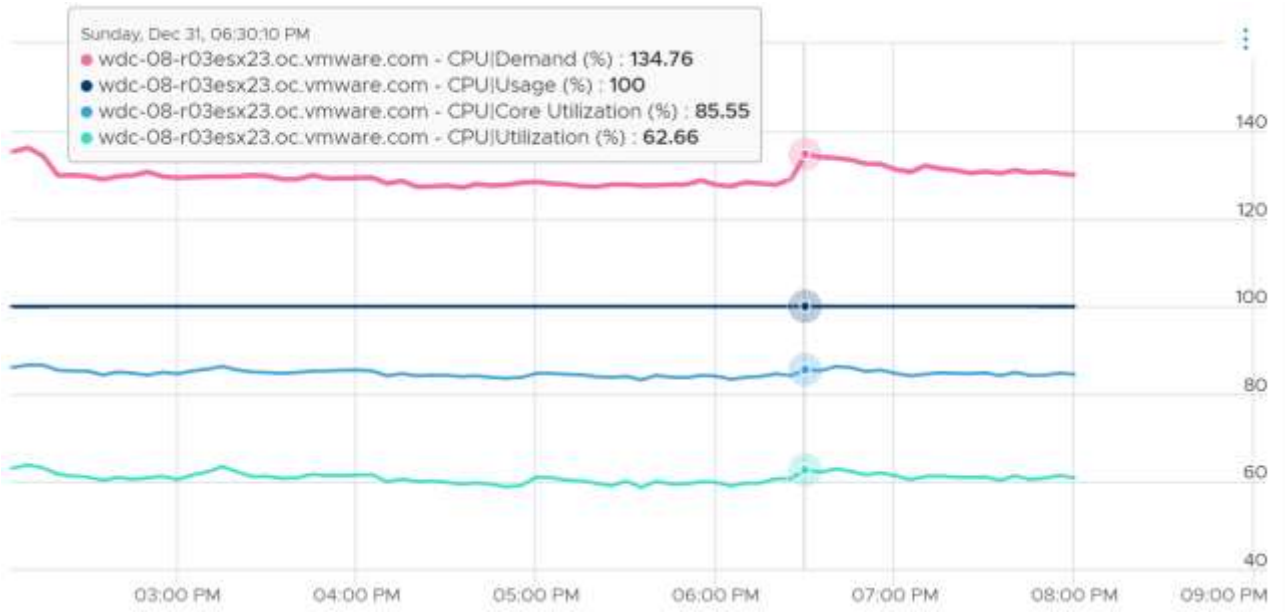
1 - 50 of 297 items < 1 2 3 4 5 6 >

Let's zoom on an ESXi with a very high utilization.

Take a look at the 4 numbers below. Demand is consistently around 130%.

With such a high demand value, it's no surprise that Usage is 100% flat out.

Do we have a performance problem? What can you deduce from the other 3 numbers?

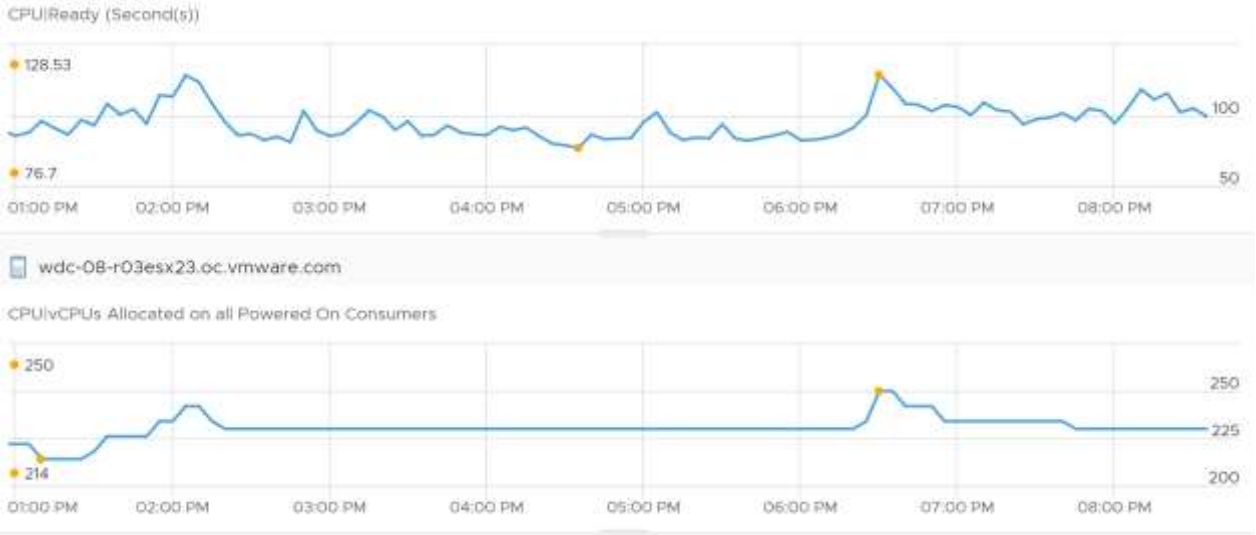


Core Utilization at 85% is not very high. The box still has 15% of the cores not running. The box has 40 cores 80 threads. Unless there is limit, there are still 6 cores available.

Utilization at 62% is moderate. This means ~38% x 80 threads, or 30 threads are available.

Based on the raw utilization above, I expect the CPU ready to be low.

Let's check. What's your conclusion from below?



The value peaks at 128 seconds.

But that's for the entire box. There are 250 vCPU at that time.

Dividing the value, it's only 514 ms per vCPU. This translates into 2.5%, a relatively low number.

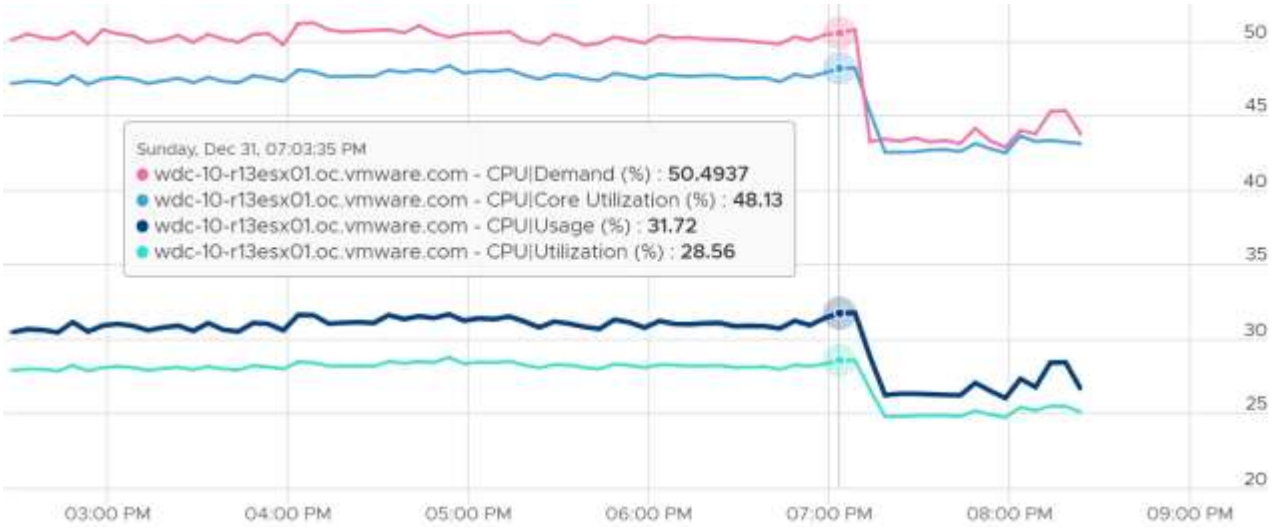
Let's validate this further by going into each VM.

I took the average of the entire 7 hour period, and got around 2% for Ready and less than 0.1% for Co-Stop. This is pretty good considering these VMs are large. They are 15.29 vCPU on average. Some of them also have limit, which could contribute to the CPU read.

Name	Ready ↑	Co-stop	Limit	vCPU
gcl-1-5C7s	1.69 %	0.0014 %	32 GHz	16
esx-hcx-setup-1-1S1V	1.94 %	0.12 %	72 GHz	36
ubuntu-groovy-20.1...	2.07 %	0.0003 %	8 GHz	4
esxmgmt-02a-Uaio	2.18 %	0.02 %	36 GHz	18
ubuntu-22.04-2023...	2.96 %	0.07 %	32 GHz	16
ubuntu-22.04-2023...	3.05 %	0.07 %	32 GHz	16
ubuntu-22.04-2023...	3.64 %	0.26 %	32 GHz	16
ubuntu-22.04-2023...	4.79 %	0.26 %	32 GHz	16
Highest	4.79 %	0.26 %	72 GHz	36
Average	1.98 %	0.06 %	30.57 GHz	15.29
Summary	-	-	-	214

Now let's pick the opposite example.

The following chart shows an ESXi with low utilization. What do you notice?



This time around, Usage is lower than Core Utilization.

This ESXi is not even 50% utilized, as the core utilization shows 48%. The kernel decides that it could complete the job with less power, and clocks down the core.

Reservation

There is only 1 metric provided, which is Reserved Capacity (MHz). Compare this with memory, which provides 3 counters. Why is that so?

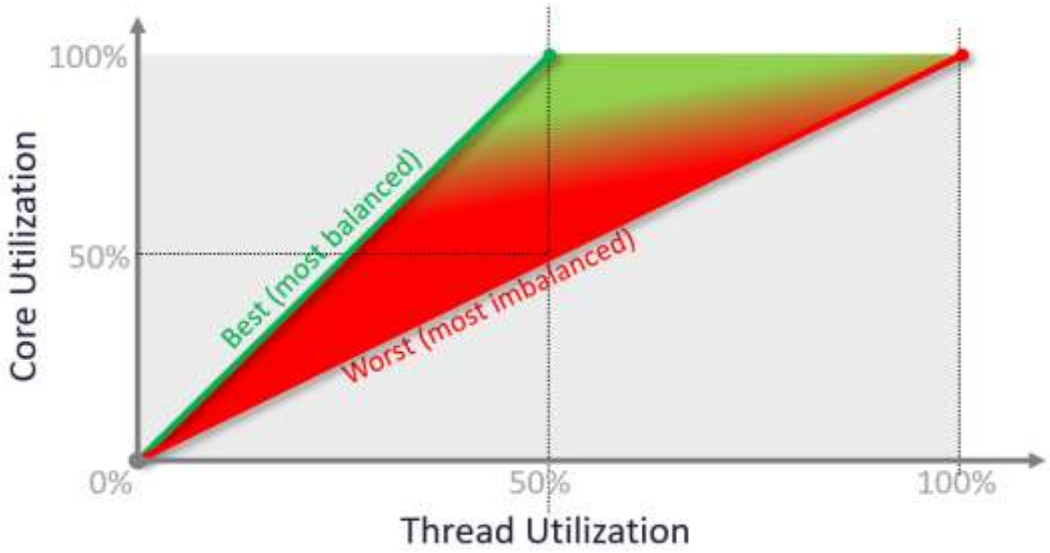
Review the System Architecture section. In short, it is not applicable to CPU as due to its highly transient nature. When the VM is not running, the reserved capacity can be used by other VM. This differs to memory, which is “sticky” as it’s a form of storage.

The metric does not include hypervisor reservation.



Imbalance

You want all cores used first, before threads are sharing a core.



Since HT is dual thread, the worst score is 2.0.

When there are idle cores, no core should be running 2 threads. Since real world is not perfect world, the diagram shows gradient. It changes from **red** to **green** as Core Utilization hits 100%.

Number of CPU Core Running	Number of CPU Thread Running																											
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29
1	1.00	2.00																											
2		1.00	1.50	2.00																									
3			1.00	1.33	1.67	2.00																							
4				1.00	1.25	1.50	1.75	2.00																					
5					1.00	1.20	1.40	1.60	1.80	2.00																			
6						1.00	1.17	1.33	1.50	1.67	1.83	2.00																	
7							1.00	1.14	1.29	1.43	1.57	1.71	1.86	2.00															
8								1.00	1.13	1.25	1.38	1.50	1.63	1.75	1.88	2.00													
9									1.00	1.11	1.22	1.33	1.44	1.56	1.67	1.78	1.89	2.00											
10										1.00	1.10	1.20	1.30	1.40	1.50	1.60	1.70	1.80	1.90	2.00									
11											1.00	1.09	1.18	1.27	1.36	1.45	1.55	1.64	1.73	1.82	1.91	2.00							
12												1.00	1.08	1.17	1.25	1.33	1.42	1.50	1.58	1.67	1.75	1.83	1.92	2.00					
13													1.00	1.08	1.15	1.23	1.31	1.38	1.46	1.54	1.62	1.69	1.77	1.85	1.92	2.00			
14														1.00	1.07	1.14	1.21	1.29	1.36	1.43	1.50	1.57	1.64	1.71	1.79	1.86	1.93	2.00	
15															1.00	1.07	1.13	1.20	1.27	1.33	1.40	1.47	1.53	1.60	1.67	1.73	1.80	1.87	1.93

Imbalance Score of 1.5 at means half the cores are running 2 threads.

For an ESXi with 64 cores, 128 threads, a score of 1.5 at 40 cores means:

- 20 cores are running 2 threads
- 20 cores are running 1 thread
- 24 cores are idle.

Ideally, it should be 60 cores running 1 thread each.

The imbalance should be used together with Core Utilization.

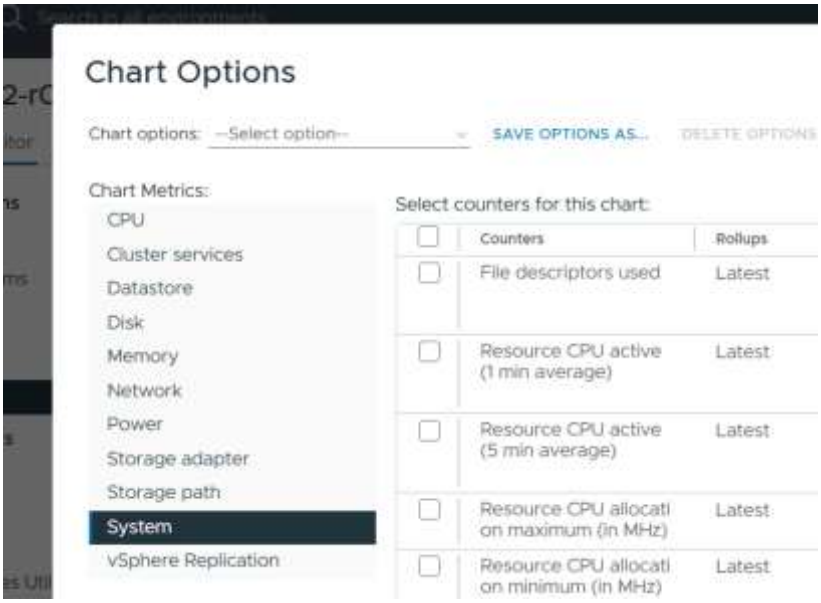
```
ESXi CPU Imbalance = Count of Running Threads / Count of Running Cores
vSphere Cluster ESXi CPU Imbalance = Average (ESXi CPU Imbalance)
```

Metrics to Avoid

Do not use the following metrics. Use the recommended metric instead.

CPU Utilization for Resources

Under the System metric group, you will see 17 metrics with names starting with "CPU Utilization for Resources".



Here is the full list.

Name in VCF Operations	Name in vSphere Client
CPU Utilization for Resources|CPU Active (1 min. average)	rescpu.actav1.latest
CPU Utilization for Resources|CPU Active (5 min. average)	rescpu.actav5.latest
CPU Utilization for Resources|CPU Active (15 min. average)	rescpu.actav15.latest
CPU Utilization for Resources|CPU Active (1 min. peak)	rescpu.actpk1.latest
CPU Utilization for Resources|CPU Active (5 min. peak)	rescpu.actpk5.latest
CPU Utilization for Resources|CPU Active (15 min. peak)	rescpu.actpk15.latest
CPU Utilization for Resources|CPU Running (1 min. average)	rescpu.runav1.latest
CPU Utilization for Resources|CPU Running (5 min. average)	rescpu.runav5.latest
CPU Utilization for Resources|CPU Running (15 min. average)	rescpu.runav15.latest
CPU Utilization for Resources|CPU Running (1 min. peak)	rescpu.runpk1.latest
CPU Utilization for Resources|CPU Running (5 min. peak)	rescpu.runpk5.latest
CPU Utilization for Resources|CPU Running (15 min. peak)	rescpu.runpk15.latest
CPU Utilization for Resources|CPU Throttled (1 min. average)	rescpu.maxLimited1.latest
CPU Utilization for Resources|CPU Throttled (5 min. average)	rescpu.maxLimited5.latest

CPU Utilization for Resources|CPU Throttled (15 min. average)	rescpu.maxLimited15.latest
CPU Utilization for Resources|Group CPU Sample Count	rescpu.sampleCount.latest
CPU Utilization for Resources|Group CPU Sample Period	rescpu.samplePeriod.latest

When the hypervisor says "for resources," it means "from the perspective of the Resource Scheduler's allocation buckets." To the ESXi kernel (VMkernel), a "resource" isn't just a virtual machine—it's any node in the allocation hierarchy that can have Shares, Reservations, or Limits applied to it.

Do not use them. They are legacy metrics.

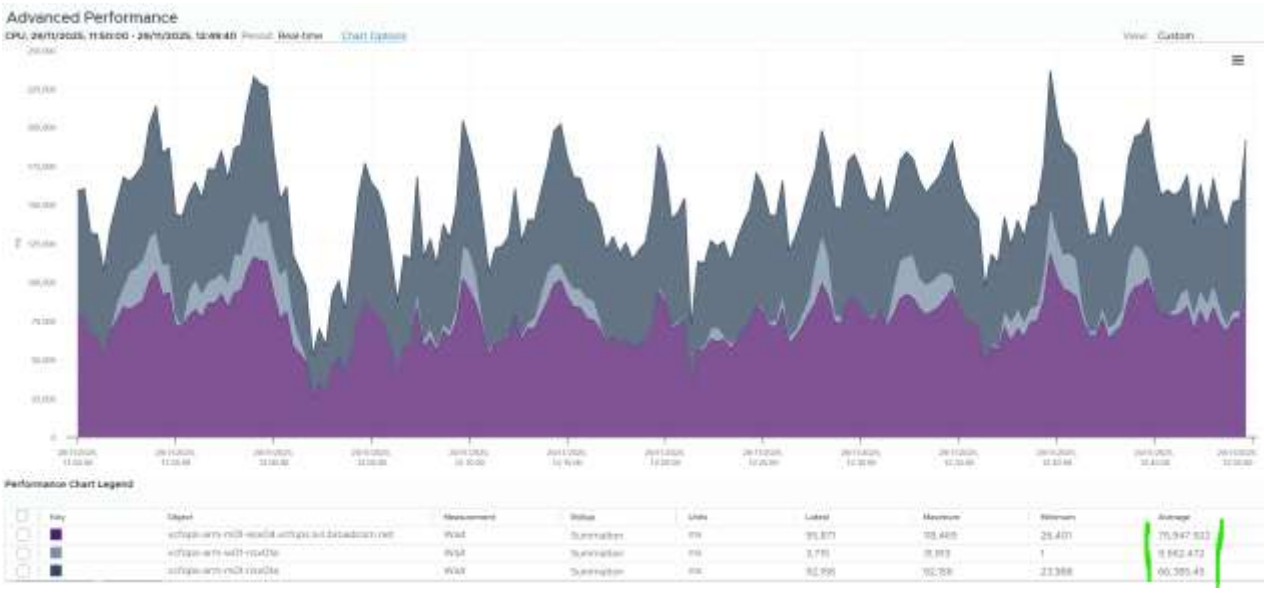
Instead, use the following:

- For CPU Active, use CPU Demand as they are the same. The 1-minute average results in the same value as the 20-second metric and the 5-minute metric, as VCF Operations averages all of them into a 5-minute value (or whatever default interval you set). So there is no need for a separate 1-minute. The 15-minute does not have a use case as you already have the 5-minute and can always roll up.
- For CPU Running, use CPU Usage.
- For CPU Throttled, use Effective Limit.

Wait Metric

You can ignore it as it's the sum of the VM Wait, which mixes idle time.

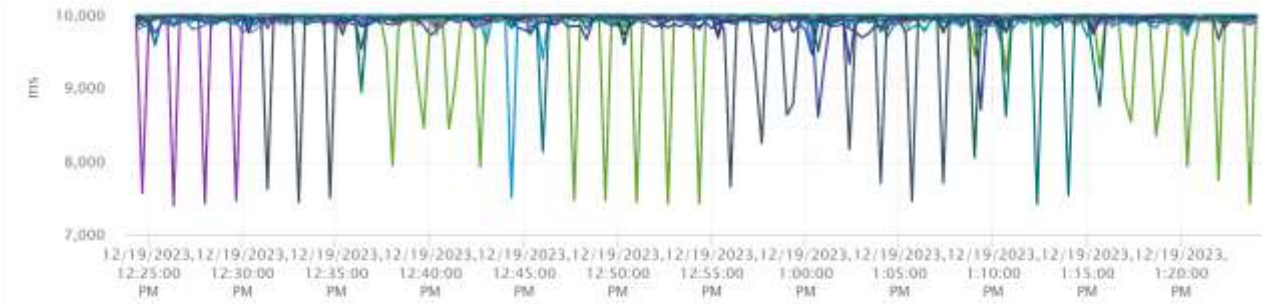
The following ESXi host has 2 VM. You can see the Wait metric is the sum of the 2 VM wait metrics.



Idle

Before we cover more advance consumption metrics, we need to cover Idle. The reason is idle + non idle should add up to 100%. Knowing what defines 100% is crucial.

If HT is enabled, Idle is capped at 10000 ms at a thread level. The following example shows an ESXi with 64 threads that is basically idle. Notice none of the threads passes 10000 ms. The table shows maximum value of 9998, not exactly 10000. I suppose the 2 ms is the kernel just humming along.

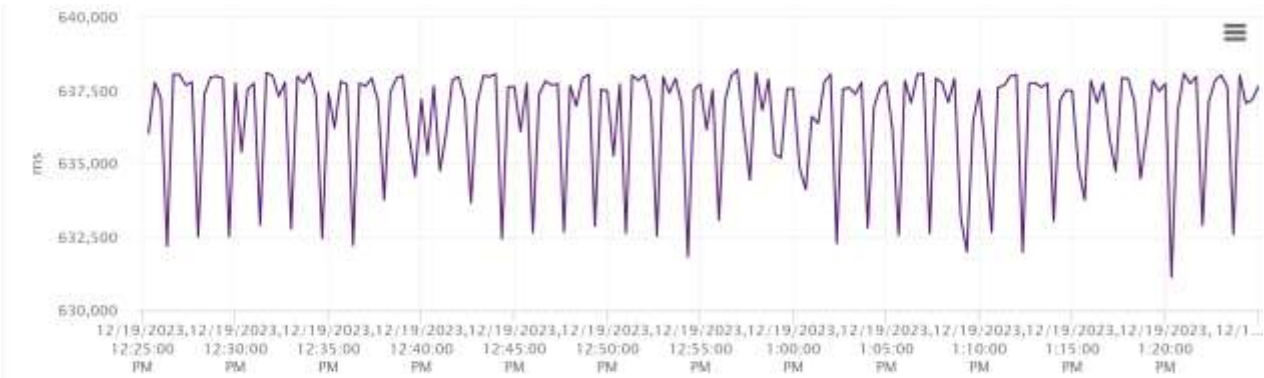


Performance Chart Legend

Key	Object	Measurement	Rollup	Units	Latest	Maximum ↓	Minimum
	22	Idle	Summation	ms	9,987	9,998	9,831
	58	Idle	Summation	ms	9,957	9,998	7,458
	44	Idle	Summation	ms	9,988	9,998	7,500
	23	Idle	Summation	ms	9,987	9,998	9,831
	59	Idle	Summation	ms	9,957	9,998	7,458

The above translates that the total per core should be 20000, since there are 2 threads per core.

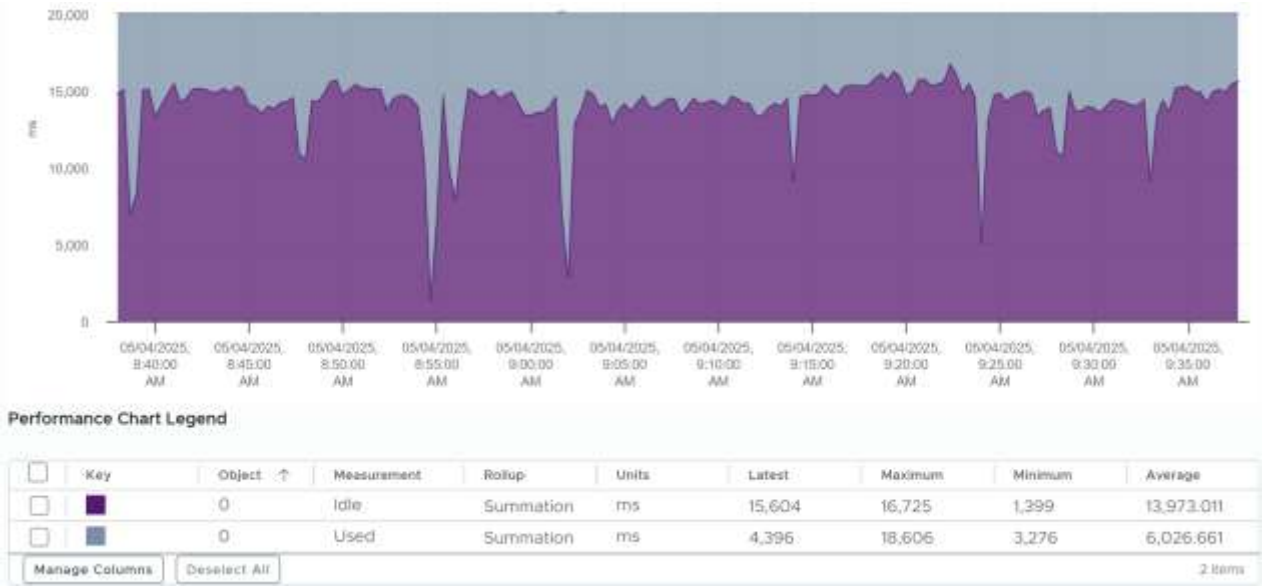
You can confirm the above ESXi is idle by plotting idle at the host level. The sum is near 640000 ms.



Performance Chart Legend

Key	Object	Measurement	Rollup	Units	Latest	Maximum ↓	Minimum
	w2-hs4-r1207.eng.vmware.com	Idle	Summation	ms	637,631	638,205	631,128

If HT is not available, then there is no counter at thread level. The counter at the core level adds up to 20000 ms.



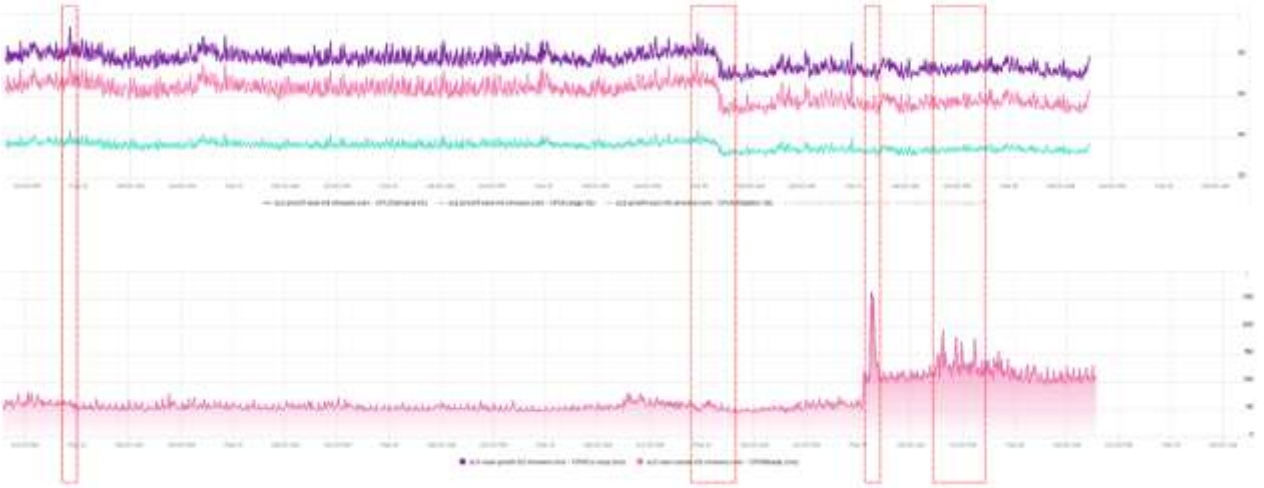
Performance Chart Legend

Key	Object	Measurement	Rollup	Units	Latest	Maximum	Minimum	Average
	0	Idle	Summation	ms	15,604	16,725	1,399	13,973.011
	0	Used	Summation	ms	4,396	18,606	3,276	6,026.661

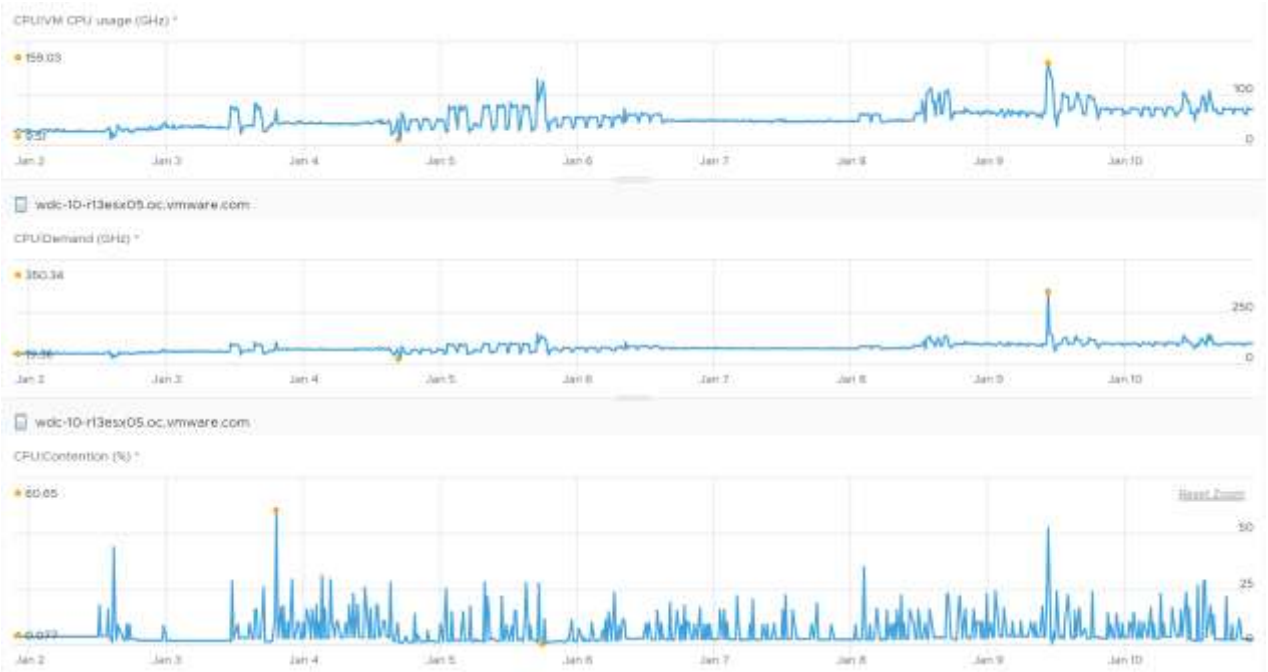
Manage Columns | Deselect All | 2 items

Contention Metrics

The nature of average is also one reason why ESXi "consumption" does not correlate to ESXi "contention". The 4 highlighted area are examples where the metrics don't correlate, even go the opposite way in some of them. Can you guess why?



The above picture is a little too small. The following ESXi has total capacity of 111 GHz. So Demand hit 315%! Sum of VM CPU Usage also unbelievably high at 143%. Sure, Ready + CoStop were high at that moment, but how do you explain these 2 exceed Total Capacity?



These are the reasons why they don't match:

- One looks at physical CPU, the other the virtual CPU. One looks at ESXi, while the other looks at VM.
- Hyperthreading and Power Management.
- Imbalance utilization. There are many VMs in this host. Their experience will not be identical.
- Limit may impact the VM, either directly or via resource pool.
- CPU pinning, although this rarely happens.

So what metrics should you use?

Here are the latency metrics provided by vSphere Client.

Counters	Rollups ↑	Units	Description
Co-stop	Summation	ms	Time the virtual machine is ready to run, but is unable to run due to co-scheduling constraints
Ready	Summation	ms	Time that the virtual machine was ready, but could not get scheduled to run on the physical CPU during last measurement interval
Latency	Average	%	Percent of time the virtual machine is unable to run because it is contending for access to the physical CPU(s)
Readiness	Average	%	Percentage of time that the virtual machine was ready, but could not get scheduled to run on the physical CPU
Wait	Summation	ms	Total CPU time spent in wait state
Swap wait	Summation	ms	CPU time spent waiting for swap-in
Idle	Summation	ms	Total time that the CPU spent in an idle state

For Co-stop and Ready, the counter does not include contention faced by non VM worlds[13].

Your operations can't wait until problem become serious. All the built-in metrics are averaged of all the running VMs. So by the time they are high, it's time to prepare your resume, not start troubleshooting 😉.

I'd provide a set of leading indicators to replace these lagging indicators. As performance management is best done holistically, I'd cover it as a whole. Find them under vSphere Cluster chapter as an ESXi is typically part of a cluster.

[13] Thanks Kristiyan for the confirmation

Part 3 Chapter 3

Architecture

The purpose of memory is to act as disk cache. So you want to utilize all the cache given to you. 100% usage results in faster performance than 99%, all else being equal.

Let's now take a trip down memory lane, pun intended.

Memory differs from CPU as it is a form of storage.

- CPU is transient in nature. Instructions enter and leave the execution pipelines in less than a nanosecond. That's why CPU reservation becomes not applicable when the VM is not using it.
- Memory behaves more like disk space. Memory reservation could remain in place even if the VM has not read or written the page in days.

As a storage, memory is basically a collection of blocks in physical DIMM. Information is stored in memory in standard block sizes, typically 4 KB or 2 MB. Each block is called a page. At the lowest level, the memory pages are just a series of zeroes and ones. MS Windows initializes its pages with 0, hence there is a zero-page counter in ESXi.

Keeping this concept in mind is *critical* as you review the memory metrics. The storage nature of memory is the reason why memory monitoring is more challenging than CPU monitoring. Unlike CPU, memory has 2 dimensions:

Speed	Nanoseconds	Speed matters in CPU, whereas in RAM we can generally ignore it. DDR5 RAM is faster than DDR4 but for general monitoring reason it can be ignored. This becomes more important in memory tiering, as it's using NVMe disk and not DIMM.
		The only counter ESXi has is Memory Latency. This counter increases when the time to read from the RAM is longer than usual. The counter tracks the percentage of memory space that's taking longer than expected. It's not tracking the *actual* latency in nanosecond. This is the opposite of Disk, where we track the actual latency, but not the percentage of amount of space that is facing latency. Both are storage, but "server people" and "storage people" measure them differently 😊
Space	Bytes	This is the bulk of the metrics
		As storage, it's relatively static. As a result, you can create a heat map that plots all your Guest OS or VMs memory consumption. You want it near 100% while making sure the page in and page out rate within normal expectation.

The 4 Layers

With virtualization, there are 2 additional layers added. This brings the total to 4 from Process → Guest OS → VM → ESXi.

No, a container does not have its own memory management system. Because a container shares the host OS kernel, it relies entirely on the parent OS kernel to allocate, track, and manage physical memory.

However, the parent OS uses specific kernel features to make the container think it is isolated and to strictly limit how much memory it can use.

The only layer that manages the actual physical memory is the last layer. This is why the term "Guest physical memory" is illogical.

Each layer has their own viewpoint. Understanding the vantage point is required to make sense of the metrics. It will prevent you from comparing metrics that are not comparable (e.g. granted vs consumed) as they have different context.

Each of these layers have their own address space. And that's where the fun of performance troubleshooting begins 😉

Virtual Memory

Virtual memory is an integral part of memory management. The following[14] shows how Windows or Linux masks the underlying physical memory from processes running on the OS.



From the process' point of view, this technique provides a contiguous address space, which makes memory management easier. It also provides isolation, meaning process A can't see the memory of process B. This isolation provides *some* level of security. The isolation is not as good as isolation by container, which in turn is inferior to isolation by VM.

Virtual memory abstraction provides the possibility to overcommit. Microsoft Windows may only have 16 GB of physical RAM, but by using pagefile the total memory available to its processes can exceed 16 GB. The process is unaware what is backing its virtual address. It does not know whether a page is backed by Physical Memory or Swap File. All it experiences is slowness, but it won't know why as there is no counter at process level that can differentiate the memory source.

[14] For details, I recommend one of Valentin Bondzio talks

On the other hand, some applications manage its own memory and do not expose to the operating system. Example of such applications as are database and Java VM. Oleg Ulyanov shared in this blog SQL Server has its own operating system called SQLOS. It handles memory and buffer management without communicating back to underlying operating system.



From the VMs point of view, it provides a contiguous address space and isolation (which is security). The underlying physical pages at ESXi layer may not be contiguous, as it's managed differently.

VM memory	Metrics tracks the VM Pages. There are 2 sets, one for each VM, and one a summation at ESXi level for all running VMs. Do not confuse the summation with ESXi memory metrics. Examples: Granted or Memory Shared
ESXi memory	Metrics tracks the ESXi Pages. There are also 2 sets, but the summation at ESXi level contains the kernel own memory and VM overhead Examples: Consumed or Memory Shared Common

ESXi uses 3 levels to manage memory:

TPS	Transparent Page Sharing (TPS) happens automatically even if ESXi has plenty of RAM as it makes sense to do so. It's not an indicative of unmet demand. Sharing the same page is the right thing to do, and not something that should be started only when physical pages are running low. See more here.
Tiered Memory	This also happens transparently to the VM. ESXi moves inactives pages to the NVMe device.
Balloon	The first sign of unmet demand. It happens proactively, before ESXi is unable to meet Demand. Ballooning reduces cache. It does not mean ESXi unable to meet Demand. Demand is not met when Contention happen. That's the only time it is not met.
Compress/Swap	This happens proactively too. It does not mean VMs were contending for RAM. It merely means ESXi Consumed is very high. That Consumed can contain a lot of cache
	If there is a host cache, ESXi uses this first.

This abstraction provides the possibility to overcommit, because the VM is unaware what is backing the physical address. It could be physical memory on DIMM, swap file, Copy-on-Write, zipped, or ballooned. The following is taken from VMware vSphere 6.5 Host Resources Deep Dive[15]. I have highlighted in green the part you need to pay attention.

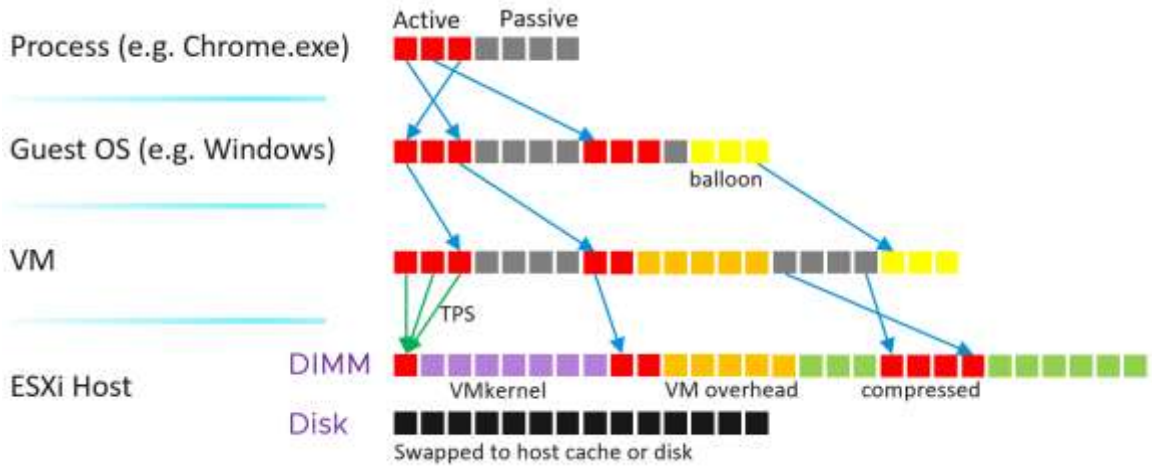
> The VMkernel manages the physical memory of an ESXi host. A small part of the system's memory is used to run the VMkernel and its processes. The remaining memory is available to the VMs. To run VMs the VMkernel abstracts the physical memory to dynamically allocate memory to the active VMs. The VMkernel manages physical memory without the knowledge of the guest OS. In turn, the guest OS manages its own memory space without informing the VMkernel.
>
> To be aware of this mutual disconnect allows you to further understand the impact of each memory reclamation technique. In turn this helps you with designing a proper consolidation ratio for your environment. Memory reclamation techniques are covered in the next chapter.

Read further and you will see that the kernel large page setting contributes more to ESXi capacity and the VM performance.

Relationship

Let's put the 4 layers in a diagram[16], showing how a page maps across the 4 layers.

What do you notice?



Yup, a few things:

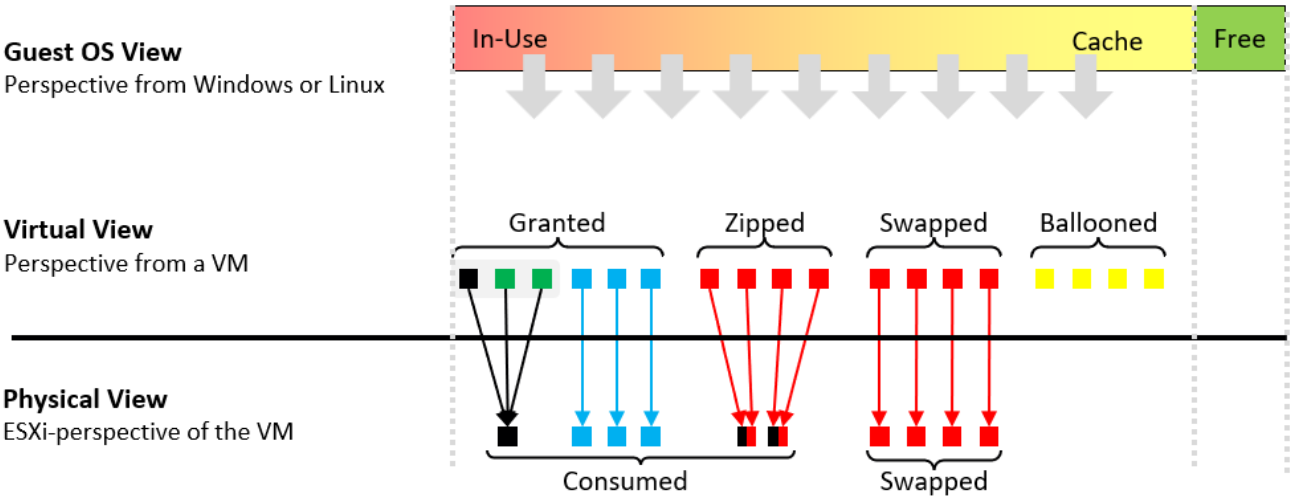
- Balloon did not map to physical layer. Notice there is no arrow on the yellow blocks
- The physical memory is "shared" between VM and hypervisor own processes, meaning you need to manage them as one. Sum their utilization to form the total consumption.

[15] Chapter 11 VMkernel Memory Management, page 243. Book by Frank Denneman and Niels Hagoort
[16] Modified from yet another diagram of Valentin Bondzio

- At the physical layer, there can be host cache and tiered memory, and not just DIMM. If you use memory tiering, be on guard when active memory metric crosses 50%. The NVMe latency should be low also.
- Just to be complete, the mapping may not be to the closest DIMM. You can have NUMA effect.

The VM Monitor for each VM maps the VM pages to the ESXi pages[17]. This page mapping is not always 1:1. Multiple VM pages may point to the same ESXi pages due to transparent page sharing. On the other hand, VM page may not map to ESXi page due to balloon and swapped. The net effect is the VM pages and ESXi pages (for that VM) will not be the same, hence we need two sets of metrics.

Let's zoom into the key metrics that form a single VM.



Take note the difference between the VM perspective and the ESXi perspective.

- The VM looks at the "input", while ESXi looks at the "output". For example, VM metrics will report 3 pages being shared, while ESXi will report 1 page of the result shared page.
- Balloon does not exist at physical view as the page has been removed. It's not pointing to any block in the ESXi memory address space.

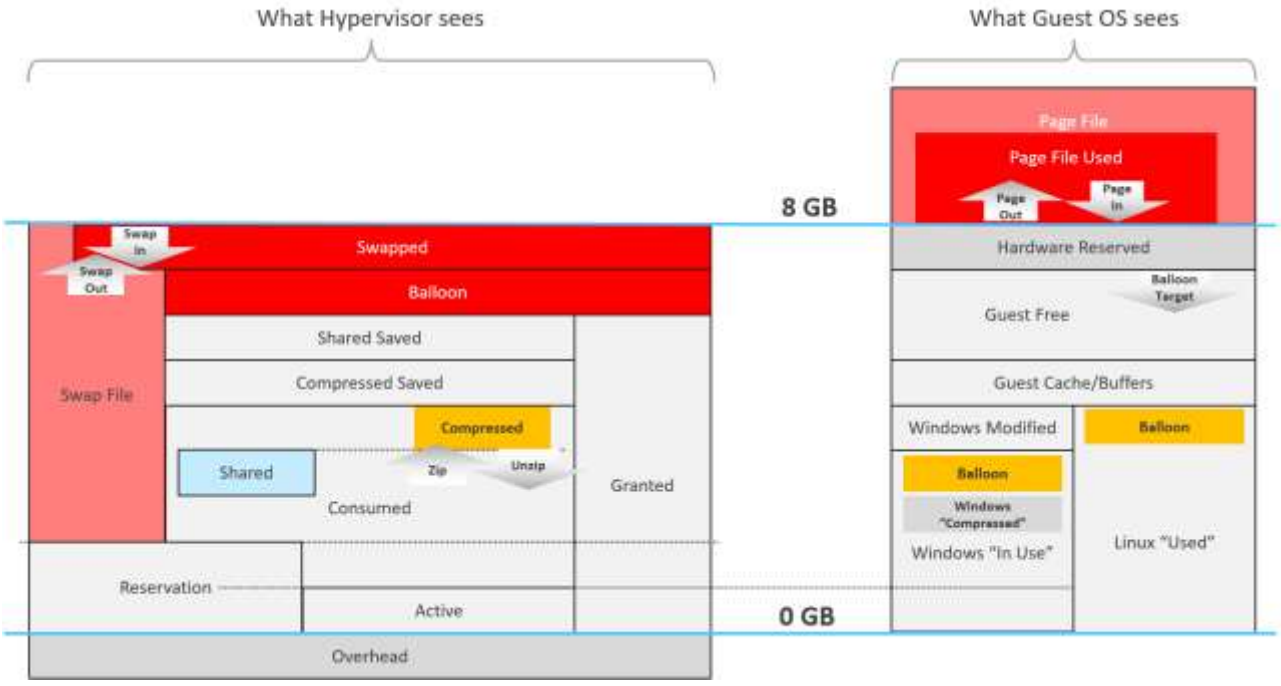
Further reading: vSphere Resource Management technical paper.

[17] Other documents use the term Guest Physical Page and Machine Page. I find it unnecessarily confusing, so I just call it VM pages and ESXi pages.

Guest OS vs VM

Both come with dozens of metrics. Compared with Guest OS such as Windows, can you notice what's missing and what's added?

The following diagram compares the memory metrics between VM and Guest OS,



Guest OS and VM metrics do not map to each other. Neither the kernel nor the Guest OS have full visibility into each other.

Right off the bat, you will notice that popular metrics such as Consumed, Shared, and Reservation do not even exist in Windows.

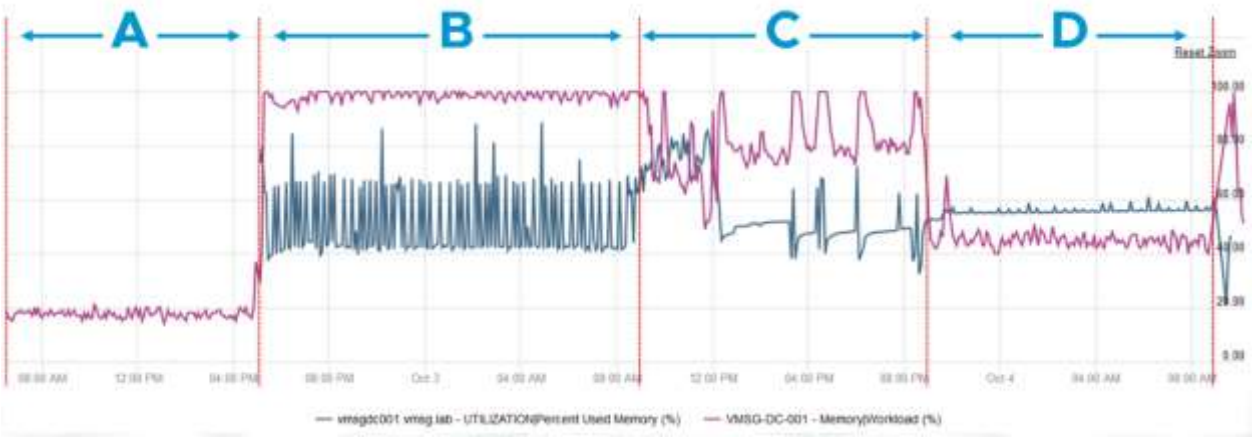
Type	Guest OS Metric	VM Metrics
Contention	Paging	None
	None	Latency
Utilization	In Use	None
	Cache	None
	Free	None
	Compressed	None
	None	Swapped or Compressed

ESXi Host cannot see how the Guest OS manages its memory pages, how it classifies the pages as Use, Modified, Cache and Free. ESXi also cannot see the virtual memory (page file).

ESXi can only see when the Guest OS performs reads or writes. That's why vSphere VM main metrics are basically what is active recently and what has been active. The first one is called Active, the second is called Consumed. All other metrics are about ESXi memory management, and not about VM memory utilization. VM memory utilization impacts ESXi memory management, but they are clearly not the same thing.

Example: Guest OS More Accurate

Let's take an example with a simple Microsoft Windows server running Active Directory. It has 4 GB of memory as it's just serving a small number of objects in the Singapore office lab. Look at the following table, where I compared the counter from inside the Guest OS and the VM memory active counter.



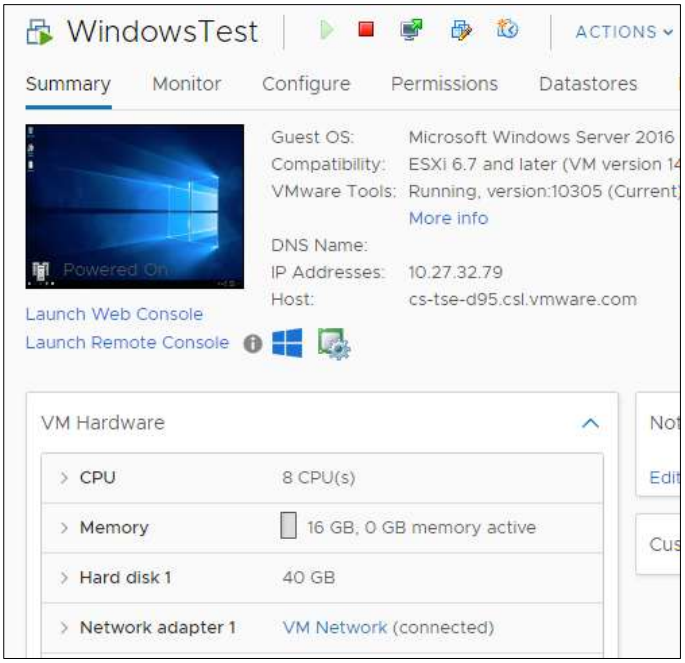
There are four periods above where I made changes inside Windows. Let's step through them.

Period	What happened
A	Microsoft AD server in normal running condition. vCenter is reporting low utilization, around 15-20%. Note vCenter users the Active metric, not Consumed.
B	I installed the VCF Operations agent, which is based on the open source Telegraf. This gives the Guest OS metric, which is shown in blue. The agent collects data every 5 minutes, hence the regular spike. So far so good. Notice the value from VM Active metric jumps to 100%. That's fine, but then it stays at 100% for more than 12 hours. All I did was installing a small collection agent and that's it. I actually got an alarm in vCenter, even though the VM does not need the RAM obviously. What happened here prove that the Active counter is based on sampling, and that sampling could be wrong.
C	The next morning, I decided to generate some load as the pattern does not change at all. Since Windows has not been patched for a long time, I started Windows patch. The entire process is mostly downloading and installing, which last for several hours. The two metrics show no correlation at all.
D	After several hours, the entire Windows update process is completed.

Example: VM More Accurate

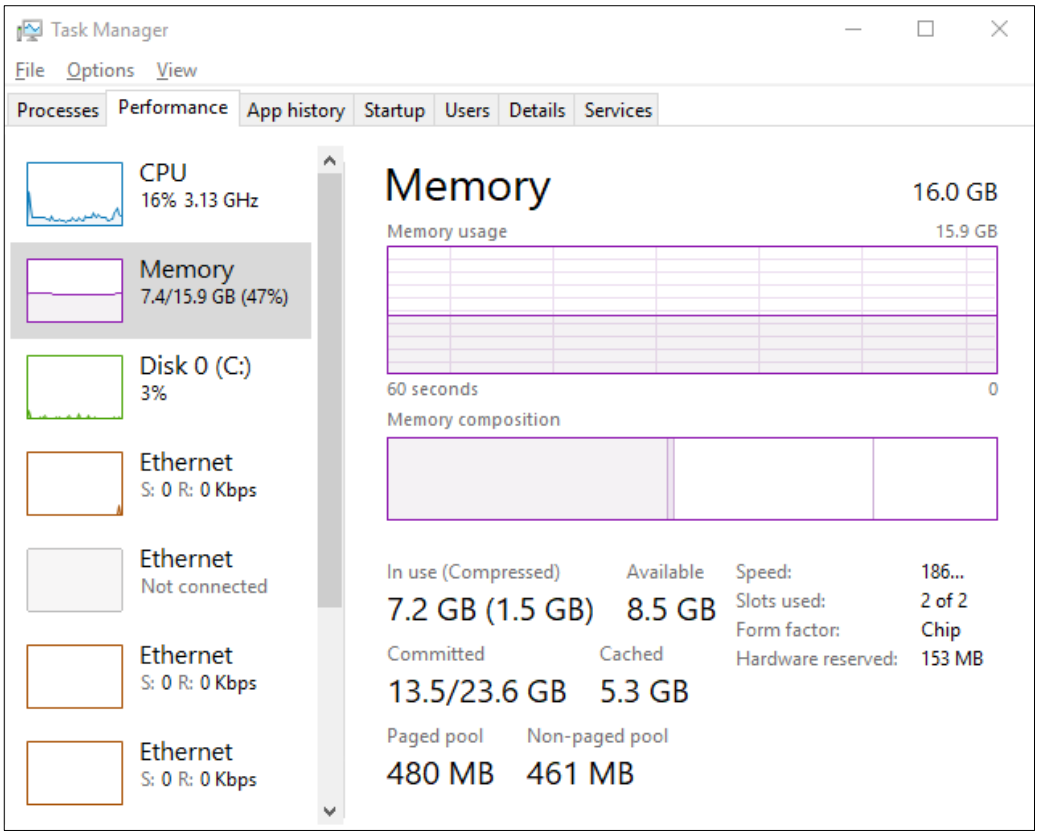
Let's now look inside the VM. I will use another VM to show a different example. This time around, I will take an idle VM so we can see how the metrics behave. An idle VM will have minimal or 0 activity.

You can see that this Windows Server 2016 VM has 16 GB, but 0 GB is active. It is expected as we know the Guest OS is idle as nothing is installed. vCenter is showing the data correctly. So far so good….



What do you think you will see inside Windows?

Will the Windows In Use counter show that it's using 0 GB or somewhere near there? You know that it won't show 0 GB as it's impossible that any OS does not use any memory while it's running. So what number will the In Use counter show?

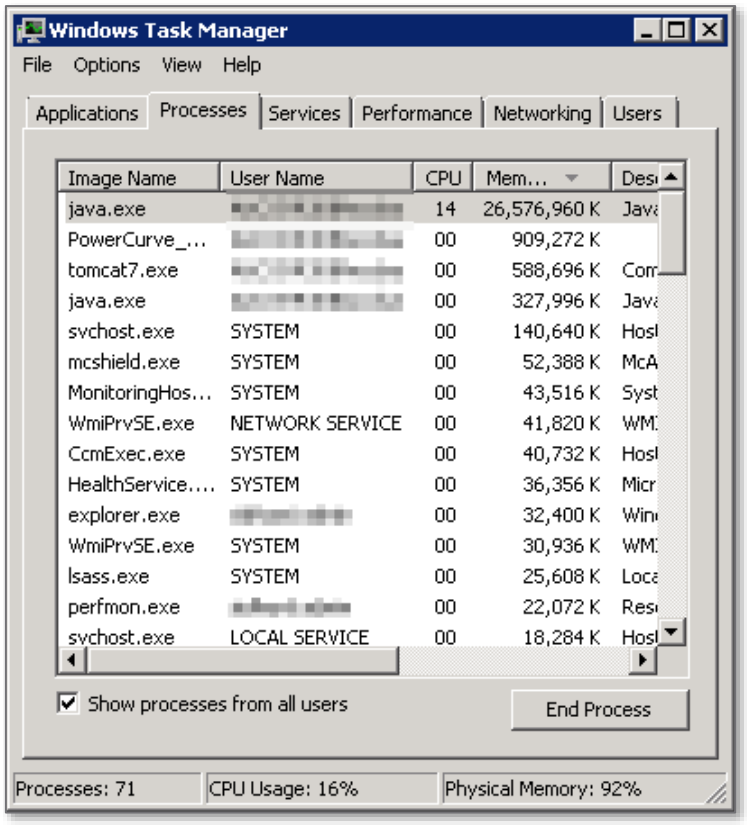


It's showing 7.2 GB! That's nowhere near 0%.

Look at the chart. What do you notice?

It portrays that it has been constantly or actively using that much of memory. In reality, we know it's idle because ESXi is the one doing the actual reading and writing. The other proof that it is idle is Windows actually compressed 1.5 GB of this 7.2 GB.

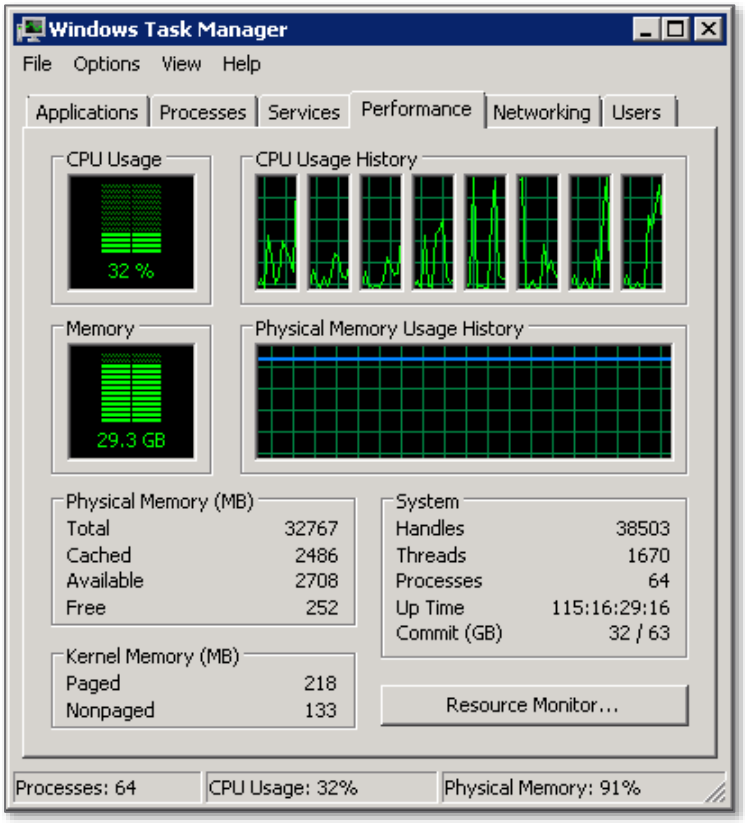
One possible reason why Windows is showing high usage when there is none is applications that manage their own memory. These applications will ask for the memory upfront in 1 contiguous block. You can see in the example below:



You can see that java.exe takes up 26 GB.

JVM (Java Virtual Machine) manages that memory and Windows can't see inside this block. Windows sees the entire block as used and committed, regardless of the application actually uses it or not.

BTW, the above is taken from old blog article of Manny Sidhu. The blog no longer available, hence I could not provide the link.



I hope the above simple experiments shows that you should use the right counter for the right purpose.

VM

Just like the case for CPU, some metrics are for the kernel consumption, not your operations.

Overview

For performance use case, the only counter tracking actual performance is Page-fault Latency.

Counters	Rollups	Units	Description
Overhead reserv...	Average	KB	Host physical memory reserved by ESXi, for its data structures, for running the virtual ma
Page-fault latency	Average	%	Percentage of time the virtual machine spent waiting to swap in or decompress guest ph
Shared	Average	KB	Am
Swap in	Average	KB	Am

Percentage of time the virtual machine spent waiting to swap in or decompress guest physical memory

Next, check for swapping as it's slower than compressed. You get 6 metrics for it

Counters ↑	Rollups	Units	Description
Swap in	Average	KB	Amount of guest physical memory that is swapped in from the swap space since th
Swap in rate	Average	KBps	Rate at which guest physical memory is swapped in from the swap space
Swap out	Average	KB	Amount of guest physical memory that is swapped out from the virtual machine to t
Swap out rate	Average	KBps	Rate at which guest physical memory is swapped out to the swap space
Swap target	Average	KB	Amount of memory that ESXi needs to reclaim by swapping
Swapped	Average	KB	Amount of guest physical memory that is swapped out to the swap space

Next is compressed

Compressed	Average	KB	Guest physical memory pages that have undergone memory compression
Compressed	Latest	KB	Amount of guest physical memory pages compressed by ESXi
Compression rate	Average	KBps	Rate of guest physical memory page compression by ESXi
Compression saved	Latest	KB	Host physical memory, reclaimed from a virtual machine, by memory compression.
Decompression rate	Average	KBps	Rate of guest physical memory decompression

Host Cache should be faster than disk (at least I assume you designed it with faster SSD), so you check it last.

Host cache consumed	Average	KB	Storage space consumed on the host swap cache for storing swapped guest physi
Host cache swap in rate	Average	KBps	Rate at which guest physical memory is swapped in from the host swap cache
Host cache swap out rate	Average	KBps	Rate at which guest physical memory is swapped out to the host swap cache

Lastly, there is the balloon.

Balloon target	Average	KB	Absolute	Desired amount of guest physical memory the balloon driver needs to reclair
Ballooned memory	Average	KB	Absolute	Amount of guest physical memory reclaimed from the virtual machine by the

Wait! Where is the Intel Optane memory metrics?

It does not exist yet, as that's supposed to be transparent to ESXi.

Performance is essentially the only use case you have at VM level. For Capacity, you should look at Guest OS. The VM capacity metrics serve as input to the host capacity and are used in determining the VM memory footprint (e.g. when migrating to another ESXi).

You've got 5 metrics, with consume being the main one.

Consumed	Average	KB	Amount of host physical memory consumed for backing up guest physical memory
Host consumed %	Average	%	Percentage of host physical memory that has been consumed
Entitlement	Average	KB	Amount of host physical memory the virtual machine deserves, as determined by E
Granted	Average	KB	Amount of host physical memory or physical memory that is mapped for a virtual m
Shared	Average	KB	Amount of guest physical memory that is shared within a single virtual machine or

I'm going to add Active next, although I don't see any use case for it. It's an internal counter used by the kernel memory management.

Active	Average	KB	Absolute	Amount of guest physical memory that is being actively read or written by guest. Acti
Active write	Average	KB	Absolute	Amount of guest physical memory that is being actively written by guest. Activeness

Lastly, you get the shared pages and 0 pages.

Overhead active	Average	KB	Estimate of the host physical memory, from Overhead consumed, that is actively re
Overhead consumed	Average	KB	Host physical memory consumed by ESXi data structures for running the virtual ma
Overhead reserved	Average	KB	Host physical memory reserved by ESXi, for its data structures, for running the virtu
Zero pages	Average	KB	Guest physical memory pages whose content is 0x00

Now that we've got the overview, let's dive into the first counter!

"Contention" Metrics

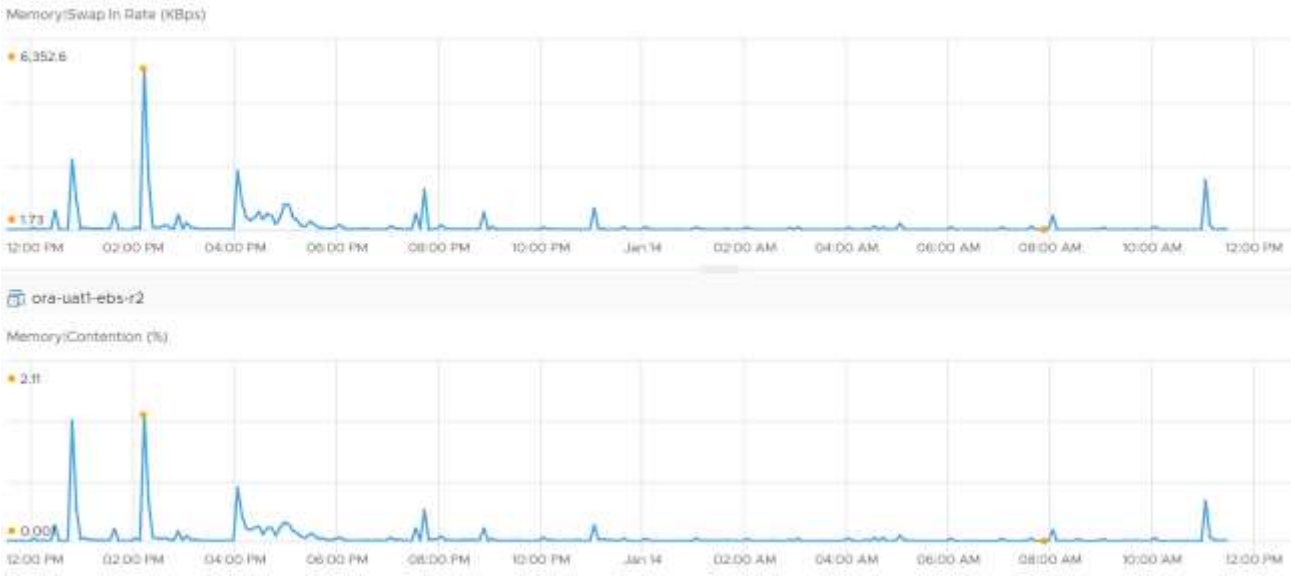
I use quote because the only true contention counter is latency. The second reason is VCF Operations has a metric called Contention[18], which is actually vCenter counter called latency.

Latency

Memory Latency, aka "Page-fault latency" is tracking the amount of time a vCPU spends waiting on the completion of a page fault. Its value is mostly swap wait, with a bit of page decompression / copy-on-write-break. The counter is called %LAT_M in esxtop.

The latency metric is highly corelated with the swap-in rate.

[18] I'm not a big fan of the name contention as it implies the VM is contending for memory, which is not the case.



This is the *only* performance counter for memory. Everything else does not actually measure latency. They measure utilization, because they measure the disk space occupied. None captures the performance, which is how fast that memory page is made available to the CPU.

Consider the hard disk space occupied. A 90% utilization of the space is not slower than 10%. It's a capacity issue, not performance.

If a page is not in the physical DIMM, the VM has to wait longer. It could be in Host Cache, Swapped or Compressed. It will take longer than usual. vSphere tracks this in 2 metrics: CPU Swap Wait and RAM Latency.

- CPU Swap Wait tracks the time for Swapped In.
- RAM Latency tracks the percentage of time VM waiting for Decompressed and Swapped In. The RAM Latency is a superset of CPU Swap Wait as it caters for more scenarios where CPU has to wait. VCF Operations VM Memory Contention metric maps to this.

Latency is >1000x lower in memory compared to disk, as it's CPU basically next to the CPU on the motherboard. Time taken to access memory on the DIMM bank is only around 200 nanoseconds. Windows/Linux does not track memory latency. The closest counter is perhaps page fault. The question is does page fault includes prefetch? If you know, let me know please.

This counter has the effect of reduced value of the Compressed metric and/or Swapped metric, and increased the value of Consumed & Granted.

Latency does not include balloon as that's a different context. In addition, the hypervisor is not aware of the Guest OS internal activity.

Actions you can do to address high value:

- Store vswp file on higher throughput, lower latency storage, such as using Host Swap Cache.
- Increase memory shares and/or reservation to decrease amount of swapping. If the VM belongs to a resource pool, ensure the resource pool has sufficient for all its VMs.
- Reduce assigned memory. By rightsizing, you reduce the size of memory reclamation, hence minimizing the risk.
- Remove VM Limit.

- Unswap the swapped memory. You cannot do this via API, but you can issue the command manually. Review this article by Duncan Epping and Valentin Bondzio.
- If possible, reboot the VM as part of regular maintenance. This will eliminate the swap file, hence avoiding future, unexpected swap wait on that swapped page. Note this does guarantee the same page to be swapped out again.

Best Practice

In an environment where you do not do memory overcommit and place limit, the chance of hitting memory contention will be basically 0. You can plot the highest VM Memory Contention counter in all clusters and you will basically see a flat line. That would be a lot of line charts, so I'm using a pie chart to analyze 2441 VM in the last 4 months. For each VM, I took the highest value in the last 4 months. Only 13 VM had its worst VM Contention above 1%.



Balloon

Balloon is an application (kernel driver to be precise) running inside the Guest OS, but it can take instruction from the kernel to inflate/deflate.

When it receives an instruction to inflate, it asks the Guest OS to allocate memory to it. This memory in the Guest OS is not backed up by physical memory in ESXi, hence it is available for other VMs. When ESXi is no longer under memory pressure, it will notify the Balloon to release its requested page inside Guest OS. This is a proactive mechanism to reduce the chance of the Guest OS doing paging. Balloon will release the page inside the Guest OS. The Balloon counter for the VM will come down to 0.

It is the Guest OS that initiates memory reallocation. Therefore, it is possible to have a balloon target value of 0 and present balloon value greater than 0. The counter **Balloon Target** tracks this target, so if you see a nonzero value in this counter, it means that the hypervisor has asked this VM to give back memory via the VM balloon driver.

Just because Balloon asks for 1 GB of RAM, does not mean ESXi gets 1 GB of RAM to be freed. It can be less if there is TPS.

Guest OS will start allocating from the Free Pages. If insufficient, it will take from Cache, then Modified, then In Use.

To use ballooning, Guest OS must be configured with sufficient swap space.

How much will be asked depends on Idle Memory Tax. I do not recommend playing with this setting.

Performance Impact

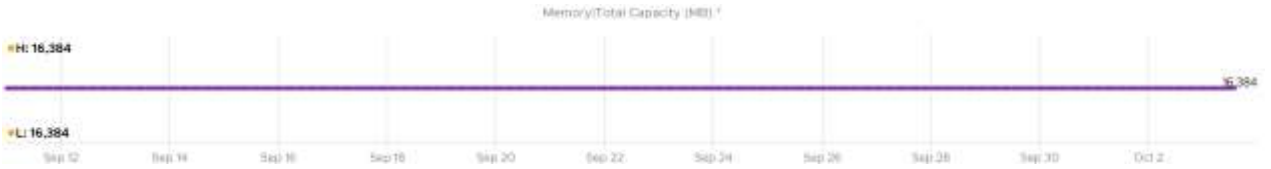
Balloon by itself does *not* cause performance problem. What will cause performance is when the ballooned page is requested by Windows or Linux. The following shows a VM that is heavily ballooned as limit was imposed on it. Notice the actual performance happens rarely.



The higher the value is for balloon, swapped, and compressed, the higher the chance of a performance hit happening in the future if the data is requested. The severity of the impact depends on the VM memory shares, reservation, and limit. It also depends upon the size of the VM's configured RAM. A 10-MB ballooning will likely have more impact on a VM with 4 GB of RAM than on one with 512 GB.

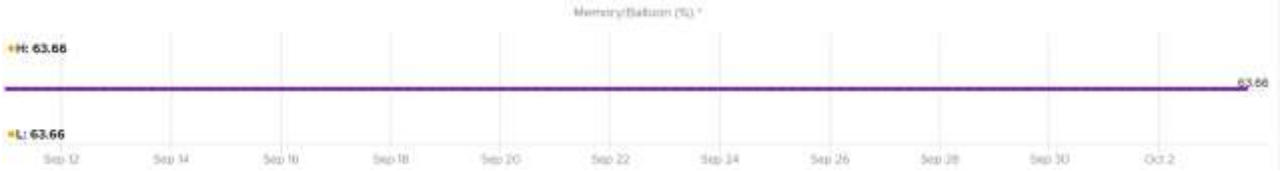
How high?

Let's take a VM and plot its value over time. The VM is configured with 16 GB memory. As you can see, the value in the last 4 weeks is a constant 16 GB.



The line is a perfect flat. Both the Highest value and Lowest value show 16,384 MB.

The VM was heavily ballooned. 63.66% of its memory was reclaimed. That's a whopping 10,430 MB!



Notice something strange.

The Ballooned did not change at all for 4 weeks.

That likely means the Guest OS is not active. It never needs any of those 10+ GB that was ballooned out.

So Guest was playing with the remaining 6 GB. It never page in those pages.

So what do you expect if we plot Granted + Swapped + Compressed?

You got it. A flat line.



Capacity Impact

Balloon is a memory request from ESXi. So it's not part of the application. It should not be included in the Guest OS sizing, hence it's not part of reclamation.

Balloon impacts the accuracy of Guest OS sizing. However, there is no way to measure it.

When Balloon driver asks for pages, Guest OS will allocate, resulting in In Use to go up. This is because the balloon driver is treated like any other processes.

If the balloon driver page comes from Free, then we need to deduct it from In Use.

If the page comes from In Use, then it gets tricky as the value of In Use does not change. The Guest OS pages out, so we need to add Page Out or Cache.

Swap + Compress

Swap and Compress go hand in hand as the block that cannot be compressed go into swapped. The size of a memory page or block is only 4 KB. When the compression does not result in savings below 2 KB, then it makes no sense to compressed and the page is moved to swapped file.

There are 2 levels of compression (4:1 and 2:1), so a 4 KB page may end up as 1 KB or 2 KB. The reason there is no 3:1 compression

If the compression result is less than that, the page will be swapped instead as that's a cheaper operation. So it's completely possible to have 0 swapped as all the pages were compressed instead.

Large page (2 MB) is not compressed directly. Instead, it will be split into 4 KB page first.

The following screenshot shows how the 2 counters moved in tandem.



The above are the main metrics that you should track. vSphere provides additional visibility as swap and compress are complex process.

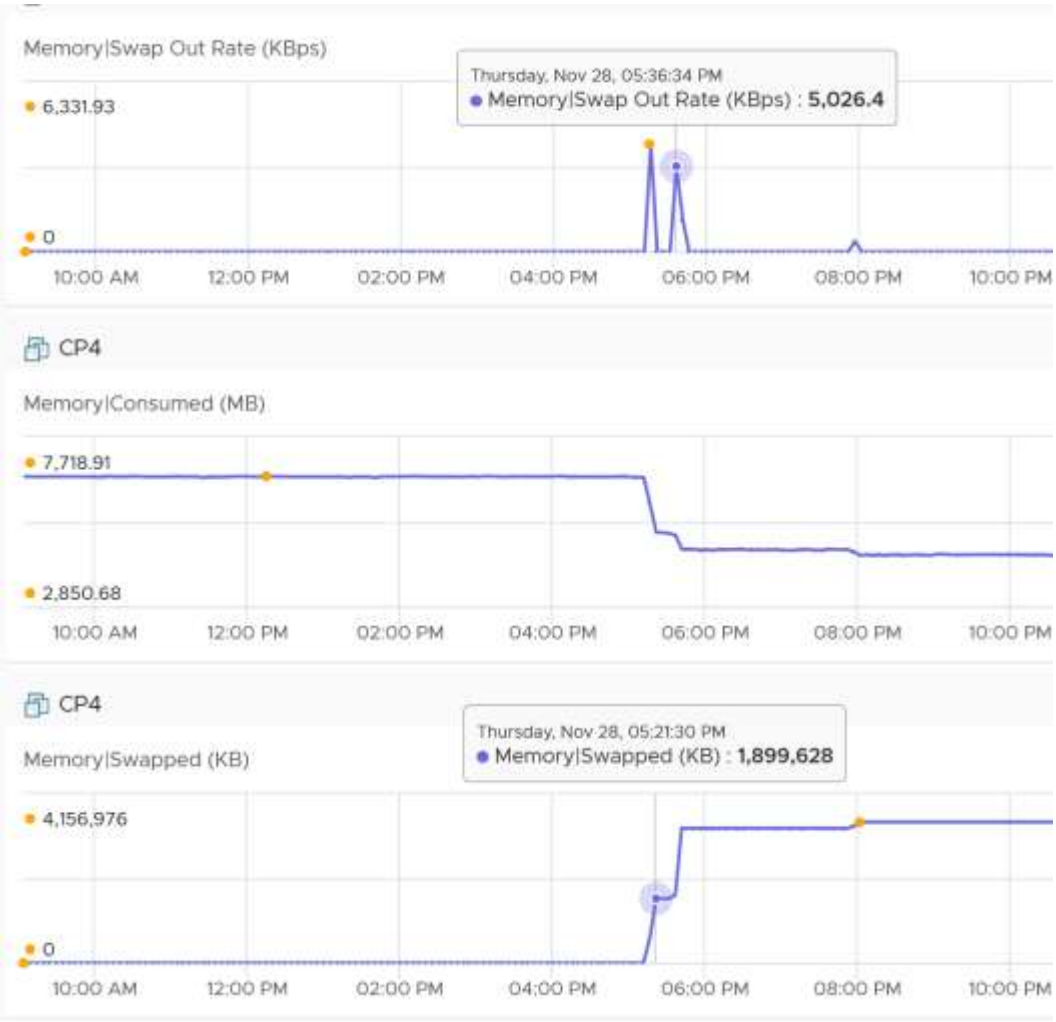
Transition

Read the following carefully, as there will be a quiz after this.

Input	This is the number of pages that will be subjected to compression and swapping.

	It's not the number of pages that was processed already. The metric is called Swap Target. Think of it as Compress-Swap Target as the page that cannot be compressed is swapped.
Process	You need metrics to track the progress as it's happening. This complements the result as it covers how much memory is compressed or swapped at any given period. A 10 MB compressed in 1 second is different to 10 KB compressed over 1000 seconds. Both results in the same amount, but the problem is different. One is an acute but short fever; the other is low grade but persistent fever. You don't want neither, but good to know what exactly you're dealing with. This is the rate of • compression • decompression • swapping from DIMM into the swap file. • swapping from the swap file into the DIMM The In and Out can happen at the same, as they typically involve different pages. You can have 10 MB being decompressed and another 10 MB being compressed. Your swap file size is unchanged but the content has changed. This is why Swapped is not Swapped In – Swapped Out. Swap-out does not mean there is contention. If you are lucky, the page being swapped is not required. However, swap out means the ESXi was under memory pressure or the VM hit a limit. Swap-in means there is contention. The page was called for, hence it was brought in. Plot them together and you will see a high correlation. Swap-in doesn't happen because there's memory pressure on the host. Swap-in just means there was memory pressure in the past and now the guest OS wants some of that data.
Output	You need a metric to track the result and the savings. For swap, 3 counters are provided • Swapped • Swapped Out • Swapped In For compress, the compressed-in and compressed-out are not provided. You do get the savings from compression.

Example



Swapped Out Rate was 6331.93 + 5026.4 + 1882.13 + 615.86. This gives a total of 13856. Since it's sustained for 300 seconds, we multiply by 300 and then divide by 1024 to get the total of 4059 MB being swapped out. This is pretty close to the 4157 MB of Swapped.

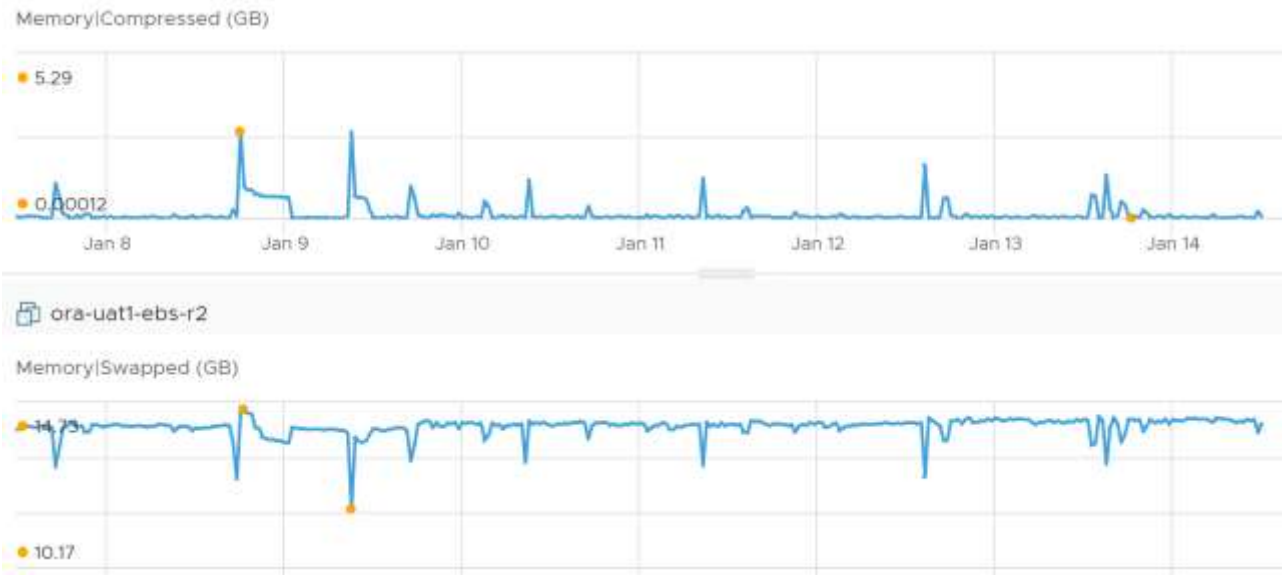
There is no Swap In. The rate metric was showing 0 during the entire period.

Consumed drops by 4868 MB (7719 – 2851). There were other factors impacting it.

Quiz

Explain why swap and compress move in opposite direction in the following chart.

This VM is configured with 64 GB of memory. So it experienced high amount of swapping and compressed in the last 7 days. It peaked at 20 GB, which is really bad.



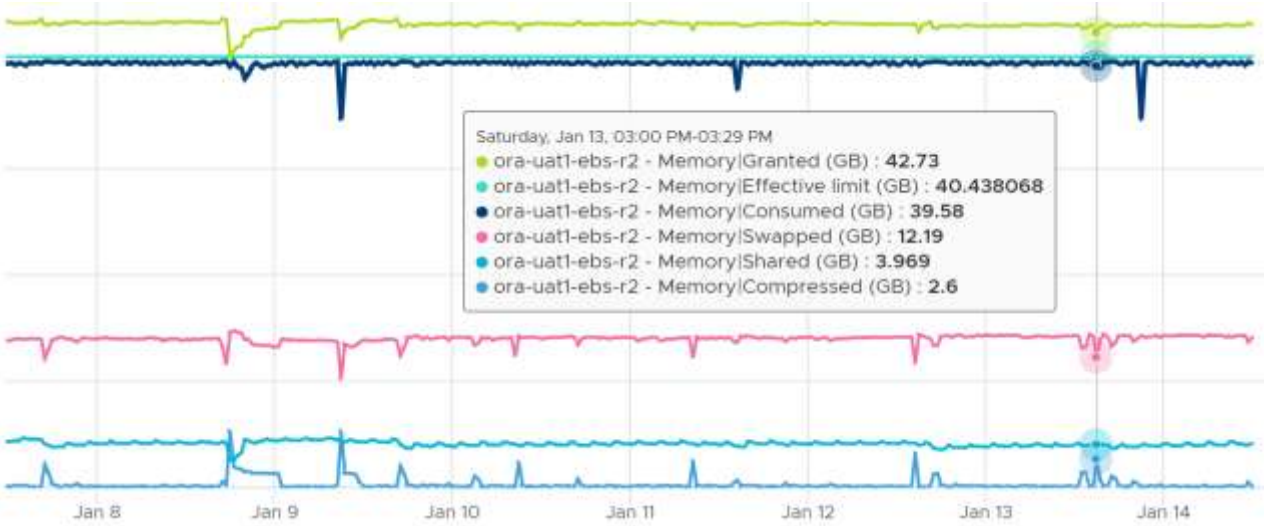
Let's plot consumed and granted. What's your conclusion?



Why is consumed basically flat for 7 days?

Granted looks rather normal as it hovers above consumed, with some movement.

If you guess Limit, you're right! Let's plot all the counters together now.



Because of Limit, Consumed can't go above it. The tiny gap is the hypervisor overhead for this VM.

Now, because Guest OS is actively using pages, this results in movement in both directions. Some pages are brought in from swapped files. To make space, other pages are subjected to compression & swap decision.

Balloon

Compressed and Swapped are different from ballooning, as the hypervisor has no knowledge of the free memory inside the Guest OS. It will *randomly* compress or swap. As a result, any value in this counter indicates that the host is unable to satisfy the VM memory requirement. This can have potential impact on performance.

It is possible to have balloon showing a zero value while compressed or swapped are showing nonzero values—even though in the order of ESXi memory reclamation techniques, ballooning occurs before compression. This indicates that the VM did have memory pressure in the past that caused ballooning, compression, and swapping then, but it no longer has the memory pressure. These events could have happened at different time. Data that was compressed or swapped out is not retrieved unless requested, because doing so takes CPU cycles. The balloon driver, on the other hand, will be proactively deflated when memory pressure is relieved.

Limit

Does limit result in Balloon?

The answer is no. Why not?

Go back to the 4 layers of memory. They are at different level on memory management. Limit results in swapped or compressed.

Let's take an example with a VM that is configured with 16 GB RAM. This is a My SQL database running on RHEL. You can see in the last 7 days, it's using around 13.4 GB and increasing to 13.6 GB.



It's given a bad limit of 2 GB.

In the last 7 days, we can see the limit is a perfectly flat line. It's 2.12 GB as it includes the overhead value.



The VM, or rather the Guest OS, did ask for more. You can see the demand by looking at the Granted or Compressed or Swapped metrics. I'm only showing Granted here:



Because of the limit, the Consumed counter did not past the 2 GB. It's constantly hovering near it as the VM is asking more than that.



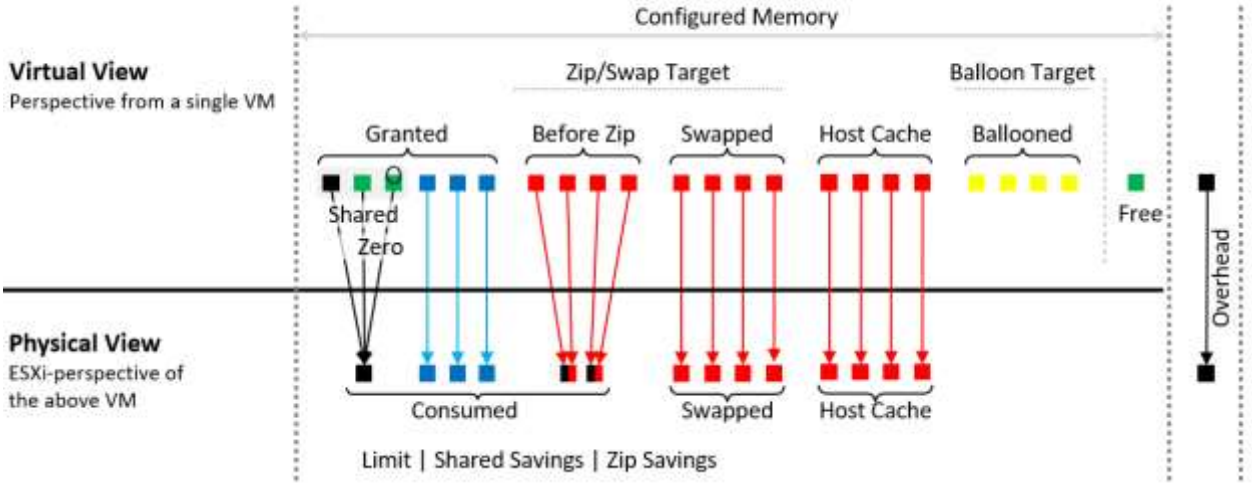
What do you expect to see the Balloon value?

If Balloon has something to do with it, it would not stay a perfectly flat line.

But this is what you got. A perfectly flat line, proving Limit had nothing to do with Balloon.



Consumption Metrics



Configured	What the Guest OS can see is what is configured by vSphere. Guest OS can't see the hypervisor memory overhead.
Granted	This is page granted to the VM, minus Ballooned, Swapped, Compressed, and swapped to host cache. Take note of the process of transitioning during swap. During this process, the value of granted includes them
Shared	The number of memory pages that are pointing to the same underlying block. Many of these pages are likely the 0000 page.
Compressed Swapped	These 2 are mutually exclusive and go together. What can't be compressed will be swapped. Compressed is preferred as unzipping memory from DRAM is faster than bringing it from SSD disk. Notice Windows even compressed its In Use pages. Compressed is the output (result of compression), not the input (memory subjected to compressed as that contains swapped also)
Entitlement	= Granted + Overhead. Overhead is mostly negligible, as it's just storing metadata or index information required by virtualization, such as the shadow page tables. Overhead value goes up as you configure more vCPUs and memory. My guess is this excludes limit as you're not entitled beyond the limit.
Ballooned	The page was reclaimed by the balloon driver. It has not been asked back by the Guest OS, hence it's just seating there collecting pixie dust. I colored the box yellow as that's not a good (green) situation. The bigger the balloon size, the higher the chance a page will be required in the future.
Overhead	Virtualization overhead is operationally negligible.
"Free"	The VM never touches or uses the page since it's powered on. After a while, as Guest OS starts paging in and out, this value will grow to near 0.

Granted

Granted and Consumed are not similar. The former looks at from consumer layer, while the later looks from provider layer.

The formula is

```
Granted = normal page + shared page + zero page + being swapped-in + tiered page
```

Normal page is single pages being backed by physical DIMM. This metric is not exposed on its own.

Being swapped in means a staging stage between unmapped and normal. Once the transition is complete, it becomes a normal page.

Tiered is the memory tiering. Unlike swapped, memory tiering is “in-line” hence it is considered as part of granted.

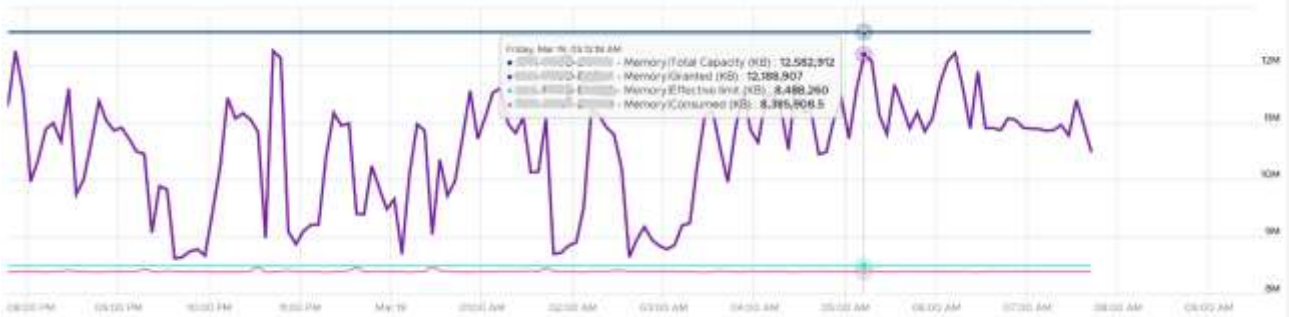
Over time, a guest OS touches a large fraction of its address space through normal operations — OS initialization, memory zeroing, page table setup, file caching, and application allocations. As pages are touched, VMkernel creates physical page frame mappings. This what Granted metric is. Many of these mapped pages end up containing zeroes or are shared across VMs via transparent page sharing (TPS), consuming negligible physical RAM — but they still count toward Granted.

If the parent ESXi is not under physical memory pressure, VMkernel does not aggressively reclaim these zero/shared pages because they are cheap to maintain. Generaly speaking, for a VM that has been running for some time, its granted counter gradually increases toward the configured size, as that’s the upper limit.

Granted does not care about page savings at physical layer its vantage point is the VM, not ESXi. The following shows Granted is perfectly flat. Consumed goes down as the amount of shared page goes up.



The same vantage point reason is why Limit impacts Consumed, but not Granted. The following is VM is a Windows 2016 server, configured with 12 GB of RAM, but was limited to 8 GB. Limit is shown as the flat line in cyan near the bottom, hovering just above the pink line).



The purple line jumping up and down is Granted. Granted ignores the limit completely and runs way above it.

Notice Consumed (KB) is consistently below Limit. Granted does not exceed 12 GB as it does not exceed configured.

Compressed + Swapped

Granted does not include Compressed + Swapped because the page is no longer directly accessible without some extra processing.

The following shows Granted move up while the other 2 metrics went down.



Summing up the above resulted in a delta of 0.7 MB.

Ballooned

Balloon driver removes page from the granted list. Granted does not include ballooned as the page is not functionally used. Technically, Guest OS memory counters include it so don't forget to exclude it when working out the Guest OS utilization.



Shared

Shared counts the amount subjected to sharing. It does not count the actual savings post sharing.

There are 2 types of sharing that can happen to a page:

Intra-VM sharing	sharing within the same VM. By default, each page is 4 KB. If Guest OS uses the Large Page, then it's 2 MB. The chance of sharing in 4 KB page is logically much higher than in 2 MB.
Inter-VM sharing	Due to security concern, this is by default disabled in vSphere
	For accounting purpose, the Shared page is counted in full for each VM. This means if you sum the number from all VMs you're going to get inflated value at the ESXi level

Example:

- VM 001 has 1 GB private.
- The 100 MB is the amount that is being shared internally within the VM. If not shared, they would consume 100 MB.
- There is additional 10 MB that is subjected to sharing with other VMs. It could be shared with 1 VM or many VM; it does not matter as far this VM 001 concern. The Shared counter merely counts that this 10 MB is being shared.

Using the above, the shared page for VM 001 is 110 MB.

Savings

The calculation has to be done on each block that points to the same destination page. For example, if there are 10 pages pointing to the same physical DIMM, then the savings is 9 pages worth of memory.

The above process is repeated for all the shared page.

The result is some savings. But how much savings?

Here is a summary from 7500 VMs.

Name	Memory|Granted	Memory|Shared	Memory|Consumed
vPodRouter-uVRr	223.25 MB	31.18 MB	200.45 MB
vPodRouter-vnBF	223.83 MB	18.71 MB	207.88 MB
vPodRouter-WXh1	224 MB	26.73 MB	203.43 MB
vPodRouter-aMEg	225.86 MB	28.04 MB	203.06 MB
Average	**14.09 GB**	**4.07 GB**	**10.86 GB**
Sum	**103.26 TB**	**29.83 TB**	**79.56 TB**

1 - 50 of 7505 items ‹ 1 2 3 4 5 … 151 ›

Review the last row. What are the total savings?

Granted – Consumed = (103.26 – 79.56) = 23.7 TB

Savings = 29.83 TB – 23.7 TB = 6.13 TB.

That's around 6% saving.

Profile your own environment. What savings do you get?

Savings does not include compressed as that is not true saving. Zipped results in performance impact, while saving should not.

Zero

A commonly shared page is certainly the zero page. A common technique to initialize space is to simply write 0.

The following screenshot shows the 2 moved in tandem over several days.



Consumed

```
Consumed = Granted – Savings.
```

Consumes tracks the ESXi Memory mapped to the VM. ESXi assigns large pages (2 MB) to VM whenever possible; it does this even if the Guest OS doesn't request them. The use of large pages can significantly reduce TLB misses, improving the performance of most workloads, especially those with large active memory working sets. The drawback is VM consumes more memory than what Guest OS does.

Consumed does not include overhead memory, although this number is practically negligible. Think of the word consume as delivering benefit to the VM. Since overhead is transparent to Guest OS, it's excluded.

Consumed does not include swapped memory, for the same reason above. The swapped pages are not readily available for use. As for compressed, I'm unsure if it includes the portion that is the DIMM. It does not include the portion that was subjected to compression. For example, a 4 KB page was compressed to 1 KB. The 0.75 KB is not in Consumed as it's no longer in the DIMM.

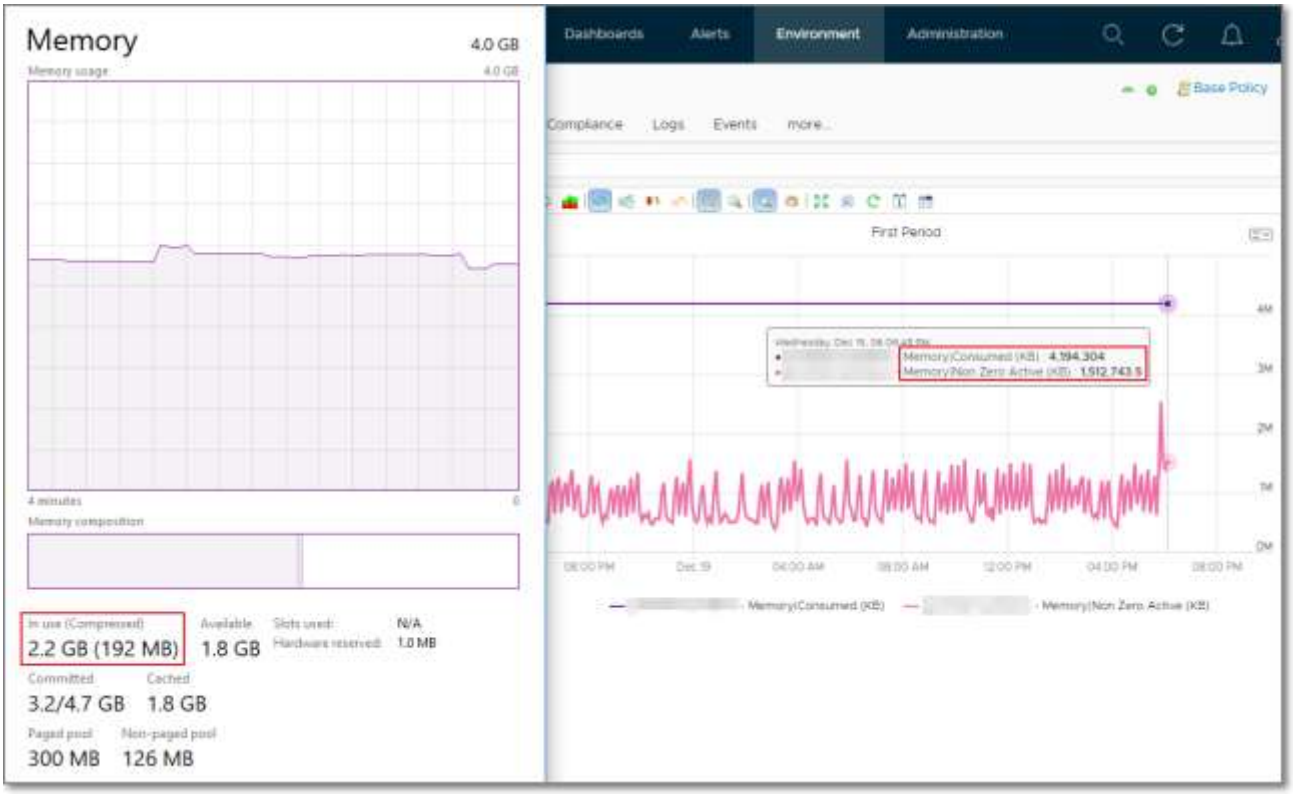
Consumed includes memory that might be reserved.

Do not use the metric Guest \ Memory Usage. That's the same as Consumed.

Guest OS

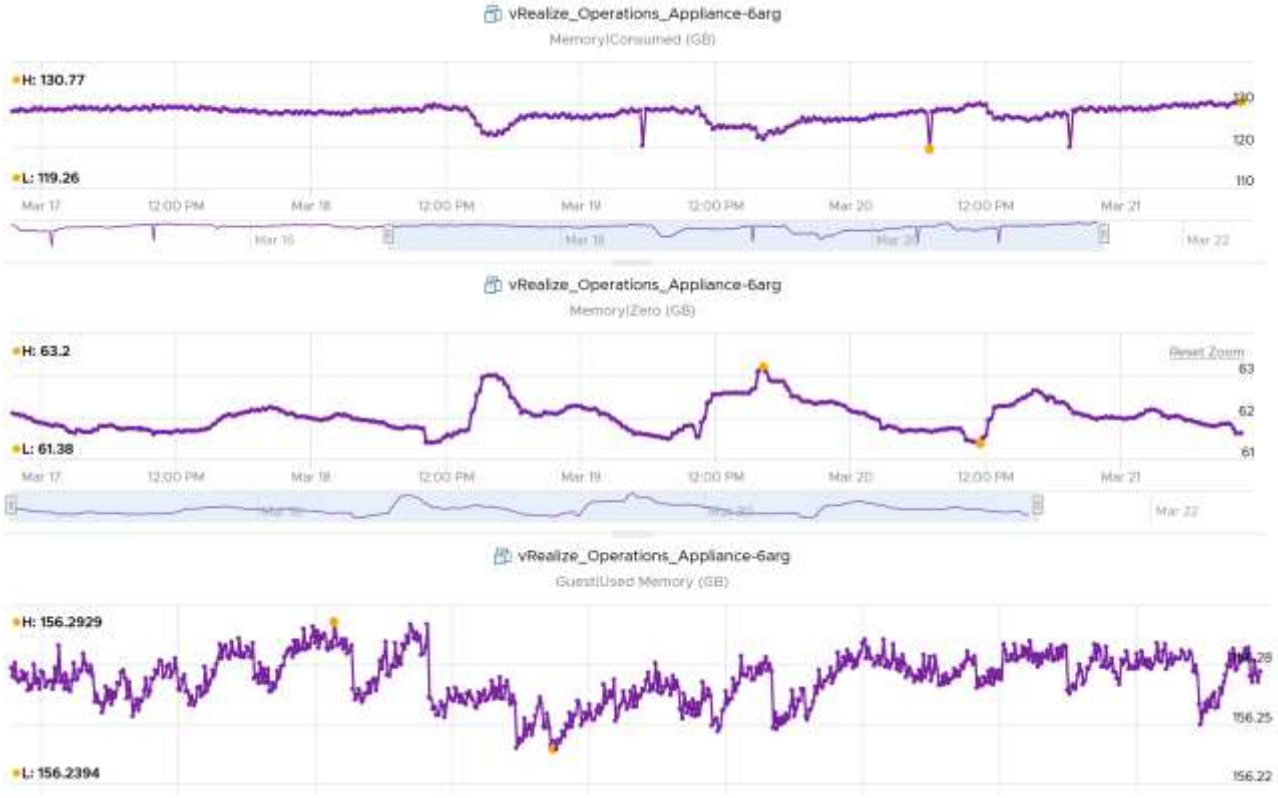
When a Guest OS frees up a memory page, it normally just updates its list of free memory, it does not actually update the content. This list is not exposed to the hypervisor, and so the physical page remains claimed by the VM. This is why the Consumed is higher than the Guest OS In Use, and it remains high when the Active counter has long dropped.

Consumed and Guest OS In Use are not related, as they are independently managed. Here is a screenshot comparing Windows 10 Task Manager memory metrics with VCF Operations Memory \ Non Zero Active (KB) and Memory \ Consumed (KB). As you can see, none of the metrics match.



When you see Consumed is lower than Guest OS Used, check if there are plenty of shared pages. Consumed does not include shared page.

The following screenshot shows Guest OS Used consistently higher. It's also constant, around 156 GB throughout. Consumed was relatively more volatile, but never exceed 131 GB. The reason for it is Shared. Notice the value of page with all 0 is high, around 61 – 63 GB.



Ballooned

This 64-bit CentOS VM runs My SQL and is configured with 8 GB of RAM.

Linux was heavily ballooned out (default limit is around 63%). Why is that so?



The answer for this VM is we set a limit to 2 GB. As a result, Consumed could not exceed 2 GB. Since the VM needed more, it experienced heavy ballooning.



Did you notice the common deep in Balloon and Consumed?

Can you explain them?

Balloon dropped by 0.46 GB then went back to its limit again. This indicated Guest OS was active.

Consumed went down from 2.09 GB to 1.6 GB, and then slowly going back up. Why did it suddenly consume 0.4 GB less in the span of 20 minutes? Both the configured limit and the runtime limit did not change. They were constant at 2 GB. This makes sense, else the Consumed would not be able to slowly go up again.



There must be activity by the VM and pages were compressed to make room for the newly requested pages. The Non Zero Active counter shows that there are activities.



The pages that are not used must be compressed or swapped. The Swapped value is negligible, but the Compressed metric shows the matching spike.



So far so good. Windows or Linux were active (2.4 GB in 5 minute at the highest point, but some pages were probably part of Consumed). Since Consumed was at 100%, some pages were moved out to accommodate new pages. The compression resulted in 0.6 GB, hence the uncompressed amount was in between 2x and 4x.

Consumed dropped by 0.4 GB as that's the gap between what was added (new pages) and what was removed (existing pages).

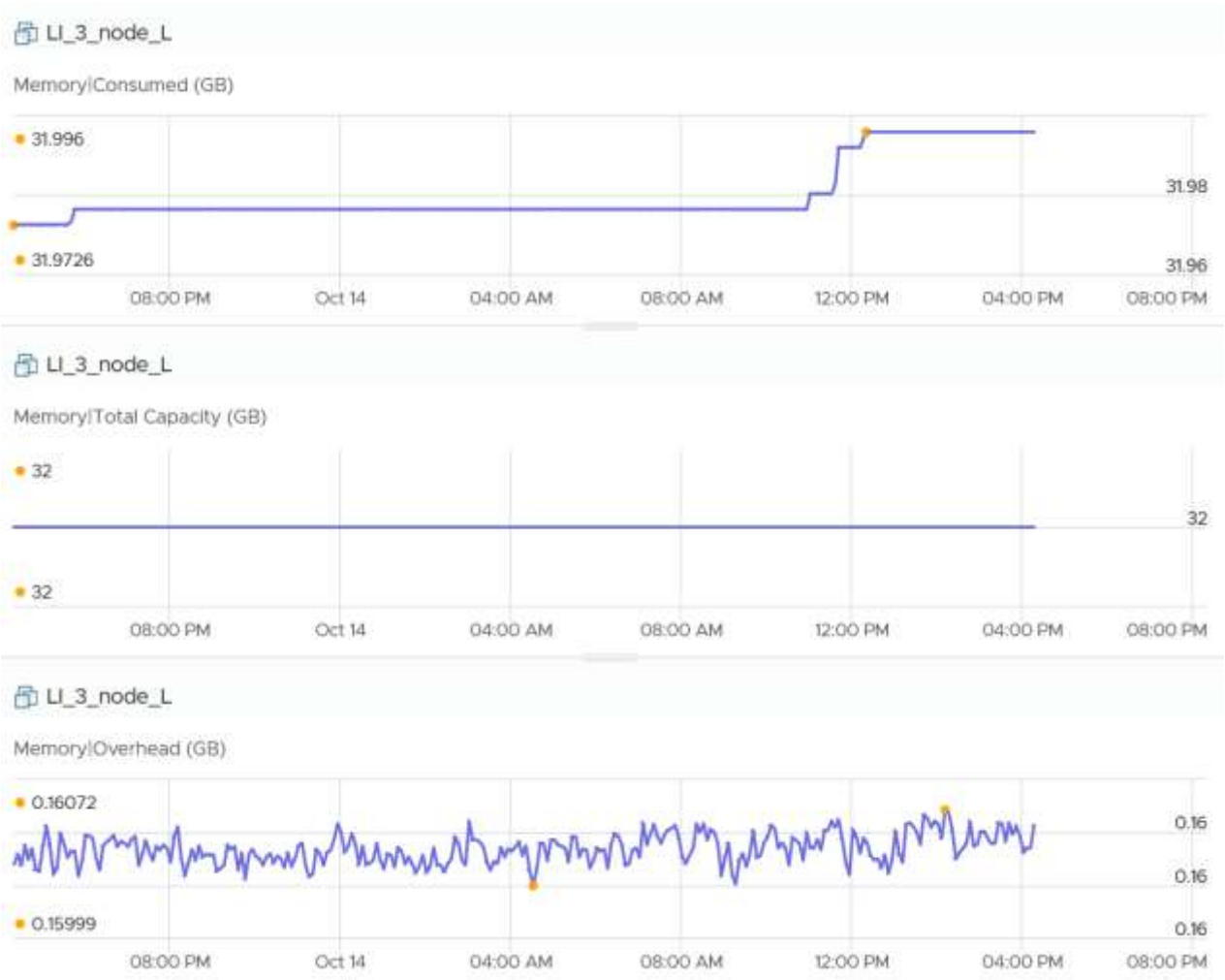
Limit

Consumed is affected by Limit. The following is a VM configured with 8 GB RAM but was limited to 2 GB.



Total

Consumed may reach but not exceed the configured memory. Both total and consumed do not include the virtualization overhead memory.



Active

This is a widely misunderstood counter. ESXi calls this Touch as it better represents the purpose of the metric. Note that vCenter still calls it Active, so I will call it Active.

This counter is often used to determine the VM utilization, which is not what it was designed for. To know why, we need to go back to fundamental. Let's look at the word **active**. It is an English word that needs to be quantified before we can use it as metric. There are 2 dimensions to consider before we apply it:

- Definition of active. In RAM context, this means read or write activity. This is similar to disk IOPS. The more read/sec or write/sec to *a page*, the more active that page is. Note that the same page can be read/written to many times in a second. Because a page may be accessed multiple times, the actual active pages could be lower. Example: a VM do 100 reads and 100 writes on its memory. However, 50 of the writes are on the page that were read. In addition, there are 10 pages that were read multiple times. Because of these 2 factors, the total active pages are far fewer than 300 pages. If the page is average 4 KB, then the total active is way less than 1200 KB.
- Active is time bound. Last week is certainly not active. Is 300 seconds ago active? What exactly, is recent? 1 second can be defended as a good definition of recent. Windows shows memory utilization in 1 second interval. IOPS is always measured per second, hence the name IOPS. So I think 1 second seems like a good definition of recent.

Applying the above understanding, the active counter is actually a rate, not a space. However, the counter reported by vCenter is in KB, not KB/s.

To translate from KB/s to KB, we need to aggregate based on the sampling period. Assuming ESXi samples every 2 seconds, vCenter will have 10 sampling in its 20 second reporting period. The 10 samplings can be sampling the same identical pages, or completely different ones. So in the 20 seconds period, the active memory can be as small as 1 sampling, or as large as 10 samplings.

Examples:

- First 2 seconds: 100 MB Active
- Next 2 seconds: 150 MB Active

In the above 4 seconds, the active page ranges from 150 MB to 250 MB.

Each sampling is done *independently*, meaning you could be sampling the same block again. But the value is then averaged it with previous samples. Because sampling and averaging takes time, Active won't be exact, but becomes more accurate over time to approximate the amount of active memory for the VM. This is why there is actually a longer version of Active, which you will see in esxtop (it is not available in vSphere Client).

VM Active is typically different from Guest OS working set estimate. Sometimes the difference may be big, because Guest OS and the kernel use different working set estimate algorithm. Also, VM has a different view of active memory, due to ballooning and host swapping. Logically, ballooned memory is considered inactive, so, it is excluded from the sampling. Active is unaware of locked and large pages.

Reference: Active Memory by Mark Achtemichuk.

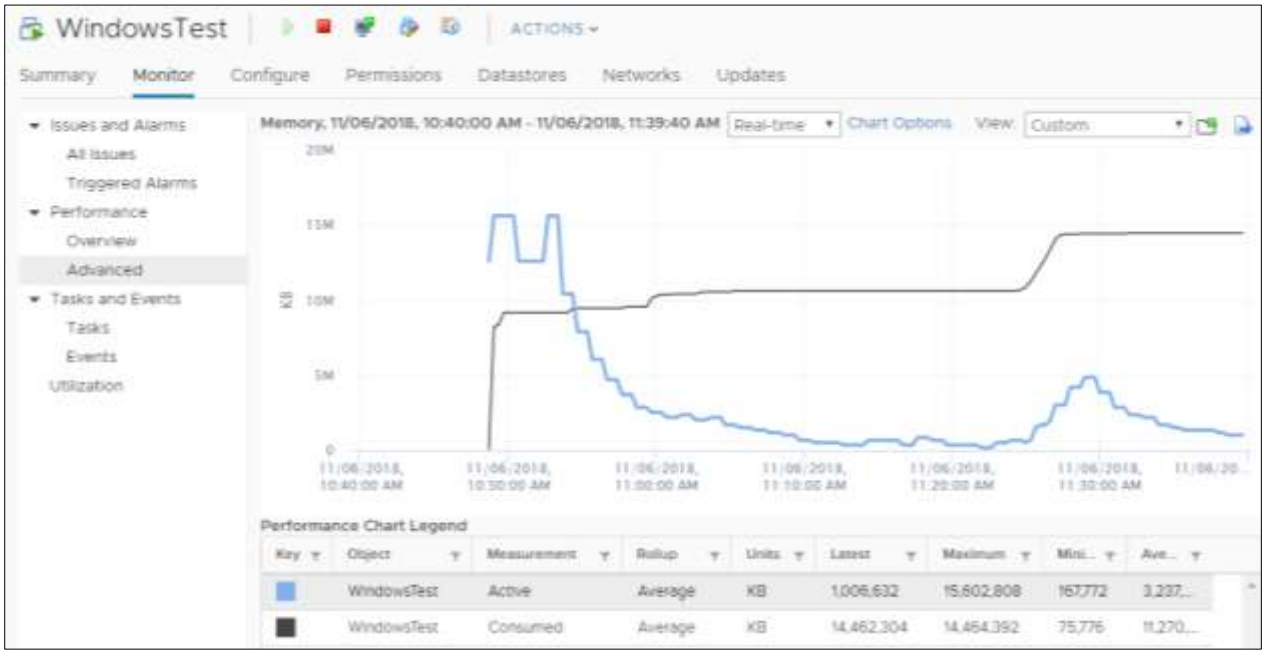
Consumed vs Active

I hope the explanation on Consumed and Active convince you that they serve different purpose. They are not calculated in a similar manner, and are not simply differ based on aggressive vs conservative.

Both Active and Consumed are not suitable for sizing the Guest OS. They are VM level metrics, with little correlation to the Guest OS memory usage.

Example 1

The following test shows Active going down while Consumed going up.



Example 2

If you plot a VCF Operations VM in vCenter real-time performance chart, you will see 12 peaks in that one-hour line chart. The reason is VCF Operations pulls, process, and writes data every 5-minutes. The chart for CPU, disk and network will sport the same pattern. This is expected.

But if you plot the memory metrics, be it total active, active write or consumed, you will not see the 12 peaks. This is what I got instead.



Consume is completely flat and high. Active (read and write) and Active Write (write only) is much lower but again the 12 peaks are not shown.

Can you figure it out?

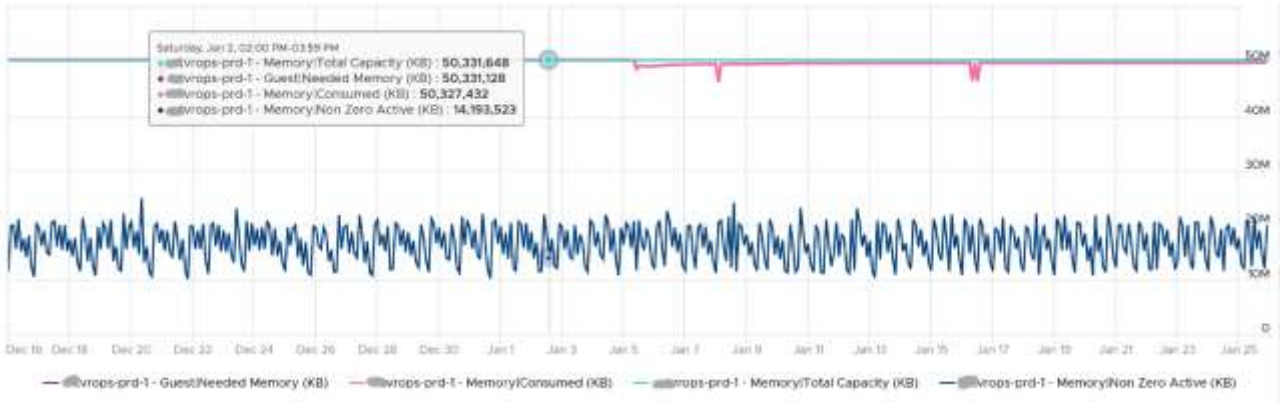
My guess is the sampling size. That's just a guess, so if you have a better answer let me know!

Now let's go to VCF Operations. In VCF Operations, this metric is called Memory \ Non Zero Active (KB).

vCenter reports in 20 seconds interval. VCF Operations takes 15 of these data and average them into a 300-second average. In the 300 second period, the same page can be read and written multiple times. Hence the active counter over reports the actual count.

Quiz: now that you know Active over reports, why is it lower than Consumed? Why is it lower than Guest OS metrics?

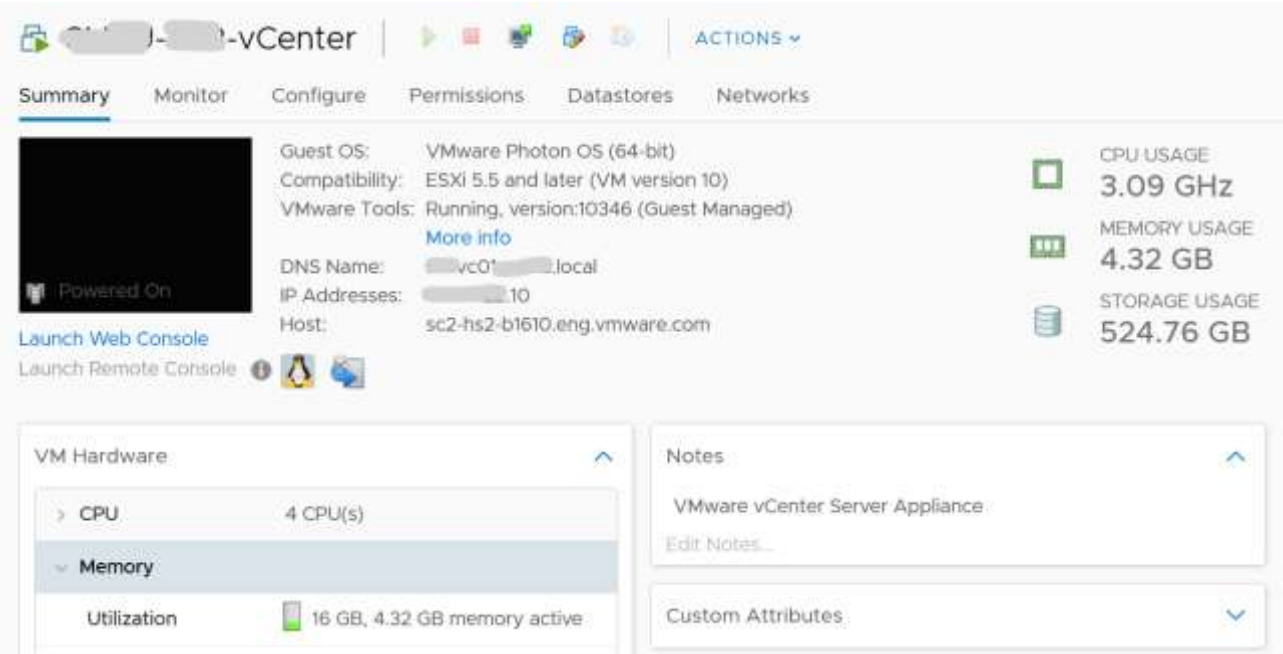
Active is lower than both metrics because these 2 metrics do not actually measure how actively the page is used. They are measuring the disk space used, so it contains a lot of inactive pages. You can see it in the general pattern of Consume and Guest OS used metrics. The following is VCF Operations appliance VM. Notice how stable the metrics are, even over millions of seconds.



Usage (%)

Usage metric in vCenter differs to Usage metric in VCF Operations.

What you see on the vCenter UI is Active, not Consumed.



Mapping to Active makes more sense as Consumed contains inactive pages. As covered earlier, neither Active nor Consumed actually measures the Guest OS memory. This is why VCF Operations maps Usage to Guest OS. The formula is:

```
VM Memory Usage (%) = Guest OS Needed Memory (KB) / VM Memory Total Capacity (KB) * 100
```

The following shows what Usage (%) = Guest OS Needed Memory over configured memory. The VM has 1 GB of memory, so 757 MB / 1024 = 74%.

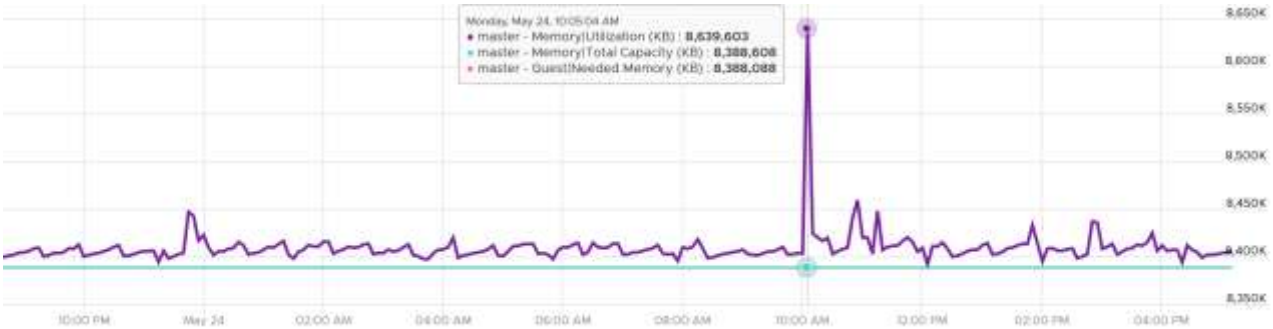
Take note that there can be situation where Guest OS metrics do not make it to VCF Operations. In that case, Usage (%) falls back to Active (notice the value dropped to 6.99%) whereas Workload (%) falls back to Consumed (notice the value jump to 98.95%).



Utilization

Utilization (KB) = Guest Needed Memory (KB) + (Guest Page In Rate per second * Guest Page Size (KB)) + Memory Total Capacity (KB) – Guest Physically Usable Memory (KB).

Because of the formula, the value can exceed 100%. The following is an example:



It's possible that VCF Operations shows high value when Windows or Linux does not. Here are some reasons:

- Guest metrics from VMware Tools are not collecting. The value falls back to Consumed (KB). Ensure your collection is reliable, else the values you get over time contains mixed source. If their values aren't similar, the counter values will be fluctuating wildly.
- Guest Physically Usable Memory (KB) is less than your configured memory. I've seen in one case where it's showing 58 GB whereas the VM is configured with 80 GB. My first guess is the type of OS licensing. However, according to this, it should be 64 GB not 58 GB.
- Low utilization. We add 5% of Total, not Used. A 128 GB VM will show 6.4 GB extra usage.
- Excessive paging. We consider this. The tricky part is excessive is relative.
- We include Available in Linux and cache in Windows, as we want to be conservative.

Demand

Can you spot a major counter that exists for CPU, but not for RAM?

That's right. It's Demand. There is no memory demand counter in vCenter UI.

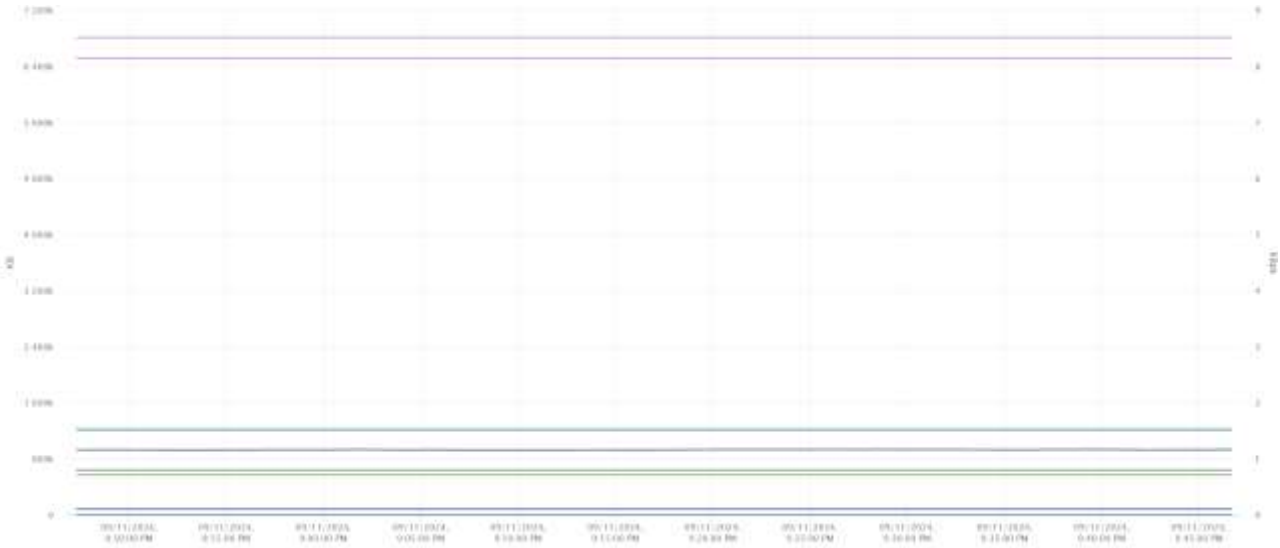
To figure out demand, we need to figure out unmet demand, as demand is simply unmet demand + used (which is met demand). Since the context here is VM, and not Guest OS, then unmet demand includes only VM level metrics. The metrics are ballooned + swapped + compressed.

Do you agree with the above?

If we are being strict with the unmet demand definition, then only the memory attributed to contention should be considered unmet demand. That means balloon, swap, or compressed memory can't be considered unmet demand. Swap in and decompression are the contention portion of memory. The problem then becomes the inability to differentiate contention due to limits using host level metrics, which means we'd need to look at VM level metric to exclude that expected contention.

Quiz!

Take a look at the following VM. The chart is hard to read, so read the table below it.



Measurement	Rollup	Units	Latest ↓	Maximum	Minimum	Average
Swap target	Average	KB	6,816,320	6,816,884	6,816,092	6,816,489
Swapped	Average	KB	6,520,964	6,520,964	6,520,964	6,520,960.5
Granted	Average	KB	1,223,088	1,223,088	1,223,088	1,223,088
Consumed	Average	KB	936,996	939,272	931,340	935,289.4
Compressed	Average	KB	644,556	644,556	644,556	644,557.7
Compressed	Latest	KB	644,556	644,556	644,556	644,557.7
Shared	Average	KB	570,756	570,756	570,756	570,754.7
Active	Average	KB	83,884	83,884	83,884	83,884
Balloon target	Average	KB	0	0	0	0
Ballooned memory	Average	KB	0	0	0	0
Active write	Average	KB	0	0	0	0
Compression rate	Average	KBps	0	0	0	0
Decompression rate	Average	KBps	0	0	0	0
Swap in	Average	KB	0	0	0	0
Swap in rate	Average	KBps	0	0	0	0
Swap out	Average	KB	0	0	0	0
Swap out rate	Average	KBps	0	0	0	0

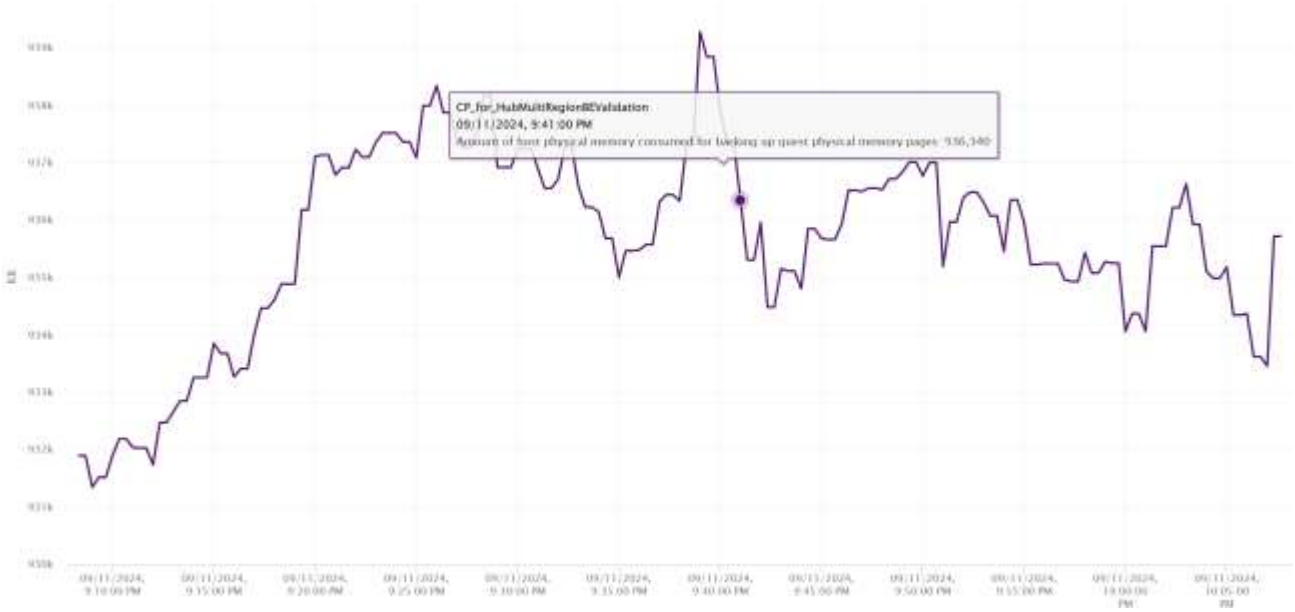
The 1st line (highest) is Swap Target. It's hovering around 6.8 million KB. It's fluctuating every 20 seconds (I did zoom in). ESXi does not have to achieve this if it does not need it.

The 2nd line is actual swapped. It's also constant in the last 1 hour. You can see the maximum and minimum are the same. The VM does not have limit, and it's not part of resource pool, so this could be historical.

The 3rd line shows the amount granted. It's also constant, indicating the VM does not ask for more.

The 4th line shows the amount consumed. It did fluctuate. I zoomed in and confirmed there were minor fluctuations. This is interesting since all other counters are 0.

The patterns of Consumed and Swap Target are mirror image. When Consume goes up, the target swap goes down.



Active is a sample. So it could be constant even though it should not be in this case as Consumed was not constant.

ESXi

Compared with CPU metrics, vCenter provides even more metrics for memory: 38 metrics for RAM plus 11 for the kernel RAM. The kernel has around 50 processes that are tracked. As a result, a cluster of 8 ESXi can have > 800 metrics just for ESXi RAM.

We will cover each metric in-depth, so let's do an overview first.

Overview

Just like the case for VM, the primary counter for tracking performance is Page-fault Latency. Take note this is normalized average, so use the Max VM Memory Contention instead.



Counters	Rollups	Units	Stat Type	Description
Page-fault latency	Average	%	Absolute	Percentage of time the virtual machine spent waiting to swap in or dec
Reclamation threshold	Average	KB	Absolute	Percentage of time the virtual machine spent waiting to swap in or decompress gues physical memory

The contention could be caused by swapping in the past. You've got only 5, not 6 metrics for swap. Which counter is missing?

Swap consumed	Average	KB	Absolute	Swap storage space consumed
Swap in	Average	KB	Absolute	Amount of guest physical memory that is swapped in from the swap s
Swap in rate	Average	KBps	Rate	Rate at which guest physical memory is swapped in from the swap sp
Swap out	Average	KB	Absolute	Amount of guest physical memory that is swapped out from the virtua
Swap out rate	Average	KBps	Rate	Rate at which guest physical memory is swapped out tc

Swap target is missing. It can be handy to see the total target at ESXi level.

Swap and Compress go hand in hand, so we should check both together. Here are the compressed metrics.

Compressed	Average	KB	Absolute	Guest physical memory pages that have undergone memory compressior
Compression rate	Average	KBps	Rate	Rate of guest physical memory page compression by ESXi
Decompression rate	Average	KBps	Rate	Rate of guest physical memory decompression

I'm unsure if Compressed measures the result of the compression, or the input. My take is the former as that's more useful from ESXi viewpoint.

Lastly, the performance could be caused by memory being read from the Host Cache. While they are faster than disk, they are still slower than physical memory.

Host cache consumed	Average	KB	Absolute	Storage space consumed on the host swap cache for storing
Host cache swap in	Average	KB	Absolute	Amount of guest physical memory swapped in from host cach
Host cache swap in rate	Average	KBps	Rate	Rate at which guest physical memory is swapped in from the
Host cache swap out	Average	KB	Absolute	Amount of guest physical memory swapped out to the host sv
Host cache swap out rate	Average	KBps	Rate	Rate at which guest physical memory is swapped out to the h

Wait! What about Balloon?

As will cover in-depth shortly, that's more of capacity than performance metrics. One can even say that other than Page-fault Latency, the rest of the metrics are actually for capacity not performance.

The famous balloon is a warning of capacity, assuming you do not play with limit.

Ballooned memory	Average	KB	Absolute	Amount of guest physical memory reclaimed from the virtual machine by t

When will ballooning kick in? There is a counter for that!

Reclamation threshold	Average	KB	Absolute	Threshold of free host physical memory below which ESXi will begin a

The memory state level shows one of the 5 possible states. You want to keep this at Clear state or High state.

Free state	Latest	num	Absolute	Current memory availability state of ESXi. Possible values are high, clear, s

For environment where performance matters more than cost, you want Balloon to be 0. That means Consume becomes your main counter for capacity. It is related to Granted and Shared.

Host consumed %	Average	%	Absolute	Percentage of host physical memory that has been consumed
Consumed	Average	KB	Absolute	Amount of host physical memory consumed for backing up guest physical memory pages
Granted	Average	KB	Absolute	Amount of host physical memory or physical memory that is mapped for a virtual machine or a host
Shared	Average	KB	Absolute	Amount of guest physical memory that is shared within a single virtual machine or across virtual machines
Shared common	Average	KB	Absolute	Amount of host physical memory that backs shared guest physical memory (Shared)

Reservation plays a big part in capacity management as it cannot be overcommitted. ESXi, being a provider of resource, has 3 metrics to properly account for reservations.

Reservation available	Average	KB	Absolute	Amount by which reservation can be raised
Reservation consumed	Average	MB	Absolute	Memory reservation consumed by powered-on virtual machines
Total reservation	Average	MB	Absolute	Total reservation, available and consumed, for powered-on virtual machines

There are a few metrics covering 0 pages and overhead. The Heap counter shows the memory used by the kernel heap and other data. This is normally a constant and small value.

Overhead consumed	Average	KB	Absolute	Host physical memory consumed by ESXi data structures for running the virtual machines
VMkernel consumed	Average	KB	Absolute	Amount of host physical memory consumed by VMkernel
Zero pages	Average	KB	Absolute	Guest physical memory pages whose content is 0x00
Heap	Average	KB	Absolute	Virtual address space of ESXi that is dedicated to its heap
Heap free	Average	KB	Absolute	Free address space in the heap of ESXi. This is less than or equal to Heap

Other Metrics

Active is not a counter for capacity or performance. It's for the kernel memory allocation.

Active	Average	KB	Absolute	Amount of guest physical memory that is being actively read or written by guest. Activeness is estimated by ESXi
Active write	Average	KB	Absolute	Amount of guest physical memory that is being actively written by guest. Activeness is estimated by ESXi

Persistent Memory

Counters	Rollups	Units	Stat Type	Description
Persistent memory available reservation	Latest	MB	Absolute	Persistent memory available reservation on a host.
Persistent memory reservation managed by DRS	Latest	MB	Absolute	Persistent memory reservation managed by DRS on a ho...

Lastly, there are a few metrics for VMFS pointer block cache. Read more here. They are internal, only used by the kernel. The only one you might be interested in the cache capacity miss ratio. Let me know if you have a real-world use case them.

Maximum VMFS PB Cache Size	Latest	MB	Absolute	Maximum size the VMFS Pointer Block Cache can grow to
Maximum VMFS Working Set	Latest	TB	Absolute	Maximum amount of file blocks whose addresses are cached
VMFS PB Cache Capacity Miss Ratio	Latest	%	Absolute	Trailing average of the ratio of capacity misses to compulsory misses for the VMFS PB Cache
VMFS PB Cache Overhead	Latest	KB	Absolute	Amount of VMFS heap used by the VMFS PB Cache
VMFS PB Cache Size	Latest	MB	Absolute	Space used for holding VMFS Pointer Blocks in memory
VMFS Working Set	Latest	TB	Absolute	Amount of file blocks whose addresses are cached in the VMFS PB Cache

"Contention" Metrics

I put the title in "quote" as none of these counters actually measure contention.

I do not cover the Latency metric as that's basically a normalized average of all the running VMs on the host.

Balloon

Balloon is a leading indicator that an ESXi is under memory pressure, hence it's one of the primary metrics you should use in capacity. Assuming you're not using Limit to artificially cap the resource, you should ensure that the balloon amount does not cause VM to experience contention.

We know that contention happens at hypervisor level, not at VM level. The VM is feeling the side effects of the contention, and the degree of contention depends on each VM's shares, reservation and utilization. ESXi begins taking action if it is running low on free memory. This is tracked by a counter called State. The State counter has five states, corresponding to the Free Memory Minimum (%) value

ESXi State	Threshold	1 TB ESXi	Example based on ESXi with 1 TB RAM
High	300%	32.4 GB	First, we calculate the Free Memory Minimum value. There is many website to help you with this, such as this. For 1 TB, the value is 10.8 GB.
Clear	100%	10.8 GB	
Soft	64%	6.9 GB. Balloon starts here	
Hard	32%	3.5 GB. Compress/Swap starts here	
Low	16%	1.7 GB. Block execution	

Using the example above, let's see at which point of utilization does ESXi triggers balloon process.

ESXi State	512 GB ESXi	1 TB ESXi	1.5 TB ESXi
Balloon Threshold	3.7 GB	6.9 GB	10.2 GB
Threshold	508.3 GB	993.1 BB	1489.8 GB
Threshold in %	99.3%	99.3%	99.3%

As you can see from all the 3 ESXi, balloon only happens after at least 99% of the memory it utilized. It's a very high threshold. Unless you are deliberately aiming for high utilization, all the ESXi should be in the **High** state.

In addition, the spare host you add to cater for HA or maintenance mode will help in lowering the overall ESXi utilization. Let's use example to illustrate

- No of ESXi in a cluster = 12
- Provisioned for HA = 11
- Target ESXi memory utilization = 99% (when HA happens or planned maintenance)
- Target ESXi memory utilization = 99% x 11 / 12 = 90.75% (during normal operations)

Using the above, you will not have any VM memory swapped as you won't even hit the ballooned stage. If you actually see balloon, that means there is limit imposed.

The **Low Free Threshold** (KB) counter provides information on the actual level below which ESXi will begin reclaiming memory from VM. This value varies in hosts with different RAM configurations. Check this value only if you suspect ESXi triggers ballooning too early.

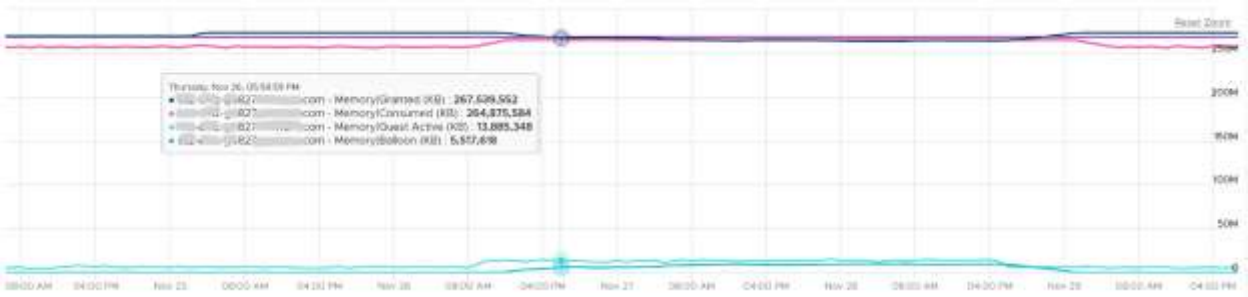
ESXi memory region can be divided into three: Used, Cached and Free

- Used is tracked by Active. Active is an estimate of recently touched pages.
- Cached = Consumed - Active. Consumed contains pages that were touched in the past, but no longer active. I'm not sure Ballooned pages are accounted in Consumed, although logically it should not. It should go to Free so it can be reused.
- Free = Total Capacity - Consumed.

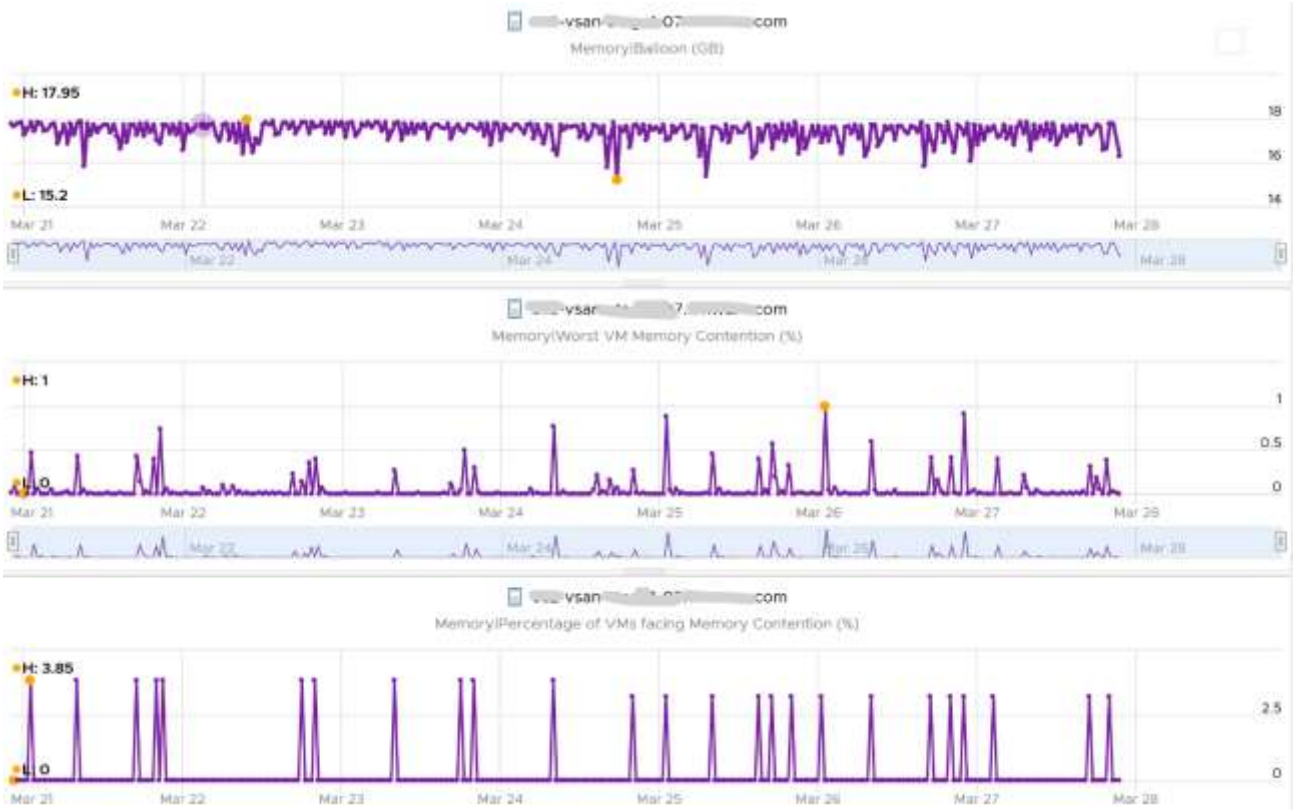
The nature of memory as cache means the active part is far lower than the non-active part. It's also more volatile. The following shows an ESXi with low memory usage, both active and consumed, in the last 3 months.



Let's look at an opposite scenario. The following ESXi is running at 100%. It has granted more memory than what it physically has. Initially, since the pages are inactive, there is no ballooning. When the active rise up, the consumed counter goes up and the balloon process kicks in. When the VM is no longer using the pages, the active counter reflects that and ESXi begin deflating the balloon and giving the pages back.



I shared in the VM memory counter that just because a VM has balloon, does not mean it experiences contention. You can see the same situation at ESXi level. The following ESXi shows a constant and significant balloon lasting at least 7 days. Yes the worst contention experienced by any VM is not even 1%, and majority of its 19 VMs were not experiencing contention at all.



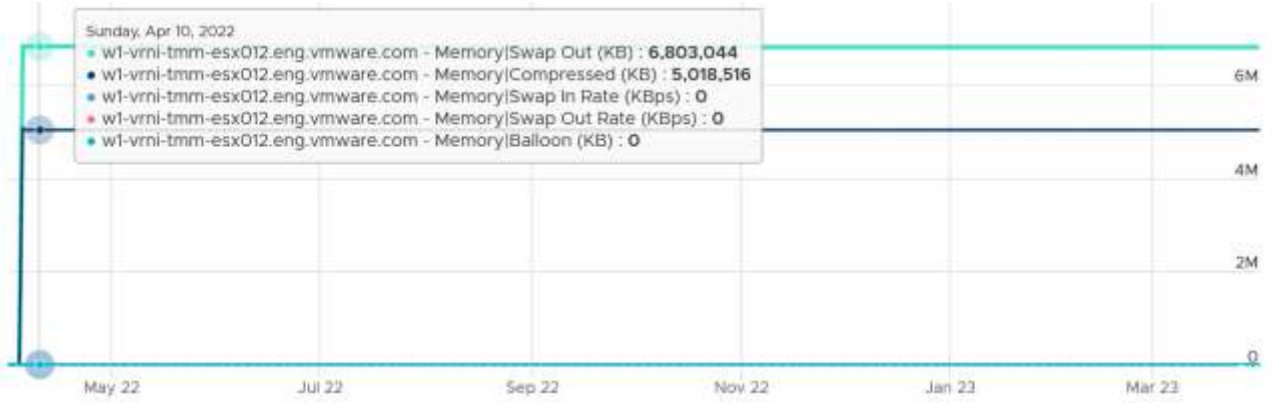
Swap + Compress

For swap, the metric is the summation of running VMs and the kernel services.

For compress, there are 2 counters at ESXi level. The first is the sum of all amounts that were subjected to compressed. The second is the resultant compressed amount.

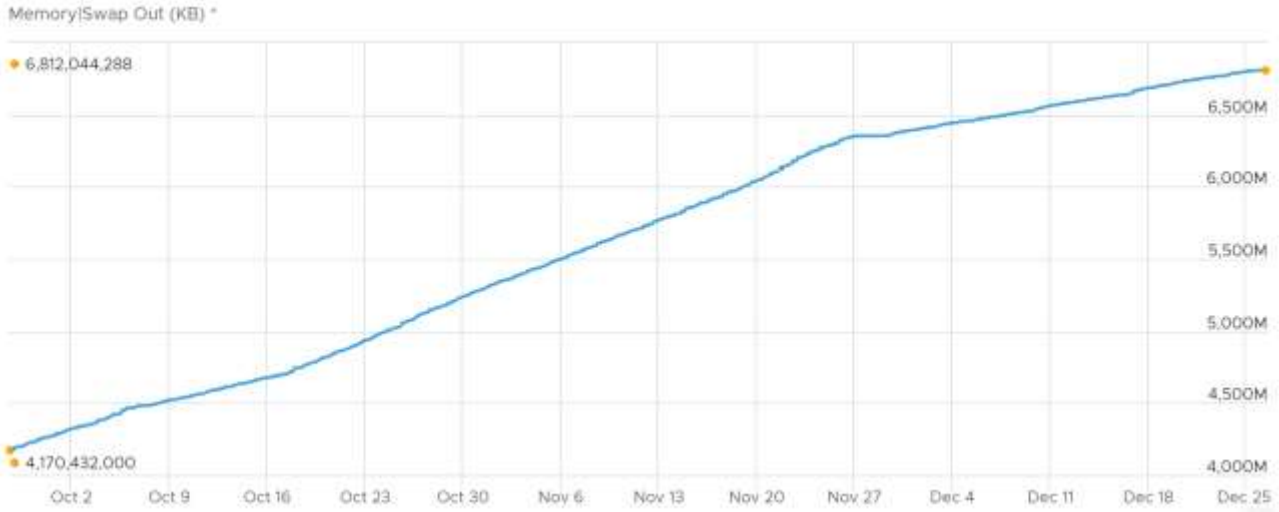
Metrics	Description
Swap Consumed	Sum of memory swapped of all powered on VMs and vSphere services on the host. This number will reduce if pages are swapped back into the DIMM. I think this is swapped out – swapped in.
Swap In	The total amount of memory that have been swapped in or out to date. Note: These counters are accumulative.
Swap Out	
Swap In Rate	I think this includes compressed, not just swapped, but I'm not 100% sure as I can't find a proof yet.
Swap Out Rate	

Pages can and will remain in compressed or swapped stage for a long time. The following screenshot shows compressed remains around 5 GB for around 1 year.

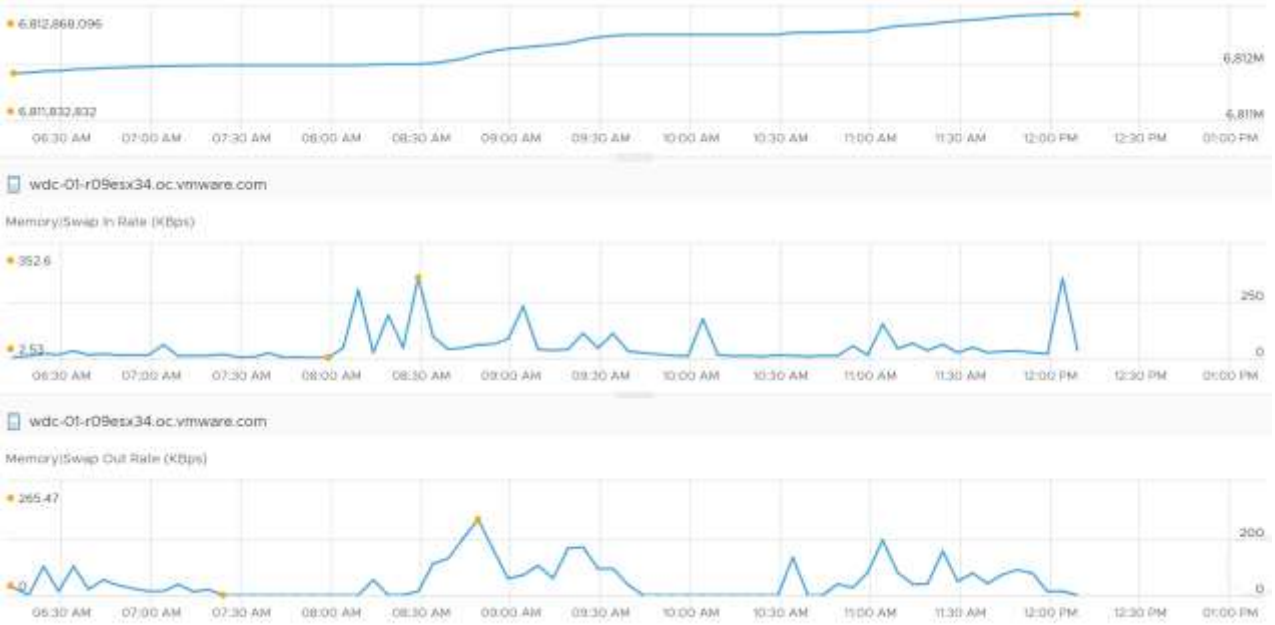


The above happened because there was no need to bring back those pages. Notice ballooning was flat 0, indicating the ESX host was not under memory pressure.

Swap Out is an accumulative counter.



Let’s zoom in, and add the swap in and swap out counters to compare.



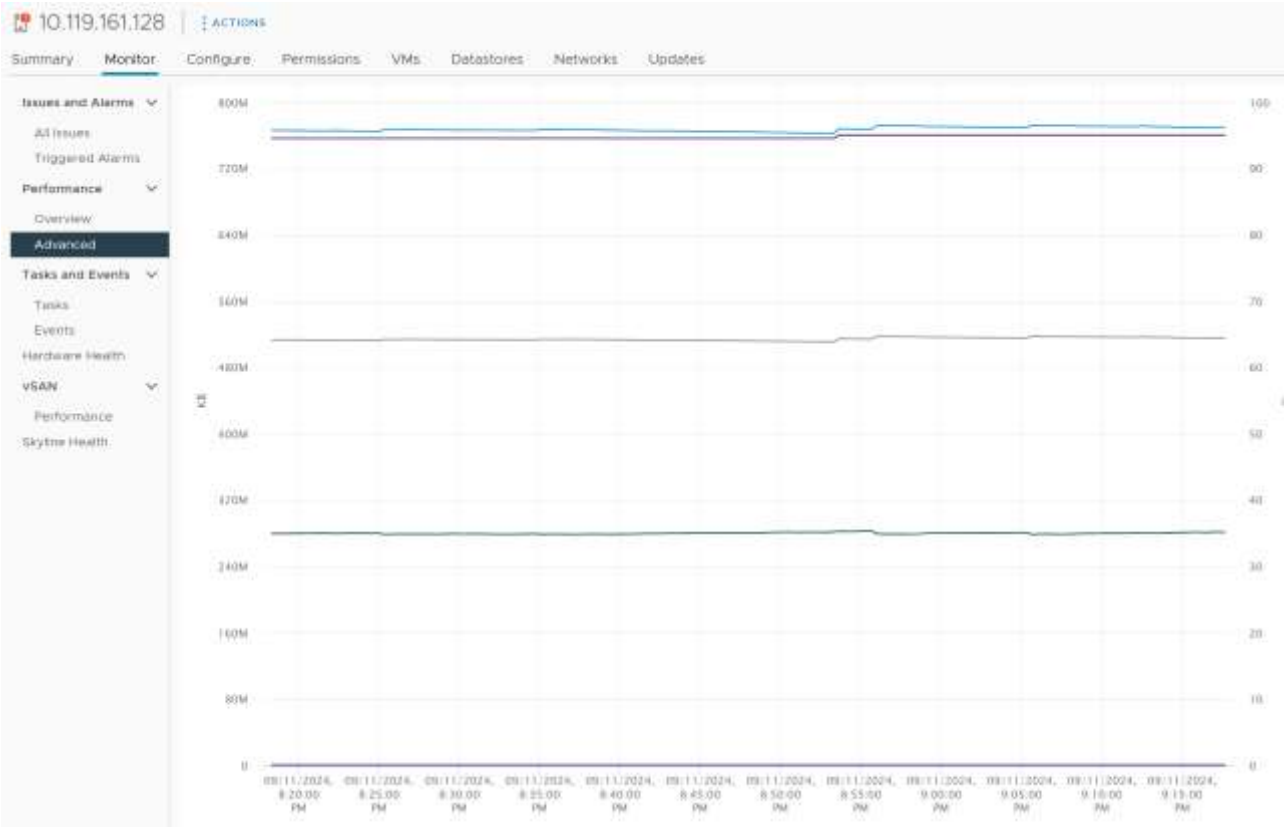
Notice the value did not go down despite swap in.

All "Together"

Balloon operates differently and works at a different layer than Swap and Compress. It takes longer to realize, and is not affected by limit. As a result, you can have 0 balloon while having swapping and zipped.

The following ESXi shows high Consume, and even higher Granted.

- The first line (highest blue line) shows Consume is hovering around 96%.
- The second line (purple line, just below the blue) shows Granted is hovering around 756 million KB.
- The third line shows consumed hovering around 514 million KB. ESXi has 511.46 GB or 536,304,680 KB of memory.



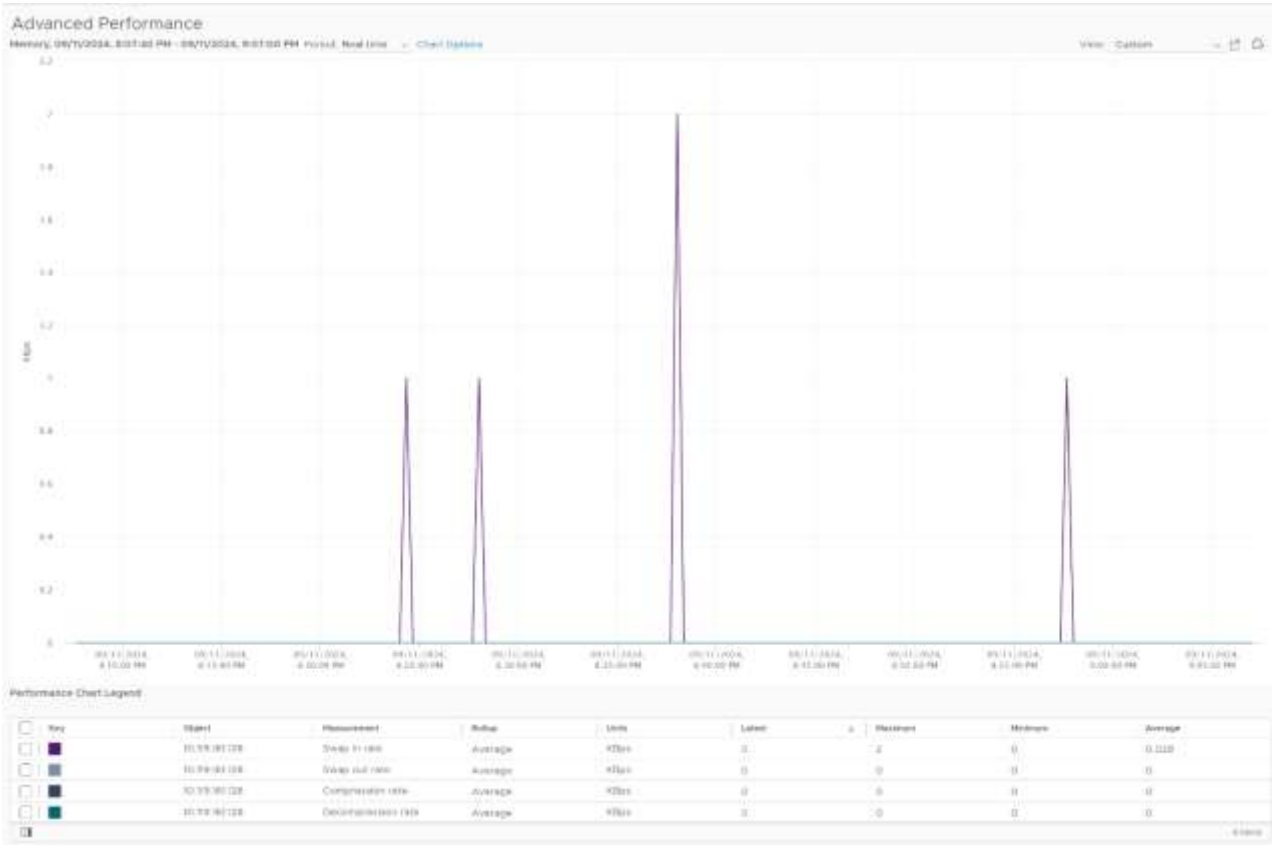
Since it's hard to see, let's show the table. What do you notice?

Measurement	Rollup	Units	Latest ↓	Maximum	Minimum	Average
Granted	Average	KB	760,780,350	760,780,350	756,805,120	758,557,570
Consumed	Average	KB	515,485,056	517,934,048	511,484,768	514,850,208
Shared	Average	KB	282,273,952	283,359,168	279,181,088	280,497,088
Swap consumed	Average	KB	1,424,896	1,425,212	1,424,896	1,425,055.2
Compressed	Average	KB	89,388	89,432	89,388	89,406.77
Host consumed %	Average	%	96.11	96.57	95.37	95.994
Ballooned memory	Average	KB	0	0	0	0

Ballooned was 0 constantly.

However, there were swapped and compressed. Let's see if they are still happening, since these 2 counters are accumulative.

As you can see from the following chart, the amount is negligible. There are 4 instances of swap in, and each time the amount is 1 KB/second or 2 KB/second. Since the number of is the average of 20 seconds, the total amount is 20 KB or 40 KB only.



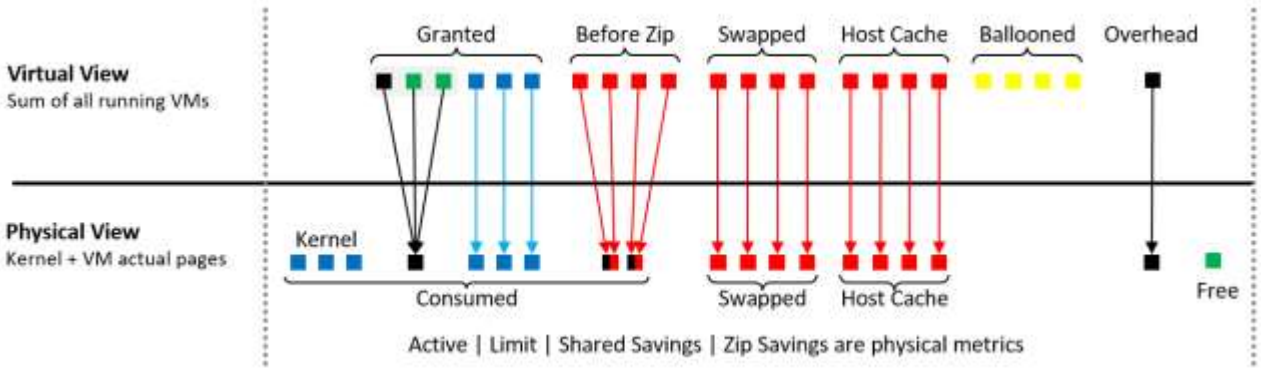
Does it mean the memory is not active?

Let’s look at the Active metric. It shows there are indeed activities, but they are within the pages already in the DIMM.



Consumption Metrics

Consumption covers utilization, reservation and allocation.



Consumed

Consumed is the primary counter for ESXi utilization but it contains a lot of cache and inactive pages. Just like any other modern-day OS, the kernel uses RAM as cache as it's faster than disk. So the Consumed counter will be near 100% in overcommit environment. This is an ideal goal, as opposed to something you need to panic.

The formula is:

```
Consumed = Physical memory – Free.
```

What does the above mean since you can overcommit?

Consumed only includes pages mapped in the physical DIMM. That means:

- It does not include ballooned. This metric is at the Guest OS level.
- It does not include swapped. The swapped pages reside on disk or host cache.
- It only includes the resultant zipped, not the amount subjected to compression.
- It only includes the resultant savings from TPS, not the amount subjected to sharing.
- It includes VM overhead
- It includes all the kernel processes, both user space and protected space.

Formula for VCF Operations Metrics:

- Usage (%) = Consumed / Total Capacity.
-

Ballooned

Consumed does not include Ballooned. This makes sense as the pages no longer backed by physical pages. The following screenshot shows consumed drops when balloon went up.



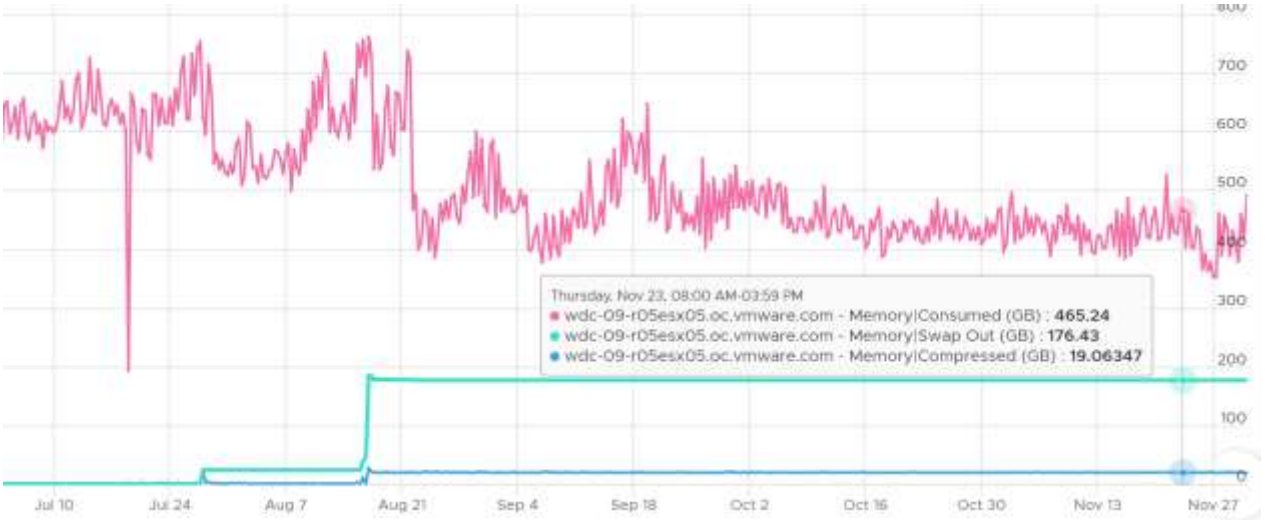
Swapped

Consumed does not include swapped. This makes sense as the page are no longer in the physical memory. The following screenshot shows consumed drops when swap out went up.



Compressed

Consumed does not include compressed. The following shows that both compressed and swap out went up by almost 200 GB, yet Consumed dropped in the same period. It's possible pages were removed from Consumed and were swapped and compressed.

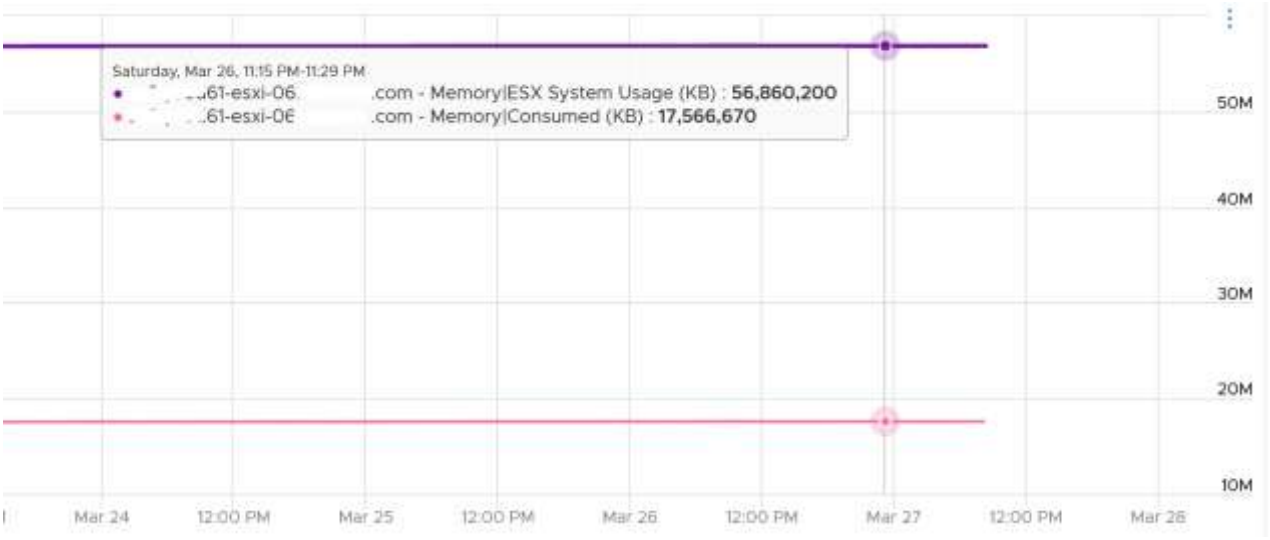


Kernel

The other part of Consumed is non VM. This means the kernel, vSAN, NSX and whatever else running on the hypervisor. Because ESXi Consumed includes non VM, it can be more than what's allocated to all running VMs, as shown below.



Take note that Consumed includes the actual consumption, not the reservation. The following ESXi has 0 running VM, so the Consumed is just made of the kernel. You can see the utilization is much lower than the reservation.

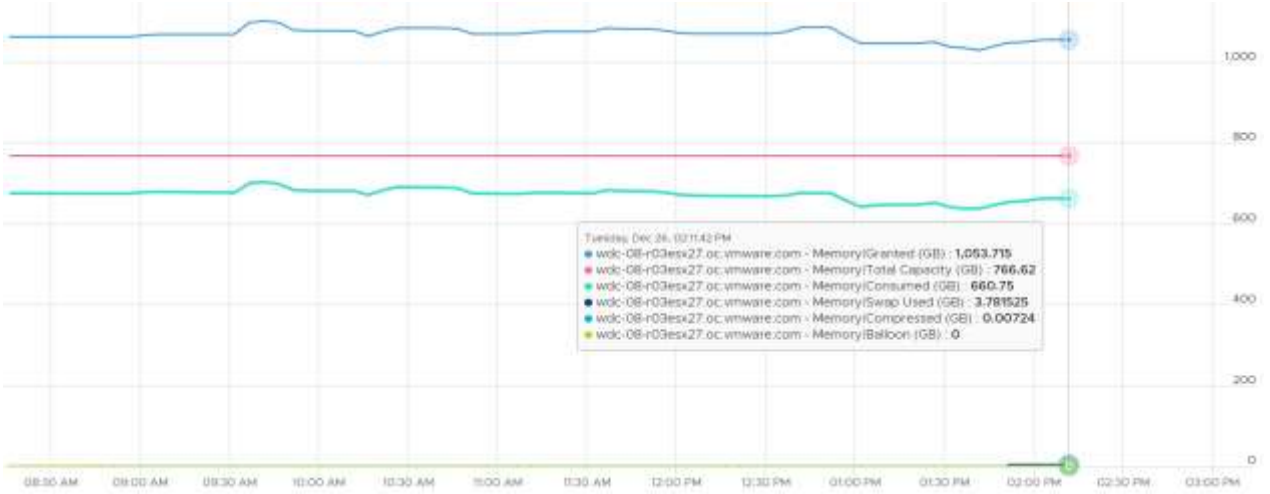


If you're wondering why it's consuming 17 GB when there is 0 VM, the likely answer is vSAN. Just because there is no VM does not mean vSAN should stop running.

Granted

Granted differs to Consumed as it excludes certain part of the kernel. It does not include the kernel space as processes at this privileged level gets what they want. They do not need the granting process, so to speak. Consumed, on the other hand, includes all both user and kernel space as both are indeed consuming pages.

Granted, being a consumer-level counter, can exceed total capacity. The following ESXi has granted 1053 GB of memory to running VMs, way above its total capacity of 755 GB.



Notice the sum of consumed + swapped + compressed is always below the total capacity.

I added balloon just in case you're curious.

The following example shows ESXi hosts with no running VM. I'm surprised to see the granted counter is not 0. My guess the extra memory is for non-VM user world process.

Total Capacity	Consumed	Granted	VMkernel	vSAN Host?	VM
382.47 GB	38.65 GB	0.56 GB	37.48 GB	Enabled	0
382.47 GB	38.56 GB	0.55 GB	37.38 GB	Enabled	0
255.62 GB	36.65 GB	0.51 GB	35.69 GB	Enabled	0
382.62 GB	38.7 GB	0.51 GB	37.65 GB	Enabled	0
255.91 GB	35.66 GB	0.5 GB	34.7 GB	Enabled	0
510.62 GB	38.96 GB	0.5 GB	37.99 GB	Enabled	0
511.91 GB	38.01 GB	0.5 GB	36.99 GB	-	0
511.91 GB	37.97 GB	0.49 GB	36.98 GB	-	0
255.62 GB	35.38 GB	0.49 GB	34.45 GB	-	0
766.62 GB	7.05 GB	0.49 GB	6.1 GB	-	0
191.75 GB	14.93 GB	0.45 GB	14.01 GB	-	0

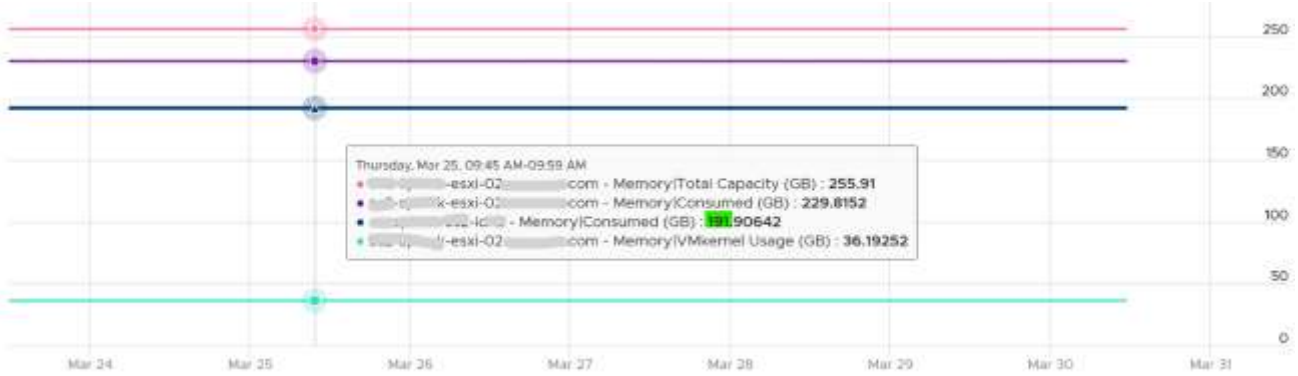
Let's take one of the ESXi to see the value over time. This time around, let's use vCenter instead.



Performance Chart Legend

Key	Object	Measurement	Rollup	Units	Latest	Maximum	Minimum	Average
	10.155.60.228	Consumed	Average	KB	8,334,308	8,439,876	8,332,708	8,339,415
	10.155.60.228	VMkernel consumed	Average	KB	7,685,100	7,702,640	7,683,448	7,689,504
	10.155.60.228	Granted	Average	KB	432,356	490,460	431,992	432,588.72

You can verify that ESXi Consumed includes its running VMs Consumes by taking an ESXi with a single running VM. The ESXi below has 255 GB of total capacity but only 229 GB is consumed. The 229 GB is split into 191 GB consumed by VM and 36 GB consumed by the kernel.



The kernel consumption is the sum of the following three resource pools.

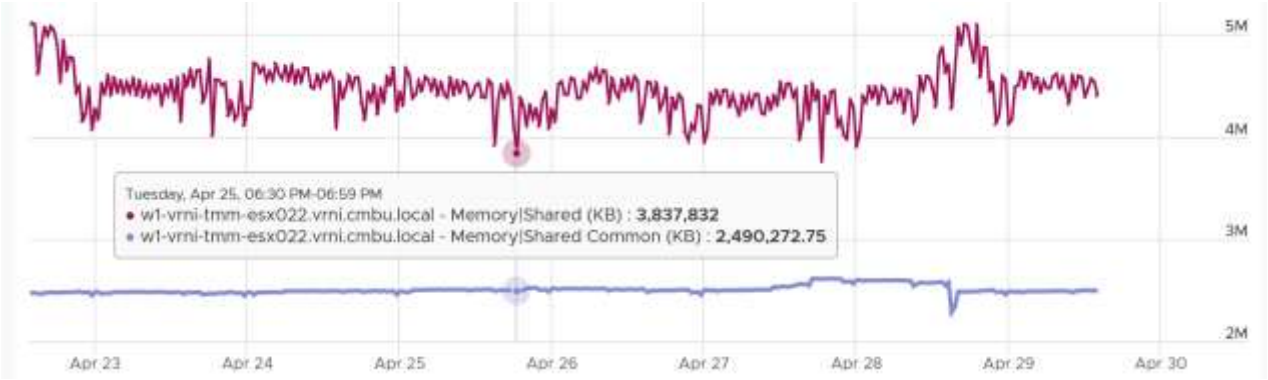


Shared

Metrics	Description
Shared	The sum of all the VM memory pages & the kernel services that are pointing to a shared page. In short, it's Sum of VM Shared + the kernel Shared. If 3 VMs each have 2 MB of identical memory, the shared memory is 6 MB.
Shared Common	The sum of all the shared pages. You can determine the amount of ESXi host memory savings by calculating Shared (KB) - Shared Common (KB)

Memory shared common is at most half the value of Memory shared, as sharing means at least 2 blocks are pointing to the shared page. If the value is a lot less than half, then you are saving a lot.

The following shows the shared common exceeding half many times in the last 7 days.



I'm not sure why. My wild guess is large pages are involved. ESXi hosts sport the hardware-assisted memory virtualization from Intel or AMD. With this technology, the kernel uses large pages to back the VM memory. As a result, the possibility of shared memory is low, unless the host memory is highly utilized. In this high consumed state, the large pages are broken down into small, shareable pages. The smaller pages get reflected in the shared common. Do let me know if my wild guess is correct.

You can also use the Memory shared common counter as leading indicator of host breaking large page into 4K. For that, you need to compare the value over time, as the absolute value may be normal for that host. The following table shows 11 ESXi hosts with various level of shared pages. Notice none of them is under memory pressure as balloon is 0. That's why you use them as leading indicator.

Memory|Shared (KB)	Memory|Shared Common (KB)	Memory|Consumed (KB)	Memory|Balloon (KB)	Memory|Capacity Available to VMs (KB)
65,607,005.87	10,357,951.73	352,640,588.27	0	529,732,198.4
70,097,646.13	7,480,631.47	343,656,066.4	0	529,059,771.73
78,135,773.33	6,784,762.93	344,631,349.07	0	529,682,909.87
11,506,141.33	3,209,230.13	153,130,568.8	0	153,209,241.6
4,457,464.53	2,489,271.73	159,532,024.27	0	153,887,129.6
8,202,879.73	1,742,066.4	173,350,765.33	0	166,842,094.93
7,425,348.8	1,447,984	167,072,416.8	0	166,758,468.27
9,779,264.27	1,378,015.2	307,635,957.07	0	344,527,872
5,428,565.07	1,231,148	173,757,243.47	0	166,856,226.13
2,512,468.53	964,596.53	56,699,406.4	0	33,570,611.2
19,168,012.53	766,276.8	126,624,427.73	0	157,945,787.73

With Transparent Page Sharing limited to within a VM, shared pages should become much smaller in value. I'm not sure if salting helps address the issue. From the vSphere manual, "With the new salting settings, virtual machines can share pages only if the salt value and contents of the pages are identical".

I'm unsure if the above environment has the salting enabled or not. Let me know what level of sharing in your environment, especially after you disable TPS.

Utilization

We've seen that Consumed is too conservative as mostly cache and Active is too aggressive as it's not even designed for memory sizing.

This calls for a metric in the middle. This is where Utilization comes in.

It's the sum of running VM Utilization metrics + the kernel reservation.

Utilization uses the reservation amount for the kernel, instead of the actual utilization. This is technically not accurate but operationally wise as it gives you buffer.

I plotted from 192 ESXi. I averaged the data to remove outlier. Based on 6840 running VMs, the Utilization counter is lower than Consumed by 122 GB. If you include Shared Common, your savings goes up to 152 GB on average.

Name	Total Capacity	Consumed	Utilization	Running VMs	Memory|Shared Common (GB)
wdc-01-r0_	3,070.46 GB	2,627.11 GB	2,304.28 GB	50	128.38 GB
wdc-01-r0_	3,070.46 GB	2,222.03 GB	1,728.51 GB	41	133.22 GB
wdc-01-r0_	3,070.46 GB	2,060.89 GB	1,537.81 GB	32	140.81 GB
wdc-01-r0_	3,070.46 GB	2,884.35 GB	2,045.38 GB	14	210.15 GB
wdc-01-r0_	3,070.46 GB	2,326.02 GB	1,695.57 GB	13	254.8 GB
wdc-01-r0_	3,070.46 GB	2,625.7 GB	1,742.37 GB	47	270.37 GB
Average	**892.05 GB**	**522.63 GB**	**401.1 GB**	**35.62**	**29.7 GB**

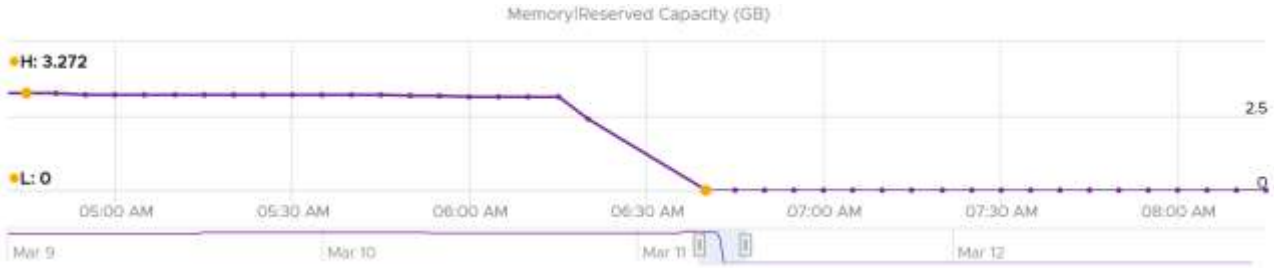
Validation

The following screenshot shows that ESXi had all its VM evacuated. Not a single VM left, regardless of power on/off status.



In the preceding chart, we could see the metric Memory Allocated on All Consumers dropped from 452 GB to 0 GB, and it remained flat after that.

Checking the Reserved Capacity metric, we can see it dropped to 0. This is expected.



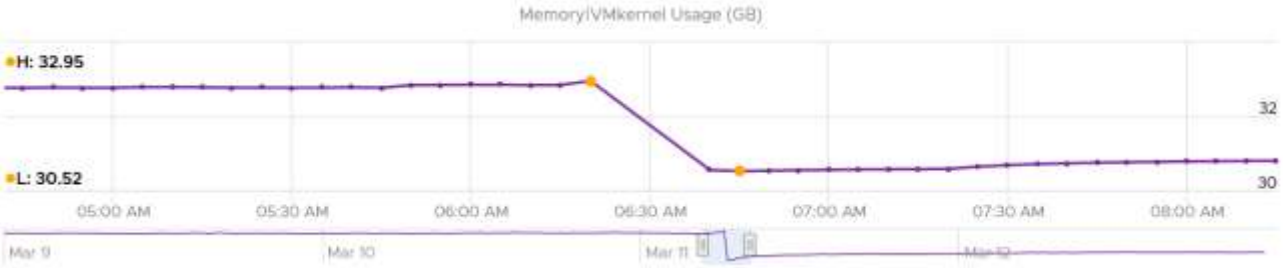
How about Consumed?



Memory Consumed also dropped. The value was 400 GB, less than 452 GB of allocated to all VM. This indicated some VM had not used the memory, which could happen.

The value dropped to 32 GB, not 0 GB. This is expected as Consumed includes every other process that runs. In this case, it is majority vSAN, which runs in the kernel.

Let's check the kernel utilization.



Notice it's a bit smaller than Consumed, indicating Consumed has other thing. I suspect it's BIOS and the console in vSphere Client UI.

How come the value didn't change much? I kind of expect some changes, based on the theory that some kernel modules memory footprint depends on the number of running VM. If you know, let me know!

How about the kernel reservation? What do we expect the value to change?



Well, it won't since the actual usage does not change.

Analysis

I compare 185 production ESXi hosts to understand the behaviour of the metrics. I averaged their results to eliminate outlier.

Total Capacity	Host Usage	Machine Demand	Utilization	Consumed ↑
382.62 GB	232.35 GB	116.34 GB	160.03 GB	265.11 GB
382.62 GB	233.74 GB	87.74 GB	131.28 GB	265.98 GB
767.22 GB	228.6 GB	194.05 GB	252.55 GB	267.5 GB
766.67 GB	231.77 GB	183.79 GB	235.66 GB	268.47 GB
382.62 GB	236.23 GB	83.42 GB	126.67 GB	268.5 GB
382.62 GB	237.47 GB	85.58 GB	129.48 GB	270.63 GB
382.62 GB	238.72 GB	236.48 GB	281.05 GB	273.31 GB
382.62 GB	240.35 GB	75.55 GB	119.63 GB	273.38 GB
766.67 GB	256.9 GB	230.85 GB	282.89 GB	293.52 GB
703.42 GB	260.93 GB	184.42 GB	234.59 GB	296.29 GB
736.7 GB	**367.07 GB**	**247.96 GB**	**299.56 GB**	**404.42 GB**

The average of all the 185 ESXi hosts have total capacity of 737 GB. This is the physical configured memory.

The metric Memory \ Usable Memory is 729 GB (not shown in above table). It's 1% less or 8 GB than Total Capacity. I suspect this maps to Managed metric in vCenter. It is the total amount of machine memory managed by the kernel. The kernel "managed" memory can be dynamically allocated for VM, the kernel, and User Worlds. I need to check what exactly this is as I don't see a use case for it.

The metric Memory \ VMkernel Usage is 7.6 GB (not shown in above table). This is much lower than the reservation, which is 51.6 GB.

Consumed is generally higher than the other 3 metrics. The only time it's lower is when there is a lot of savings from shared pages.

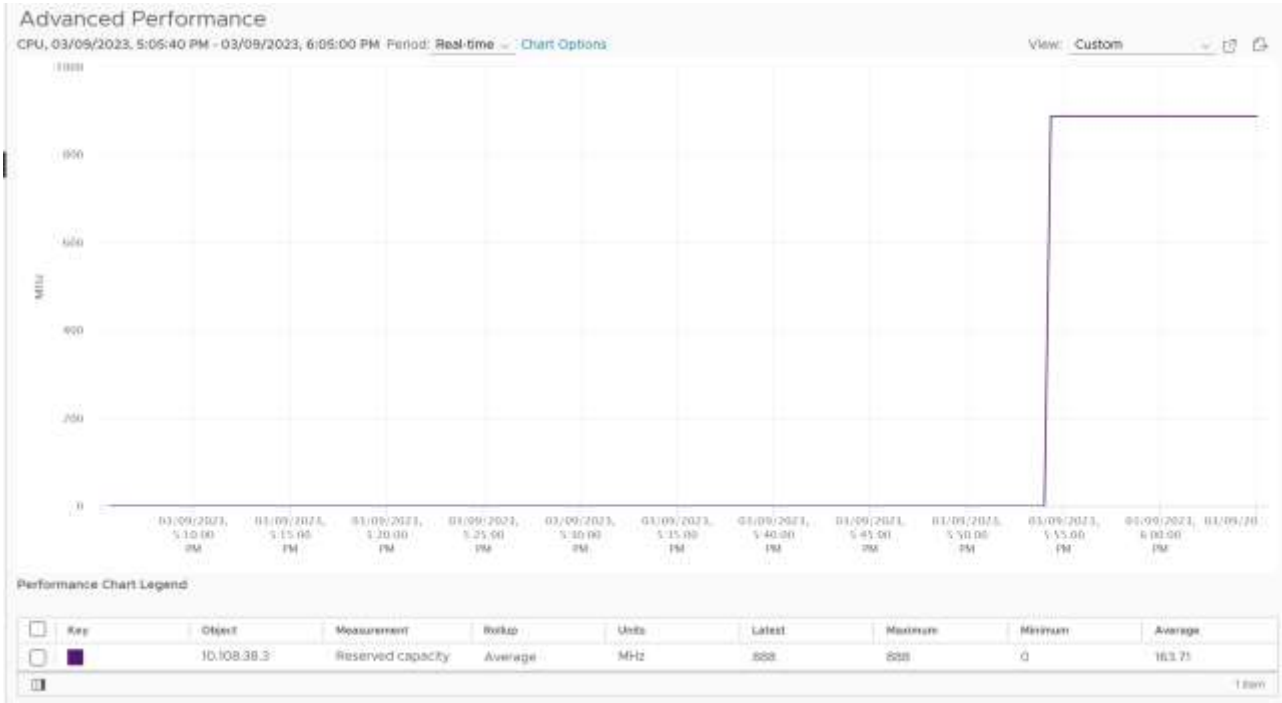
What are these?

- Host Usage. Sum of VM Consumed. ESX System Usage is not included. Use case is only for migration, where we don't want the ESXi consumption.
- Machine Demand. The sum of VM Utilization
- Utilization. Machine Demand + ESX System Usage. You can see that the value equals ESX System Usage when there is 0 running VM.
- Workload = Utilization against Usable

Reservation

Total reservation (MB)	This is the amount reserved. Note it does not mean it's actually used by the VM. It only counts reservation by powered on VM. It does not include powered off VM and the kernel reservation. See screenshot below. This metric is also labelled as Reserved Capacity.
Reservation consumed	The actual consumption. If this number if consistently lower than the reserved capacity, it indicates over reservation.
Reservation available	This is the amount that is not even reserved. That means it is available for new reservation.

The following screenshot shows an ESXi where the CPU reservation was flat 0 MHz. I then set one of its VM reservation to 888 MHz. Notice the immediate yet constant change.



Part 3 Chapter 4

Architecture

To understand storage metrics, be aware that there are 2 dimensions of metrics (speed and space). Both the speed and space dimensions have consumption metrics, but they are completely different.

For speed, the counters are IOPS and throughput, where throughput = IOPS x block size.

For space, the counters are disk space.

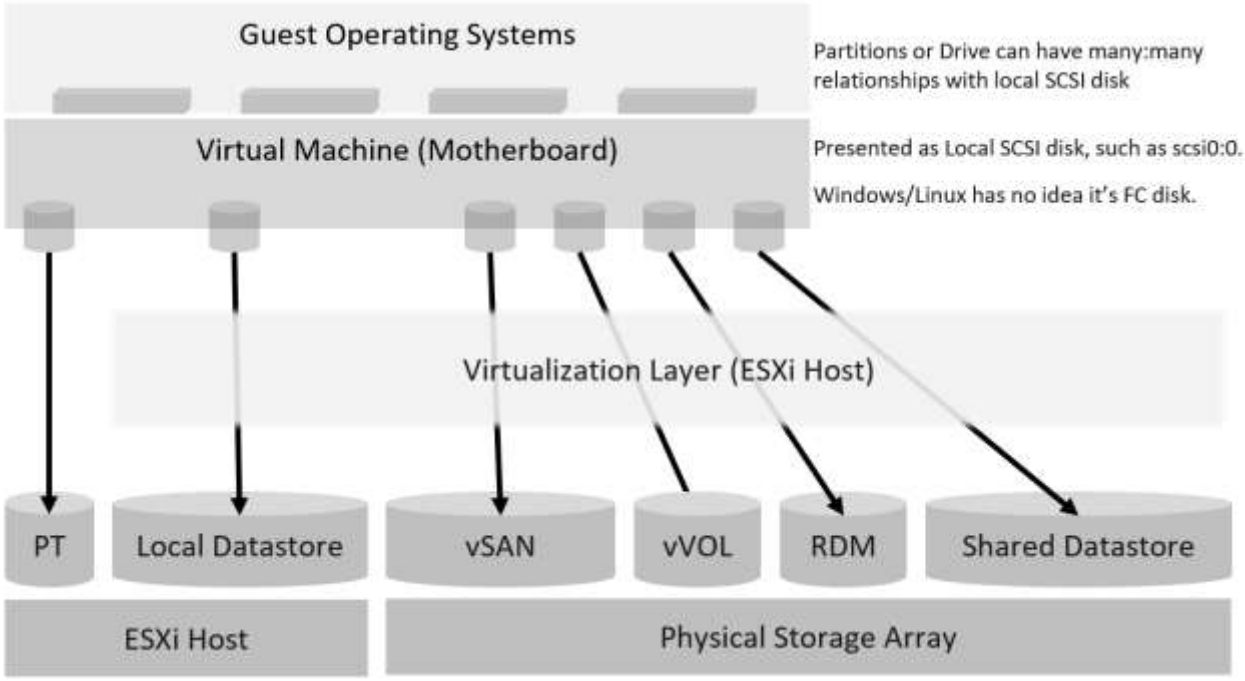
Speed	Performance is measured in 2 ways (contention and utilization). Contention can happen at all 3 stages of IO processing: • pre-processing: each layer has their own queue or outstanding IO. • processing: aborted SCSI commands, dropped frame, etc. • post-processing: latency.
Space	Capacity has no impact of slowness in modern, SSD based storage as access to data is no longer relying on spinning platter. 1% disk space full is not slower/faster than 99% full as defragmentation is no longer causing latency. However, it impacts availability. At 100% full, the storage will stop processing IO and your application will crash as a result. Capacity, as in disk space, is measured in bytes. Storage differs to compute as reality overwrites projection. In compute, you use a projected capacity remaining number, which takes into account the past. In storage, if you have 0 bytes left, the number overwrites whatever number shown by capacity engine. You should also focus on reclamation as the amount tends to be substantial.

Storage Layers

Virtualization increases the complexity in both storage capacity and performance. Just like memory, where we have more than 1 level, we have multiple layers of storage and each layer only has control over its own. In addition, each layer may not use the same terminology. For example, the term disk, LUN and device may mean the same thing. A device is typically physical (something you can hold, like an SSD card). LUN is typically virtual, a striping across physical devices in a volume.

The layers present challenge in management, as they create limitation in end-to-end visibility and raise different questions. You lose VM-level visibility in the physical adapter (for example, you cannot tell how many IOPSs on vmhba2 are coming from a particular VM) and physical paths (for example, how many disk commands travelling on that path are coming from a particular VM).

Storage in VMware IaaS is presented as datastore. In some situation, RDM and network file shares are also used by certain VM.

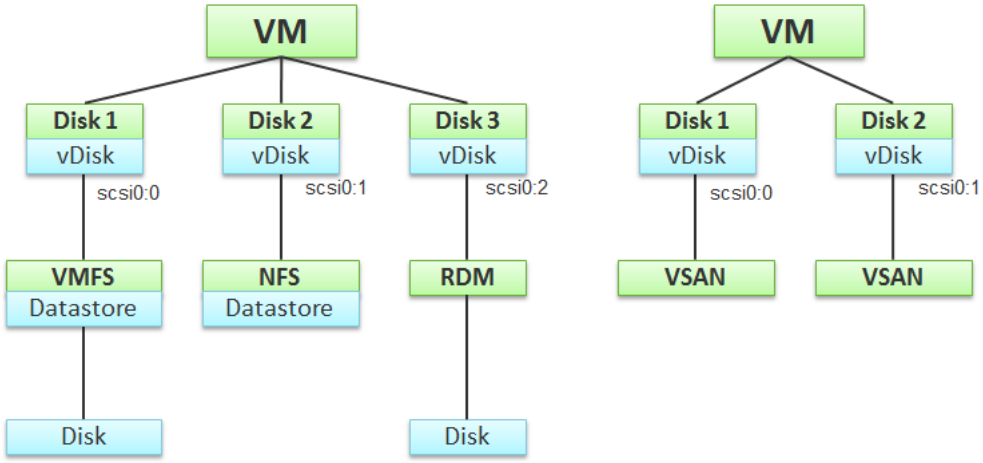


Underlying storage protocol (FC, iSCSI, vSAN, vVOL) is not exposed. Guest OS needs not support the protocol as it's transparent.

Layer	Description
Guest OS	The most upper layer is the Guest OS. It sees virtual disks presented by the VM motherboard. Guest OS typically has multiple partition or drive. Each partition has its own filesystem, serving different purpose such OS drive, paging file drive, and data drive. A large database VM will have even more partitions. Partition may *not* map 1:1 to the virtual disk. There is no visibility to this mapping. This makes calculating unmapped blocks accurately an impossible task in the case of RDM disk.
	To make it more complex, there is also networked drive. Windows or Linux mounts them over protocol such as SMB. These filesystems are not visible to the hypervisor; hence they are not virtual disk. The disk IO is not visible to the VM as it goes out via vNIC.
VM	See the VM Layer section below
ESXi	See the ESXi Layer section below
Datastore	See the Datastore Layer section below
Storage Subsystem	This can be virtual (e.g. vSAN) or physical (e.g. physical array). If it's NFS, it can be virtual server or physical. The datastore is normally backed one to one by a LUN, so what we see at the datastore level matches with what we see at the LUN level. Multiple LUNs reside on a single array. Datastores that share the same underlying physical array can experience problem at the same time. The underlying array can experience a hot spot on its own, as it is made of independent magnetic disks or SSD.

VM Layer

The preceding storage layers resulted in 3 different layers of "disks" from the VM downwards. The blue boxes show the 3 layers.



Remember the 3 blue boxes. You will see them, no more no less, in vSphere Client UI. To enable us to focus on the blue boxes, I've excluded non virtual disk files such as snapshot and memory swap in the preceding diagram.

The first layer is virtual disk. This exists because the VM does not see the underlying shared storage. It sees as simple local SCSI disks only, as presented by the VM motherboard. This explains why MS-DOS can run on fibre channel, because it's unaware of the underlying storage architecture.

A virtual disk is identified as scsiM:N, starting with scsi0:0, where M is the adapter bus number.

A virtual disk can sit on top of different underlying storage architecture. I've shown 4 in the preceding diagram: NFS, VMFS, vSAN, and RDM. A VM can also be presented with ESXi local physical disk as direct passthrough, although that means the VM cannot run on other ESXi host.

For block datastore (read: VMFS), it can span multiple disks. It is called extend in vSphere. Avoid doing this as it complicates operations. That's why in the preceding diagram I show a 1:1 relationship.

The discrepancy between VM layer and Guest OS utilization happens because each layer works differently.

- If there is RDM or thick VMDK, VM can't see the actual used inside Guest OS. It simply sees 100% used, regardless of what Windows or Linux uses.
- If there is unmapped block, Guest OS can't see this overhead.

We are interested in data both at the VM aggregate level, and at the individual virtual disk level. If you are running a VM with a large data drive (for example, Oracle database), the performance of the data drive is what the VM owner cares about the most. At the VM level, you get the average of all drives; hence, the performance issue could be obscured.

Metric Mapping

While there is a 1:1 mapping between Guest OS and its underlying VM, not all metrics map. The following table explains it:

Type	Guest OS Metrics	VM Metrics
Contention	OS Queue	None, as VM is not an OS. Because it is a motherboard, the driver is specific to the Guest OS.
	Driver Queue	
	Latency	Latency. They should be similar, especially when plotted over time.
Utilization	IOPS	They should match with the metrics at VM level, especially when plotted over time. If not, there is something wrong.
	Throughput	

Performance

Latency can happen when IOPS and throughput are not high, because there are multiple stacks involved and each stack has their own queue. It begins with a process, such as a database, issuing IO request. This gets processed by Windows or Linux storage subsystem, and then send to the VM storage driver.

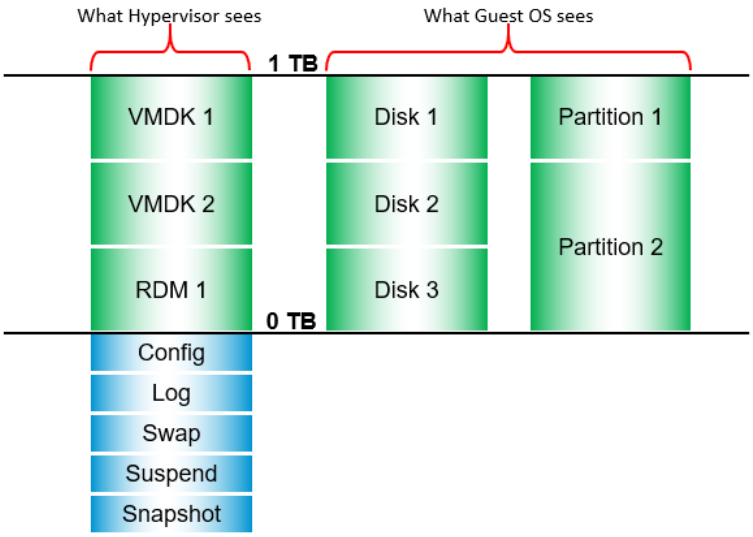
Ensure that you do not have packet loss for your IP Storage, dropped FC frames for FC protocol, or SCSI commands aborted for your block storage. They are a sign of contention as the datastore (VMFS or NFS) is shared. The metrics Bus Resets and Commands Aborted should be 0 all the time. As a result, it should be fine to track them at higher level objects. Create a super metric that tracks the maximum or summation of both, and you should expect a flat line.

Once you have ensured that you do not have packet loss on IP Storage or aborted commands on block storage, use the latency counter and outstanding IO for monitoring. For troubleshooting, you will need to check both read latency and write latency, as they tend to have different patterns and value. It's common to only have read or write issue, and not both.

Total Latency is not (Read Latency + Write Latency) / 2. It is not a simple summation. In a given second, a VM issues many IOPS. For example, the VM issues 100 reads and 10 writes in a second. Each of these 110 commands will have their own latency. The "total" latency is the average of these 110 commands. In this example, the total latency will be more influenced by the read latency, as the workload is read dominated.

If you are using IP storage, take note that Read and Write do not map 1:1 to Transmit (Tx) and Receive (Rx) in Networking metrics. Read and Write are both mapped to Transmit counter as the ESXi host is issuing commands, hence transmitting the packets.

Capacity



A VM can consume storage via:

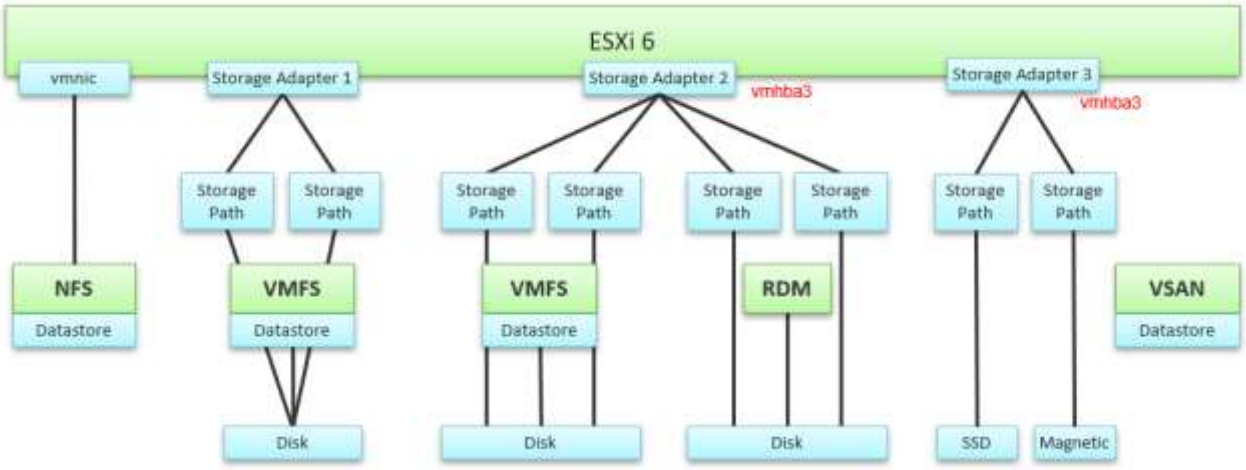
- Virtual disk.
  Each virtual disk has label, type (RDM or VMDK), provisioning type (thin, lazy zero, eager zero). If it's RDM, need to know additional properties such as RDM type (physical or virtual).
- Compute virtualization.
  Snapshots, Swapped, Logs. Guest OS can't see them.This can be overhead and non-overhead. This is not visible to the Guest OS. They are shown in blue in the following diagram.
- Storage virtualization.
  This includes vSAN protection, deduplication and decompression. We need this number to reported separately as it's not applicable in non vSAN.

There are more file types than shown above. However, from monitoring and troubleshooting viewpoint, the above is sufficient.

ESXi Layer

Storage at ESXi is more complex than storage at VM level. Reason is ESXi virtualizes the different physical storage subsystem, and VM simply consumes all of them as local SCSI drive. The kernel does the IO on behalf of all the VMs. It also has its own kernel modules, such as vSAN.

Typically, multiple VMs run on the same ESXi, and multiple ESXi hosts mount a shared datastore. This creates what is popularly termed "IO Blender" effect. Sequential operations from each VM and kernel modules become random when combined together. The opposite is when the kernel rearranges these independent IOs and try to sequence them, so on average the latency is lower.

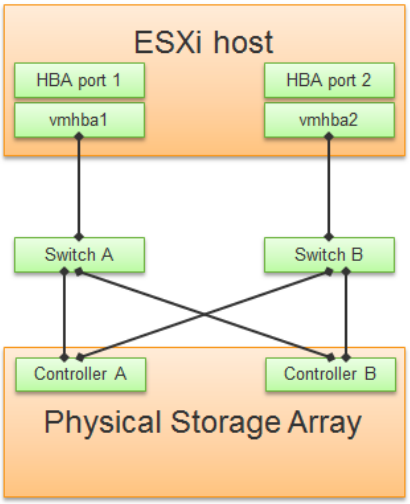


The green boxes are what you are likely to be familiar with. You have your ESXi host, and it can have NFS Datastore, VMFS Datastore, vSAN Datastore, vVOL datastore or RDM objects. vSAN & vVOL present themselves as a VMFS datastore, but the underlying architecture is different. The **blue** boxes represent the metric groups you see in vCenter performance charts.

Just like compute virtualization, there is no more association to VM for metrics at physical layers.

In the central storage architecture, NFS and VMFS datastores differ drastically in terms of metrics, as NFS is file-based while VMFS is block-based.

- For NFS, it uses the vmnic, and so the adapter type (FC, FCoE, or iSCSI) is not applicable. Multipathing is handled by the network, so you don't see it in the storage layer.
- For VMFS or RDM, you have more detailed visibility of the storage. To start off, each ESXi adapter is visible and you can check the metrics for each of them. In terms of relationship, one adapter can have many devices (disk or CDROM). One device is typically accessed via two storage adapters (for availability and load balancing), and it is also accessed via two paths per adapter, with the paths diverging at the storage switch. A single path, which will come from a specific adapter, can naturally connect one adapter to one device. The following diagram shows the four paths:



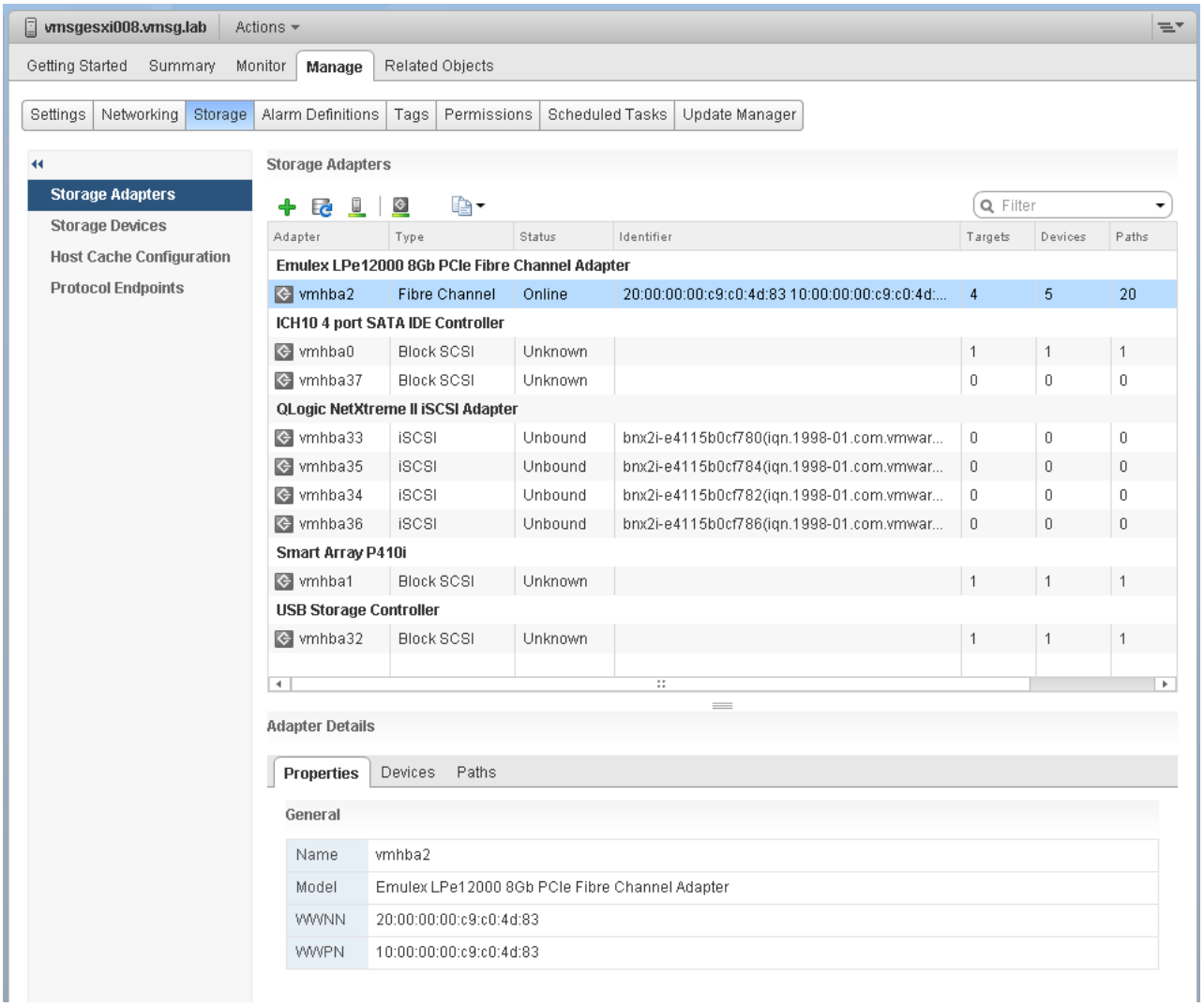
The counter at ESXi level contains data from all VMs and the kernel overhead. There is no breakdown. For example, the counter at vmnic, storage adapter and storage path are all aggregate metrics. It's not broken down by VM. The same with vSAN objects (cache tier, capacity disk, disk group). None of them shows details per VM.

Can you figure out why there is no path to the VSAN Datastore?

We'll do a comparison, and hopefully you will realize how different distributed storage and central storage is from performance monitoring point of view. What look like a simple change has turned the observability upside down.

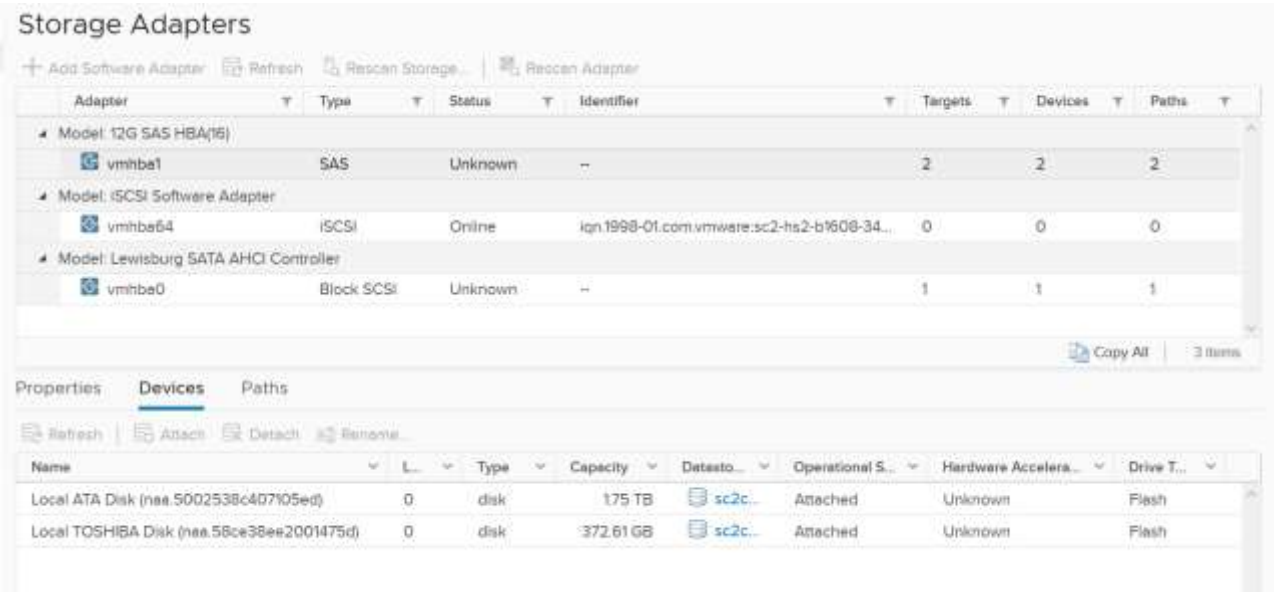
Storage Adapter

The screenshot shows an ESXi host with the list of its adapters. We have selected vmhba2 adapter, which is an FC HBA. Notice that it is connected to 5 devices. Each device has 4 paths, giving 20 paths in total.

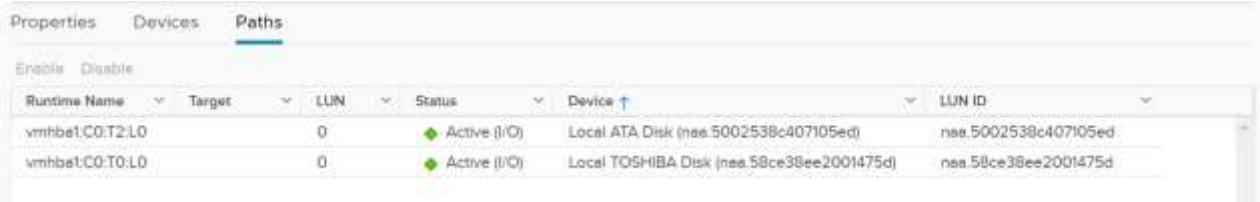


What do you think it will look like on vSAN? The following screenshot shows the storage adapter vmhba1 being used to connect to two vSAN devices. Both devices have names begin with "Local". The storage adapter has 2 targets, 2 devices and 2 paths. If you are guessing it is 1:1 mapping among targets, devices and paths, you are right.

We know vSAN is not part of Storage Fabric, so there is no need for Identifier, which is made of WWNN and WWPN.

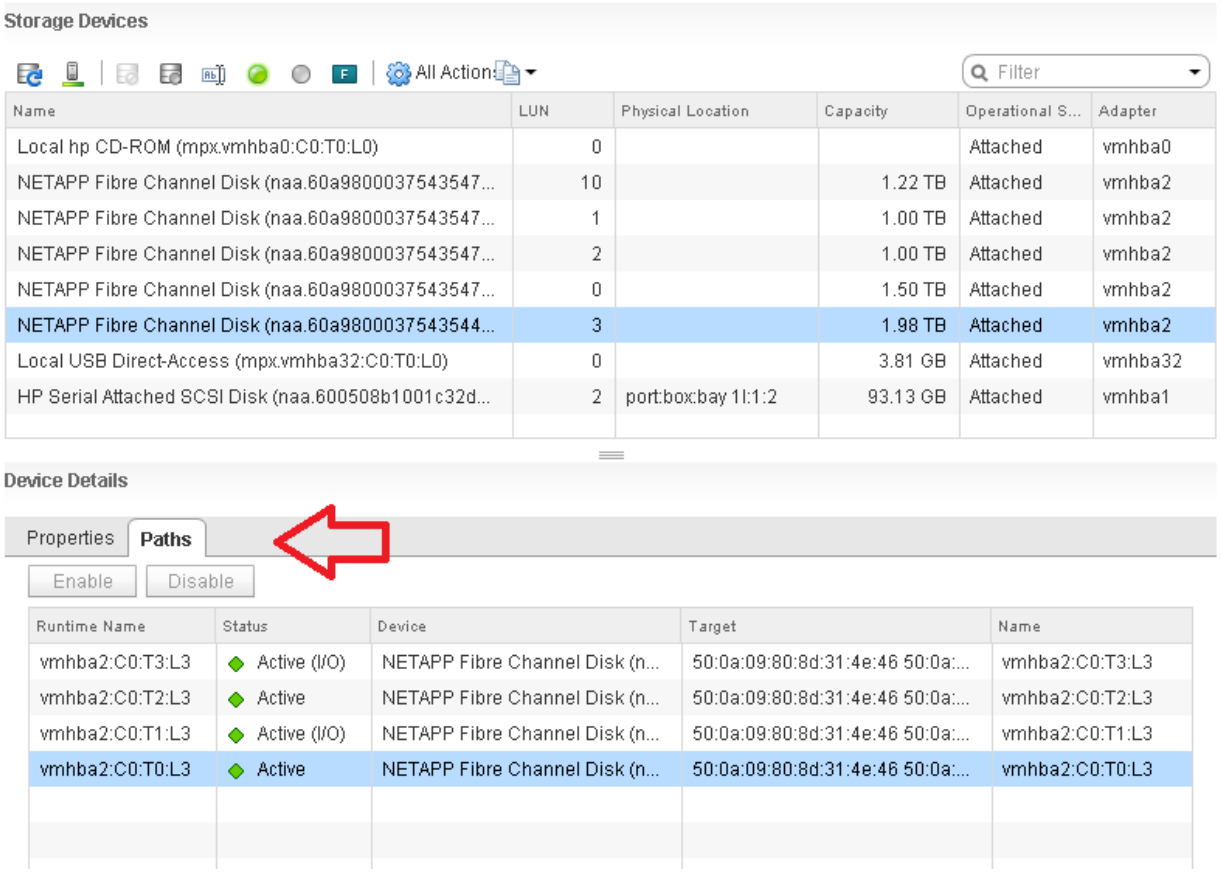


Let's expand the **Paths** tab. We can see the LUN ID here. This is important. The fact that the hypervisor can see the device is important. That means the kernel can report if there is an issue, be it performance or availability. This is different if the disk is directly passed through to the VM. The hypervisor loses visibility.



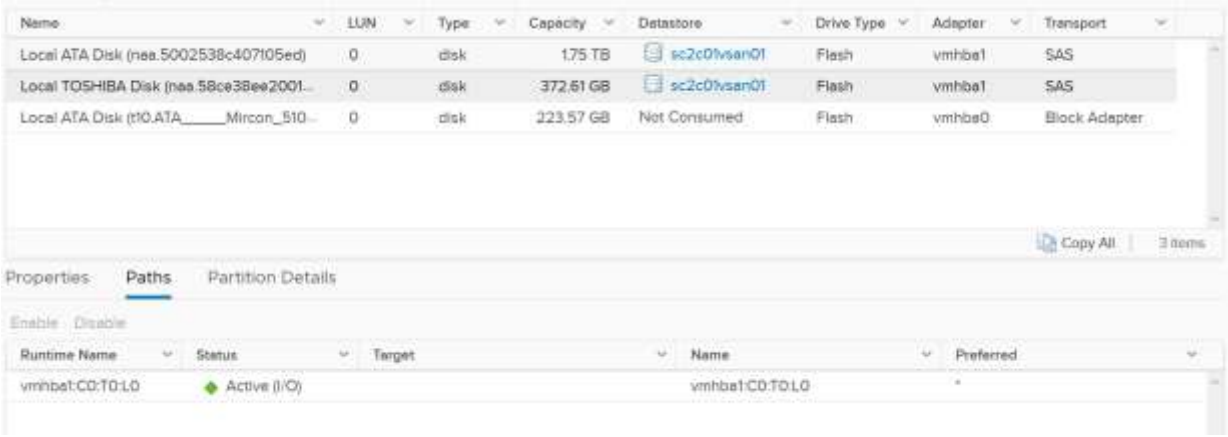
Storage Path

Continuing our comparison, the last one is Storage Path. In a fibre channel device, you will be presented with the information shown in the next screenshot, including whether a path is active or not.



Note that not all paths carry I/O; it depends on your configuration and multipathing software. Because each LUN typically has four paths, path management can be complicated if you have many LUNs.

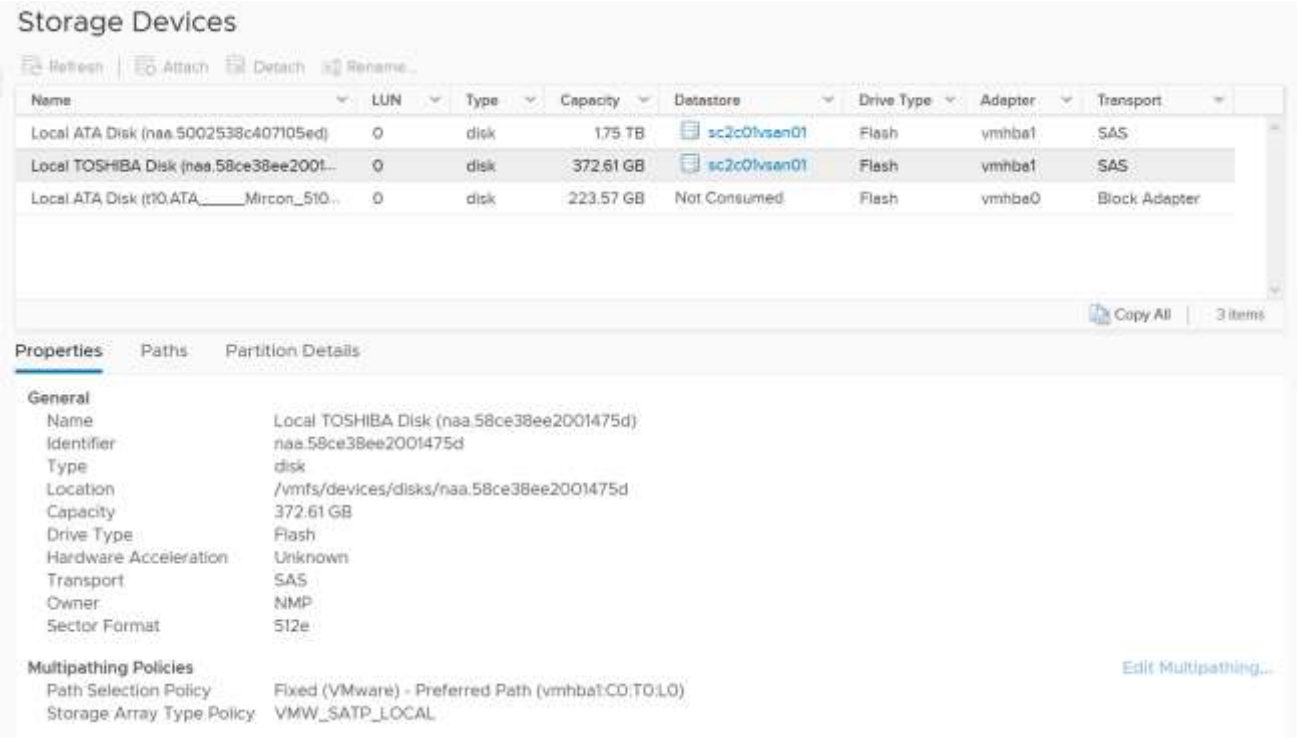
What does Path look like in vSAN? As shared earlier, there is only 1 path.



Storage Devices

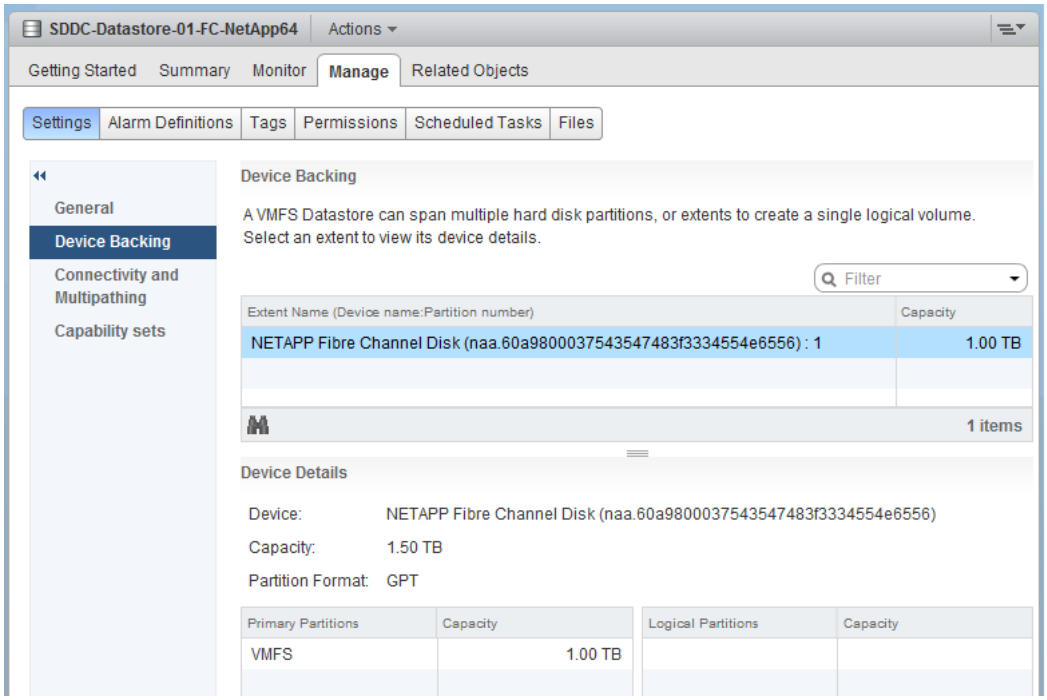
The term drive, disk, device, storage can be confusing as they are often used interchangeably in the industry. vSphere Client uses the terms device and disk interchangeably. In vSphere, this means a physical disk or physical LUN partition presented to ESXi host.

The following shows that the ESXi host has 3 storage devices, all are flash drive and the type = disk. The first two are used in vSAN datastore and are accessed via the adapter vmhba1.



A storage path takes data from ESXi to the LUN (the term used by vSphere is Devices), not to the datastore. So if the datastore has multiple extents, there are four paths per extent. This is one reason why you should not use more than one extent, as each extent adds 4 paths. If you are not familiar with VMFS Extent, Cormac Hogan explains it here.

For VMFS (non vSAN), you can see the same metrics at both the Datastore level and the Disk level. Their value will be identical if you follow the recommended configuration to create a 1:1 relationship between a datastore and a LUN. This means you present an entire LUN to a datastore (use all of its capacity). The following shows a VMFS datastore with a NetApp LUN backing it.



In vSAN, there is no connectivity and Multipathing menu. There is also no Capability Sets menu. vSAN datastore is not mapped to a LUN. It is supported by disk groups.

Datastore Layer

What you can see at this level, and hence how you monitor, depends on the storage architecture.

The underlying storage protocol can be files (NFS) or blocks (VMFS). vSAN uses VMFS as its *consumption* layer as the underlying layer is unique to vSAN, and hence vSAN requires its own monitoring technique. Because vSAN presents itself as a VMFS datastore you need to know that certain metrics will behave differently when datastore type is vSAN.

For VMFS datastore, the VM IO commands are passed down as is to the Pluggable Storage Architecture (PSA) layer. There is one queue (schedQ) made for each vmdk on each VM. The metrics at this layer is are collected at that layer.

For NFS datastore, as it is network file share (as opposed to block), you have no visibility to the underlying storage. The type of metrics will also be more limited, and network metric becomes more critical.

Relationship

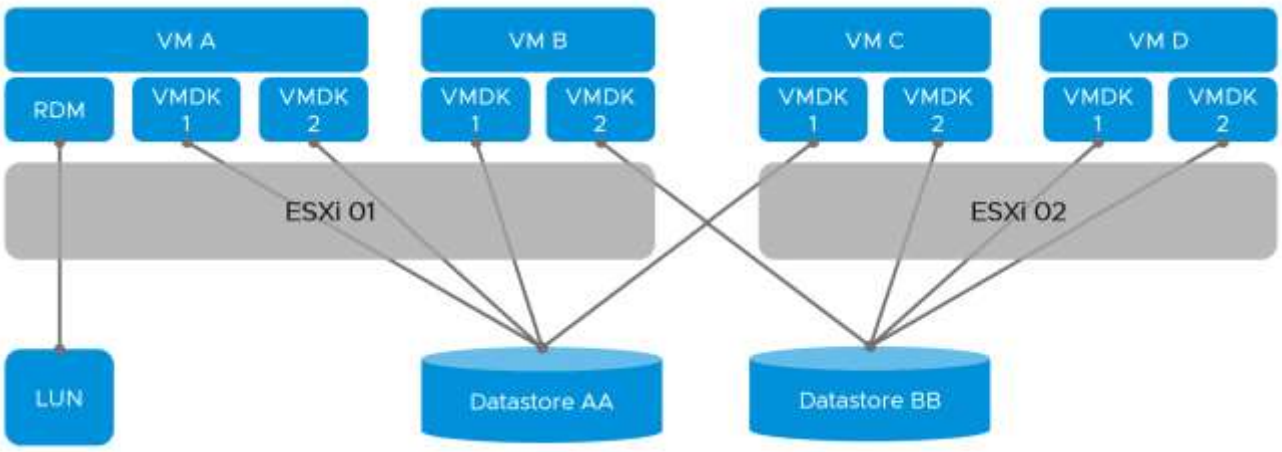
A datastore has relationship to 3 other objects:

- VM
- ESXi
- Cluster

A datastore typically has 1:M relationship to VM. It is also typically shared by multiple ESXi. If you design such that a VM spans multiple datastores, and a datastores spans multiple clusters, you create a trade-off in terms of observability.

The value of datastore metric excludes VMDK that is *not* on the datastore. It includes every files in the datastore, including orphaned files outside the VM folder. Logically, it also excludes RDM.

I created a simple diagram below, with just 4 VMs on 2 ESXi and 2 datastores. What complications do you see?



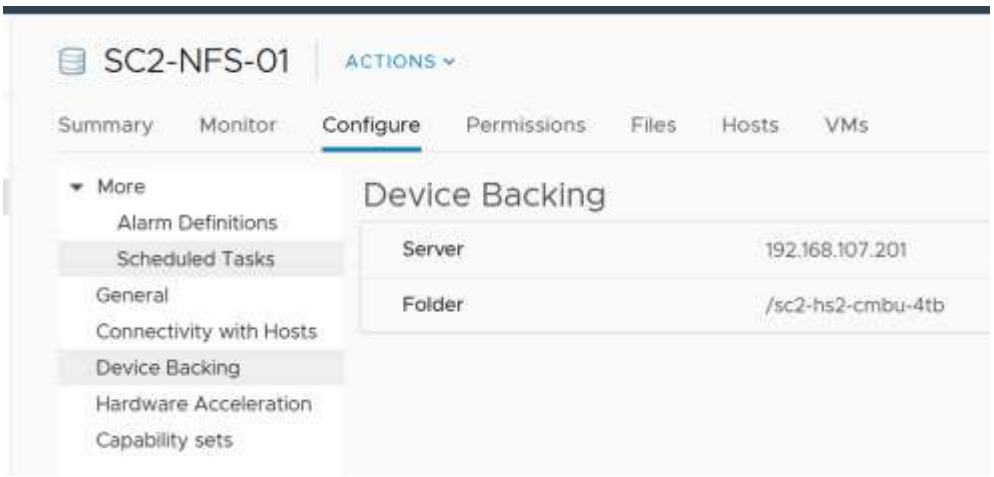
Performance and capacity become complex due to many to many relationships. The metrics at ESXi level and cluster will not match the metrics at datastore level. How do you then aggregate the cluster storage capacity when its datastores are shared with other clusters?

You're right. You can't.

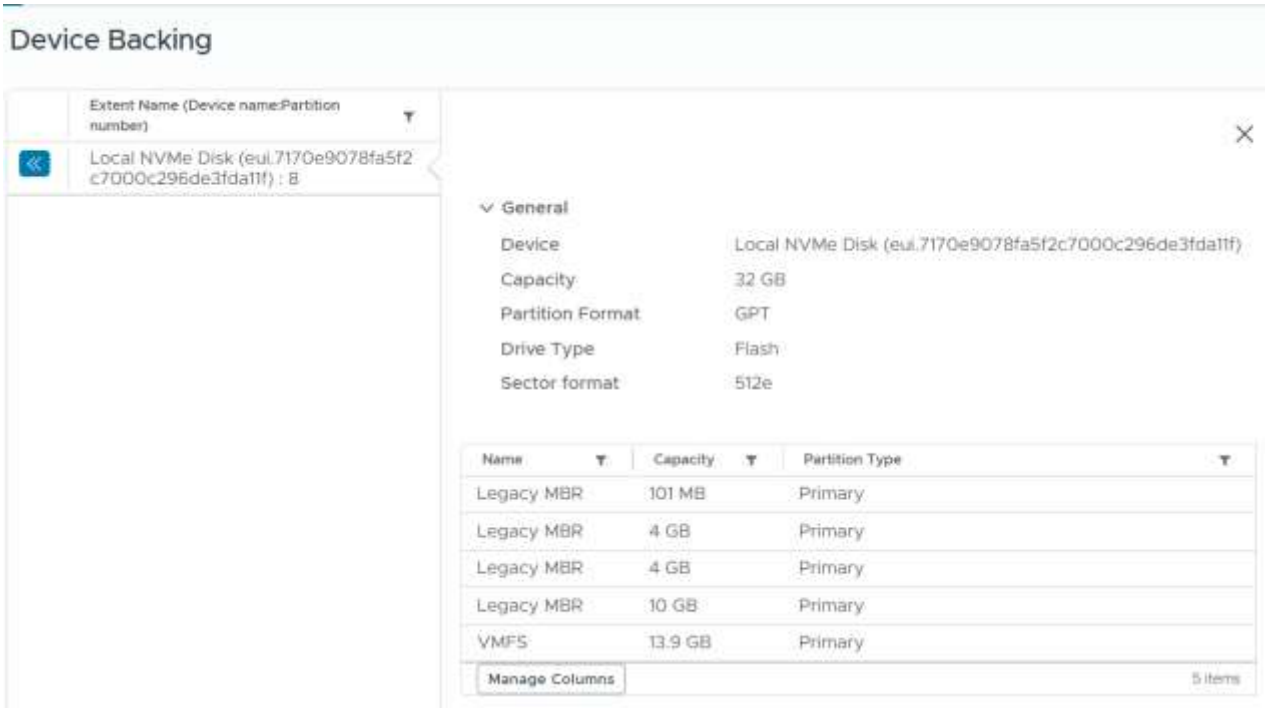
In summary, while there are use cases where you should separate the VMDK into multiple datastores, take note of the observability compromise.

Backing Device

Since datastore is a filesystem, it's necessary to monitor the backing device. This can be NFS or LUN. For NFS, it looks something like this:



For a local disk, it looks something like this:



Since the underlying device is outside the realm of vSphere, you need to login to the storage provider and build the relationship. Compare the metrics by deriving a ratio. Investigate if this ratio shows unexpected value.

Take for example, a datastore on a FC LUN. If you divide the IOPS at the LUN level with the IOPS at the datastore, what value do you expect?

Assuming they are mapped 1:1, then the ratio should be 1.

If the value is > 1, that means there are IO operations performed by the array. This could be array level replication or snapshot.

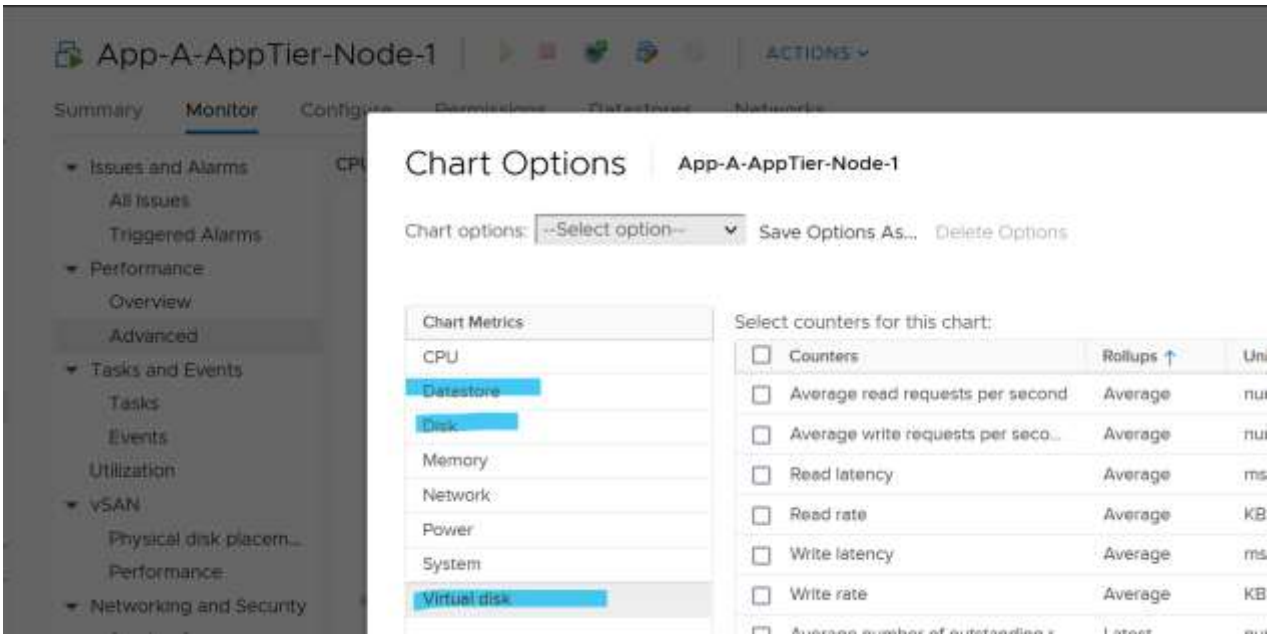
What about NFS datastores? The troubleshooting is different as you now need to at files as opposed to block. In both cases, you need to monitor the filer or array directly.

VM

We covered earlier that storage differs to compute as it covers both dimensions (speed and space). As a result, we cannot simply use the contention and consumption as grouping. Instead we would group by performance and capacity. This is also good as operationally you manage performance and capacity differently.

Overview

Recall the 3 layers of storage from VM downward. As stated, the 3 blue boxes appear in the vSphere Client UI as virtual disk, datastore and disk.



Among the 3, which one is the most important?

You're right, virtual disk.

It is the closest to the VM and it is the most detail in terms of observability.

Virtual Disk

Use the virtual disk metrics to see VMFS vmdk files, NFS vmdk files, and RDMs.

However, you don't get data for anything other than virtual disk. For example, if the VM has snapshot, the metric does not include the snapshot data.

A VM typically has multiple virtual disks, typically 1 Guest OS partition maps to 1 virtual disk. The following VM has 3 virtual disks.



As you can see in the preceding screenshot of vSphere Client UI, there is no aggregate number at VM level. You need to add them manually in vCenter. In VCF Operations, you use the "aggregate of all instances" metric to see the rest.

The following properties is available in VCF Operations for each virtual disk:

Property Name	Values
Virtual Device Node	Virtual disks SCSI bus location. Virtual disks are enumerated starting with the first controller and moving along the bus.
Compatibility Mode	Physical Virtual Virtual mode specifies full virtualization of the mapped device. Physical mode specifies minimal SCSI virtualization of the mapped device.
Disk Mode	Dependent Independent – Persistent Independent – Nonpersistent
SCSI Bus Sharing	None Physical Virtual
SCSI Controller Type	BusLogic Parallel LSI Logic Parallel LSI Logic SAS VMware Paravirtual
Encryption Status	
Is RDM	true false False means the virtual disk is a VDMK not RDM.
Virtual Disk Sharing	Unspecified

	No Sharing
	Multi-Writer

The property "Number of VDMK" excludes RDM, as the name implies. The metric "Number of RDMs" only includes RDM attached to the VM.

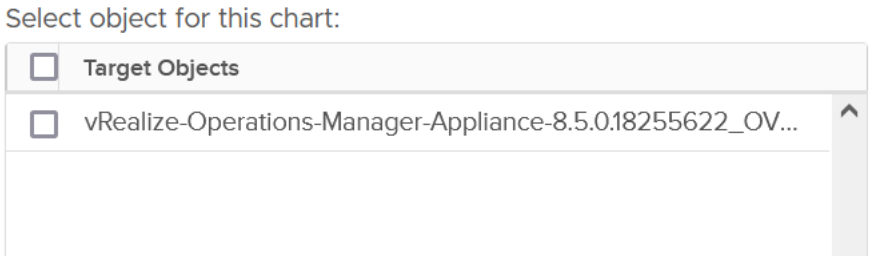
Pro Tip: sum the property "Number of RDMs" from all the VMs in a single physical storage array. Compare the result with the number of LUNs in the array that are carved out for RDM purpose. If there are more LUNs than this number, you have unused RDM.

You need to do the above per physical array, so you know which array needs attention.

Disk

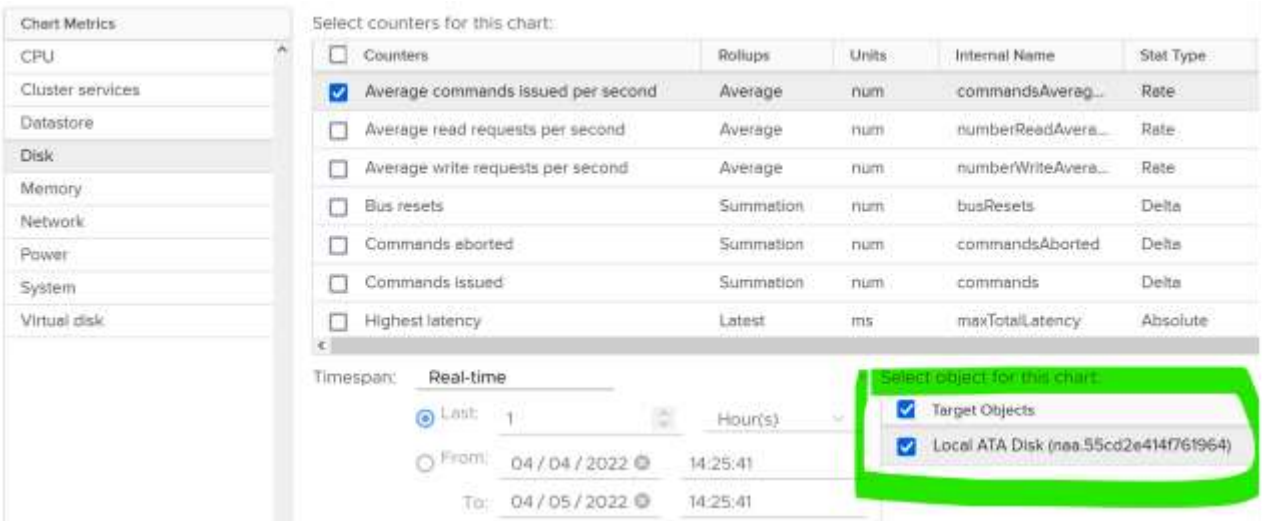
This should be called Physical Disk or Device, as a simple terminology "disk" sounds like a superset of virtual disk.

Disk means device, so we're measuring at LUN level or RDM level. It's great to know that we can associate the metrics back to the VM. Notice we can't associate it to specific virtual disk as they are different layers.



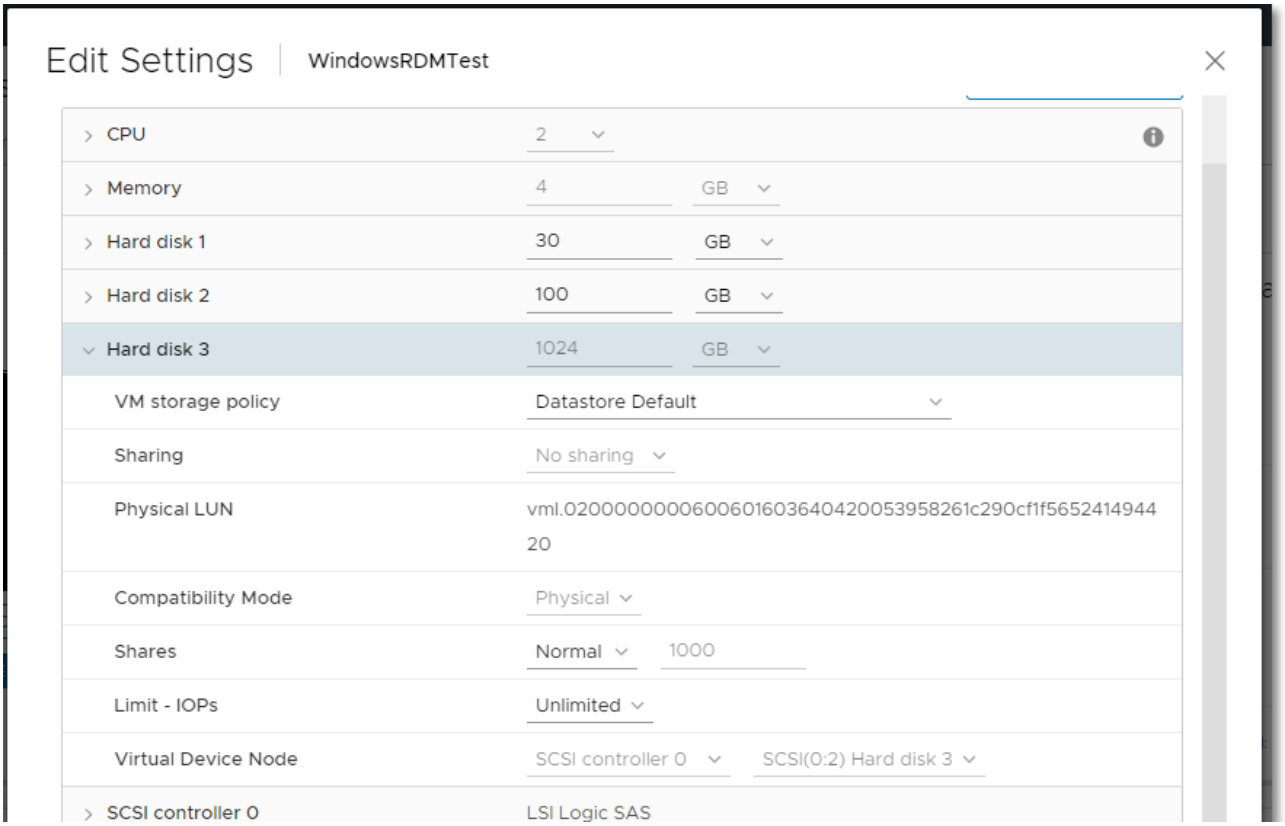
Use the disk metrics to see VMFS and RDM, but not NFS. The data at this level should be the same as at Datastore level because your blocks should be aligned; you should have a 1:1 mapping between Datastore and LUN, without extents. It also has the Highest Latency counter, which is useful in tracking peak latency

The metric is at the disk level. So I'm not 100% sure if the value is per VM or per disk (which typically has many VM).



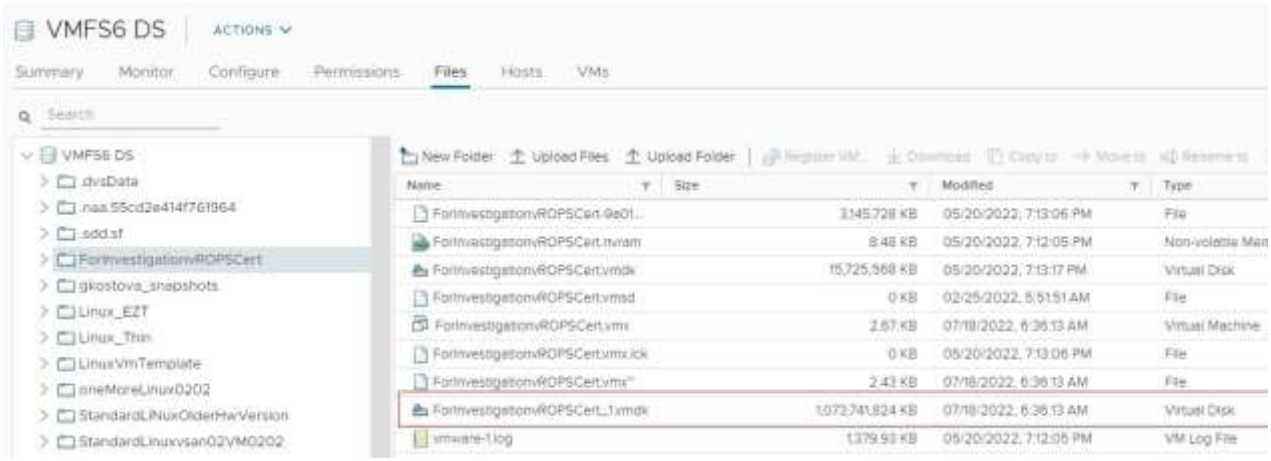
Raw Device Mapping

RDM appears clearly as LUN in the VM Edit Settings dialog box:



But what does it appear when you browse the VM folder in the parent datastore?

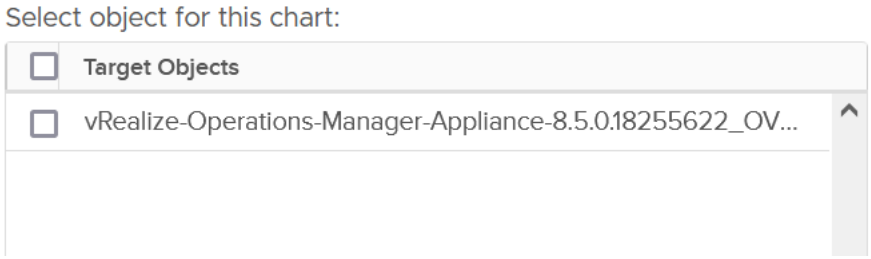
RDM appears like a regular VMDK file. There is no way to distinguish it in the folder.



Datastore

Use the datastore metrics to see VMFS and NFS, but not RDM. Because snapshots happen at Datastore level, the counter will include it. Datastore figures will be higher if your VM has a snapshot. You don't have to add the data from each virtual disk together as the data presented is already at the VM level. It also has the Highest Latency counter, which is useful in tracking peak latency.

Just like LUN level, we lose the breakdown at virtual disk. The metric is only available at VM level.



Mapping

If all the virtual disks of a VM are residing in the same datastore, and that datastore is backed by 1 LUN, then all the 3 layers will have fairly similar metrics. The following VM has 2 virtual disks (not shown). Notice all 3 metrics are identical over time.

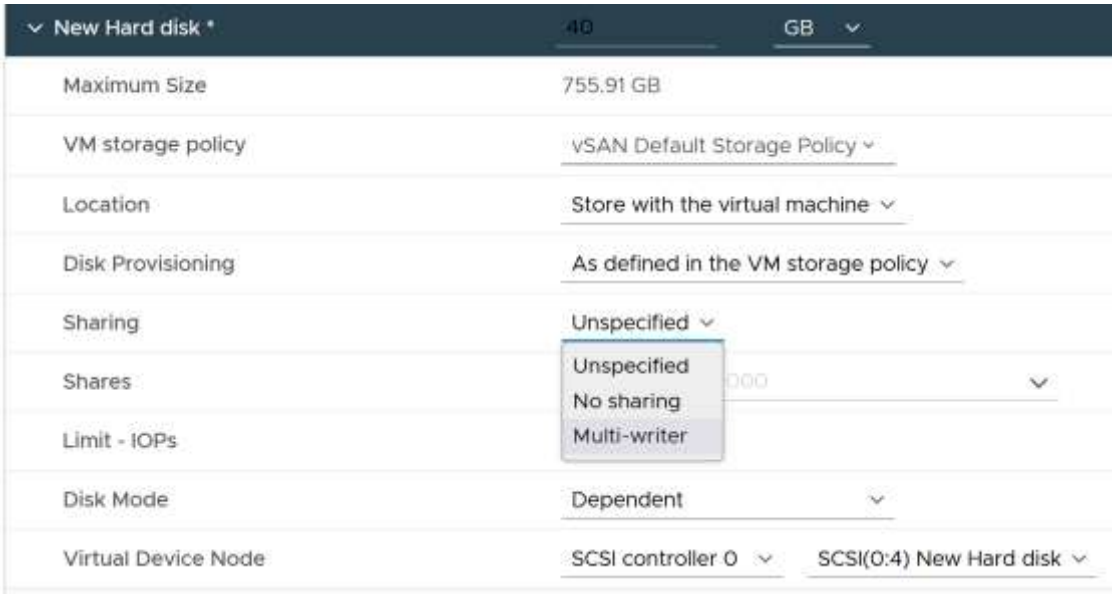


The difference comes from files outside the virtual disks, such as snapshot, log files, and memory swap.

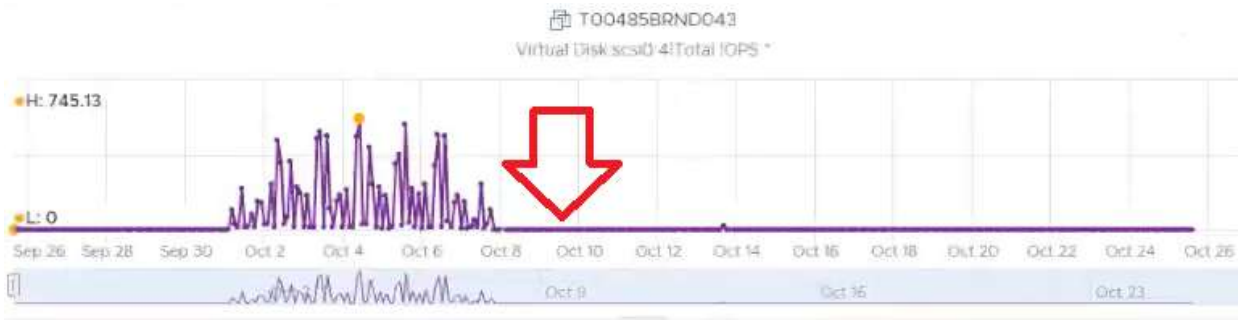
Multi-Writer Disk

In application such as database, multiple VMs need to share the same disk.

Shared disk can be either shared RDM or VMDK. The following screenshot shows the option when creating a multi-writer VMDK in vCenter Client.



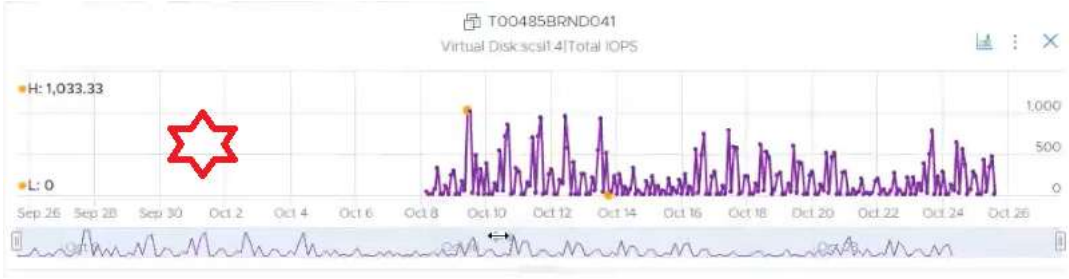
When multiple VMs are sharing the same virtual disk or RDM, it creates additional challenge in capacity, cost and performance management. In the following example, notice the metric become flat 0. See the red arrow.



The above is obviously wrong as IOPS are typically not flat 0.

What happens here is typical Active/Passive pair of VM. The application fail over to the second VM, so the first VM becomes passive. vCenter API returns 0 instead of blank, hence you see 0 in VCF Operations.

You can see from the following screenshot that the second VM took over at the time the first VM failover. Notice it started showing IOPS metrics on the same time.



Do you notice something inconsistent?

The first VM returns 0 after failover. The second VM returns blank (no data) before taking over.

Metrics

Name	Description
Disk Space | Active Not Shared (GB)	The total amount of disk space from all the VMDK and RDM that are exclusively owned by this VM. Active only cover the virtual disks. Snapshot is considered as non-active files hence it's not counted. Formula: Disk Space|Not Shared (GB) - Disk Space|Snapshot Space (GB)

Performance Metrics

Just like CPU and memory, we would cover contention type of metric first, then the consumption type of metrics.

Contention Metrics

Contention could happen due to the VM itself (e.g. IOPS Limit has been placed) or the underlying infrastructure.

We will cover them from virtual disk first, then datastore and disk.

Virtual Disk

The main metrics for tracking performance is latency. They are provided in both ms and microsecond.

Read latency	Average	ms	The average time a read from the virtual disk takes
Write latency	Average	ms	The average time a write to the virtual disk takes
Read Latency (us)	Latest	µs	Read latency in microseconds
Write Latency (us)	Latest	µs	Write latency in microseconds

A related counter to latency is Outstanding IO.

This is the number of I/Os that have been issued, but not yet completed. They are waiting in the queue, indicating a bottleneck

Average number of outstanding read requests	Latest	num	Average number of outstanding read requests to the virtual d
Average number of outstanding write requests	Latest	num	Average number of outstanding write requests to the virtual c

The relationship is

```
Average Latency = Average Outstanding IO / Average IOPS
```

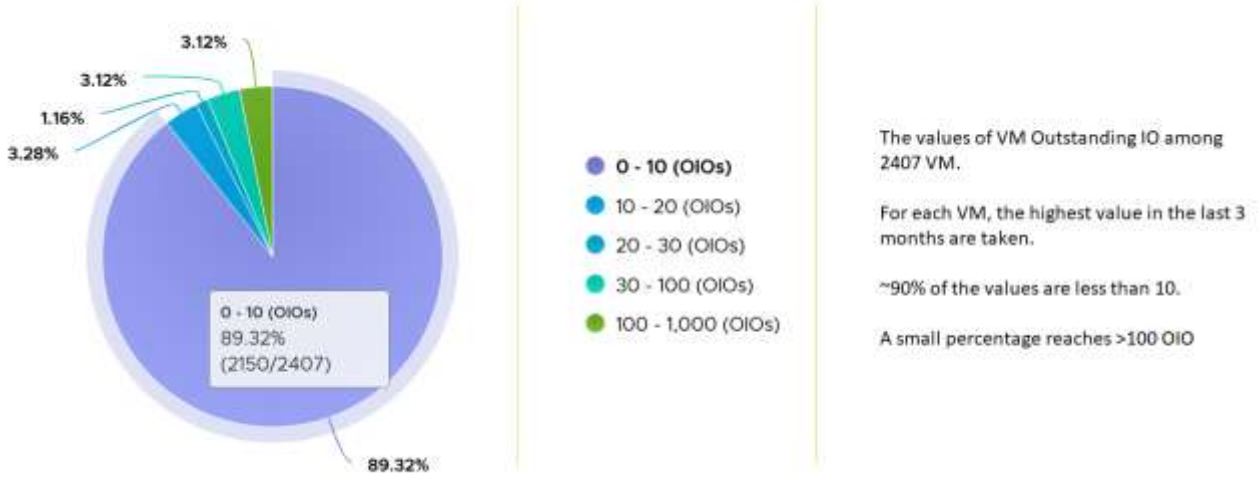
Outstanding IO should be seen in conjunction with latency. It can be acceptable to have high number of IO in the queue, so long the actual latency is low.

Since your goal is maximum IOPS and minimum latency, the metric is less useful as its value is impacted by IOPS. See this KB article for VSAN specific recommendation on the expected value.

What should be the threshold value?

That depends on your storage, because the range varies widely. Use the profiling technique to establish the threshold that is suitable for your environment.

In the following analysis, we take more than 63 million data points (2400 VM x 3 months worth of data). Using data like this, discuss with the storage vendor if that's in line with what they sold you.



Disk

As the physical disk layer, there are 2 error metrics. I always find their values to be 0 all the time, so if you've seen a non-zero value let me know.

Bus resets	Summation	num	Number of SCSI-bus reset commands issued during the collection interval
Commands aborted	Summation	num	Number of SCSI commands aborted during the collection interval

For latency, there is no breakdown. It's also the highest among all disks. Take note the roll-up is latest, so it's the single value at the end of the collection period.

Highest latency	Latest	ms	Highest latency value across all disks used by the host

Datastore

At the datastore layer, the only metric provided for contention is latency. There is no outstanding IO.

Counters	Rollups	Units ↓	Stat Type	Description
Highest latency	Latest	ms	Absolute	Highest latency value across all datastores used by the host
Read latency	Average	ms	Absolute	The average time a read from the datastore takes
Write latency	Average	ms	Absolute	The average time a write to the datastore takes

The highest latency is useful for VMs with multiple datastores. But take note the roll-up is Latest, not average.

For the read and write latency, the value in VCF Operations is a raw mapping to these values datastore.totalReadLatency.average and datastore.totalWriteLatency.average

Consumption metrics

A typical suspect for high latency is high utilization, so let's check what IOPS and throughput metrics are available.

Virtual Disk

As you can expect, you're given both IOPS and throughput metrics at virtual disk level.

Counters	Rollups	Units	Description
Average read requests per second	Average	num	Average number of read commands issued per second to the virtual dis
Average write requests per second	Average	num	Average number of write commands issued per second to the virtual di
Read rate	Average	KBps	Rate of reading data from the virtual disk
Write rate	Average	KBps	Rate of writing data to the virtual disk

VM Disk IOPS and throughput vary widely among workload. For a single workload or VM, it also depends on whether you measure during its busy time or quiet time.

Take note that vSphere Client does not provide summary at VM level. Notice the target objects are individual scsiM:N, and there is no aggregation at VM level as the option in Target Objects column below.



In the following example, I plotted from a 3500 production VMs. They are sorted by the largest IOPS on any given 5 minute. What's your take?

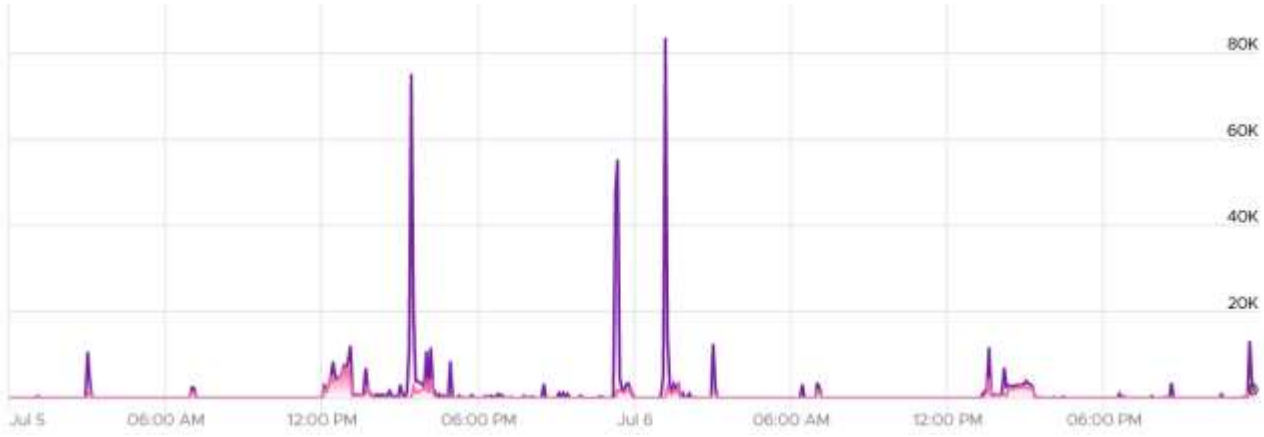
Name	Bursty IOPS (5 minute)	Sustained IOPS (1 hour)	Throughput (1 hour)	Throughput (5 minute)	vCPU	vDisks
CO...	83,252.4	11,452.97	4.74 Gbps	11.94 Gbps	14	8
mg...	80,345.47	28,663.92	1 Gbps	2.85 Gbps	6	5
cus...	67,323.53	16,539.19	7.85 Gbps	32.48 Gbps	16	33
FIN...	58,115.46	23,226.41	-	-	4	5
mg...	58,043.54	11,099.42	0.74 Gbps	2.66 Gbps	4	7
gss...	53,387.27	4,985.2	0.68 Gbps	6.27 Gbps	12	3
myl...	51,759.8	35,013.87	6.16 Gbps	17.47 Gbps	8	6
ora...	44,594.2	31,705.37	22.93 Gbps	33.91 Gbps	24	26
cp-...	41,501.07	12,253.99	0.05 Gbps	0.48 Gbps	4	1

I think those numbers are high. At 1000 IOPS averaged over 5 minutes, that means 300,000 total IO commands that need to be processed. So 10K IOPS translates into 3 millions commands, which must be completed within 300 seconds.

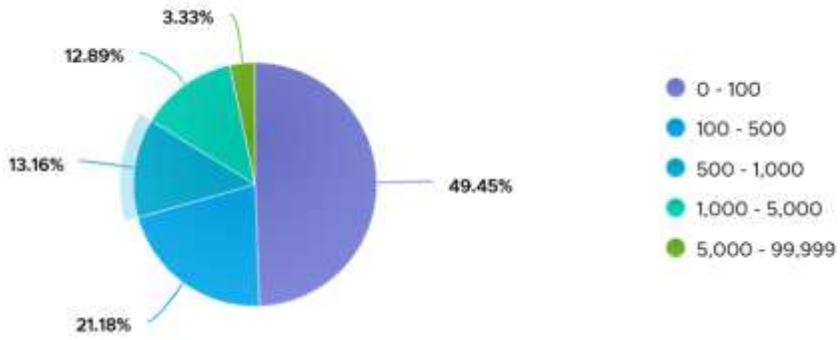
A high IOPS can also impact the pipe bandwidth, as it's shared by many VMs and the kernel. If a single VM chews up 1 Gb/s, you just need a handful of them to saturate 10 Gb ethernet link.

There is another problem, which is sustained load. The longer the time, the higher the chance that other VMs are affected.

In the following example, it's a burst IOPS. Regardless, discuss with the application team if it is higher than expected. What's normal from one application may not be for another.



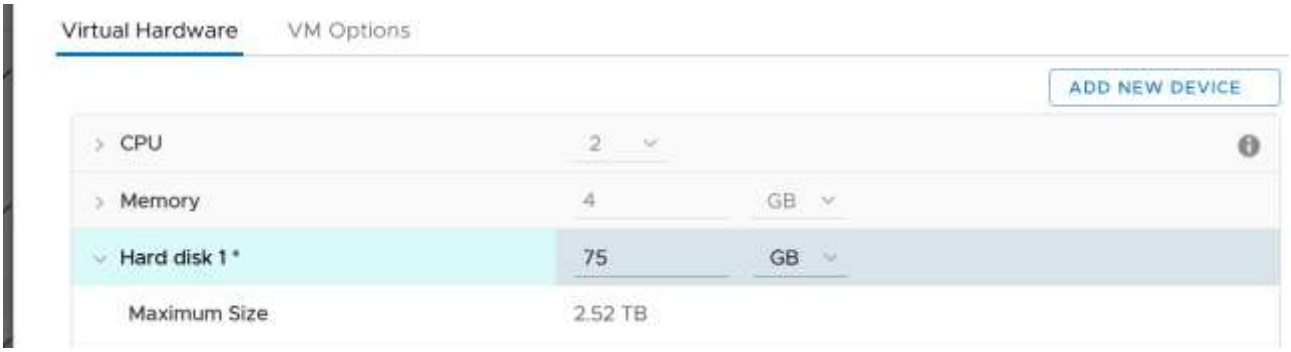
While there is no such thing as normal distribution or range, you can analyse your environment so you get a sense. I plotted all the 3500 VMs and almost 85% did not exceed 1000 IOPS in the last 1 week. The ones hitting >5K IOPS only form around 3%.



If the IOPS is low, but the throughput is high, then the block size is large. Compare this with your expected block size, as they should not deviate greatly from plan. You do have a plan, don't you 😉

Read request size	Latest	num	Average read request size in bytes
Write request size	Latest	num	Average write request size in bytes

You can set the limit for individual virtual disk of VM.



A few rows below, and you will see the following.



The default setting is no limit, which is what I recommend.

Note that the limit on a virtual disk, not the whole VM. That means you cannot set limit on non virtual disk such as snapshot and memory swap. This makes sense as they are part of IaaS.

Take note that since the limit is applied at VM level, the metrics that will show high latency is at Guest OS levels. The VM metric will not show high latency, as the IO that were allowed to pass was not affected by this limit. This is no different to any problem at Guest OS layer. For example, if LSI Logic or PVSCSI driver is causing problem, the VM will not report anything as it's below the Guest OS driver.

VCF Operations have the following related data at each virtual disk

- IOPS Limit property.
- IOPS per GB metric.

Disk

There are 2 sets of metrics for IOPS. Both are basically the same. One if the total number of IO in the collection period, while the other one is average of 1 second.

Counters	Rollups	Units	Description
Average commands issued per second	Average	num	Average number of SCSI commands issued per second during the collection interval
Average read requests per second	Average	num	Average number of disk reads per second during the collection interval
Average write requests per second	Average	num	Average number of disk writes per second during the collection interval
Commands issued	Summation	num	Number of SCSI commands issued during the collection interval
Read requests	Summation	num	Number of disk reads during the collection interval
Write requests	Summation	num	Number of disk writes during the collection interval

There are the usual metrics for throughput.

Read rate	Average	KBps	Average number of kilobytes read from the disk each second during the collection interval
Usage	Average	KBps	Aggregated disk I/O rate. For hosts, this metric includes the rates for all virtual machines running on the host during the collection interval.
Write rate	Average	KBps	Average number of kilobytes written to disk each second during the collection interval

It will be great to have block size, especially the maximum one during the collection period.

Datastore

For utilization, both IOPS and throughput are provided.

Counters	Rollups	Units	Stat Type	
Average read requests per second	Average	num	Rate	Average number of read commands issued per second to the datastore during the collection interval
Average write requests per second	Average	num	Rate	Average number of write commands issued per second to the datastore during the collection interval
Read rate	Average	KBps	Rate	Rate of reading data from the datastore
Write rate	Average	KBps	Rate	Rate of writing data to the datastore

For the IOPS, the value in VCF Operations is a raw mapping to these values datastore.numberReadAveraged.average and datastore.numberWriteAveraged.average in vCenter.

Review the following screenshot. Notice something strange among the 3 metrics?



Yes, the total IOPS at datastore level is much lower than the IOPS at physical disk and virtual disk levels. The IOPS at physical disk and virtual disk are identical over the last 7 days. They are quite active.

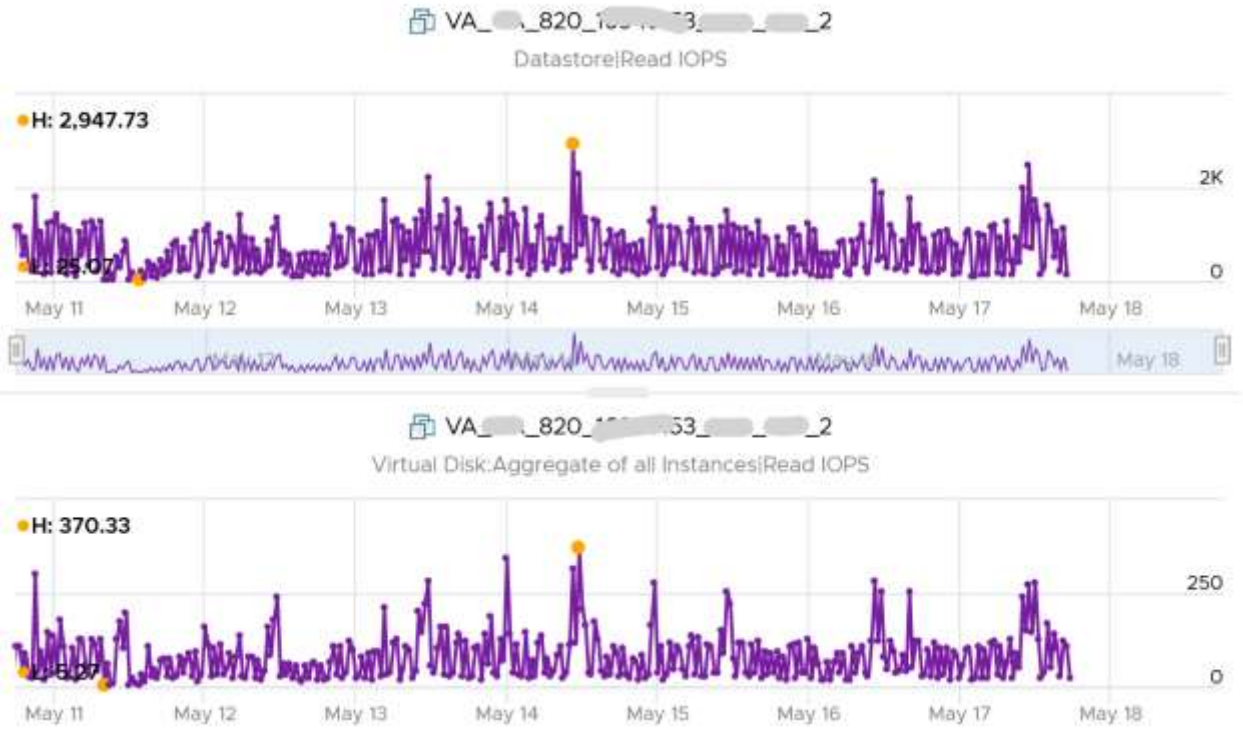
The IOPS at datastore level is much lower, and only spike once a day. This VM is an Oracle EBS VM with 26 virtual disks. Majority of its disks are RDM, hence the IOPS hitting the datastore is much less.

Snapshot requires additional read operations, as the reads have to be performed on all the snapshots. The impact on write is less. I'm not sure why it goes up so high, but logically it should be because many files are involved. Based on the manual, a snapshot operation creates .vmdk, -delta.vmdk, .vmsd, and .vmsn files. Read more here.

For Write, ESXi just need to write into the newest file.

Name	vDisk Reads/sec	Datastore Reads/sec	vDisk Writes/sec	Datastore Writes/sec	Snapshot Size	Snapshot Age
ev...	593.2	594.2	101.33	101.8	0 GB	-
vR...	197.27	199.2	148.4	150.6	0 GB	-
VA...	164.27	720.67	93.4	171.13	30.45 GB	38
vR...	156.67	158.33	154.4	156.87	0 GB	-
vR...	151.73	153.53	142.13	144.33	0 GB	-
vR...	130.07	131.73	138.67	141.07	0 GB	-
VA...	95.93	1,064.13	36.4	60.33	58.9 GB	47
VA...	60.2	60.53	14.47	15.27	0 GB	-
VA...	55.87	472.93	27.8	39.27	53.07 GB	47
VA...	41.4	88.93	26.6	32.93	36.47 GB	47

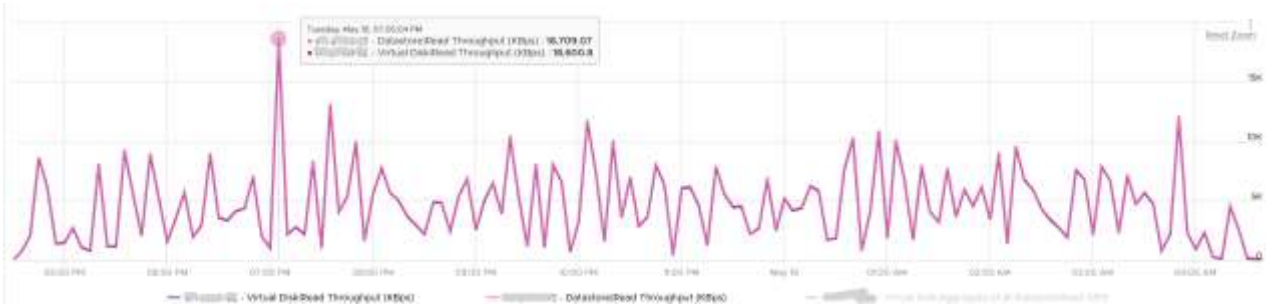
The pattern is actually identical. I take one of the VM and show it over 7 days. Notice how similar the 2 trend charts in terms of pattern.



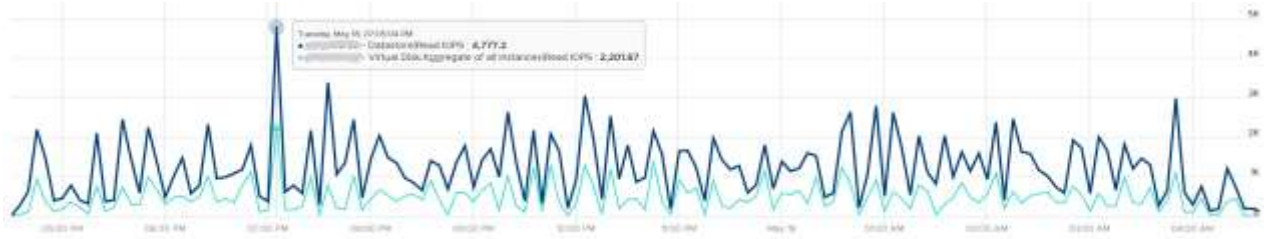
You can validate if snapshot causes the problem by comparing before and after snapshot. That's exactly what I did below. Notice initially there was no snapshot. There was a snapshot briefly and you could see the effect immediately. When the snapshot was removed, the 2 lines overlaps 100% hence you only see 1 line. When we took the snapshot again, the read IOPS at datastore level is *consistently* higher.



How I know that's IOPS effect as the throughput is identical. The additional reads do not bring back any data. Using the same VM but at different time period, notice the throughput at both levels are identical.



And here is the IOPS on the same time period. Notice the value at datastore layer is consistently higher.



For further reading, Sreekanth Setty has shared best practice here.

In addition of latency and IOPS, snapshot can also consume more than the actual space consumed by the virtual disk, especially if you are using thin and you take snapshot early while the disk is basically empty. The following VM has 3 virtual disks, where the snapshot file _1-00001.vmdk is much larger than the corresponding vmdk.

kmargaryan-dual-stack-D-0de45800.hlog	0.28 KB	File
kmargaryan-dual-stack-D.db	9 KB	File
kmargaryan-dual-stack-D.db-journal	3.52 KB	File
kmargaryan-dual-stack-D.vmsd	0.58 KB	File
kmargaryan-dual-stack-D.nvram	8.48 KB	Non-volatile Memory File
kmargaryan-dual-stack-D-000001.vmdk	1,143,808 KB	Virtual Disk
kmargaryan-dual-stack-D.vmdk	4,241,408 KB	Virtual Disk
kmargaryan-dual-stack-D_1-000001.vmdk	77,038,592 KB	Virtual Disk
kmargaryan-dual-stack-D_1.vmdk	9,224,192 KB	Virtual Disk
kmargaryan-dual-stack-D_2-000001.vmdk	1,024 KB	Virtual Disk
kmargaryan-dual-stack-D_2.vmdk	2,148,352 KB	Virtual Disk
kmargaryan-dual-stack-D.vmx	2.73 KB	Virtual Machine
vmware.log	3,826.39 KB	VM Log File
kmargaryan-dual-stack-D-Snapshot2.vmsn	19.29 KB	VM Snapshot

Storage DRS

Lastly, there are storage DRS metric and seek size.

Number of large seeks	Latest	num	Number of seeks during the interval that were greater than 8192 LBNs apart
Number of medium seeks	Latest	num	Number of seeks during the interval that were between 64 and 8192 LBNs apart
Number of small seeks	Latest	num	Number of seeks during the interval that were less than 64 LBNs apart
Read workload metric	Latest	num	Storage DRS virtual disk metric for the read workload model
Write workload metric	Latest	num	Storage DRS virtual disk metric for the write workload model

Capacity Metrics

Disk space metrics are complex due the different types of consumption in a single Virtual Disk.

- Actual used by Guest OS
- Unmapped block
- vSAN protection (FTT)
- vSAN savings (dedupe and compressed).

Let's break it down, starting with understanding the files that make up a VM.

VM Files

At the end of the day, all those disk space appear as files in the VMFS filesystem, including the RDM pointer files. You can see them when you browse the datastore. The following is a typical example of what vSphere Client will show.

Name	Size	Type
bitnami-pimcore-h1-36bf279d.hlog	0.42 KB	File
bitnami-pimcore-h1-4d7460c5.vswp	0.26 KB	File
bitnami-pimcore-h1-4d7460c5.vswp.lck	0 KB	File
bitnami-pimcore-h1.vmsd	0 KB	File
bitnami-pimcore-h1.vmx	3.17 KB	File
bitnami-pimcore-h1.vmx.lck	0 KB	File
bitnami-pimcore-h1.vmx~	3.17 KB	File
vmx-bitnami-pimcore-h1-be20ac105ac193dda7c3450de0094e70c8992809-1.vswp	81,920 KB	File
.sdd.sf		Folder
bitnami-pimcore-h1.nvram	8.48 KB	Non-volatile Memory File
bitnami-pimcore-h1.vmdk	13,520,896 KB	Virtual Disk
vmware-1.log	147.43 KB	VM Log File
vmware-2.log	158.77 KB	VM Log File
vmware-3.log	227.79 KB	VM Log File
vmware-4.log	335.87 KB	VM Log File
vmware.log	4,505.53 KB	VM Log File

Yes, a lot of files 😊

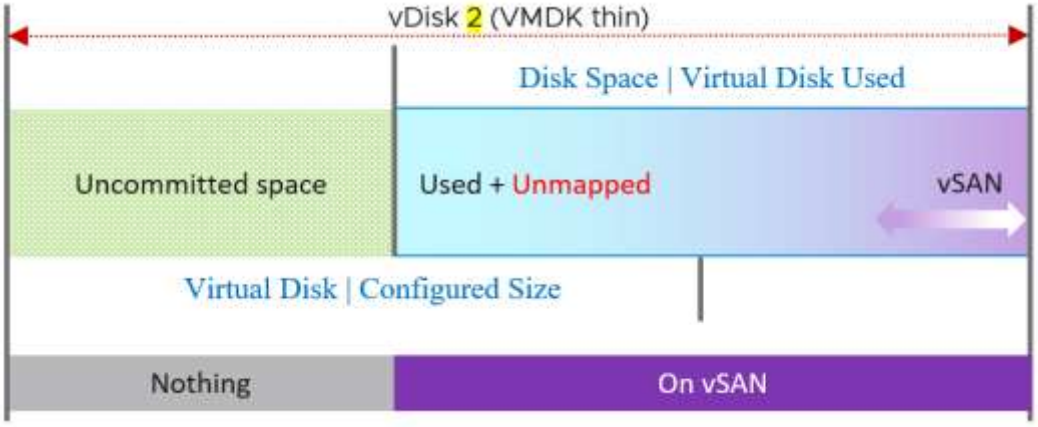
We can categorize them into 4 from operations viewpoint:

Disk	Virtual disk or RDM. This is typically the largest component. This can be thin provisioned, in which case the provisioned size tends to be larger than the actual consumption as Guest filesystem typically does not fill 100%. All virtual disks are made up of two files, a large data file equal to the size of the virtual disk and a small text disk descriptor file which describes the size and geometry of the virtual disk file. The descriptor file also contains a pointer to the large data file as well as information on the virtual disks drive sectors, heads, cylinders and disk adapter type. In most cases these files will have the same name as the data file that it is associated with (i.e. MyVM1.vmdk and MyVM1-flat.vmdk). A VM can have up to 64 disks from multiple datastores.
Snapshot	Snapshot protects 3 things: • VMDK • Memory • Configuration For VMDK, the snapshot filename uses the syntax **MyVM-000001.vmdk** where MyVM is the name of the VM and the six-digit number 000001 is just a sequential number. There is 1 file for each VMDK. Snapshot does not apply to RDM. You do that at storage subsystem instead, transparent to ESXi. If you take snapshot with memory, it creates a **.vmem** file to store the actual image. The **.vmsn** file stores the configuration of the VM. The **.vmsd** file is a small file, less than 1 KB. It stores metadata about each snapshot that is active on a VM. This text file is initially 0 bytes in size until a snapshot is created and is updated with information every

	time snapshots are created or deleted. Only 1 file exists regardless of the number of snapshots running as they all update this single file. This is why your IO goes up. Name / Size a vSAN Space Test-000001.vmdk 36,864 KB a vSAN Space Test.vmx 2.29 KB a vSAN Space Test_1.vmdk 42,815,488 KB a vSAN Space Test-Snapshot1.vmsn 10.39 KB a vSAN Space Test.vmsd 0.42 KB
Swap	The memory swap file (**.vswp**). A VM with 64 GB of RAM will generate a 64 GB swap file (minus the size of memory reservation) which will be used when ESXi needs to swap the VM memory into disk. The file gets deleted when the VM is powered off. You can choose to store this locally on the ESXi Host. That would save space on vSAN. The catch is vMotion as the swap file must be transferred too. There is also a smaller file (in MB) storing the VMX process swap file. But I'm unsure about this and have not seen it yet.
Others	All other files. They are mostly small, in KB or MB. So if this counter is large, you've got unneeded files inside the VM directory. Logs files, configuration files, and BIOS/EFI configuration file (**.nvram**) Note that this includes any other files you put in the VM directory. So if you put a huge ISO image or any file, it gets counted.

Single VMDK

Let's review with a single virtual VMDK disk. In the following diagram, vDisk 2 is a thin provisioned VMDK file. It still has uncommitted space as it's not yet fully used up.



Because vSAN is a software-defined storage, the storage-layer gets mixed up. What operational complexity do you spot from the above diagrams?

- Your unmapped file is also protected by vSAN.
- The uncommitted part does not include vSAN, as it's yet written.

- You can see that the 2 metrics are not aware of vSAN. vSAN protection (Failure To Tolerate) is shown in purple.

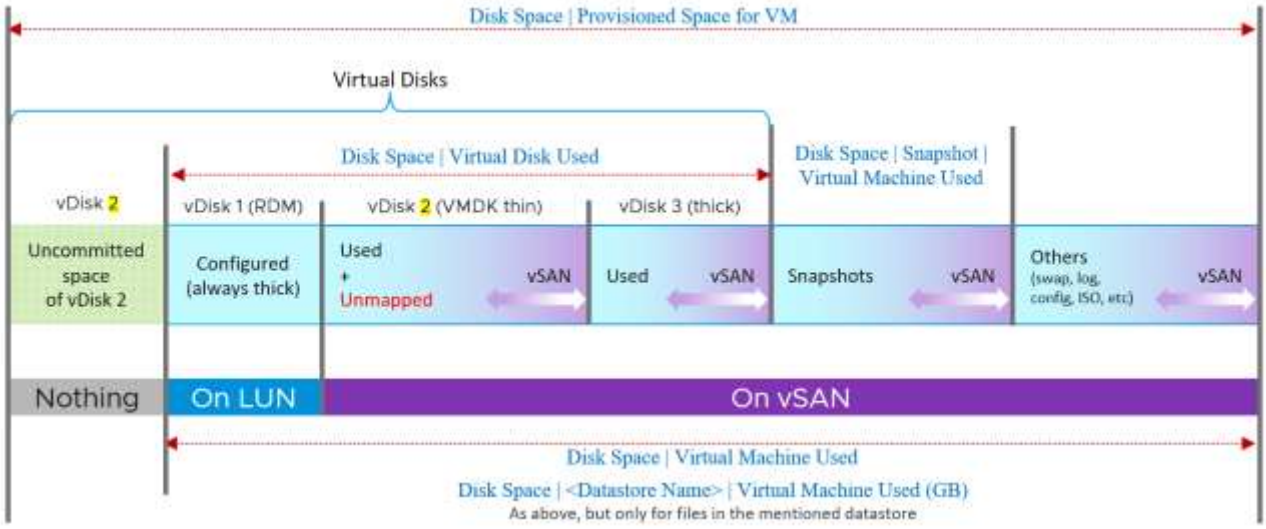
There are 2 metric, shown in Times New Roman font:

Metric	Description
Disk Space | Virtual Disk Used (GB)	The actual consumed size of the VMDK files. It excludes other files such as snapshot files. Note: For RDM the used space is the configured size of the RDM, unless the LUN is thin provisioned by the physical storage array. So its disk space consumption at VM level works like a thick provisioned disk. If this is higher than Guest OS used, and you're using thin provisioned, then run unmap to trim the unmapped blocks.
Virtual Disk | Configured Size	This metric does not include the vSAN part as it's taken from consumer layer.

All VM Files

Let's take an example of a VM with 3 virtual disks, so we can cover all the combinations.

- Thin provisioned
- Thick provisioned
- RDM. Physical or virtual is not relevant.



The boxes with blue line show the actual consumption at VM layer. Let's go through each rectangle.

RDM	It's not on vSAN as RDM can't be on a VMFS datastore. It's mapped to a LUN backed by an external storage. It's always thick provisioned, regardless of what Windows or Linux uses. The LUN itself could be thin provisioned but that's another issue and transparent to ESXi (hence VM).
Thin VMDK	We blended vSAN protection into a single box as you can't see the breakdown. It's inside the same file (so there is only 1 file but inside there is actual data + vSAN

	protection - vSAN dedupe - vSAN compressed). Thin Provisioned can accumulate unmapped block over time. You should reclaim them by running a trim operation. Uncommitted space is the remaining amount that the VMDK can grow into. Since it's not yet written, it does not have vSAN overhead yet.
Thick VMDK	The Used size equals the configured size as it's fully provisioned regardless of usage by Guest OS. I'm not sure the final outcome of dedupe and compression. If the Guest OS has not written to it, then I expect the saving will be near 100% in both lazy zero and eager zero.

vSAN protection applies to every file in the datastore. Yes, even your snapshot and log files are protected by default.

All the metrics are under Disk Space metric group. The key ones are:

Metric	Description
Disk Space | Provisioned Space for VM	Just like the Disk Space | Virtual Machine Used (GB), but thin provisioned is based on configured not actual usage. So this metric will have higher value if the thin provisioned is not fully used. This metric is useful at the datastore level. When you overcommit the space and want to know what the total space would be when all the VMs grow to the full size. This metric is not useful for capacity as it mixes both allocation and utilization. BTW, there can be case where the number here is reported as much higher number. See KB 83990. This is fixed in 7.0.2 P03 or 7.0 U2c, specifically in PR 2725886.
Disk Space | Virtual Machine Used (GB)	Just like above, but includes files other than virtual disks. So this metric is always larger. The actual consumed size of the VM files + the configured size of the RDM files. It includes all files in the VM folder in the datastore(s). Formula: Sum ([layoutEx.file] uniqueSize != null ? uniqueSize : size) / (1024 * 1024 * 1024)
Disk Space | <Datastore Name> | Virtual Machine Used (GB)	Just like above, but only includes files in that specific datastore only. For VM that only resides in 1 datastore, the value will be identical to above.

Snapshot

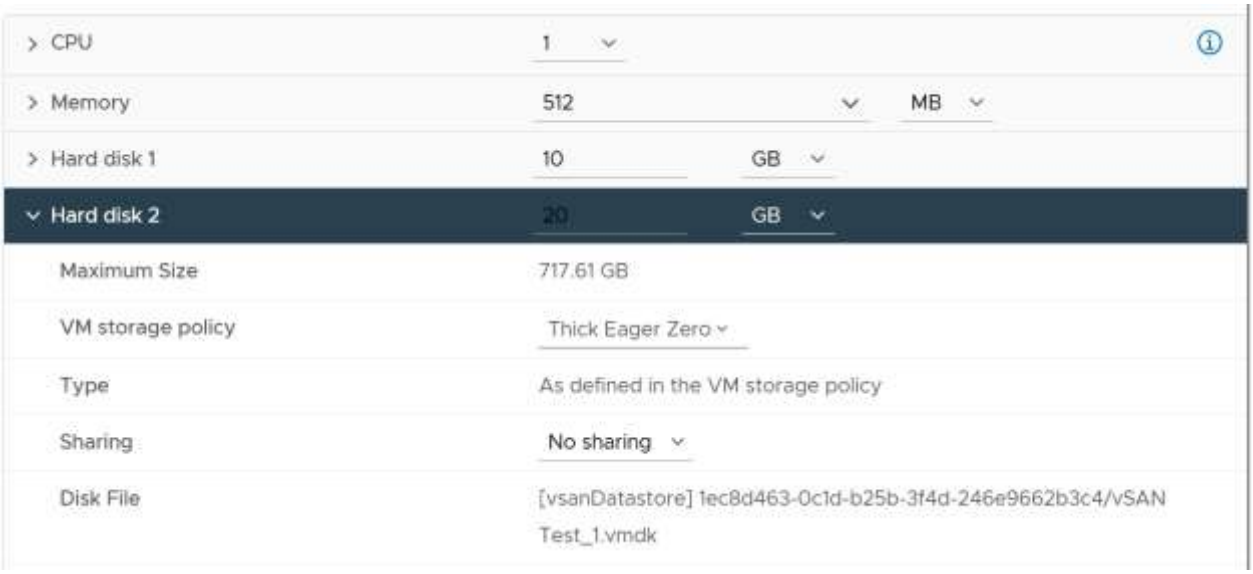
Disk Space | Snapshot | Virtual Machine Used (GB)	Disk Space used by all files created by snapshot (vmdk and non vmdk). This is the total space that can be reclaimed if the snapshot is removed. Use this to quickly determine which VMs have large snapshot. Formula: Sum of all files size / (1024 * 1024 * 1024) where aggregation is only done for snapshot files. A file is a snapshot file if

	its layoutEx file type equals to snapshotData, or snapshotList or snapshotMemory
Disk Space | Snapshot | Access Time (ms)	The date and timestamp the snapshot was taken. Note you need to format this.

vSphere Client UI

I'm adding this just in case you got curious 😊

Let's start with the basic and progress quickly. In the following example, I would create a small VM from scratch, with 2 VMDK disk.

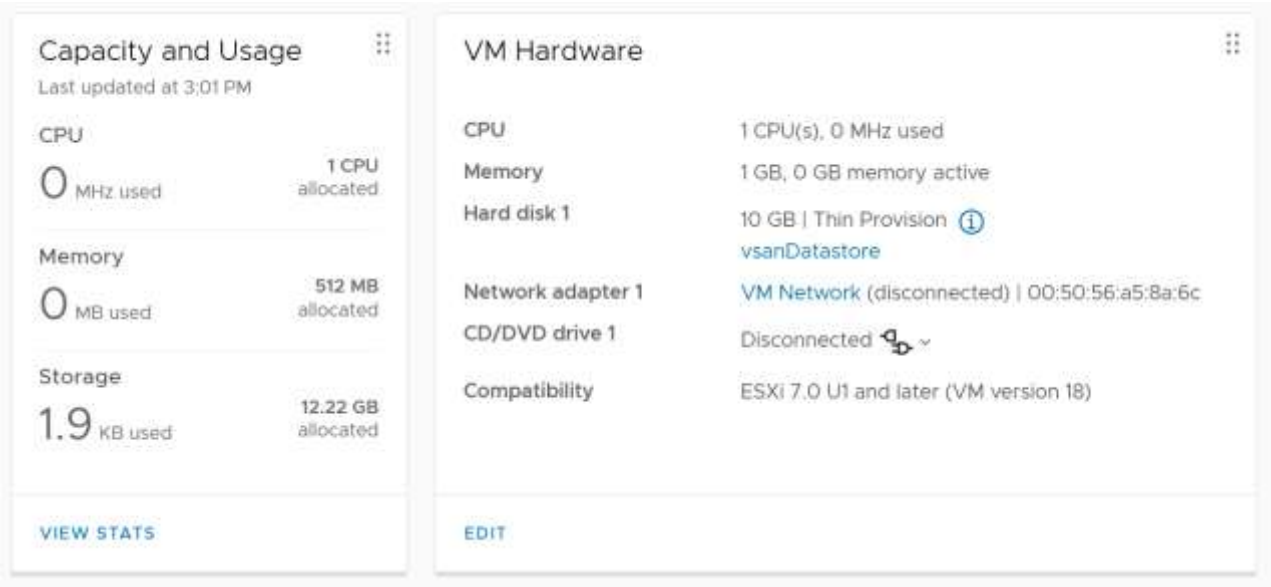


Hard disk 1 is 10 GB. Thin Provisioned. On vSAN.

The VM is powered off. All other settings follow default setting.

I created the VM with just the first disk, to validate the metrics value that will be shown upon creation. What do you expect to see on the vCenter UI?

Here is what I got on vSphere 7.



You get 2 numbers, used and allocated, as shown in the Capacity and Usage section.

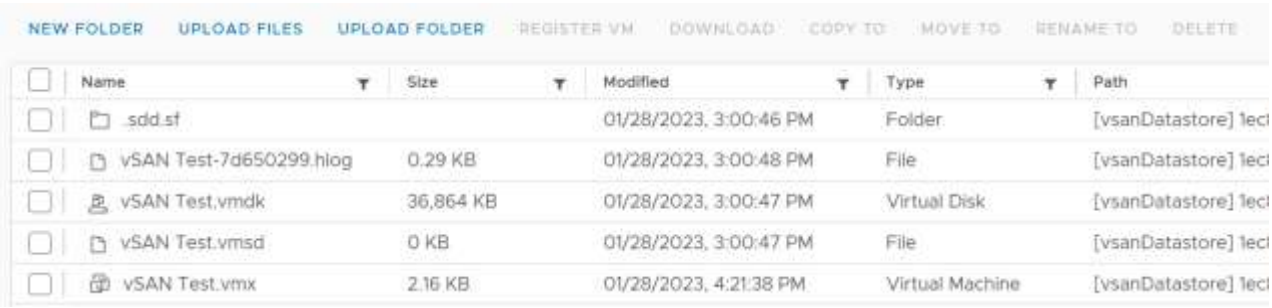
Used is only 1.9 KB. This is expected as it's thin provision and the VM is powered off. This is very low, so let's check the next number….

Allocated is 12.22 GB. This is 10 GB configured + 2.22 GB used. The hard disk 1 size shows 10 GB not 20 GB. This is what is being configured, and what Guest OS see. It is not impacted by vSAN as it's not utilization.

So you have 2 different numbers for the use portion: 1.9 KB and 2.22 GB.

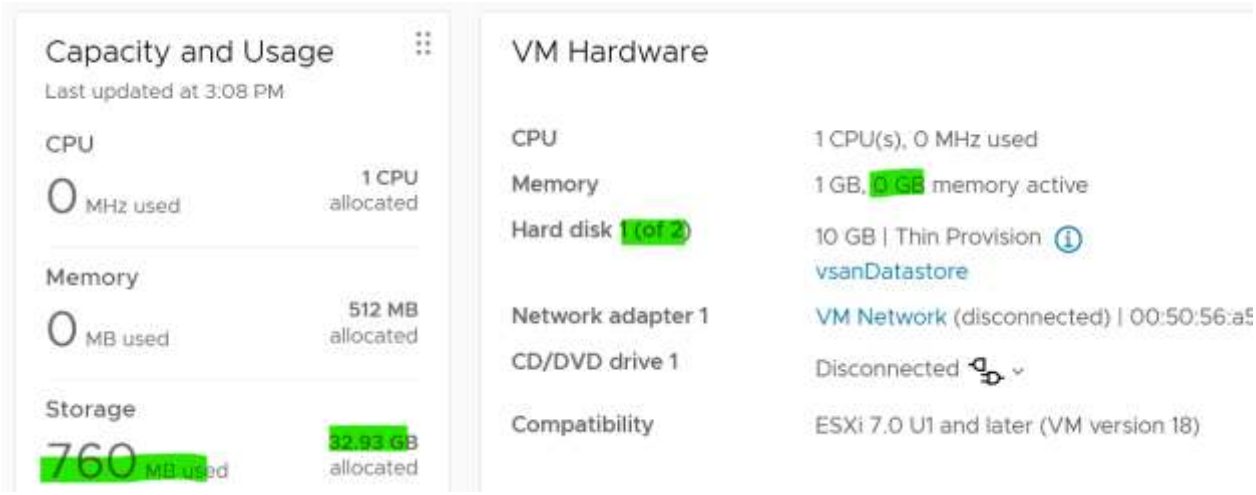
Why 2 different values?

Let's see what the files are. We can do this by browsing the datastore and find the VM folder.



The total from the files above is 36 MB. This does not explain 1.9 KB nor 2.22 GB.

Let's continue the validation. This time I added Hard disk 2 and configure it with 20 GB. Unlike the first disk, this is Thick Provisioned so we can see the impact. It is also on vSAN.



Used has gone up from 1.9 KB to 760 MB. As this is on vSAN, it consists of 380 MB of vSphere + 380 MB of vSAN protection. The vSAN has no dedupe nor compression, so it's a simple 2x.

Allocated is 32.93 GB as it consists of 30 GB configured and 2.93 GB. This 2.93 is half vSphere overhead + vSAN protection on the overhead.

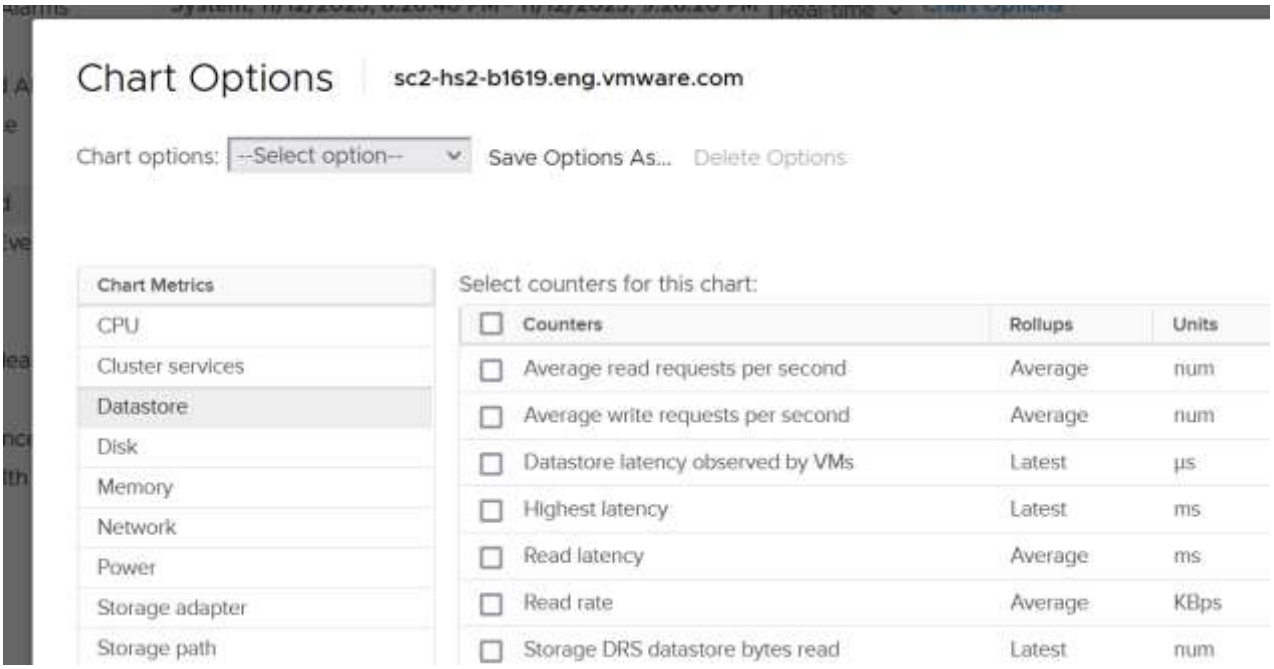
Looking at the datastore level, the second hard disk is showing 40.86 GB. It maps to hard disk 2.

Name	Size	Modified	Type
.sdd.sf		01/28/2023, 3:00:46 PM	Folder
vSAN Test-7d650299.hlog	0.29 KB	01/28/2023, 3:00:48 PM	File
vSAN Test.vmdk	36,864 KB	01/28/2023, 3:00:47 PM	Virtual Disk
vSAN Test.vmsd	0 KB	01/28/2023, 3:00:47 PM	File
vSAN Test.vmx	2.16 KB	01/28/2023, 4:21:38 PM	Virtual Machine
vSAN Test_1.vmdk	42,815,488 KB	01/28/2023, 4:21:38 PM	Virtual Disk

From this simple example, you can see that Allocated in vCenter UI actually contains used and allocated. By allocated it means the future potential used, which is up to the hard disk configured size. The used portion contains vSAN consumption if it's on vSAN, while the unused portion does not (obviously since vSAN has not written any block).

ESXi

The following screenshot shows the ESXi metric groups for storage in the vCenter performance chart.



As expected, there are 4 metrics groups for storage

- Datastore
- Disk
- Storage adapter
- Storage path.

We've covered earlier how they work together. We also covered how vSAN impacts them.

Disk or Device

There are 3 layers from VM to physical LUN

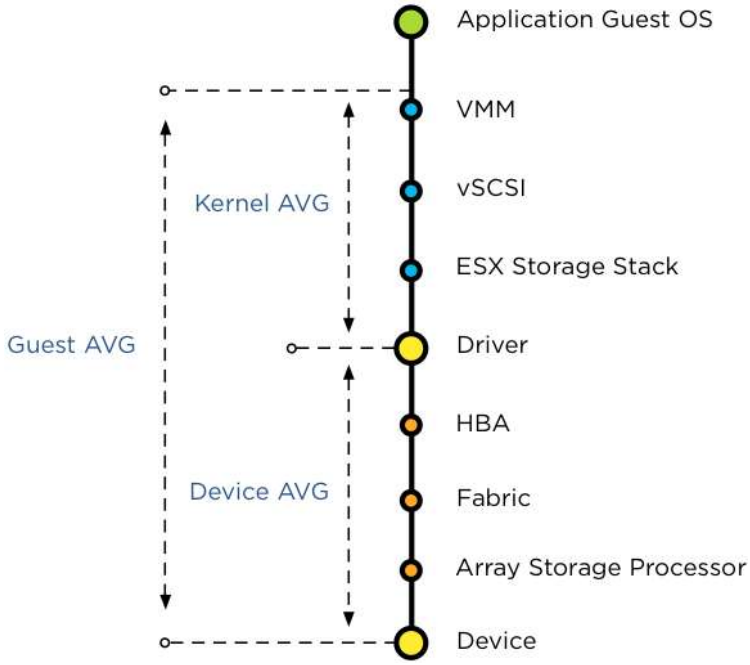
- VM.
- The kernel. This is measured by the KAVG counter and QAVG counter.
- Device.

Compared with Adapter or Path, you get a lot more metrics for disk or device as there is capacity metric.

As expected, there is no breakdown as the kernel cannot actually see anything in between the HBA and the device. So no metrics such as number of hops as it's not even aware of the fabric topology.

Contention Metrics

Frank Denneman, whose blog and book are great references, shows the relationship among the counters using the following diagram:



For further reading, review this explanation by Frank, as that's where I got the preceding diagram from.

Guest Average	GAVG	Guest here means VM, not Guest OS as the counter starts from VMM layer not Windows or Linux.
Kernel Average	KAVG	ESXi is good in optimizing the IO, so in a healthy environment, the value should be within 0.5 ms. This is computed from Guest AVG and Device AVG, which are raw counters.
	QAVG	QAVG, which is queue in the kernel, is part of KAVG. If QAVG is high, check the queue depths at each level of the storage stack. Cody explains why QAVG can be higher than KAVG here. In short, QAVG measures both VM IO and the kernel IO, while KAVG only includes VM IO.
Device Average	DAVG	The average time from ESXi physical card to the array and back. Typically, there is a storage fabric in the middle. The array typically starts with its frontend ports, then CPU, then cache, backend ports, and physical spindles. So if DAVG is high, it could be the fabric or the array. If the array is reporting low value, then it's the fabric of the HBA configuration. I'm unsure what DAVG measures when it's vSAN and the data happens to be local.

For each of the above 4 sets, you expect read latency, write latency and the combined latency. That means 12 counters and here are what they are called in vSphere Client UI:

Device	Physical device command latency	Average	ms
	Physical device read latency	Average	ms
	Physical device write latency	Average	ms
Kernel	Kernel command latency	Average	ms
	Kernel read latency	Average	ms
	Kernel write latency	Average	ms
Queue	Queue command latency	Average	ms
	Queue read latency	Average	ms
	Queue write latency	Average	ms
Guest	The counters are not prefixed with Guest, so they are simply called: • Command Latency • Write Latency • Ready Latency		

With the above understanding, let's validate with real world values.

Name	99P Device Latency	99P Kernel Latency	99P Queue Latency	95P Kernel Latency	95P Device Latency	95P Queue Latency
wdc-06-r...	0.42 ms	0.1 ms	0.03 ms	0.04 ms	0.27 ms	0 ms
wdc-06-r...	0.44 ms	0.27 ms	0.01 ms	0.1 ms	0.27 ms	0 ms
wdc-06-r...	0.45 ms	0.14 ms	0 ms	0.08 ms	0.29 ms	0 ms
wdc-06-r...	0.46 ms	0.29 ms	0.01 ms	0.17 ms	0.32 ms	0 ms
wdc-06-r...	0.46 ms	0.33 ms	0.14 ms	0.21 ms	0.38 ms	0.0037 ms
wdc-06-r...	0.46 ms	0.14 ms	0 ms	0.09 ms	0.26 ms	0 ms
wdc-06-r...	0.48 ms	0.2 ms	0.0044 ms	0.1 ms	0.3 ms	0 ms
wdc-06-r...	0.49 ms	0.36 ms	0.1 ms	0.3 ms	0.44 ms	0.05 ms
wdc-06-r...	0.49 ms	0.34 ms	0.04 ms	0.18 ms	0.4 ms	0 ms
wdc-06-r...	0.51 ms	0.0013 ms	0 ms	0 ms	0.02 ms	0 ms
wdc-06-r...	0.81 ms	0.59 ms	0.02 ms	0.43 ms	0.62 ms	0 ms
Average	0.06 ms	0.03 ms	0.0029 ms	0.02 ms	0.03 ms	0.0004 ms
Max	0.81 ms	0.59 ms	0.14 ms	0.43 ms	0.62 ms	0.05 ms

I chose the last ESXi since that's the one with worst latency.

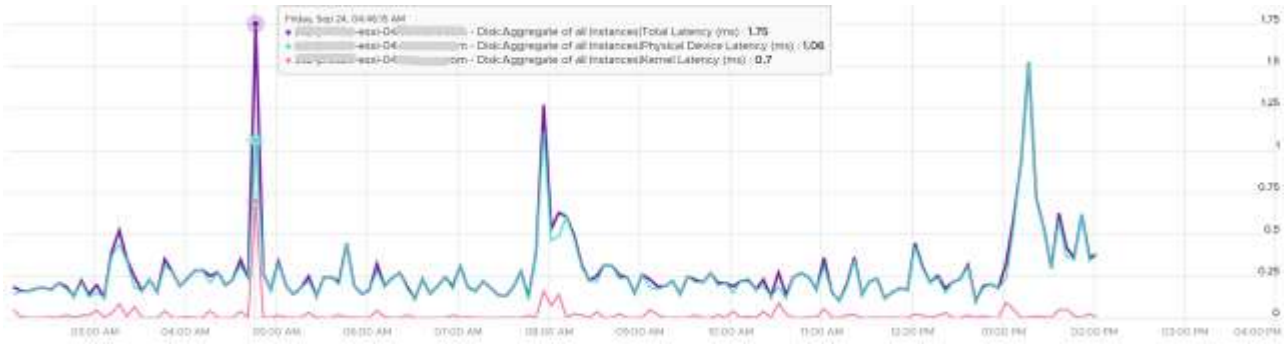
I plotted Kernel vs Device.

What do you notice? Can you determine which is which?

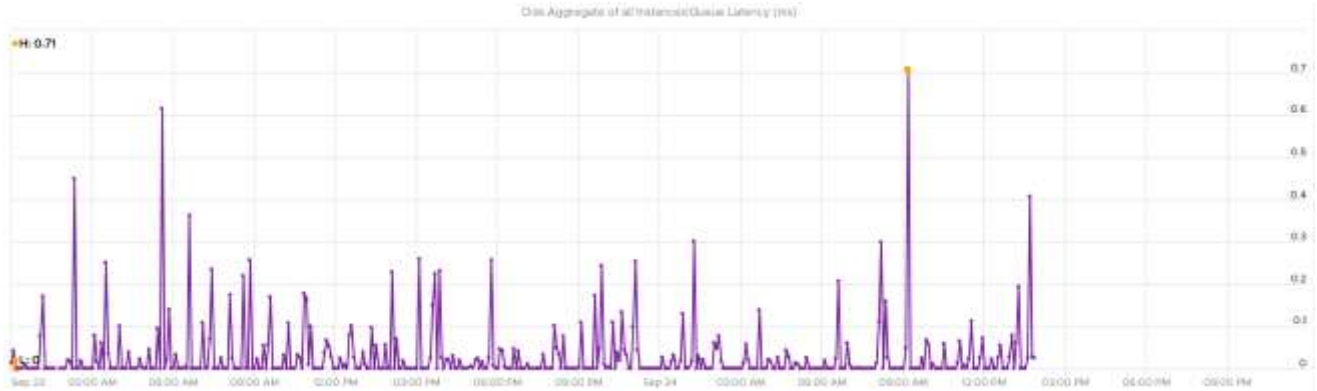


They don't correlate. This is expected since both have reasonably good value (my expectation is below 0.5 ms).

The bulk of the latency should come from the Device. In a healthy environment, it should be well within 5 ms. With SSD, it should be even lower. As you can see below, it's below 1.75 ms. Notice the kernel latency is 0.2 ms at all times except in 1 spike.



What about the Queue latency? It's part of the kernel latency, so it will be 100% within it. When the kernel latency value is in the healthy range, the 2 values should correlate, as the value is largely dominated by the Queue. Notice the pattern below is basically identical.





Other Metrics

I find the value of Bus Resets and Commands Aborted are always 0

Object	Measurement	Rollup	Units	Latest	Maximum	Minimum
Local ATA Disk (naa.5002538c4089117f)	Bus resets	Summation	num	0	0	0
Local ATA Disk (t10.ATA_____Micron_5100_MTFDDAV240TCB_____...)	Bus resets	Summation	num	0	0	0
Local TOSHIBA Disk (naa.58ce38ee2001a845)	Bus resets	Summation	num	0	0	0
Local ATA Disk (naa.5002538c4089117f)	Commands aborted	Summation	num	0	0	0
Local TOSHIBA Disk (naa.58ce38ee2001a845)	Commands aborted	Summation	num	0	0	0

If you've seen a non zero let me know.

Highest latency	Latest	ms	Highest latency value across all disks used by the host
Maximum queue depth	Average	num	Maximum queue depth

I'm not sure what highest latency refers to (Guest, Kernel, or Device).

Maximum Queue Depth is more of a property than a metric, as it's a setting.

Object	Measurement	Rollup	Units	Latest	Maximum	Minimum
Local TOSHIBA Disk (naa.58ce38ee2001a845)	Maximum queue depth	Average	num	254	254	254
Local ATA Disk (naa.5002538c4089117f)	Maximum queue depth	Average	num	32	32	32
Local ATA Disk (t10.ATA_____Micron_5100_MTFDDAV240TCB_____...)	Maximum queue depth	Average	num	31	31	31

Consumption Metrics

You get the standard IOPS and Throughput metrics.

IOPS	Average commands issued per second	Average	num
	Average read requests per second	Average	num
	Average write requests per second	Average	num
Throughput	The counters names are • Read Rate • Write Rate • Usage All their units are in KB/s		

Total IO	This is just the number of Read or Write in the time interval. The counters names are • Read Requests • Write Requests • Commands Issued

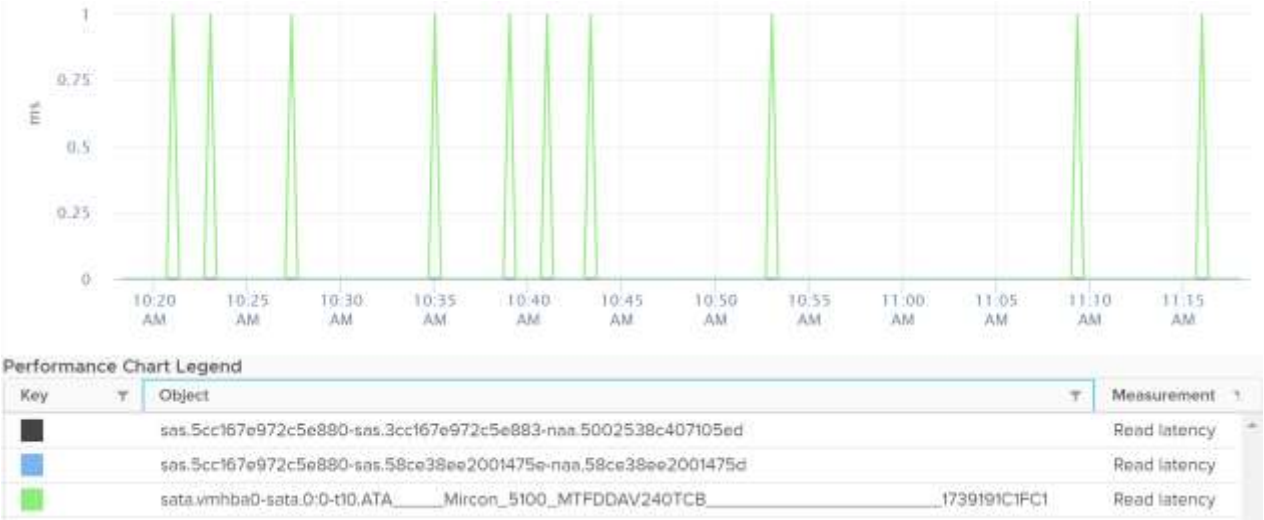
Storage Adapter & Path

They have identical set of counters, hence I'm documenting them together. Ideally, adapter should include metrics such as adapter queue length and commands aborted.

The following screenshot shows the metrics provided:

Counters	Rollups	Units	Stat Type
Average commands issued per second	Average	num	Rate
Average read requests per second	Average	num	Rate
Average write requests per second	Average	num	Rate
Highest latency	Latest	ms	Absolute
Read latency	Average	ms	Absolute
Read rate	Average	KBps	Rate
Write latency	Average	ms	Absolute
Write rate	Average	KBps	Rate

For storage path, the counters may appear that they are measuring the device, as the object name is not based on the friendly name.



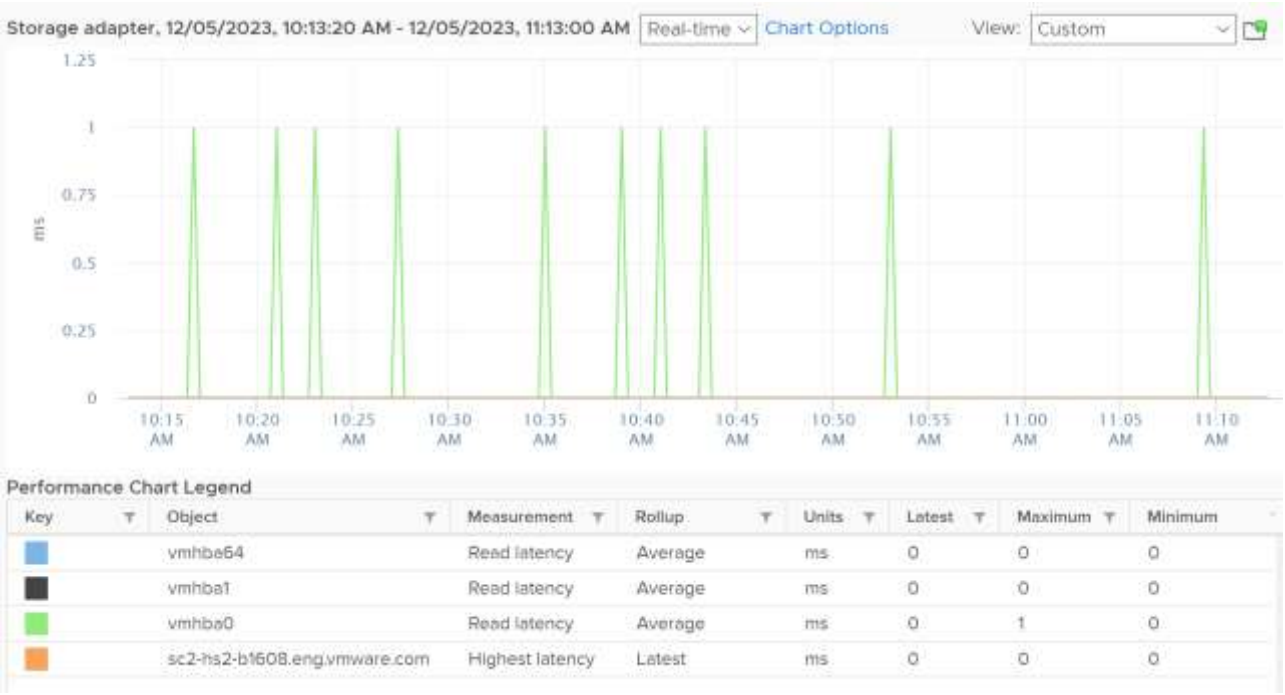
The object above is the path, not the device.

Contention Metrics

There are 3 metrics provided:

- Read latency
- Write latency
- Highest latency

The highest latency metric takes the worst value among all the adapters or the paths. This can be handy compared to tracking each of them one by one. However, it averages each adapter first, so it's not the highest read or write. You can see from the following screenshot that its value is lower than the read latency or vmhba0. What you want is the highest read or write among all the adapters or paths.



Key	Object	Measurement	Rollup	Units	Latest	Maximum	Minimum
	vmhba64	Read latency	Average	ms	0	0	0
	vmhba1	Read latency	Average	ms	0	0	0
	vmhba0	Read latency	Average	ms	0	1	0
	sc2-hs2-b1608.eng.vmware.com	Highest latency	Latest	ms	0	0	0

Analysis

I plotted 192 ESXi host and checked the highest read latency and highest write latency among all their adapters. As the data was returning mostly < 1 ms, I extended to 1 week and took the worst in that entire week. You can see that the absolute worst of write latency was a staggering 250 ms. But plotting the 95th percentile value shows 0.33 ms, indicating it's a one off occurrence in that week. The 250 ms is also likely an outlier as the rest of the 191 ESXi shows maximum 5 ms, and with much lower value at 95th percentile.

Name	Highest Read Latency	95P Read Latency	Highest Write Latency ↓	95P Write Latency
wd...	3 ms	0 ms	250.93 ms	0.33 ms
wd...	0.87 ms	0 ms	5.27 ms	0.07 ms
wd...	0.2 ms	0 ms	5.27 ms	0.07 ms
wd...	1.6 ms	1 ms	5.27 ms	0.21 ms
wd...	0.27 ms	0 ms	5.2 ms	0.07 ms
wd...	0.27 ms	0 ms	5.2 ms	0.07 ms
wd...	1.4 ms	0.4 ms	5.2 ms	0.07 ms
wd...	0.2 ms	0 ms	5.2 ms	0.07 ms
wd...	0.4 ms	0 ms	5.13 ms	0.07 ms
wd...	78.8 ms	0 ms	5.13 ms	0.07 ms
wd...	1 ms	0 ms	5.13 ms	0.07 ms

Plotting the value of the first ESXi over 7 days confirmed the theory that it's a one off, likely an outlier.



Does it mean there is no issue with the remaining of the 191 ESXi hosts?

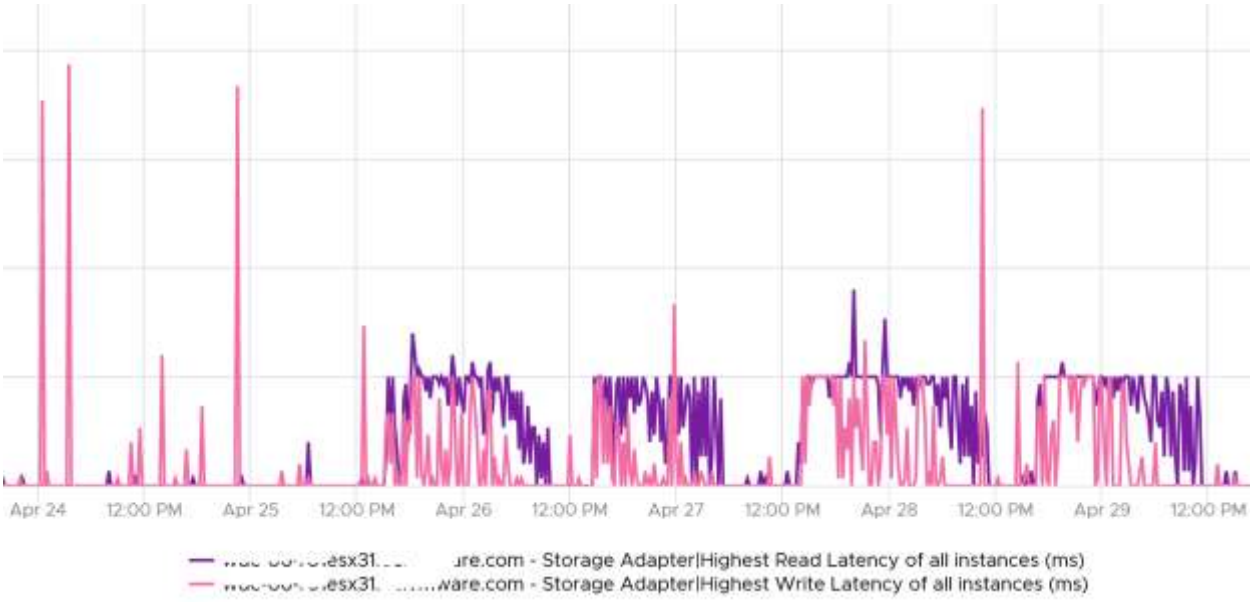
Nope. The values at 95th percentile is too high for some of them.

I modified the table by changing Maximum with 99th percentile to eliminate an outlier. I also reduced the threshold so I can see better. The following table shows the values, sorted by the write latency.

Name	99P Read Latency	95P Read Latency	99P Write Latency ↑	95P Write Latency
wdc-06-r...	0.07 ms	0 ms	0.73 ms	0.07 ms
wdc-06-r...	1 ms	1 ms	0.73 ms	0.2 ms
wdc-06-r...	1 ms	1 ms	0.78 ms	0.13 ms
wdc-06-r...	0.99 ms	0.13 ms	0.78 ms	0.07 ms
wdc-06-r...	0.13 ms	0 ms	0.79 ms	0.07 ms
wdc-06-r...	0.07 ms	0 ms	0.84 ms	0.07 ms
wdc-06-r...	0.2 ms	0.07 ms	0.86 ms	0.33 ms
wdc-06-r...	1 ms	1 ms	0.87 ms	0.47 ms
wdc-06-r...	0.07 ms	0 ms	0.93 ms	0 ms
wdc-06-r...	1 ms	1 ms	0.93 ms	0.73 ms
wdc-06-r...	1 ms	1 ms	0.99 ms	0.07 ms
wdc-06-r...	1.07 ms	1 ms	1 ms	1 ms
Average	0.22 ms	0.12 ms	0.26 ms	0.12 ms
Max	1.07 ms	1 ms	1 ms	1 ms

151 - 192 of 192 items ‹ 1 2 3 4

The table revealed that there are indeed latency problem. I plotted one of the ESXi and saw the following.



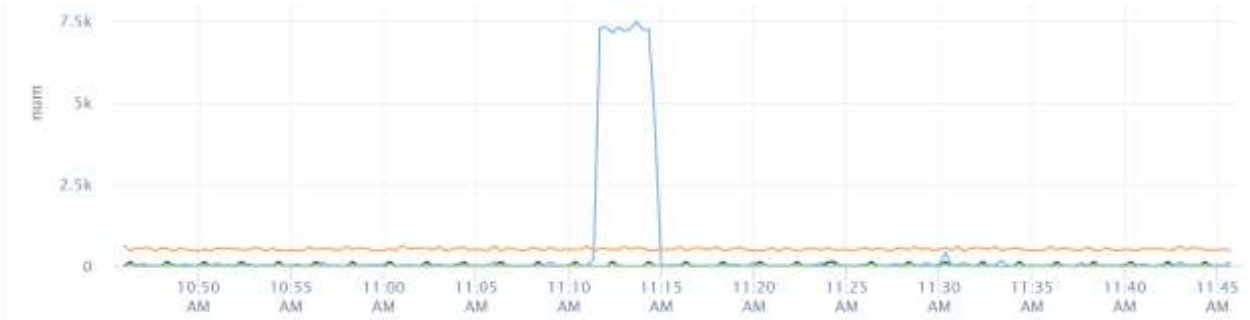
From here, you need to drill down to each adapter to find out which one.

Consumption Metrics

For each adapter, there are 4 metrics provided:

- Read IOPS, tracking the number of reads per second.
- Write IOPS
- Read throughput
- Write throughput.

The following screenshot is an example of what you get from vSphere Client UI.



Performance Chart Legend

Key	Object	Measurement	Rollup	Units	Latest	Maximum	Minimum	Average
	vmhba1	Average read requests per second	Average	num	122	7,481	11	422.956
	vmhba1	Average write requests per second	Average	num	480	643	447	524.844
	vmhba0	Average read requests per second	Average	num	1	141	0	25.05
	vmhba0	Average write requests per second	Average	num	1	66	0	11.133

If the block size matters to you, create a super metric in VCF Operations.

Datastore

For shared datastore, the metrics do not show the same value with the one at datastore object. All these metrics are only reporting from this ESXi viewpoint, not the sum from all ESXi mounting the same datastore. As a result, I'd cover only performance. Capacity will be covered under the datastore chapter.

For each datastore, you get the usual IOPS, throughput and latency. They are split into read and write, so you have 3 x 2 = 6 metrics in total. These are the actual names:

Counters	Rollups	Units	Description
Average read requests per second	Average	num	Average number of read commands issued per second to the datasto...
Average write requests per second	Average	num	Average number of write commands issued per second to the datasto...
Datastore latency observed by VMs	Latest	µs	The average datastore latency as seen by virtual machines
Highest latency	Latest	ms	Highest latency value across all datastores used by the host
Read latency	Average	ms	The average time a read from the datastore takes
Read rate	Average	KBps	Rate of reading data from the datastore
Write latency	Average	ms	The average time a write to the datastore takes
Write rate	Average	KBps	Rate of writing data to the datastore

There is no block size but you can derive it by dividing Throughput with IOPS.

You also get 2 additional counters:

- Datastore latency observed by VMs
- Highest latency.

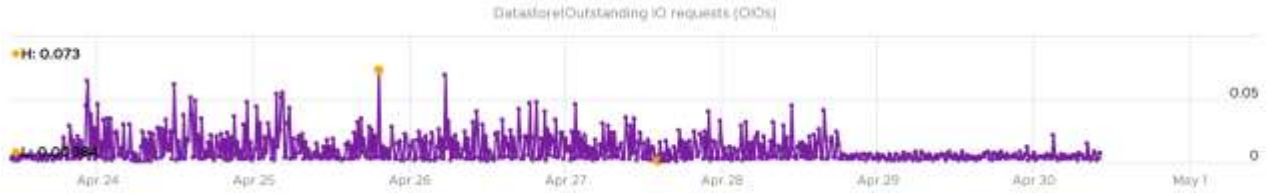
I plotted their values and to my surprise the metric Datastore latency observed by VMs is much higher. You can see the blue line below. It makes me wonder what the gap is as there is only the kernel in between.



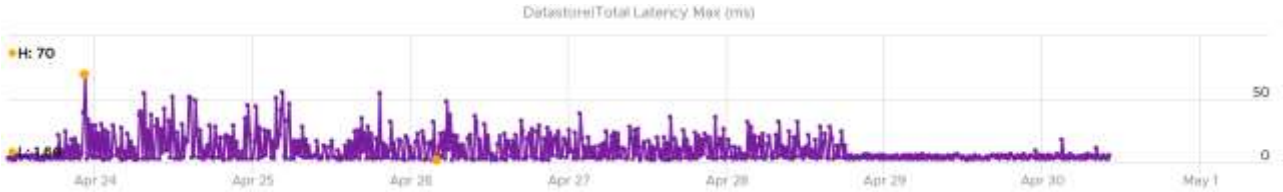
The metric Highest Latency is a normalized averaged of read and write, hence it can be lower.

Outstanding IO

You can derive the outstanding IO metric from latency and IOPS. I think latency counter is more insightful. For example, the following screenshot shows hardly any IO being in the queue:

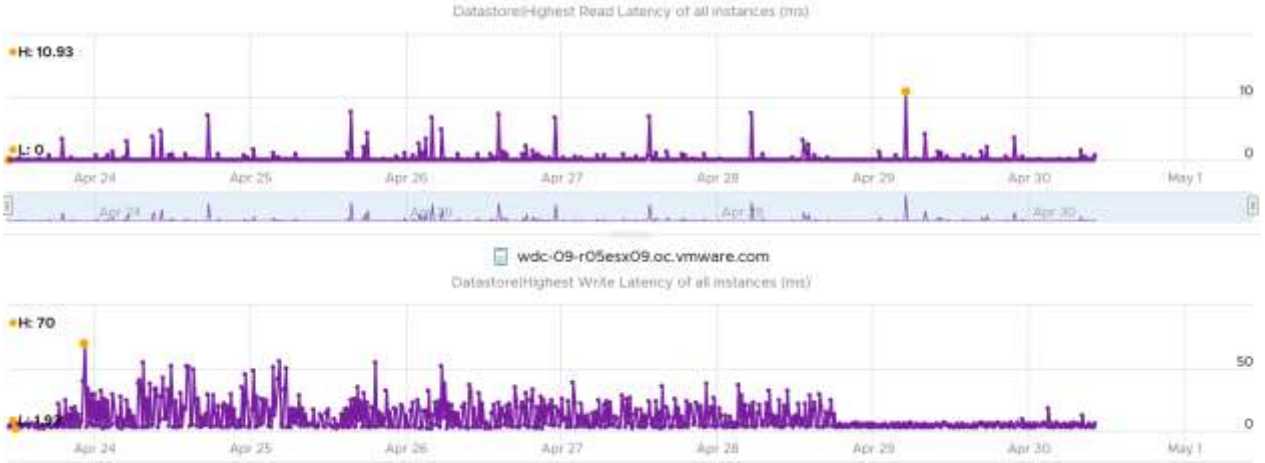


However, if you plot latency, you get same pattern of line chart but with higher value.



You can check whether it's read or write by plotting each.

The following screenshot shows it's caused by write latency. It's expected if your read is mostly served by cache.



Queue Depth

You can also see the queue depth for each datastores (I think this is actually the backing LUN, but unsure if there are extent). Ensure that the settings are matching your expectation and are consistent. You can list them per cluster and see their values.

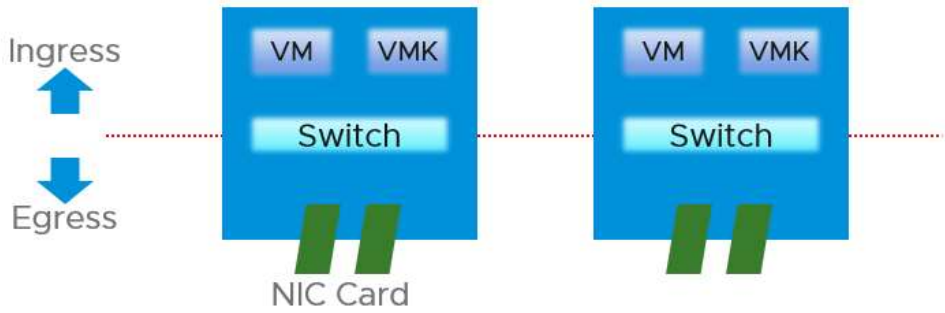
Name ↑	Instance Name	Max Queue Depth
blr-01-r06esx30.oc.vmware.com	148341	4,294,967,296
blr-01-r06esx30.oc.vmware.com	148384	4,294,967,296
blr-01-r06esx31.oc.vmware.com	148341	4,294,967,296
blr-01-r06esx31.oc.vmware.com	148384	4,294,967,296
blr-01-r06esx32.oc.vmware.com	148341	4,294,967,296
blr-01-r06esx32.oc.vmware.com	148384	4,294,967,296
Average	-	**4,294,967,296**
Max	-	**4,294,967,296**

Part 3 Chapter 5

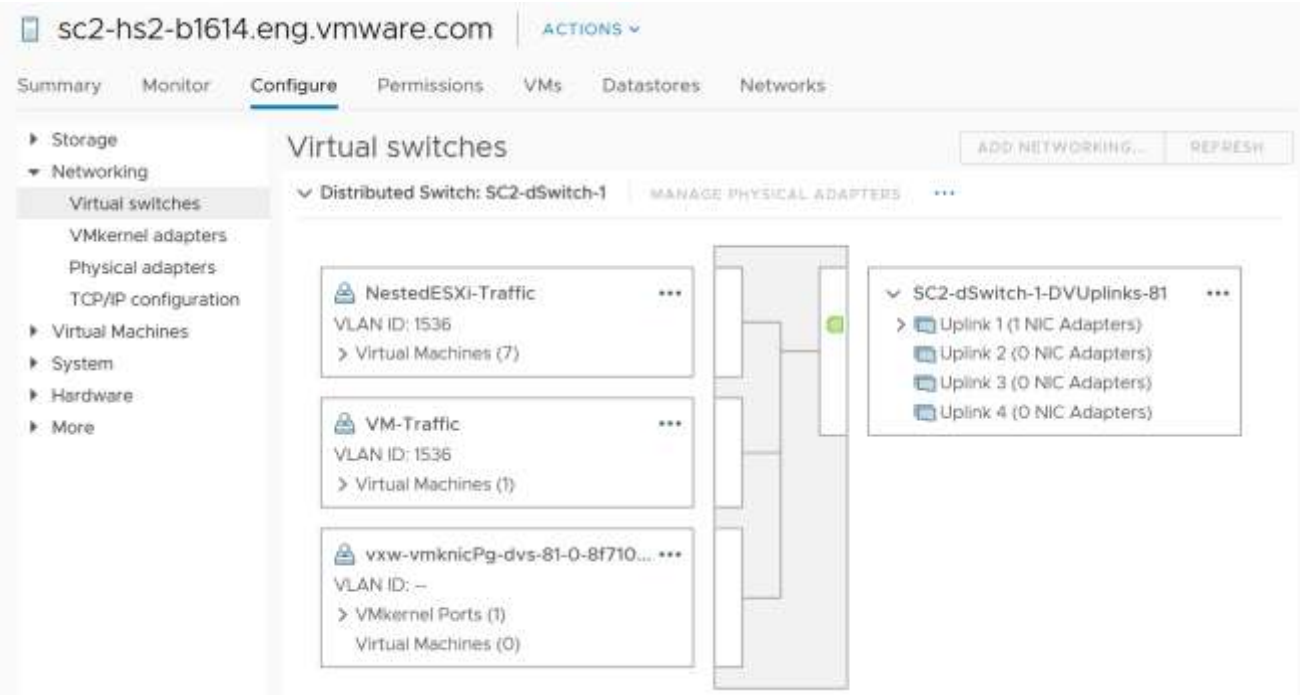
Architecture

Network monitoring is complex, especially in large data centers. Adding network virtualization takes the complexity of performance troubleshooting even higher.

Just like CPU, Memory and Disk, there is also a new layer introduced by virtualization. There are virtual network cards on each VM, and software-based switch on each ESXi bridging the VM card to the physical NIC card. The various ESXi kernel modules also do not "talk" directly to the physical card. Basically, what used to be the top of rack switch are now living inside each ESXi as a software switch.



vSphere Client shows the 2 layers side by side (personally I prefer up and down, with the physical layer placed below).



Virtual Network	There are 2 types: • Kernel • VM They do not mix, for security reason. The kernel port group runs specific traffic, such as vMotion and vSAN. The VM port group runs VM. If the traffic is just a VM-to-VM traffic, within the

	same ESXi, the packets does not reach the physical network, hence the metrics do not register it. This is why the values in virtual do not always match physical. The virtual network does not have the limit that physical network does, if the traffic remains in the box. This makes it harder to troubleshoot as the total capacity is not statically defined. So instead of just monitoring the throughput metric, you should also check the packet per second metric.
Physical Network	They are called vmnic instead of pNIC Metrics at this level do not have per-VM breakdown, or per kernel interface breakdown

Unique Characteristics

From performance and capacity management point of view, network has different fundamental characteristics to compute or storage. The key differences are summarized below.

	Compute or Storage	Network
Nature	A node	An interconnect
Hardware	Single purpose	Multi-purpose
Location	Fewer	Many
Upper Limit	Yes	No
Net available resource to VM	Relatively high	Low
Resource allocation at VM level	Granular	Coarse
Monitoring	Simpler	Harder
Workload Type	1	Many
Primary Unit	Byte	Bit

Nature of Network

Compute and storage are nodes. They are dots, while network are lines.

When you have a CPU or RAM performance issue on one host, it doesn't typically impact another host on a different cluster. The same thing happens with storage. When a physical array has a performance issue, generally speaking it does not impact other arrays in the data center.

Network is different. A local performance issue can easily be a data center-wide problem. Here is a good read by shared Ivan Pepelnjak. To give a recent example (H2 2021), here is one from a world-class network operator[19]:

[19] The name of this Internet giant is irrelevant for this purpose, as it could have happened to anyone. It happens more often on smaller companies. BTW, notice how they made the text grey so it's harder to read!

> To all the people and businesses around the world who depend on us, we are sorry for the inconvenience caused by today's outage across our platforms. We've been working as hard as we can to restore access, and our systems are now back up and running. The underlying cause of this outage also impacted many of the internal tools and systems we use in our day-to-day operations, complicating our attempts to quickly diagnose and resolve the problem.
>
> Our engineering teams have learned that configuration changes on the backbone routers that coordinate network traffic between our data centers caused issues that interrupted this communication. This disruption to network traffic had a cascading effect on the way our data centers communicate, bringing our services to a halt.

Being an interconnect, it also connects users and servers to the Internet. If you have a global operation, you likely have multiple entry points, provided by different providers. These connectivity needs to be secured and protected with HA, preferably from 2 different ISPs.

There are typically many paths and routes in your network. You need to ensure they are available by testing the connectivity from specific points.

Hardware

The networking hardware itself can provide different functionalities.

For compute, you have servers. While they may have different form factors or specifications, they all serve the same purpose—to provide processing power and a set of working memory for hypervisor or VM.

For network, you have a variety of network services (firewall and load balancer) in addition to the basic network functionalities (switch, router, and gateway). You need to monitor all of them to get a complete picture. These functionalities can take the form of software or hardware.

Unlike storage, network has concept of duplex. A full duplex means it has 100% on both directions. For example, an ESXi with a 25 Gb port can theoretically handle 25 Gb TX + 25 Gb RX as its full duplex.

Blade servers and other HCI form factors blur the line between server and network.

Location

Server and storage tend to be located fewer places. Even in the ROBO office, they are typically located in a rack, with proper cooling and physical security. Network switch, especially Wireless Access Points, need to be placed in multiple places within the building, if that's required to provide enough network coverage.

Solution such as SDWAN even requires a network device to be deployed at employee home. I actually have the Dell edge device at my home.

Total Capacity

CPU or RAM workload have a per VM physical limit. This makes capacity management possible, and aids in performance troubleshooting.

While network has a physical limit, it can be misleading to assume it is available to all VMs all the time. Because the physical capacity of the network is shared, you have a dynamic upper limit for each workload. The VM Network port group will have more bandwidth when there is no vMotion happening. Furthermore, each VM has a dynamic upper limit as it shares the VM Network port group with other VMs.

The resource available to VM also varies from host to host. Within the same host, the limit changes as time progresses. Unlike Storage I/O Control, Network I/O Control does not provide any metrics that tell you that it has capped the bandwidth.

In many situations, the bandwidth within the ESXi host may not be the smallest pipe between the originating VM and its destination. Within the data center, there could be firewalls, load balancers, routers, and other hops that the packet has to go through. Once it leaves the data center, the WAN and Internet are likely to be a bottleneck. This dynamic nature means every VM has its own practical limit.

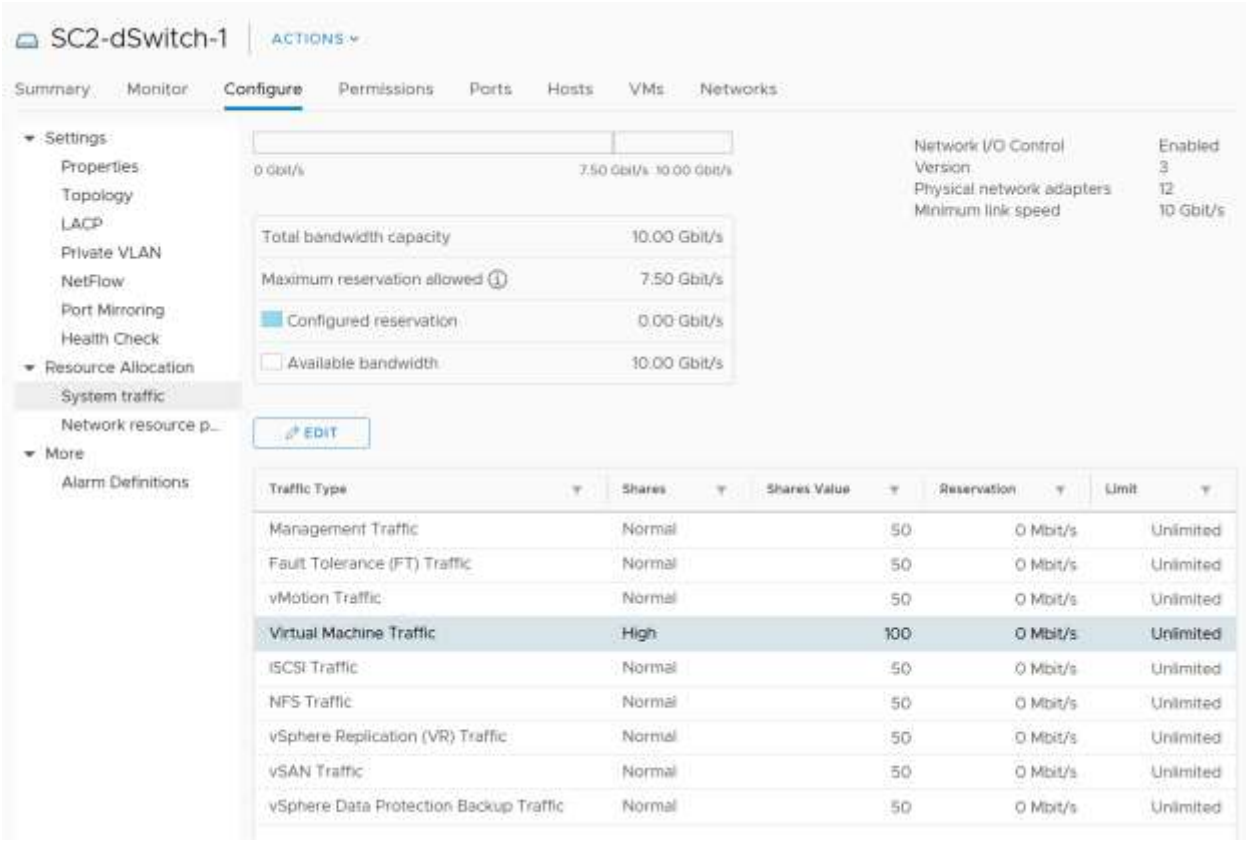
Net Available Resource

At the end of the day, the net available resources to the VMs are what we care about. What the IaaS platform used is considered an overhead. The more ESXi kernel, NSX, vSAN, vSphere Replication use, the lesser you have left for the business workload.

An ESXi host has a fixed specification (for example, 2 CPUs, 60 cores, 1 TB RAM, 2 x 50 GE NIC). This means we know the upper physical limit. How much of that is available to the VMs? Another word, what is the usable capacity for the business workload?

For compute, the hypervisor consumes a relatively low proportion of resources. Even if you add a software-defined storage such as vSAN, you are looking at around 10% total utilization but depends on many factors.

The same cannot be said about network. Mass vMotion (for example, when the host enters maintenance mode), storage vMotion (in IP storage case), VM provisioning or cloning (for IP storage), and vSAN all take up significant network bandwidth. In fact, the non-VM network takes up the majority of the ESXi resources. If you have 2 x 25 GE NIC, majority of it is *not* used by VM. The following screenshot shows that VM only gets 100 shares out of 500 shares. So the overhead can be as high as 80%.

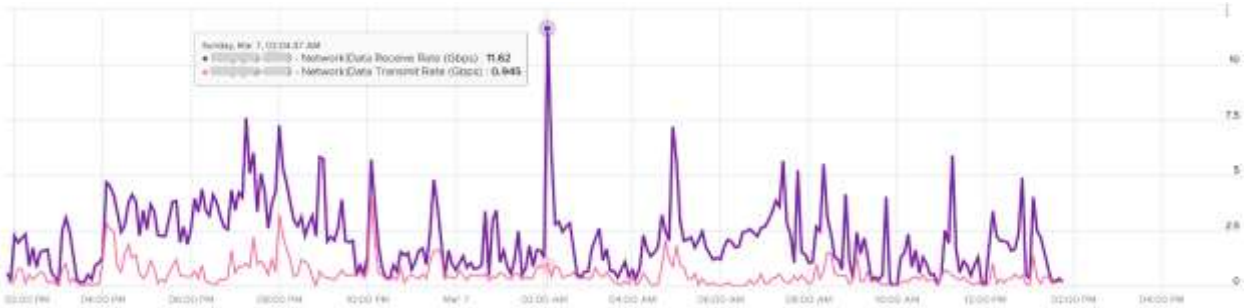


Resource Allocation

This means the resource that is given to a single VM itself. For compute, we can configure a granular size of CPU and RAM. For the CPU, we can assign one, two, three, four, etc. vCPUs.

With network, we cannot specify the vNIC speed. It takes the speed of the ESXi vmnic assigned to the VM port group. So each VM will either see 1 GE or 10 GE or 25 GE (you need to have the right vNIC driver, obviously). You cannot allocate another amount, such as 500 Mbps or 250 Mbps in the Guest OS. In the physical world, we tend to assume that each server has 10 GE and the network has sufficient bandwidth. You cannot assume this in a virtual data center as you no longer have 10 GE for every VM at the physical level. It is shared and typically oversubscribed.

A network intensive VM can easily hit 1 Gbps for both egress and ingress traffic. The following chart shows a Hadoop worker node receiving more than 5 Gbps worth traffic multiple times. You need to be careful in sizing the underlying ESXi if you want to run multiple VMs. While you can use Network I/O Control and vSphere Traffic Shaping, they are not configuration property of a VM.



Monitoring and Troubleshooting

A distributed system is harder to monitor than a single node, especially if workload varies among the components that make up the system.

The network resource available to VM also varies from host to host. Within the same host, the limit changes as time progresses. Unlike Storage I/O Control, Network I/O Control (NIOC) does not provide any metrics that tell you that it has capped the bandwidth.

NIOC can help to limit the network throughput for a particular workload or VM. If you are using 10 GE, enable NIOC so that a burst in one network workload does not impact your VM. For example, a mass vMotion operation can saturate the 10 Gb link if you do not implement NIOC. In vCenter 7, there is no counter that tracks when NIOC caps the network throughput. You may need to check the log for that.

The primary contention metrics are

- Latency.
- Dropped Packets
- Retransmit Packets. For TCP, dropped packets will be retransmitted.
- Jitter. This measures the inconsistency of the latency. An application may tolerate poor latency better than variable latency.

Note there is no latency and retransmit metrics in vSphere.

Remember that Storage has 2 metrics (IOPS and Throughput) for consumption? Network also has these 2 types, except the more popular one is the throughput. The PPS (packet per second) is less popular although they are useful in gaining insight into your network. It takes up a significant CPU time to process high number of packets with low latency, as you can see in NSX edge VM.

Workload Type

In network, not all packets are of the same type. You can have unicast, multicast and broadcast.

Majority of traffic should be unicast, as ESXi or VM should not be broadcasting to all IP addresses in the network or multicasting to many destinations. The challenge is there are purposes for each type so you need to monitor if the broadcast and multicast happen at the wrong time in the wrong network.

Storage and Server only have 1 type of workload. From operations management viewpoint, for almost all customers, A CPU instruction is a CPU instruction. You do not care what it is. The same goes with memory access and disk IO commands.

Conclusion

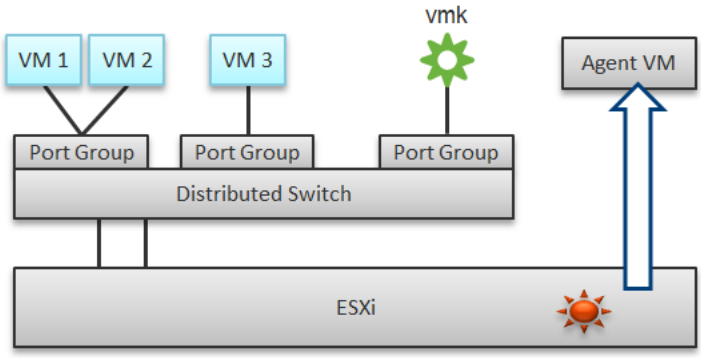
Because of all these differences, the primary unit is bit, not byte. Storage uses byte as it focuses on the amount of disk space consumed by the data.

The way you *approach* network monitoring should also be different. If you are not the network expert in your data center, the first step is to partner with experts.

BTW, there are other things which I did not cover. For example, in network there are basic services such as DNS and NTP. All these services need to be monitored, typically for availability and reliability.

Network Observability

The arrival of software-defined infrastructure services also changes the way you monitor your network. The following diagram shows a simplified setup of an ESXi host.



In a single ESXi host, there are 4 areas that need to be monitored for a complete network monitoring:

- VM network
- The kernel network
- ESXi kernel modules
- Agent VMs

There are 2 layers of networking.

- The virtual network consists of VM and the kernel (e.g. vMotion). If the traffic is a VM to VM traffic within the same ESXi, the packets does not reach the physical network, hence the vmnic metrics do not register it. The virtual network does not have the limit that physical network does, if the traffic remains in the box. This makes it harder to use this metric as the 100% is not statically defined. So instead of just monitoring the throughput metric, you should also check the packet per second metric.
- The physical network means traffic going through the physical network card. At this level it's no longer aware of VM and the kernel.

In the preceding example, we have 3 VMs running in the host. VM 1 and VM 2 are connected to the same VXLAN (or VLAN). VM 3 is on a different VXLAN (or VLAN), hence it is on a different port group. Monitoring at port group level complements monitoring at VM level and ESXi level.

Traffic at Distributed Switch level carries more than VM traffic. It also carries the kernel traffic, such as vMotion and VSAN. Both the kernel network and VM network tend to share the same physical uplinks (ESXi vmnic). As a result, it's easier to monitor at port group level.

Sounds good so far. What is the limitation of monitoring at distributed port group level?

The hint is at the word **distributed**.

Yes, the data is the aggregate of all the ESXi hosts using that distributed port group!

By default, VM 1 and VM 2 can talk to each other. The traffic will not leave the ESXi. Network monitoring tools that are not aware of this will miss it. Traffic from VM 3 can also reach VM 1 or VM 2 if NSX Distributed Logical Router is in place. It is a kernel module, just like the NSX Distributed Firewall. As a result, monitoring these kernel modules, and the host overall performance, becomes an integral part of network monitoring.

The 4th area we need to monitor is Agent VM. An Agent VM is mapped to 1 ESXi Host. It does not need HA protection as every ESXi host has one, hence it typically resides on the host local datastore.



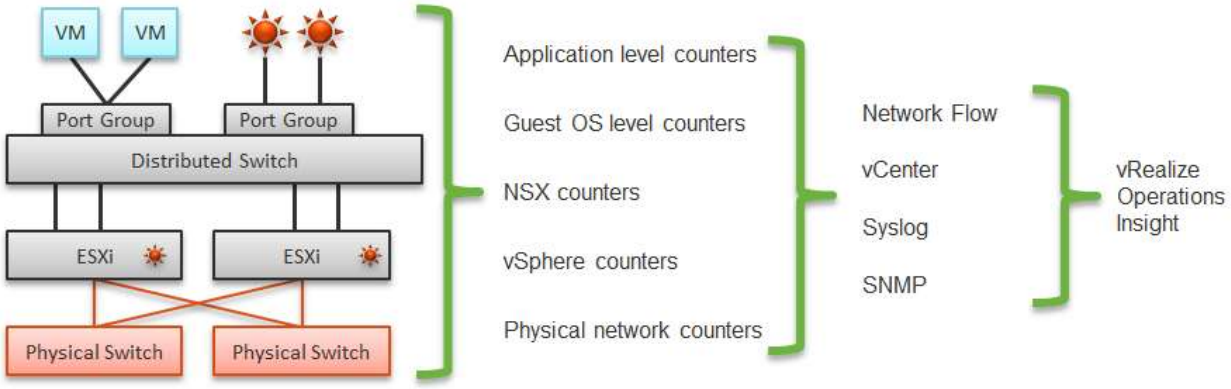
The above example shows an ESXi host with 3 agent VMs. The first VM provides a storage service (an example is Nutanix CVM), the second VM provides Network service, and the 3rd VM provides a Security VM.

Let's use the Security service as an example. A popular example here is Trend Micro Deep Security virtual appliance. It is in the data path. If the Business VMs are accessing files on a fileserver on another network, the files have to be checked by the security virtual appliance first. If the agent VM is slow (and it could be due to factor that is not network related), it will look like a network or storage issue as far as the business VMs are concerned. The Business VMs do not know that their files have been intercepted for security clearance, as it is not done at the network level. It is done at the hypervisor level.

Source of Data

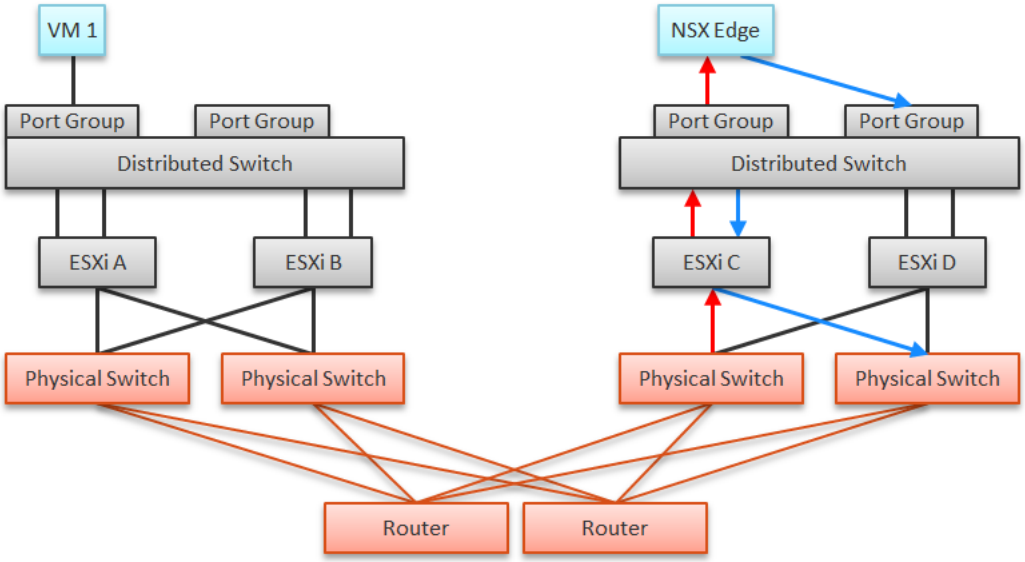
A complete network monitoring requires you to get the data from 5 different sources, not just from vSphere. In SDDC, you should also get data from the application, Guest OS, NSX and NetFlow/sFlow/IPFIX from VDS and physical network devices. For VDI, you need to get data at application level. We have seen packet loss at application-layer (Horizon Blast protocol) when Windows sees no dropped packet. The reason was the packet arrives out of order and hence unusable from protocol viewpoint.

The following shows a simplified stack. It shows the five sources of data and the 4 tools to get the data. It includes a physical switch as we can no longer ignore physical network once you move from just vSphere to complete SDDC.



The network packet analysis comes in 2 main approaches: Header analysis and full packet analysis. The header analysis is certainly much lighter but lack the depth of full analysis. You use this to provide overall visibility as it does not impose heavy load on your environment.

The impact of virtualization on network monitoring goes beyond what we have covered. Let's add NSX Edge into the above, so you can see the traffic flow when the edge services are also virtualized. You will see that a network problem experienced by a VM on one ESXi could be caused by another VM running on another ESXi. The following diagram is a simplified setup, showing a single NSX Edge residing on another cluster.



In the above example, let's say VM 1 needs to talk to outside world. An NSX Edge VM provides that connectivity, so every TCP/IP packet has to go through it. The Edge VM has 2 virtual NICs, one for each network. If the NSX Edge VM has CPU issue, or the underlying ESXi has RAM issue, it can impact the network performance of VM 1.

Type	Guest OS Metric	VM Equivalent
Contention	Dropped Packet	None, as the dropped packet at VM is a different level.
	Latency	None, no such metric available in vSphere
Utilization	Throughput	They should match with the metrics at VM level. If not, there is

	Packet/second	dropped packets. They should be reported based on traffic type: unicast, multicast, broadcast.

Traffic Flow

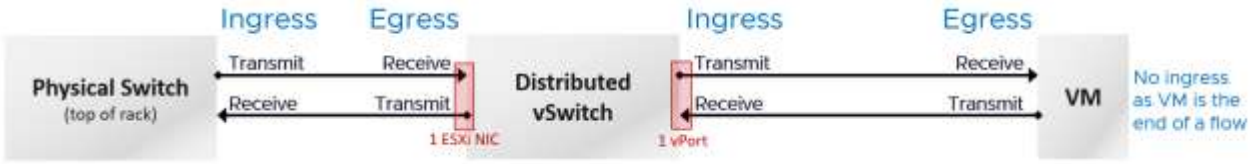
The terminology Egress and Ingress are used in context of an environment. For example, the flow from a VM to the Internet, where it will hop through different types of network devices. Typically, going further "inside" is considered ingress. That's why a VM does not have ingress traffic.

The terminology receive/RX and transmit/TX/send are used in context of a single port. A port has both incoming and outgoing traffic.

The following shows a simple example:



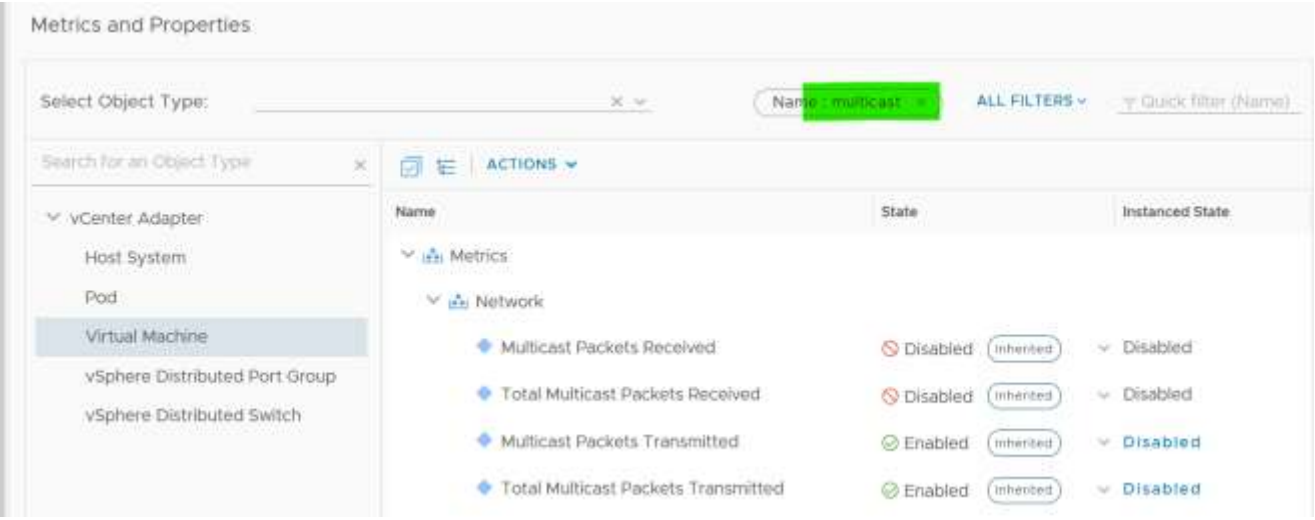
The following applies that to vSphere distributed vSwitch. Notice the ESXi host is not shown as it's not part of the hop. The ESXi host physical NIC card is the distributed vSwitch uplink.



Traffic Type

VCF Operations provides these metrics at VM, ESXi, Distributed Port Group and Distributed Switch level. As vSphere Tanzu Pod is basically a VM, it also has the metric.

BTW, one way to check what objects in what adapter have the specific metric is in the VCF Operations policy. Open any policy, and search the metric using its name. The list of matching metrics will be shown, grouped by the objects.



As you can see from above, there is no aggregation at higher level, so create super metric for the time being. I have not created those metric out of the box as I'm yet to use them in dashboard or alert.

Packet Size

It's typically 1600 byte with NSX, or 9000 bytes if you enable jumbo frames).

Special purpose packet such as ping test is smaller. But they should be a small percentage of your network.

Track the packet sizes and compare them with your expectation.

VM

VM is not an Operating System, so it has far less networking metric than Windows or Linux.

Overview

We will cover each metric in-depth, so let's do an overview first.

As usual, we start with contention. All we have is the dropped packet metrics.

Counters	Rollups	Units	Description
Receive packets dropped	Summation	num	Number of receives dropped
Transmit packets dropped	Summation	num	Number of transmits dropped

Next, you check if there are unusual traffic. Your network should be mostly unicast, so it's good to track the broadcast and multicast packets. They might explain why you have many dropped packets. If packets are broadcast packets, it might be dropped by the network.

Broadcast receives	Summation	num	Number of broadcast packets received during the sampling interval
Broadcast transmits	Summation	num	Number of broadcast packets transmitted during the sampling interval
Multicast receives	Summation	num	Number of multicast packets received during the sampling interval
Multicast transmits	Summation	num	Number of multicast packets transmitted during the sampling interval

Next you check utilization. There are 6 metrics, but I think they are triplicate.

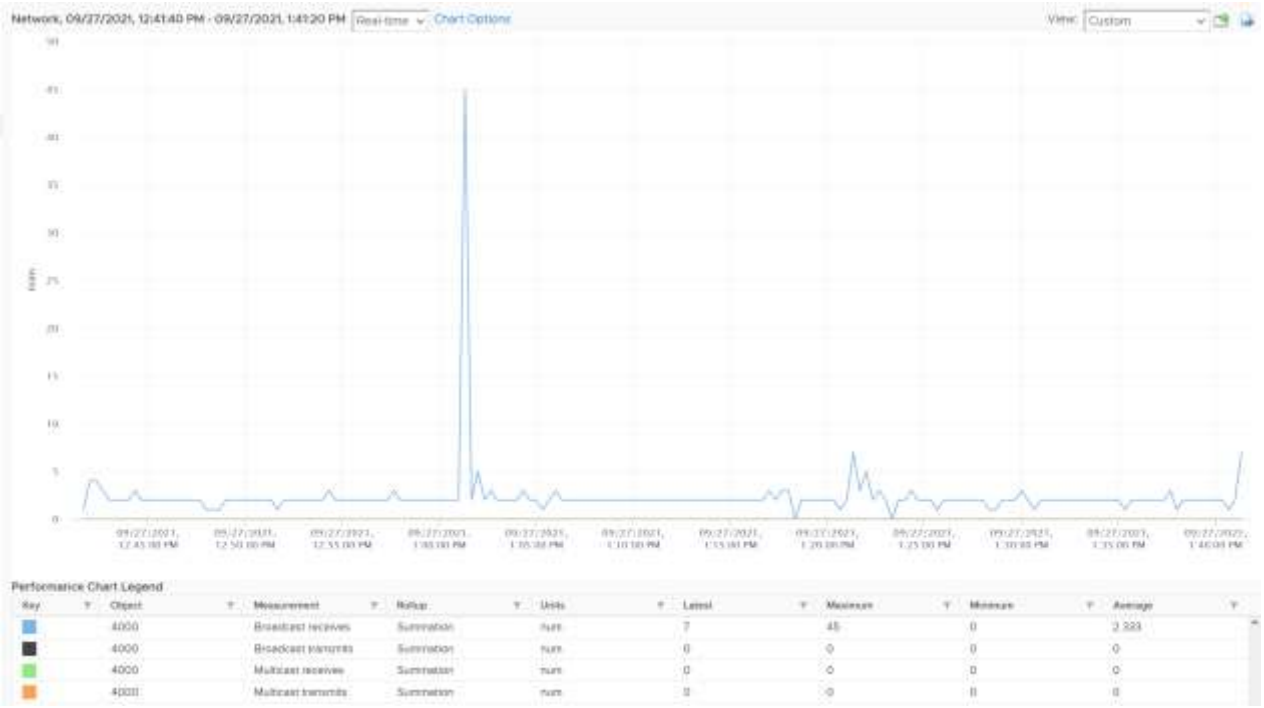
Data receive rate	Average	KBps	Average amount of data received per second
Data receive rate	Average	KBps	Average rate at which data was received during the interval
Data transmit rate	Average	KBps	Average rate at which data was transmitted during the interval
Data transmit rate	Average	KBps	Average amount of data transmitted per second
counter.net.pnicBytesRx.label	Average	KBps	counter.net.pnicBytesRx.summary
counter.net.pnicBytesTx.label	Average	KBps	counter.net.pnicBytesTx.summary

Each packet takes up CPU for processing, so it's good to check if the packet per second becomes too high

Packets received	Summation	num	Number of packets received during the interval
Packets transmitted	Summation	num	Number of packets transmitted during the interval

The metrics are available at each individual vNIC level and at the VM level. Most VMs should only have 1 vNIC, so the data at VM level and vNIC level will be identical.

The vNICs are named using the convention "400x". That means the first vNIC is 4000, the second vNIC is 4001, and so on. The following is a vCenter VM. Notice it receives a few broadcast packets, but it's not broadcasting (which is what you expect). It also does not participate in multicast, which is again expected.



The metrics are grouped into 2:

- Transmit for outgoing
- Receive for incoming.

For each group, the following metrics are provided:

Broadcast packets	Count of packets. It is the sum during the sampling window, not the rate (which is packet/second). Multicast packet and broadcast packet are listed separately. This is handy as they are supposed to low for most VM. Understand the nature of the applications so you can check if the behaviour is normal or not.
Multicast packets	
Packet dropped	
Total packets	The total includes the broadcast and multicast, but not the dropped ones.
Throughput per second	This is measured in kilobyte, as packet length is typically measured in bytes. While there are other packet sizes, the standard packet is 1500 bytes. BTW, esxtop measures in megabit. I assume this includes broadcast and multicast, but not the dropped packet.

Guess what metrics are missing?

- Retransmit. This can be useful in troubleshooting TCP packet. It naturally does not apply to UDP traffic.

- Latency.
  A normalized latency would help, especially if it's broken into internal network and external network. Network latency could be impacted by CPU. CPU might not fast enough to process the packet. In VM, this could also be due to the VM having CPU contention.
  If the latency is caused by too many hops and firewall, optimize the traffic using tools such as vRealize Network Insight.
- Packets per second. This can be derived by packet count / sampling window. If you have 200 packets in 20 seconds, that means 10 packets per second.
- Packet size. This can be computed by throughput / packet count. Expect this to be around 1500 byte.

BTW, if you see a pair of metrics with identical name, but one of them is prefixed with "Total", avoid the one without "Total." They are averaged over 15 data points, so their value is 15x lower.



Contention Metrics

As usual, let's approach the metrics starting with Contention. We covered earlier that the only contention metric is packet loss.

For TCP connection, dropped packet needs to be retransmitted and therefore increases network latency from application point of view. The counter will *not* match the values from Guest OS level. RX packets are dropped before it's handed into Guest OS, and TX packets are dropped after it left the Guest OS. ESXi dropped the packet because it's not for the Guest OS or it violates the security setting you set.

The following summary proves that receive packet gets dropped many more times than transmit packet. This is based on 3938 VMs. Each shows the last 1 month, so approximately 35 million data points in total. The average of 35 million data points show that dropped RX is significantly higher than dropped TX. This is why it's not in the SLA.

Name	Average Drop	RX Drop	TX Drop	RX	TX
corpm-p...	1.71 %	1,927.87	0.13	321.96	312.82
portal-p...	1.62 %	1,925.33	0.67	316.52	234.51
web-pro...	1.6 %	1,862.8	0.07	161.09	155.99
portal-p...	1.56 %	1,968.8	0.2	312.76	215.73
Summary	0.06 %	6,095.61	941.5	7,781.52	5,178.8

1 - 20 of 3938 items ‹ 1 2 3 4 5 … 197 ›

The following table shows that the drop is short and spiky, which is a good thing. The value at 99th percentile is 35x smaller than the value at 100th percentile.



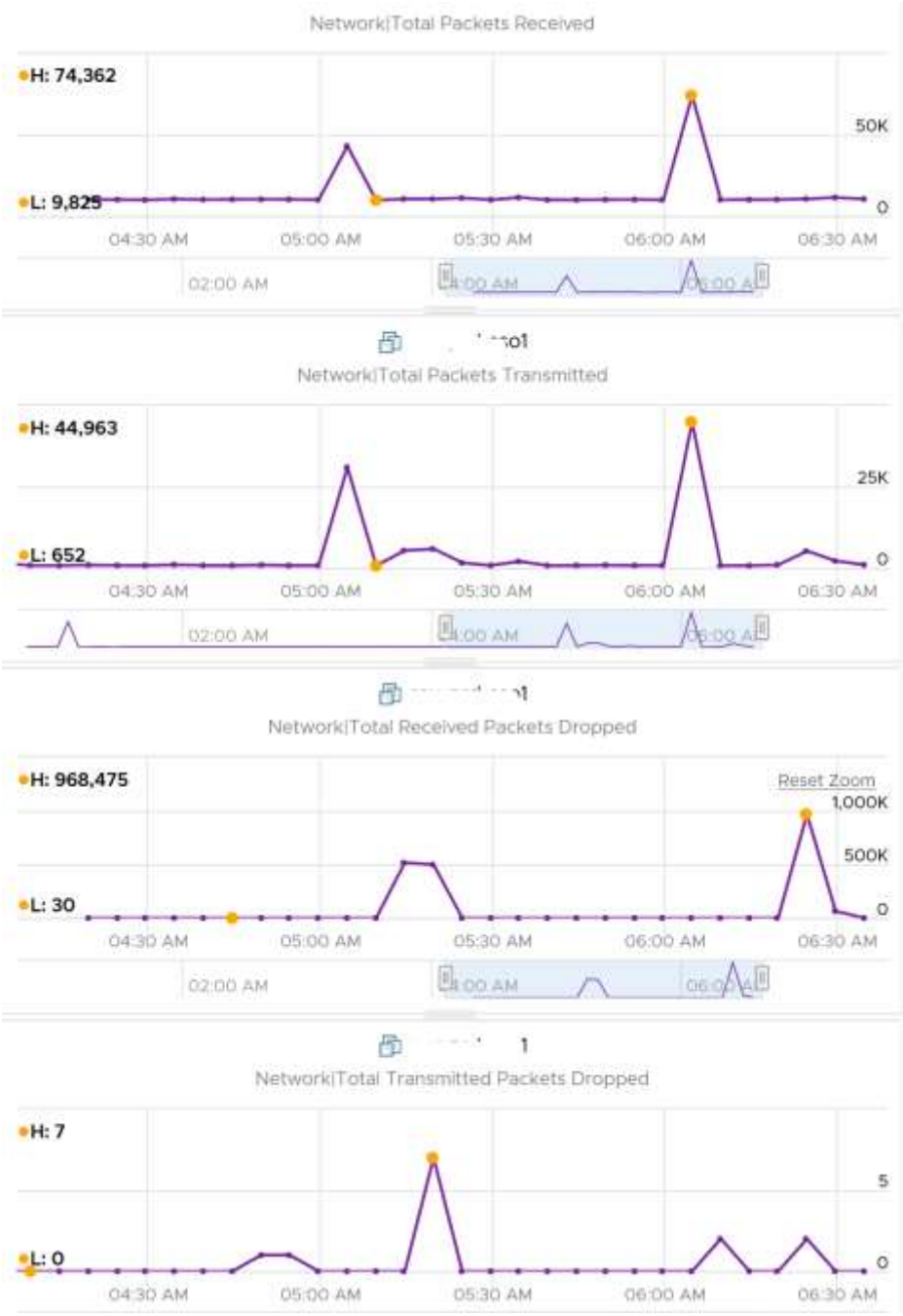
The high value in receive can impact the overall packet dropped (%) counter, as it's based on the following formula

```
dropped = Network|Received Packets Dropped + Network|Transmitted Packets Dropped
delivered = Network|Packets Received + Network|Packets Transmitted
Network|Packets Dropped (%) = dropped / (dropped + delivered) * 100
```

I've seen multiple occurrences where the packet dropped (%) jumps to well over 95%. That's naturally worrying. They typically do not last beyond 15 minutes.



In this, plot the following 4 metrics. You will likely notice that the high spike is driven by *low* network throughput and *high* received packet dropped.



Because of the above problem, profile your VM dropped packets, focusing on the transmit packets. The following is one way to do it, giving surprising results like this:

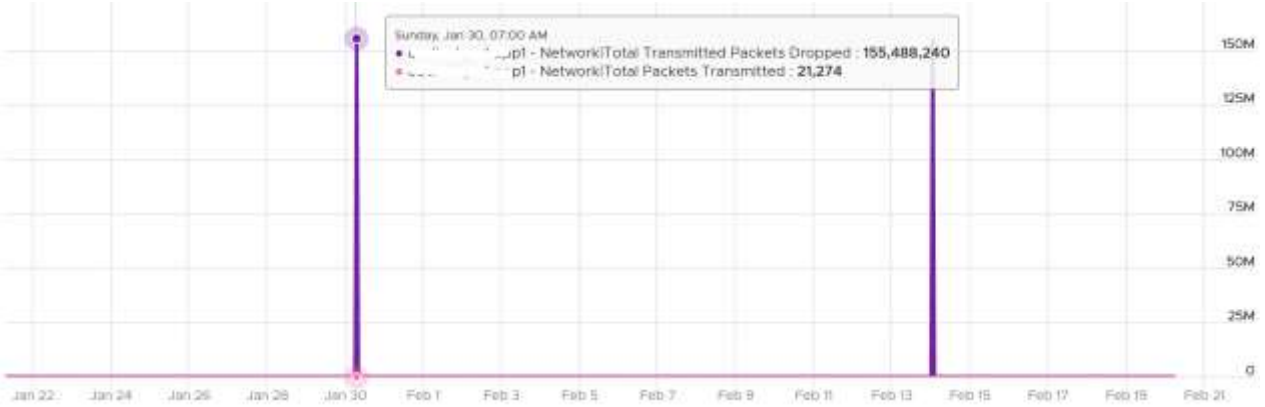
Name	All Drop at 99th Percentile	Sum of TX drop	Count of TX drop at 99th Percentile
wv...	2.26 %	6,026,306	4,050
CB...	9.07 %	3,062,394	1,865.76
SC...	13.13 %	2,746,787	1,344
ph...	0.01 %	833,283	1,216.8
ph...	0.02 %	795,154	1,214.9
PA...	0.04 %	707,092	869.77
PA...	0.03 %	687,848	834.78
bi-...	0.27 %	1,348,039	797.17
SC...	0.04 %	788,193	782
m...	1.15 %	922,267	750.84
te-...	0.48 %	4,069,541	617
SC...	0.05 %	12,406,013	604.77

The design of the preceding table is:

- First column calculates the percentage packets dropped. I took 99th percentile else many of the results will be 100%.
- Second column sums all the transmitted dropped packets (actual packet counts).
- Third column takes the 99th percentile maximum of dropped packet within any 300 seconds. Each network packet is typically 1500 bytes. Using 1.5 KB packet size, 1 thousand packets dropped = 1500 MB worth of packets within 300 seconds.

I don't expect dropped packets in data center network, so to see millions of dropped packets over a month needs further investigation with network team. Moreover, those metrics are Transmit, not Received. So the VM sent them but they got dropped. No one seem to complain, because packets are automatically retransmitted.

What I typically notice is the spike rarely happens. They look like an outlier, especially when the number is very high. The following is an example. I only showed in the last 1 month as the rest of the 6 months had similar pattern. The jump is well cover 100 million packets, and they were all dropped. Assuming each packet is 1 KB, since VCF Operations reports every 5 minutes, that's 333 MB per second sustained for 300 seconds.



I also notice regular, predictable pattern like this. This is worth discussing with network team. It's around 3800 packets each 5-minute, so it's worth finding out.



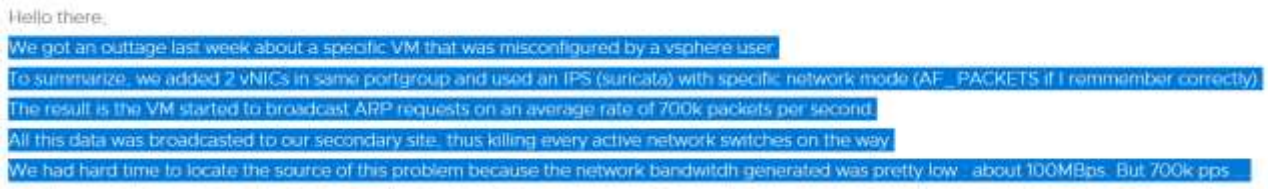
False positive on TX dropped packet because NSX firewall reject the outgoing packet. See this KB article.

Packet loss in Guest OS using VMXNET3: When using the VMXNET3 driver, you may see significant packet loss during periods of very high traffic bursts. The VM may even freeze entirely. This issue occurs when packets are dropped during high traffic bursts. This can occur due to a lack of receive and transmit buffer space or when receive traffic which is speed constrained.

Consumption Metrics

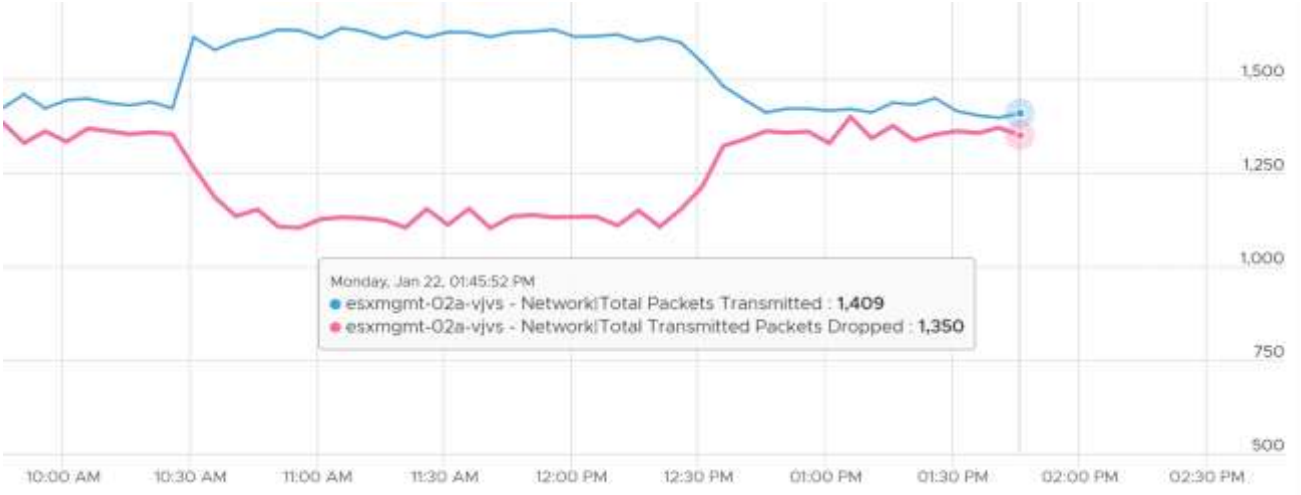
There are 2 main metrics to measure utilization: throughput and packets.

Both metrics matter as you may still have bandwidth but unable to process that many packets per second. This outage shows 700K packets per second that only consumes 800 Mbps as the packet is small. The broadcast packet is only 60 bytes long, instead of the usual 1500 bytes.

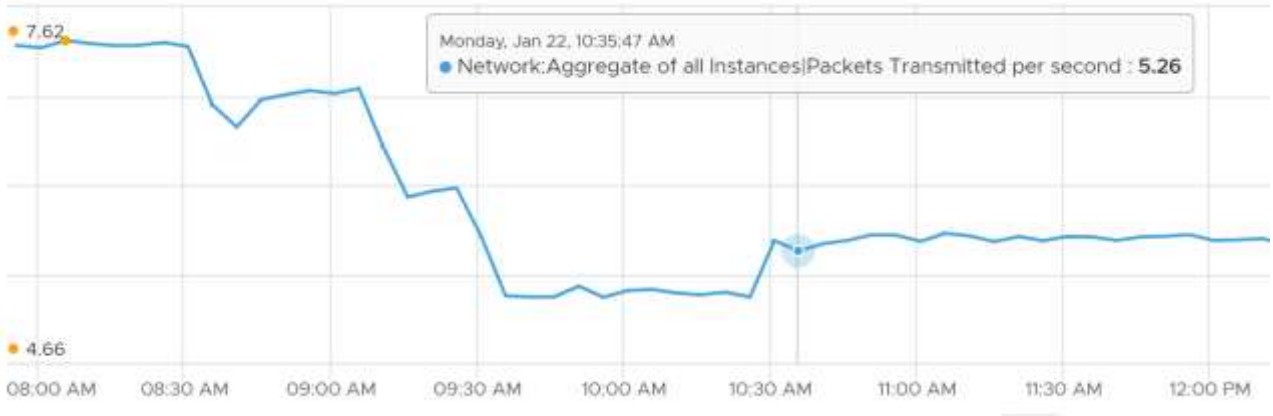

Hello there,
We got an outtage last week about a specific VM that was misconfigured by a vsphere user
To summarize, we added 2 vNICs in same portgroup and used an IPS (suricata) with specific network mode (AF_PACKETS if I remmember correctly)
The result is the VM started to broadcast ARP requests on an average rate of 700k packets per second
All this data was broadcasted to our secondary site, thus killing every active network switches on the way
We had hard time to locate the source of this problem because the network bandwitdh generated was pretty low - about 100MBps. But 700k pps

The packets transmitted does not include those dropped packets. Another word, it only counts packets that were successfully transmitted.

The following diagram proves the above relationship.



As a consequence, the packets transmitted per second = Total Packets Transmitted / 300 seconds.



ESXi

In vSphere Client, you can't see the virtual network traffic. The following shows that you can only see the physical network card.



The metrics are provided at both physical NIC card and ESXi level. The counter at host level is basically the sum of all the vmnic instances. There could be small variance, which should be negligible.

Key	Object	Measurem...	Rollup	Units	Latest	Maximum	Minimum	Average
	vmnic0	Usage	Average	KBps	1	25	1	3.067
	vmnic1	Usage	Average	KBps	5	139	4	13.022
	10.217.66.45	Usage	Average	KBps	7	164	5	16.556

Just like vCenter, VCF Operations also does not provide the metrics at the Standard Switch and its port groups. This means you cannot aggregate or analyze the data from these network objects point of view. You need to look at the parent ESXi one by one. Create a dashboard with interaction to cycle through the ESXi hosts.

Contention Metrics

In addition to the dropped packet, there are 2 other metrics tracking contention. They are error packets and unknown protocol frames.

Error Metrics

Packet receive errors	Summation	num	Number of packets with errors received during the sampling interval
Packet transmit errors	Summation	num	Number of packets with errors transmitted during the sampling interval
Unknown protocol frames	Summation	num	Number of frames with unknown protocol received during the sampling interval

A packet is considered unknown if ESXi is unable to decode it and hence does not know what type of packet it is. You need to enable this metric in VCF Operations as it's disabled by default.

Expect these error packets, unknown packets and dropped packets to be 0 at all times. The following shows from a single ESX:

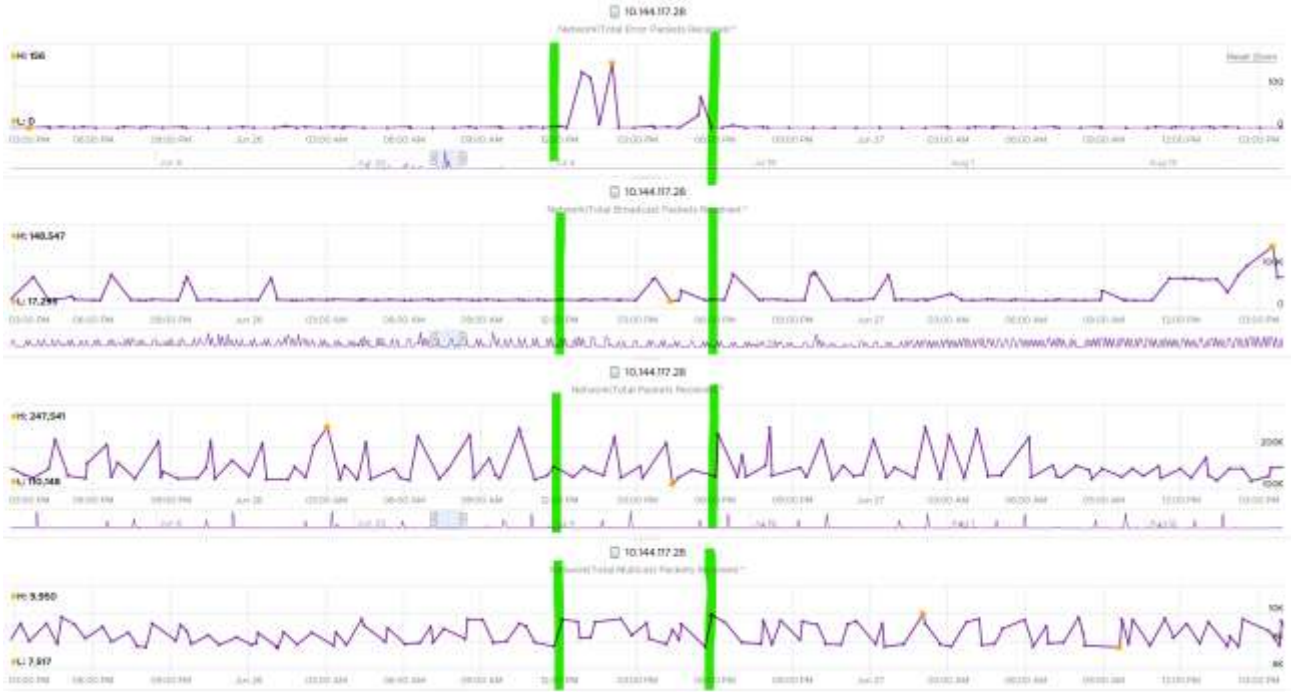
Object	Measurement	Rollup	Units	Latest	Maximum	Minimum	Average
evn1-hs1-0804.eng.vmware.com	Packet receive errors	Summation	num	0	0	0	0
evn1-hs1-0804.eng.vmware.com	Packet transmit errors	Summation	num	0	0	0	0
evn1-hs1-0804.eng.vmware.com	Receive packets dropped	Summation	num	0	0	0	0
evn1-hs1-0804.eng.vmware.com	Transmit packets dropped	Summation	num	0	0	0	0
evn1-hs1-0804.eng.vmware.com	Unknown protocol frames	Summation	num	0	0	0	0

To see from all your ESXi, use the view "vSphere \ ESXi Bad Network Packets".

Name	Error Packets Received	Error Packets Transmitted	Received Packets Dropped	Transmitted Packets Dropped	Hardware Model
10.144.117..	2,841	0	0	0	Dell Inc. PowerEdge R630
w2-haas..	2,834	0	0	0	Dell Inc. PowerEdge R630
w2-haas..	2,834	0	0	0	Dell Inc. PowerEdge R630
w2-haas..	2,834	0	0	0	Dell Inc. PowerEdge R630
w2-haas..	2,834	0	0	0	Dell Inc. PowerEdge R630
w2-haas..	2,834	0	0	0	Dell Inc. PowerEdge R630
w2-haas..	2,834	0	0	0	Dell Inc. PowerEdge R630
10.144.117..	2,834	0	0	0	Dell Inc. PowerEdge R630
10.144.117..	2,834	0	0	0	Dell Inc. PowerEdge R630
w2-haas..	2,834	0	0	0	Dell Inc. PowerEdge R630
10.144.117..	2,834	0	0	0	Dell Inc. PowerEdge R630
w2-haas..	2,834	0	0	0	Dell Inc. PowerEdge R630
w1-hs2-g..	168	0	0	0	Dell Inc. PowerEdge R740
w1-hs2-g..	160	0	0	0	Dell Inc. PowerEdge R740
w1-hs2-g..	160	0	0	0	Dell Inc. PowerEdge R740
10.127.16...	0	0	0	0	VMware, Inc. VMware Virtual Platform
10.127.16.11	0	0	0	0	VMware, Inc. VMware Virtual Platform
10.127.16...	0	0	0	0	VMware, Inc. VMware Virtual Platform
10.127.16...	0	0	0	0	VMware, Inc. VMware Virtual Platform
w1-hs2-g..	0	0	0	0	Dell Inc. PowerEdge R740
Summary	34,503	0	1	0	-

The hosts with error RX spans across different clusters, different hardware models and different ESXi build number. I can't check if they belong to the same network.

If you see a value, drill down to see if there is any correlation with other types of packets. In the following example, I do not see any correlation.



What I see though, is a lot of irregular collection. I marked with red dots some of the data collection.



You can see they are irregular. Compare it with the Error Packet Transmit counter, which shows a regular collection.



esxcli

The metrics at vCenter UI only shows the total. If you want to see the details, go to ESXi host console and issue an esxcli command.

The syntax is:

```
$ watch esxcli network nic stats get -n vmnic#
```

Take note:

- The counter is accumulative. That means the issue could have happened in the past.

See this KB for details. I'd copy the summary for convenience.

The error counters are:

Total receive errors	I think these 2 are what you see in the vCenter UI.

Total transmit errors	It's the summation of all the error below, including checksum error (which is not reported separately).
	Based on this KB article, the csumerr counter, available inside the driver's private statistics, cover the following: • Layer 3, which is introduced by hardware issue (typically cabling), • Layer 4, which cover Ingress packet has Layer4 checksum (packet is encapsulated, e.g. in vxlan or overlay), • Receive packet rate on the vmnic is too high and the hardware is unable to perform the checksum calculations in timely manner. This traffic will be passed on to the ESXi for delivery to the VM/vmknic (which is expected to do it's own checksum validations), but this traffic will be declared as error.
Receive length errors	The actual size of the packet does not match with the size of the packet being reported via the packet header
Receive over errors	Count the packets that are discarded by the hardware buffer of the card. This includes CRC error.
Receive CRC errors	The CRC value calculated by receiving ESXi does not match the value in the FCS field. If you're unfamiliar with CRC, see this.
Receive frame errors	
Receive FIFO errors Transmit FIFO errors	The physical network card unable to process due to RX ring buffe size is full
Receive missed errors	The physical network card unable to store or process packets due to hardware limitations
Transmit aborted errors	
Transmit carrier errors	
Transmit heartbeat errors	
Transmit window errors	

Dropped Packet

Receive packets dropped	Summation	num	Number of receives dropped
Transmit packets dropped	Summation	num	Number of transmits dropped

You've seen the dropped packet situation at VM. That's a virtual layer, above the ESXi. What do you expect to see at ESXi layer, as it's physically cabled to the physical top of rack switches? The counter tracks packets that are dropped prior to the packet reaching the ESXi kernel. According to this KB, "quite often this counter is a combination of the values from other counters that can be found in the Private Statistics section of the nicinfo.sh.txt file that is contained in the commands directory of ESXi host log bundles."

I plotted 319 production ESXi hosts, and here is what I got for Transmit. What do you think?

Name	Max TX Dropped	99P TX Dropped	Average TX	Peak TX
atl...	362	0	0.6 Gbps	8.38 Gbps
atl...	55	0	0.46 Gbps	8.95 Gbps
atl...	46	0	0.45 Gbps	7.95 Gbps
atl...	41	0	0.44 Gbps	5.64 Gbps
atl...	36	0	1.26 Gbps	8.52 Gbps
atl...	29	0	0.35 Gbps	8.16 Gbps
atl...	24	0	0.46 Gbps	7.49 Gbps
atl...	21	0	0.09 Gbps	4.02 Gbps
atl...	16	0	0.04 Gbps	6.02 Gbps
atl...	12	0	0.48 Gbps	7.26 Gbps
Su...	**2.24**	**0**	**0.29 Gbps**	**6.67 Gbps**

1 - 10 of 319 items < 1 2 3 4 5 ... 32 >

There are packet drops, although they are very minimal. Among 319 hosts, one has 362 dropped transmit packet in the last 3 months. That host was doing 0.6 Gbps on average and peaked at 8.38 Gbps.

As expected, the dropped packet rarely happened. At 99th percentile, the value is perfectly 0.

I tested with another set of ESXi hosts. Out of 123 servers, none of them has any dropped TX packet in the last 6 months. That's in line with my expectation. However, a few of them experienced rather high dropped RX packets.

Name	Received Packets Dropped ↓	Transmitted Packets Dropped
wdc-0...	2,574,166	0
wdc-0...	1,945,080	0
wdc-0...	935,837	0
wdc-it...	50,356	0
wdc-it...	30,938	0
wdc-0...	7,265	0
wdc-0...	906	0
wdc-0...	624	0
wdc-0...	550	0



The dropped only happened since the ESXi had an increased load



If you see something like this, you should investigate which physical NIC card is dropping packet, and which VMK interface is experiencing it.

While the number is very low, many hosts have packet drops, so my take is I should discuss with network team as I expect data center network should be free of dropped packets.

Received

What do you think you will see for Received?

Remember how VM RX is much worse than VM TX? Here is what I got:

Name	Max RX Dropped	99P RX Dropped	Average RX	Max RX	ESXi	Max TX Dropped	Peak TX
sc...	7,415,656	0	0.37 Gbps	4.42 Gbps	6.7.3	0	5.98 Gbps
sc...	5,633,351	2,121,175.76	0.3 Gbps	4.22 Gbps	6.7.3	0	6.68 Gbps
sc...	5,355,623	0	0.13 Gbps	4.78 Gbps	6.7.3	0	0.56 Gbps
sc...	3,149,483	1,424,756.4	0.24 Gbps	5.81 Gbps	6.7.3	0	8.57 Gbps
sc...	2,664,772	77,857.48	0.35 Gbps	4.23 Gbps	6.7.3	0	6.95 Gbps
sc...	2,156,437	1,045,323.22	0.22 Gbps	3.49 Gbps	6.7.3	0	5.07 Gbps
sc...	1,326,358	0	0.07 Gbps	4.83 Gbps	6.7.3	0	4.58 Gbps
sc...	936,407	9,269.8	0.37 Gbps	10.14 Gbps	6.7.3	0	2.98 Gbps
sc...	881,526	0	0.4 Gbps	6.66 Gbps	6.7.3	0	3.51 Gbps
sc...	202,110	0	1.65 Gbps	9.52 Gbps	6.7.3	0	3.77 Gbps
Su...	93,873.13	14,666.01	0.42 Gbps	7.39 Gbps	-	2.24	6.67 Gbps

1 - 10 of 319 items ‹ 1 2 3 4 5 … 32 ›

Surprisingly, the situation is the same for ESXi.

Some of them have >1 million packet dropped in 5 minute. Within these set of ESXi, some have regular packet dropped, as the value at 99th percentile is still very high. Notice none of the ESXi is dropping any TX packet.

I plotted the 2nd ESXi from the table, as it has high value at 99th percentile. As expected, it has sustained packet dropped lasting 24 hours. I marked the highest packet drop time, as it mapped to the lowest packets received.



vsish

vsish provides more information that is not available in vSphere Client UI and VCF Operations.

```
vsish -e get /net/portsets/DvsPortset-0/ports/67109026/clientStats
port client stats {
   pktsTxOK:154121
   bytesTxOK:63326625
   droppedTx:0
   pktsTsoTxOK:0
   bytesTsoTxOK:0
   droppedTsoTx:0
   pktsSwTsoTx:0
   droppedSwTsoTx:0
   pktsZerocopyTxOK:45817
   droppedTxExceedMTU:0
   pktsRxOK:339700
   bytesRxOK:257901191
   droppedRx:2620 ← the reason will appear on the next output below
   pktsSwTsoRx:0
   droppedSwTsoRx:0
   actions:0
   uplinkRxPkts:0
   clonedRxPkts:0
   pksBilled:0
   droppedRxDueToPageAbsent:0
   droppedTxDueToPageAbsent:0
}
```

We saw dropped packets, so we probe deeper for the reason

```
vsish -e get /net/portsets/DvsPortset-0/ports/67109026/vmxnet3/rxSummary
stats of a vmxnet3 vNIC rx queue {
   LRO pkts rx ok:0
```

```
   LRO bytes rx ok:0
   pkts rx ok:340093
   bytes rx ok:257984247
   unicast pkts rx ok:253678
   unicast bytes rx ok:245663220
   multicast pkts rx ok:42220
   multicast bytes rx ok:7497292
   broadcast pkts rx ok:44195
   broadcast bytes rx ok:4823735
   running out of buffers:2620     ← the reason for 2620 packets dropped
   pkts receive error:0
   1st ring size:512
   2nd ring size:512   ← the ring size is on the small side. I'd say set to 2K.
   # of times the 1st ring is full:354    ← this line shows the first ring is full 354x
   # of times the 2nd ring is full:0
   fail to map a rx buffer:0        ← other reasons look good
   request to page in a buffer:0
   # of times rx queue is stopped:0      ← other reasons look good
   failed when copying into the guest buffer:0     ← other reasons look good
   # of pkts dropped due to large hdrs:0
   # of pkts dropped due to max number of SG limits:0
   pkts rx via data ring ok:0
   bytes rx via data ring ok:0
   Whether rx burst queuing is enabled:0
   current backend burst queue length:0
   maximum backend burst queue length so far:0
   aggregate number of times packets are requeued:0
   aggregate number of times packets are dropped by PktAgingList:0
   # of pkts dropped due to large inner (encap) hdrs:0
   number of times packets are dropped by burst queue:0
   number of times packets are dropped by rx try lock queueing:0
   number of packets delivered by burst queue:0
   number of packets dropped by packet steering:0
   number of memory region lookup pass in Rx.:0
   number of packets dropped due to pkt length exceeds vNic mtu:0
   number of packets dropped due to pkt truncation:0
}
```

Networking VMs, such as firewall and routers, or any high VMs expecting high packet rates, check if the VM is requesting NetQ RSS.

Consumption Metrics

As expected, you get the 2 types of throughput:

- bits/second
- packets/second.

For bits/second, the metrics are:

Counters	Rollups ↑	Units	Description
Data receive rate	Average	KBps	Average amount of data received per second
Data receive rate	Average	KBps	Average rate at which data was received during the interval
Data transmit rate	Average	KBps	Average rate at which data was transmitted during the interval
Data transmit rate	Average	KBps	Average amount of data transmitted per second
Usage	Average	KBps	Network utilization (combined transmit-rates and receive-rates) during the interval

I'm unsure why there are duplicates metrics.

We covered earlier that full duplex means the aggregated metric can exceed the physical speed. Notice the Usage Rate is the sum of Receive and Transmit on the following screenshot.



You can also plot each vmnic one by one. Since you may not know which one to plot for a given ESXi, you can show them all in table first.

Name	vmnic	TX Rate	RX Rate
10.119.161.131	vmnic0	0 Gbps	0 Gbps
10.119.161.131	vmnic1	0 Gbps	0 Gbps
10.119.161.131	vmnic2	0.03 Gbps	0.01 Gbps
10.119.161.131	vmnic3	0.67 Gbps	0.48 Gbps
10.119.161.131	vmnic4	0 Gbps	0 Gbps
10.119.161.131	vmnic5	0 Gbps	0 Gbps

For packets/second, the metrics are:

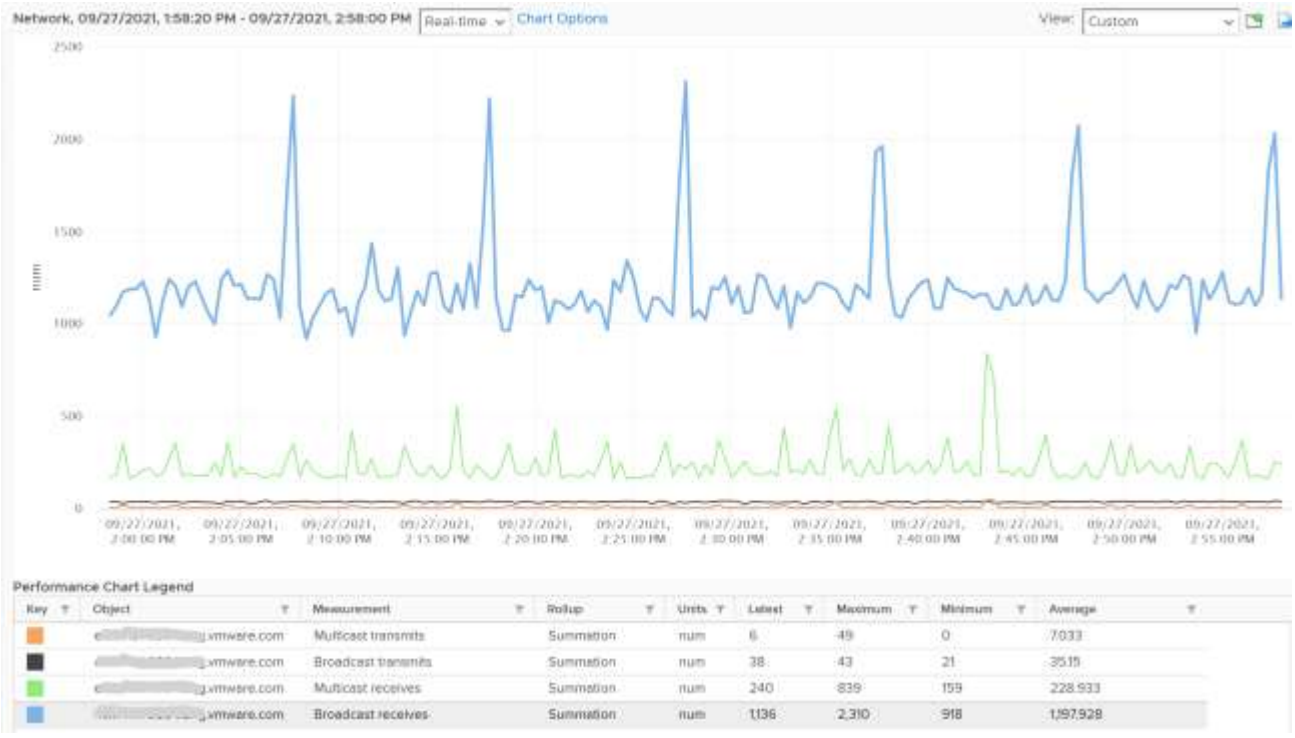
Packets received	Summation	num	Number of packets received during the interval
Packets transmitted	Summation	num	Number of packets transmitted during the interval

It's interesting to divide the packet/second with the bits/second, as you get the packet size. If this number change drastically in large environment, it's something worth investigating.

Unusual Packets

Broadcast receives	Summation	num	Number of broadcast packets received during the sampling interval
Broadcast transmits	Summation	num	Number of broadcast packets transmitted during the sampling interval
Multicast receives	Summation	num	Number of multicast packets received during the sampling interval
Multicast transmits	Summation	num	Number of multicast packets transmitted during the sampling interval

Your VM network should be mostly unicast traffic. So check that broadcast and multicast are within your expectation. Your ESXi Hosts should also have minimal broadcast and multicast packets.



Performance Chart Legend

Key	Object	Measurement	Rollup	Units	Latest	Maximum	Minimum	Average
	[redacted].vmware.com	Multicast transmits	Summation	num	6	49	0	7.033
	[redacted].vmware.com	Broadcast transmits	Summation	num	38	43	21	35.15
	[redacted].vmware.com	Multicast receives	Summation	num	240	839	159	228.933
	[redacted].vmware.com	Broadcast receives	Summation	num	1,136	2,310	918	1,197.928

Consumer Objects

Part 3 Chapter 6

Performance: VM

This section covers metrics that cut across the 4 elements of infrastructure, or metrics that are outside these 4 elements.

VM Performance (%)

With so many metrics, how do you monitor at scale? Say you have 5000 VMs, and you want to monitor every 5 minutes and see the performance trend in the last 24 hours. That would be far too many trend charts.

Enter Performance (%) metric.

Now that we've mastered the raw metrics, we're in the position to combine them to answer complex question. When we combine, we need to be mindful of which persona needs this summary metric. The answer is VM Owner. The metric is not designed for infrastructure team.

The following diagram put together all the metrics from Guest OS and VM. VM KPI includes Guest OS metrics as operationally we troubleshoot them as one, due to their 1:1 relationship.

For completeness, I added the utilization metrics to act as leading indicators.



Metric Used

Here is what I recommend, including their threshold.

	Metric	Green	Yellow	Orange	Red
Guest OS Contention	Peak CPU Run Queue Length per vCPU	0 – 2	2 – 4	4 – 8	8 – 16
	Peak Disk Queue Length (count)	0 – 20	20 – 40	40 – 80	80 – 160
Guest OS Utilization	Free Memory (MB)	512 – 1024	256 – 512	128 - 256	0 - 128
	RAM Page-out Rate (page/sec)	0 – 1000	1K - 2K	2K – 4K	4K – 8K
VM Contention	Peak CPU Co-Stop among any vCPU (%)	0 – 1%	1 – 2	2 – 4	4 – 8%
	Peak CPU Ready among any vCPU (%)	0 – 2%	2 – 4%	4 – 8%	8 – 16%
	Peak CPU Overlap (%)	0 – 0.5%	0.5 – 1	1 – 2	2 – 4%
	Peak RAM Contention (%)	0 – 0.5%	0.5 – 1	1 – 2	2 – 4%
	Peak Read/Write Latency among vDisk (ms)	0 – 10 ms	10 – 20 ms	20 – 40 ms	40 – 80 ms
	Peak Network TX Dropped packet (%)	0% - 0.25%	0.25 – 0.5%	0.5 - 1%	1 - 2%
VM Utilization	CPU Net Run	0 – 95%	95 - 97%	97 – 99%	99 - 100%
IaaS High Utilization	VM Memory Ballooned (%)	0 – 2%	2 – 4%	4 – 8%	8 – 16%
	VM Memory Compressed + Swapped (%)	0% - 0.25%	0.25 – 0.5%	0.5 - 1%	1 - 2%

Q: Why do we use 4 metrics for CPU instead of 1? Wouldn't that quadruple their share?

A: Yes, it will. This is a known limitation. We cannot combine them into a single metric as they don't peak at the same time within the 20-sec period.

Q: Why do the 4 components of CPU contention have different threshold since their impact is the same?

A: They have different level of sensitivity. The gap is sizable. To standardize will result in compromise. For example, Ready tends to have 2x the value of Co-Stop.

We can only add a metric in the table if it can be quantified into the 4 brackets. If a metric cannot be bucketized, it could do a disservice to the KPI. Hence majority of utilization metrics (e.g. disk IOPS, network throughput) are not here. I do not recommend you add them as they can mask out the real problem.

Guest OS	Guest OS metrics are included as they do not have VM equivalent, and they change the course of troubleshooting.
	CPU run queue is set to higher than normal to reduce false positive.
VM CPU	The complete set of CPU contention is provided. There are 4 metrics tracking the different type of contention or wait that CPU experiences.
	Do you know why we use CPU Net Run as opposed to CPU Usage? Read the VM CPU Metrics section. Theoretically, we should have chosen a higher threshold for CPU utilization. Practically, it's rare the CPU Run goes to 95% as CPU Used would have reached the VM CPU capacity. I set 60% as the starting line so there is no penalty for utilization below 60%.
	Memory Contention is the *only* counter tracking if the *VM* has memory

VM Memory	problem. VM and Guest OS can have memory problem independently. In future, we should add Guest OS memory performance metrics, if we find a good one. Linux and Windows do not track memory latency, only track memory disk space consumption, throughput and IOPS. These 2 OSes do not track latency, which unfortunately is the main counter for performance.
	Memory Ballooned, Swapped, Compressed are added even though their presence does not indicate real performance as they are leading indicators. That's why their threshold is higher than memory contention.
	Swapped and Compressed are combined as they are the result of the same action. Together they tell the complete picture. As they may not indicate actual problem, their threshold is 2x of memory contention
Network	vCenter does not have latency and re-transmit. So we have to resort to dropped packet metric.
	The threshold is not low as dropped packets are automatically restranmitted. Practically, this means the application team may not notice it.
	Take note of known limitation. While the packet dropped is based on the peak 20-second, the packet transmitted is based on the average of the collection period. The reason is we don't know which data points to take among the 15 data points in the default collection period.

With so many metrics, it can get difficult to debug. Also, the number will tend to be high as a single number will not have enough weightage to bring down. These are 2 things you can do:

- Combine the 4 VM CPU contention into a single metric. The drawback is there is possibility that the 4 peak values happened at different time within the 5 minute window.
- Split the table into 2 and take the lowest. There are 2 ways, both have their pros and cons.
    - Approach 1 is by layer. Table 1 is for Guest OS (4 metrics), while Table 2 is for the rest (7 metrics).
    - Approach 2 is by importance. Move less important metrics to Table 2.

Metrics Not Used

What metrics are missing from the tables?

Most utilization, such as disk IOPS, are excluded as we can map their values to the 4 brackets. There is no univeral mapping that works for all VMs.

The following metrics are not included, along with the reason why:

Guest OS : VM Ratio	Guest OS IOPS : VM IOPS Ratio. They should be near 1 or a stable number, as the block size should be identical. The actual numbers may not match, as Guest OS tends to report the last value, while VM tends to report average value. If they fluctuate greatly, something amiss. I do not include as I do not have the data yet.
Guest OS	CPU Context Switch is not included as the profiling shows this metrics has a very wide band, making it impractical.

	Memory page-in could contain Windows or Linux application binary, so its value could be over reported. Based on our profiling of 3300 production VM, the page-in is more volatile so I'm less confident of applying a threshold
	Guest OS swap file remaining free capacity is not included as I'm not sure if low value impacts performance
	Number of dead processes is not included as I'm not sure what value to set for each bracket. We need to profile Windows and Linux separately
Memory	Popular VM metrics such as Consumed, Active, Granted, etc are not shown as they do *not* indicate performance problem
	RAM Page Fault is not included as we do not have the metric.
	Swapped File Remaining is not included as it may corelate with available memory?
	Memory page-in is not included as it contains proactive fetch
Storage	Adding Disk Outstanding IO will be duplicating as it's a function of IOPS x latency
	Snapshot latency is excluded as the metric VM Other Wait already covers it, so no need to double count
Network	Network packets, such as broadcast and multicast, are not desirable. However, they do not actually cause performance.
	Error packets are excluded as Tools do not have it.
	Network queue is excluded as no such metric.
	RX Dropped Packets is excluded as too many false positive.
	Packets per second is excluded as we can't quantify the 100%.
VM DRS Score	Niels Hagoort states here that "a VM running a lower score is not necessarily running properly. It is about the execution efficiency, taking all the metrics/costs into consideration."
	The metric considers CPU, Memory (including Memory Tiering), Network, Fault Tolerance, vSAN, Storage and Maintenance. VM DRS score = Performance Goodness * 0.8 / CPU Demand 0.8 is a threshold of performance degradation.
	Reading the blog and other material, this metric is more about the cluster performance than the individual VM performance. Plus, it's using metrics that are already included in the KPI, so it's double counting.
vMotion	I do not have enough data to decide the value to put for each range of either downtime or stun time. It should be within 0.2 second for green, but what about yellow? Typically, I used 2K – 4K VMs over 3 months to convince myself that the thresholds are representing real world. Use Log Insight to profile this.
CPU Other Wait	Too many counters for CPU, which incorrectly inflates the importance of CPU factor. Plus, there is a rare bug which results in false positive.

Notice all of them are VM or Guest OS metrics. No ESXi, Resource Pool, Datastore, Cluster, etc metrics. Why?

The reason is the metrics at these "higher-level" objects are mathematically an average of the VMs in the object. A datastore with 10 ms disk latency represents a normalized/weighted average of all the VMs in the datastore. Another word, these metrics give less visibility than the 12 above, and they can be calculated from the 12.

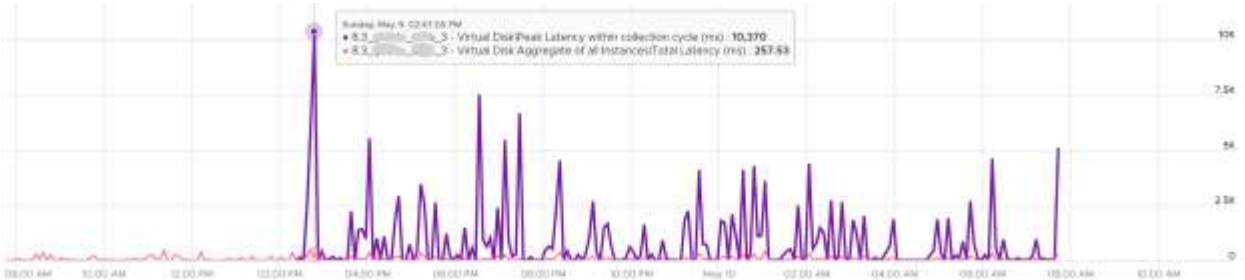
And 1 more reason: You troubleshoot VM, not infrastructure. If there is no VM, there is no problem 😊

20-second Peak Metrics

VCF Operations collects and stores data every 5 minutes. This is good enough for monitoring use case, but not for troubleshooting. 300-second average is not granular enough, as performance problem may not be sustained that long. Even a performance issue that last days may consist of repeated microbursts. I check if repeated bursts exist by profiling a few thousand VMs. Here are some of the results. I compare 3 metrics (disk latency, network throughput and CPU context switch).

Disk Latency	Peak Latency ↓	Network Usage	Peak Usage ↓	CPU Context Switch	Peak CPU Context Switch
22.67 ms	272 ms	7,792.97 Mbps	18,636.24 Mbps	6,563.73	38,560
35.43 ms	271 ms	8,771.29 Mbps	18,472.47 Mbps	35,597.4	36,850
12.8 ms	235 ms	7,052.6 Mbps	15,402.34 Mbps	30,561.87	32,575
17.33 ms	228 ms	9,192.6 Mbps	14,639.09 Mbps	25,765.4	31,806
0 ms	175 ms	57.59 Mbps	311.38 Mbps	27,520.6	30,648
35.74 ms	168 ms	51.26 Mbps	296.35 Mbps	12,995.2	29,401
7.48 ms	148 ms	43.02 Mbps	241.71 Mbps	11,259.2	27,697
11.5 ms	129 ms	19.29 Mbps	141.88 Mbps	13,083.93	24,938
8.47 ms	113 ms	23.78 Mbps	139.78 Mbps	9,610.47	23,703
3.95 ms	110 ms	22.28 Mbps	113.8 Mbps	20,738.27	22,910
10.8 ms	106 ms	65.7 Mbps	107.31 Mbps	5,632.93	22,263
2.66 ms	102 ms	7.84 Mbps	50.59 Mbps	18,679.8	20,752

The peak column is based on 20-second average. So it's 15x sharper than the 300-second average. It gives better visibility into the microbursts. If the burst exists, you will see something like this, where the 20-second shows much worse value consistently.



Are you surprised to see that the 20-second peak is a lot worse than 15x worse? The preceding chart shows 10370 ms latency at 20-second vs 257 ms at 300 second.

The huge gap is due to 2 things

- There is only 1 or 2 microbursts, and it's much higher than the average. This can happen on counter such as disk latency and CPU context switch, where the value can be astronomically high.
- There are many sets. A VM can have many disks. For example, a database VM with 20 virtual disks will have 40 sets of metrics. Each set has 15 datapoints, giving a total of 600 metrics. The peak is reporting the highest of 600 metrics. If the remaining is much lower, then the gap will naturally be high.

How are they chosen?

Take a look at the table below. It shows a VM with 2 virtual disks. Each disk has its own read latency and write latency, giving us a total of 4 metrics.

	15 datapoints collected in 5 minutes. Each is 20 seconds average.					Summary	
	0 - 20 sec	20 - 40 sec	40 - 60 sec	.....	280 - 300 sec	Peak	Average
VM 1: Disk 1 Read Latency	1 ms	7 ms	100 ms	.....	14 ms	100 ms	31 ms
VM 1: Disk 2 Read Latency	3 ms	28 ms	51 ms	.....	29 ms	51 ms	28 ms
VM 1: Disk 1 Write Latency	21 ms	19 ms	9 ms	.....	38 ms	38 ms	22 ms
VM 1: Disk 2 Write Latency	15 ms	46 ms	8 ms	.....	15 ms	46 ms	21 ms
Summary at VM level (Average)							25 ms
Summary at VM level (Peak)						100 ms	

What VCF Operations 8.3 does is to add a new metric. It does not change the existing metric, because both have their own purpose. The 5-minute average is better for your SLA and performance guarantee claim. If you guarantee 10 ms disk latency for every single IOPS, you'd be hard pressed to deliver that service. These new metrics act as early warning. It's an internal threshold that you use to monitor if your 5-minute SLA is on the way to be breached.

VCF Operations 8.3 takes the peak of these 15 data points, and stores them every 5 minutes. It does *not* store all 15 data points, because that will create a lot more IOPS and consume more storage. It answers the question "Does the VM or Guest OS experience any performance problem in any 20-second period?"

Having all 20-second data points are more natural to us, as we're used to 1 second in Windows and 20 second in vCenter performance charts. But how does that additional 14 data points change the end remediation action? If the action you take to troubleshoot is the same (e.g. adjust the VM size), why pay the price of storing 15x more data points?

In the case of virtual disk (as opposed to say memory), a VM can have many of them. A database VM with 20 virtual disks will have 40 peak metrics. That also means you need to check each one by one. So VCF Operations 8.3 takes the peak among all virtual disks read and write. It does the same thing with vCPU. A monster VM with 64 vCPU will only have 1 metric, but this metric is the highest among 64 virtual CPU. There is no need to have visibility into each vCPU as the remediation action is the same. Whether it's vCPU 7 or vCPU 63 that has the problem, it does not change the conclusion of troubleshooting in most cases.

	5-minute Average	20-second Average
Guest OS	CPU Run Queue	Peak CPU Queue within collection cycle

	CPU Context Switch	Peak CPU Context Switch within collection cycle
	Memory Page-out Rate	Peak Guest OS Page-out Rate within collection cycle
	Disk Queue Length	Peak Disk Queue within collection cycle
VM CPU	Ready (%)	Peak vCPU Ready within collection cycle
	Co-Stop (%)	Peak vCPU Co-Stop within collection cycle
	Other Wait (%)	Peak Other Wait within collection cycle
	Overlap (second)	Peak vCPU Overlap within collection cycle
VM Memory	Contention (%)	Peak Memory Contention within collection cycle
VM Disk	Read Latency (ms)	Peak Latency within collection cycle
	Write Latency (ms)	
VM Network	Usage Rate (KB/s)	Peak Usage Rate within collection cycle
	Packet/sec	Peak Network Packets/sec within collection cycle
	Transmitted Packet Dropped	Peak Transmitted Packet Dropped within collection cycle

What's the limitation?

- You can't see a pattern within the 300 seconds window as you only have 1 data point. This is largely mitigated by having the average counter also. If the delta between the maximum and average is high that means the maximum is likely a one-off occurrence. The pattern can also be seen over longer period of time.
- The peak can be from a different 20-second block. That means you can't associate that the contention is caused by high utilization as the 2 metrics can come from different 20-second block within that 5-minute time period.

Implementation

The following shows the part of the formula where we label the metrics used. Label is handy to shorten the overall length while improving readability.

It shows the 12 metrics used.

```
sum([
({This Resource: guest|20_sec_peak_cpu_queue}                  as cpuQ
({This Resource: guest|20_sec_peak_disk_queue}                 as DiskQ
({This Resource: cpu|20_sec_peak_readyPct}                     as Ready
({This Resource: cpu|20_sec_peak_costopPct}                    as coStop
({This Resource: cpu|20_sec_peak_iowaitPct}                    as oWait
({This Resource: cpu|20_sec_peak_overlap}                      as Overlap
({This Resource: mem|20_sec_peak_host_contentionPct}           as mLatc
({This Resource: virtualDisk|20_sec_peak_totalLatency_average} as dLatc
({This Resource: guest|mem.free_latest}                        as mFree / 1024
({This Resource: guest|page.outRate_latest}                    as PageO
({This Resource: mem|balloonPct}                               as Balloon
({This Resource: cpu|net_run}                                  as Run
({This Resource: Super Metrics|vSphere VM Memory Zip Swap}     as ZipSwap
({This Resource: net|droppedTxPct_max}                         as netTXd
])
```

I copied into an external editor to show each metric as 1 line so you can see them more easily.

Free memory measures the disk space. It's more important to see the latest value. Average of last 5 minutes is better than lowest 20-second period within that 5 minute period.

I've rearranged the metrics so you can see the 20 second peak metrics first.

Guest OS Free Memory is in KB, so we need to divide by 1024. Note the division can't be part of the label.

There is no metric that sums the VM zipped page and swapped page, so I created a super metric for it.

The Green Range

See the concept of the 4 ranges in the Performance Modelling section of Chapter 1.

To see the threshold more easily, I rearranged the metrics from the smallest threshold, and have skipped the metrics with special logic. They will be covered later.

```
netTXd      <    0.25 ? (100 - ( netTXd  /    0.25 * 25 ) )
Overlap     <    0.5  ? (100 - ( Overlap /    0.5  * 25 ) )
mLatc       <    0.5  ? (100 - ( mLatc   /    0.5  * 25 ) )
ZipSwap     <    0.5  ? (100 - ( ZipSwap /    0.5  * 25 ) )
cpuQ        <    1    ? (100 - ( cpuQ    /    1    * 25 ) )
coStop      <    1    ? (100 - ( coStop  /    1    * 25 ) )
oWait       <    1    ? (100 - ( oWait   /    1    * 25 ) )
Ready       <    2    ? (100 - ( Ready   /    2    * 25 ) )
Balloon     <    2    ? (100 - ( Balloon /    2    * 25 ) )
dLatc       <   10    ? (100 - ( dLatc   /   10    * 25 ) )
DiskQ       <   20    ? (100 - ( DiskQ   /   20    * 25 ) )
PageO       < 1000    ? (100 - ( PageO   / 1000    * 25 ) )
```

Let's go through the first line. It checks if network TX dropped is below 0.25%, it will prorate the value to a number between 75% and 100%. To prorate, we do this logic:

- Divide the value with the green range size to get the fraction.
- Multiply the result by 25 to adjust the scale to 25 points.
- Subtract the result from 100 (perfect score), the starting line for green.

BTW, I didn't add the * 1 weightage multiplier for green at the end to keep the formula simpler. Mathematically it's not required.

Now let's look at CPU Run, as it's a special case.

```
Run < 80  ?  ( 100 - ( max([Run - 60,0]) / 20 * 25 ) )
```

The range of green is 60% – 80%, not 0 – 80. This means utilization below 60% is given a perfect score of 100%.

Why do we take the maximum of Run – 60 and 0? To ensure we don't have negative value when the CPU Net Run metric is below 60%. This max() function will return 0, hence giving us 100 – 0 = 100%.

Now let's look Guest OS Free Memory.

```
mFree/1024 > 512  ?  min ([mFree/1024 - 512,512]) / 512 * 25 + 75
```

This one is special as the threshold is going descending. We also need to convert the unit from KB to MB.

We want the value of free memory > 512 MB of RAM to translate into 100%. Since the formula returns 0% - 25%, I had to add 75.

The Yellow Range

When the metric value does not within green threshold, it gets evaluated for yellow.

The outcome is 50% - 75%, hence the formula deducts from 75.

Just like the green range, let's start with the easy ones. You notice the following looks similar to the green range.

```
netTXd  <    0.5 ? (75 - (( netTXd  - 0.25 ) /    0.5 * 25 )) *2
Overlap <    1   ? (75 - (( Overlap - 0.5  ) /    0.5 * 25 )) *2
mLatc   <    1   ? (75 - (( mLatc   - 0.5  ) /    0.5 * 25 )) *2
ZipSwap <    1   ? (75 - (( ZipSwap - 0.5  ) /    0.5 * 25 )) *2
cpuQ    <    2   ? (75 - (( cpuQ    - 1    ) /    1   * 25 )) *2
coStop  <    2   ? (75 - (( coStop  - 1    ) /    1   * 25 )) *2
oWait   <    2   ? (75 - (( oWait   - 1    ) /    1   * 25 )) *2
Ready   <    4   ? (75 - (( Ready   - 2    ) /    2   * 25 )) *2
Balloon <    4   ? (75 - (( Balloon - 2    ) /    2   * 25 )) *2
dLatc   <   20   ? (75 - (( dLatc   - 10   ) /   10   * 25 )) *2
DiskQ   <   40   ? (75 - (( DiskQ   - 20   ) /   20   * 25 )) *2
PageO   < 2000   ? (75 - (( PageO   - 1000 ) / 1000   * 25 )) *2
```

Let's go through the first line. It checks if network TX dropped is below 0.5%, as it's definitely above 0.25%.

It then prorates the value to a number between 50% and 75%. To prorate, it applies a logic similar to the green range.

- Divide the value with the yellow range size to get the fraction.
- Multiply the result by 25 to adjust the scale to 25 points.
- Subtract the result from 75, the starting line for yellow.

The * 2 at the end is the weightage multiplier. Green is given a 1x weightage, while yellow is 2x, orange is 4x and red is 8x. The value is amplified, and then normalized again at a later portion (not shown in the screenshot).

As we're not dealing with the edge of the range, the next 2 metrics have simpler logic.

For CPU Run, the logic becomes consistent with the simpler metrics. Just need to map 80% - 90% Run value to 75% - 50% range.

```
Run < 90  ?  ( 75 - ((Run - 80)/10 * 25) ) * 2
```

For Memory Free, we have to keep on dividing by 1024.

```
mFree/1024 > 256  ?  ( (mFree/1024- 256)/256 * 25 + 50 ) * 2
```

The Orange Range and Red Range

I'm showing them together as it's an IF THEN ELSE. Whatever not caught by the orange is handled by red.

```
netTXd   <    1 ? ( 50 - (( netTXd   -   0.5) /    1 * 25 )) *4 : (25 - (min ([netTXd   -    2,   2]) /    2*25))*8)
Overlap  <    2 ? ( 50 - (( Overlap  -     1  ) /    1 * 25 )) *4 : (25 - (min ([Overlap  -    2,   2]) /    2*25))*8)
mLatc    <    2 ? ( 50 - (( mLatc    -     1  ) /    1 * 25 )) *4 : (25 - (min ([mLatc    -    2,   2]) /    2*25))*8)
ZipSwap  <    2 ? ( 50 - (( ZipSwap  -     1  ) /    1 * 25 )) *4 : (25 - (min ([ZipSwap  -    2,   2]) /    2*25))*8)
cpuQ     <    4 ? ( 50 - (( cpuQ     -     2  ) /    2 * 25 )) *4 : (25 - (min ([cpuQ     -    4,   4]) /    4*25))*8)
coStop   <    4 ? ( 50 - (( coStop   -     2  ) /    2 * 25 )) *4 : (25 - (min ([coStop   -    4,   4]) /    4*25))*8)
oWait    <    4 ? ( 50 - (( oWait    -     2  ) /    2 * 25 )) *4 : (25 - (min ([oWait    -    4,   4]) /    4*25))*8)
Ready    <    8 ? ( 50 - (( Ready    -     4  ) /    4 * 25 )) *4 : (25 - (min ([Ready    -    8,   8]) /    8*25))*8)
Balloon  <    8 ? ( 50 - (( Balloon  -     4  ) /    4 * 25 )) *4 : (25 - (min ([Balloon  -    8,   8]) /    8*25))*8)
dLatc    <   40 ? ( 50 - (( dLatc    -    20  ) /   20 * 25 )) *4 : (25 - (min ([dLatc    -   40,  40]) /   40*25))*8)
DiskQ    <   80 ? ( 50 - (( DiskQ    -    40  ) /   40 * 25 )) *4 : (25 - (min ([DiskQ    -   80,  80]) /   80*25))*8)
PageO    < 4000 ? ( 50 - (( PageO    - 2000  ) / 2000 * 25 )) *4 : (25 - (min ([PageO    - 4000,4000]) / 4000*25))*8)
```

What do you notice in the red range?

It has a min() function. This is to guard against a very large value. For example, in CPU Ready, any value above 8% will be given a score of 0%, not negative.

The formula for CPU Run becomes simpler as it cannot go above 100. We do not need to do the min() function for the red range.

```
Run < 95 ? ( 50 - ((Run - 90) /5 * 25 )) *4 : (25 - ((Run - 95) / 5 * 25)) *8)
```

The expression 50 – caps the value of orange at 50%, while the red is capped at 25.

The * 4 at the end is the weightage multiplier for orange. The * 8 at the end is the weightage multiplier for red.

Now let's look at the remaining metrics.

For Guest OS Memory Free, since the threshold is descending, the logic is simpler as the value cannot be less than 0.

```
mFree/1024 > 128  ?  ( (mFree/1024 - 128) / 128 * 25 + 25) * 4  :  mFree/1024/128 * 25 * 8)
```

vMotion

There are 3 metrics to watch. In order of importance, they are:

1. Downtime
2. Stun time
3. Copy Bandwidth

The first 2 are contention metrics. You should set alarm for high values.

The last one is a consumption metric.

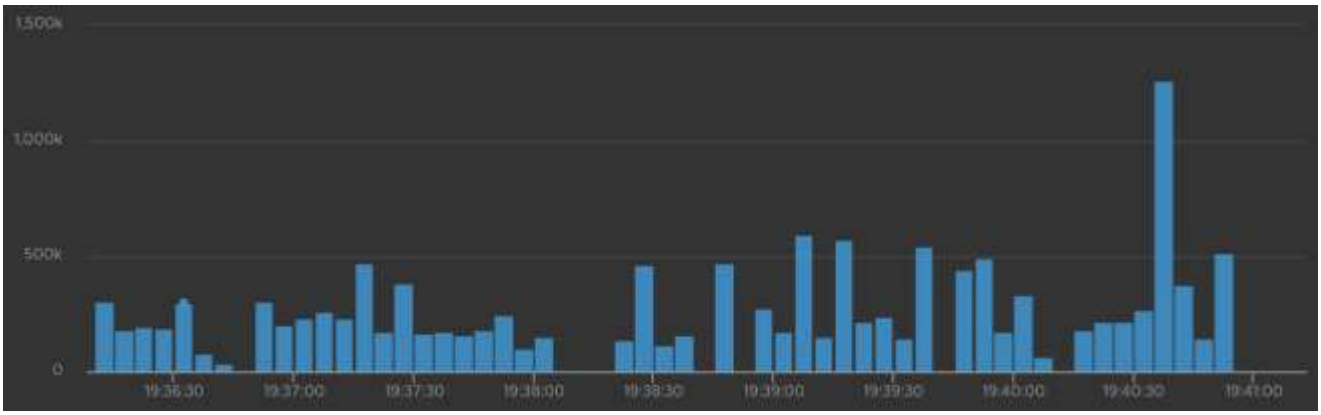
Since the values for each vMotion migration can vary, monitor both the outlier and the average, and match them to your expectation.

Downtime

Log Insight explains it well, so I will use it with some modification:

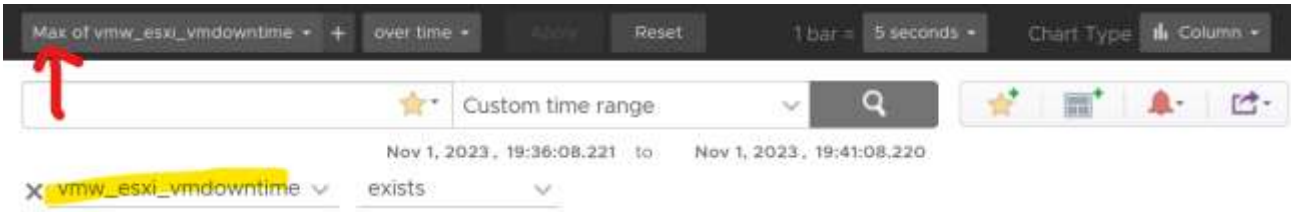
During the final phase of a vMotion operation, the VM is momentarily quiesced on the source ESXi host. This enables the last set of memory changes to be copied to the target ESXi host. After that, the VM is resumed on the target host. The guest briefly pauses processing during this step. Although the duration of this phase is generally less than 200 milliseconds, it is where the largest impact on guest performance (an abrupt, temporary increase of latency) is observed. The impact depends on a variety of factors such as network infrastructure, shared storage configuration, host hardware, vSphere version, and dynamic guest workload.

The following screenshot shows trend towards unhealthy range.



The time unit is in microseconds. The expected time is 200 milliseconds. Anything over one million microseconds (one second) is a cause for concern.

Log Insight has a metric called vmw_esxi_vmdowntime. Plot the worst value as it's a leading indicator.

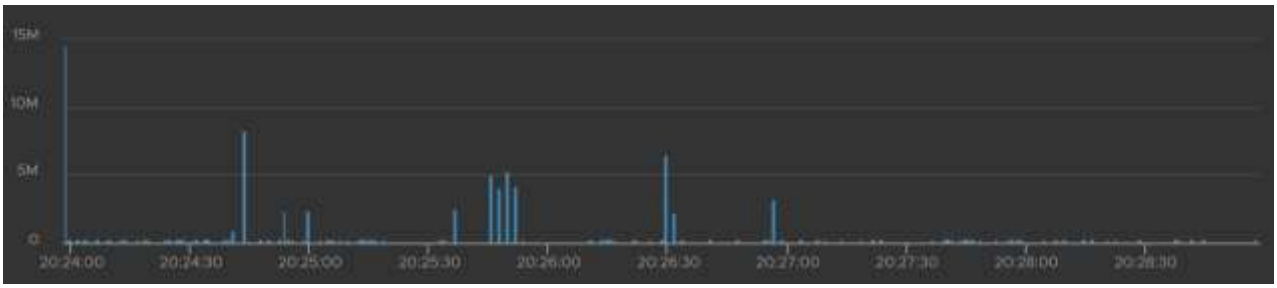


Stun Time

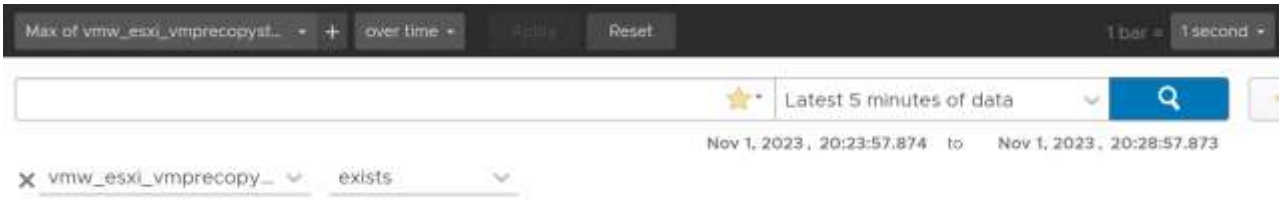
Stun Time is the period the VM gets a checkpoint stun, where no guest instruction is executed on the same vmx file. Operations that incur checkpoint are device reset, disk branching, disk promotion, snapshot take, and snapshot consolidate. A long stun time will impact Guest OS performance.

During the first phase of a vMotion operation, a snapshot is created for the VM. Snapshot means a delta VMDK is being created, which requires a stun operation. The VM then switches its write operations to the delta disk.

The time unit is in microseconds. 200 millisecond is a good threshold, and anything above 1 second should be investigated.



Log Insight has a metric called vmw_esxi_vmprecopystuntime. Plot the highest value as it's a leading indicator.



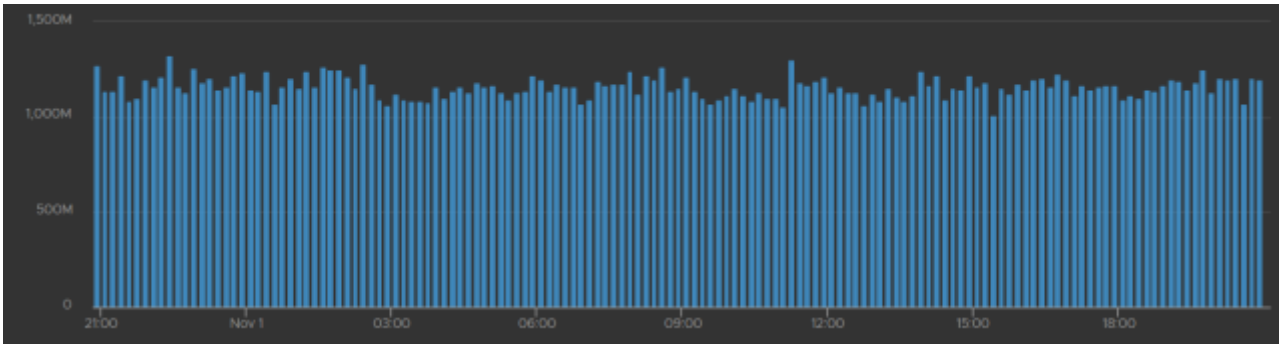
Complement the above trend chart with a table that shows the ESXi host. Note the table does not show the VM name.



Copy Bandwidth

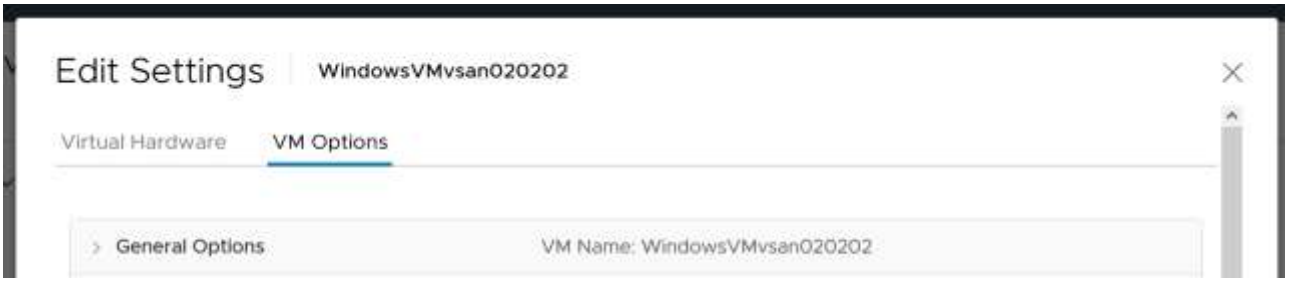
Since this is a consumption metric, ensure the values are not too low, as that will slow down vMotion progress.

The bandwidth (Gb/s) should be relatively stable and matches the assigned capacity for vMotion traffic.



Latency Sensitivity

You can reduce the latency and jitter caused by virtualization by essentially "reserving" the physical resource to a VM. In the vSphere Client UI, edit VM settings, and go to VM Options tab.



Scroll down until you see this option.



What happens to the metrics when you set Latency Sensitivity = High?

CPU Impact

The CPU "pipeline" has to be made available. In a sense, the CPU is scheduled 100% of the time. This prevents any wakeup or scheduling latencies that result of having to schedule a vCPU when it wakes up in the first place. Yes, the exclusive bit of exclusive affinity is literal.

Let's see what it looks like in esxtop. I've removed unnecessary information so it's easier to see. What do you notice?

```
     GID NAME                          %USED     %RUN     %SYS    %WAIT  %IDLE
  153670 vmx                            0.03     0.03     0.00   100.00   0.00
  153670 NetWorld-VM-2127520            0.00     0.00     0.00   100.00   0.00
```

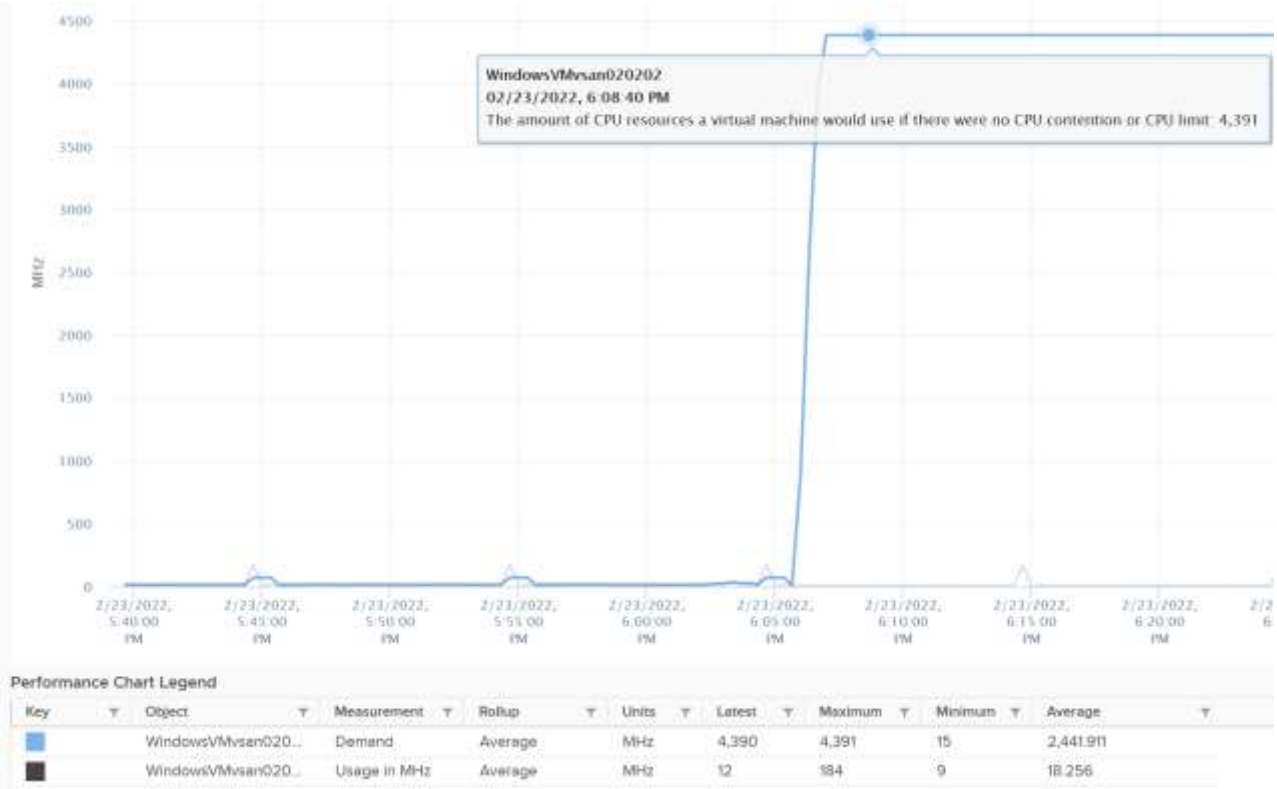
```
153670 NUMASchedRemapEpochInitialize     0.00    0.00    0.00  100.00    0.00
153670 vmast.2127520                     0.00    0.00    0.00  100.00    0.00
153670 vmx-vthread-212                   0.00    0.00    0.00  100.00    0.00
153670 vmx-filtPoll:WindowsTest          0.00    0.00    0.00  100.00    0.00
153670 vmx-mks:WindowsTest               0.00    0.00    0.00  100.00    0.00
153670 vmx-svga:WindowsTest              0.00    0.00    0.00  100.00    0.00
153670 vmx-vcpu-0:WindowsTest            0.31  100.21    0.00    0.00    0.00
153670 vmx-vcpu-1:WindowsTest            0.16  100.21    0.00    0.00    0.00
153670 vmx-vcpu-2:WindowsTest            0.15  100.21    0.00    0.00    0.00
153670 vmx-vcpu-3:WindowsTest            0.15  100.21    0.00    0.00    0.00
153670 LSI-2127520:0                     0.00    0.00    0.00  100.00    0.00
153670 vmx-vthread-212:WindowsTest       0.00    0.00    0.00  100.00    0.00
```

We can see Run shot up to 100%. This means Wait has to go down to 0%.

Strangely, Used remains low, so we can expect that Usage remains low too. This means the formula that connect Run and Used do not apply in this extreme scenario. You're basically cutting a physical core to the VM.

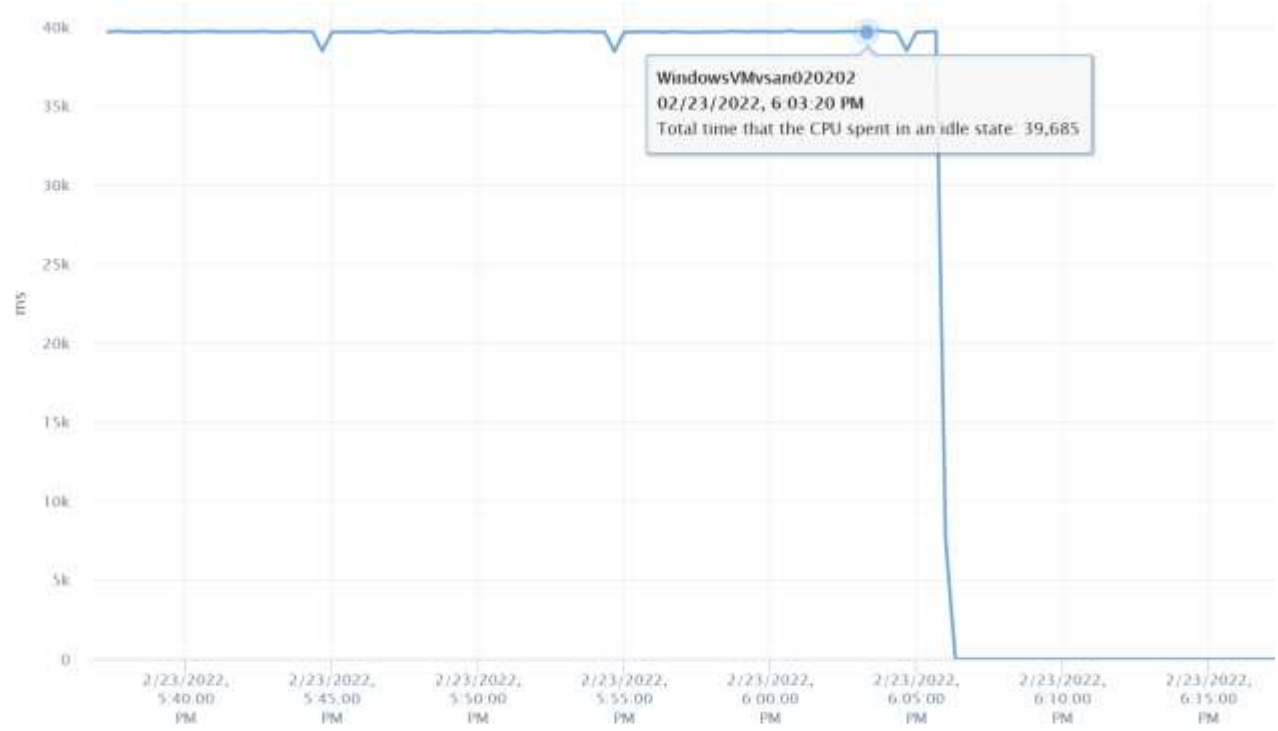
But what about Demand?

Demand shot up to 100% flat out.



So you have an interesting situation here. Demand is 100%, Usage is 0%, yet Contention is 0%.

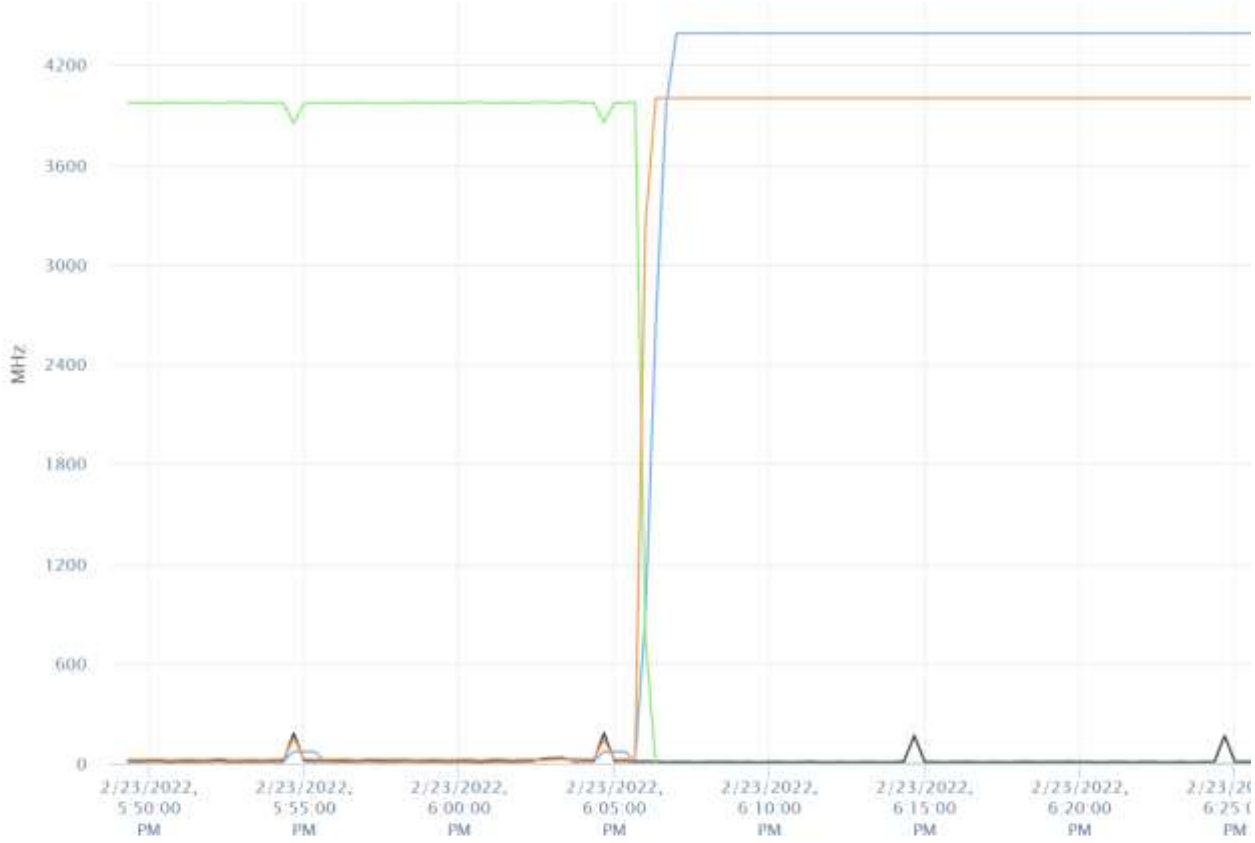
Now let's plot what happened to Wait and Idle. Notice both went from 100% to 0%.



erformance Chart Legend

Key	Object	Measurement	Rollup	Units	Latest	Maximum	Minimum	Average
	WindowsVMvsan020...	Idle	Summation	ms	0	39,774	0	18,993.805
	WindowsVMvsan020...	Wait	Summation	ms	0	39,781	0	19,001.406

So if you combine Run, Demand, Wait and Usage metrics, you can see basically Run and Demand shot up to 100% as Wait drops to 0%, while Usage is oblivious to the change.



Performance Chart Legend

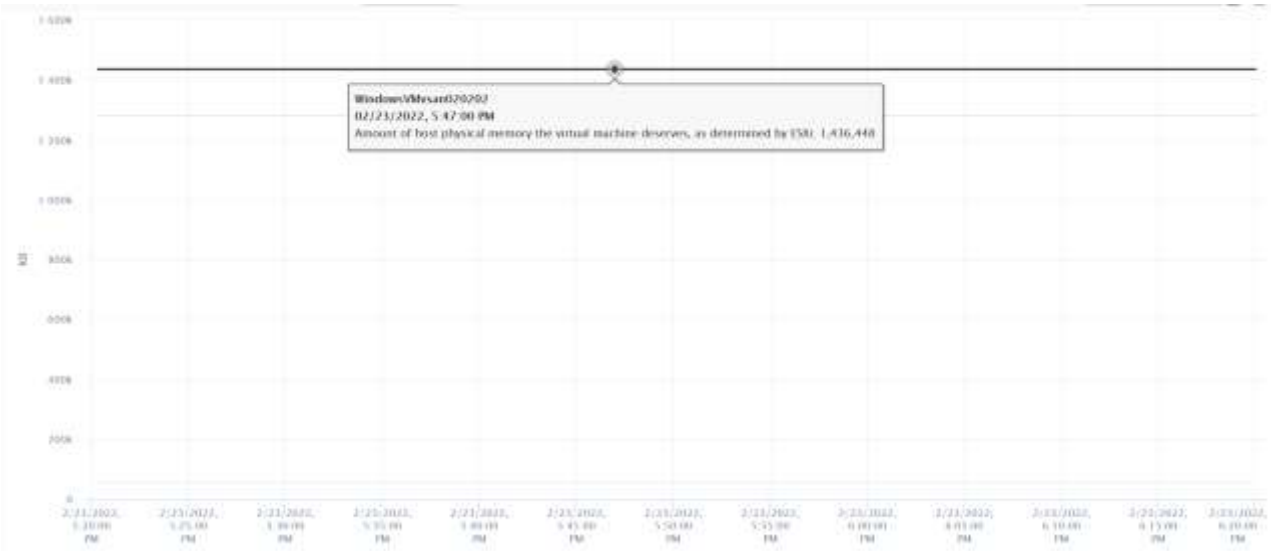
Key	Object	Measurement	Rollup	Units	Latest	Maximum	Minimum
	WindowsVMvsan020...	Demand	Average	MHz	4,390	4,391	15
	WindowsVMvsan020...	Usage in MHz	Average	MHz	11	184	9
	WindowsVMvsan020...	Wait	Summation	ms	0	39,781	0
	WindowsVMvsan020...	Run	Summation	ms	40,005	40,005	207

Just for documentation purpose, System and Ready are obviously not affected.

Key	Object	Measurement	Rollup	Units	Latest	Maximu...
	Windows...	Run	Summation	ms	40,005	40,005
	Windows...	Used	Summation	ms	75	1,650
	Windows...	System	Summation	ms	0	3
	Windows...	Ready	Summation	ms	0	35
	Windows...	Wait	Summation	ms	0	39,781
	Windows...	Idle	Summation	ms	0	39,774

Memory Impact

Memory is fundamentally storage. So I do not expect any of the counters to go up. They will go up when the VM actually needs them.



The above VM has 4 GB of RAM, fully reserved. But since it's basically idle, there is no change on the counter.

Capacity: VM

Benefits

Rightsizing is important for a VM, more so than for a physical server.

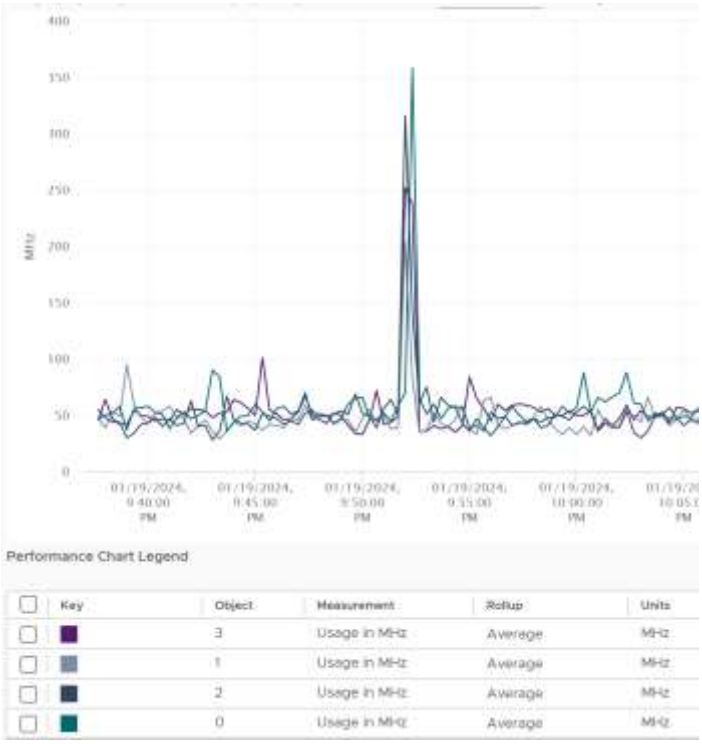
Here is an example where accuracy in right-sizing matters:

- The ESXi CPU has 2 socket, 16 cores each.
- The VM has 16 vCPU. It runs at 90%, so it fits into a single socket.
- The queue inside Linux is below 2 per vCPU for 95% of the time. This means the 16 vCPU is working hard, but able to serve all processes well.
- However, you decide to use the CPU Usage counter, which runs 25% higher due to Turbo.
- 16 vCPU x 90% x 125% = 18 vCPU
- Based on the above, you incorrectly increase the size to 18 vCPU

- The VM now spread into 2 NUMA nodes. You can have either 9 per side, or 10 vs 6. You do not like the idea of running odd numbers, plus you think it's safer to give buffer, so you bump the number to 20 vCPU.
- The VM vCPU will now be spread into 2 CPU. This means the memory will be spread too. The result is NUMA effect.

With CPU having CCD complex within a single socket, the NUMA effect does happen within a socket albeit with less penalty.

Another benefit of rightsizing is potentially faster speed from higher CPU frequency. Less vCPU means less physical threads to run. This means ESXi is able to boost the active threads by keeping unused cores. To see this, choose the CPU Usage (MHz) counter and show all the vCPU.



Lower co-stop and ready time. Even if not all vCPU is used by the application, the Guest OS will still demand all the vCPU be provided by the hypervisor.

Faster snapshot time, especially if memory snapshot is included.

Faster vMotion. Windows and Linux use memory as cache. The more it has, the more it uses, all else being equal.

Faster boot time. If a VM does not have a reservation, vSphere will create a swap file the size of the configured RAM. This can impact the boot time if the storage subsystem is slow.

Guest OS Sizing

	CPU Metrics	Memory Metrics
Guest OS	**Run Queue** Not enough CPU in Linux/Windows. Processes are queued. Need to account for this.	**In Used Page** This is the primary consumption.
	Context Switch Exclude as it can't be translated into vCPU.	**Ballooned** The memory was taken by Tools.
		Hard Page Fault Unsure if we should include them.
		Cache Page Exclude as the nature of memory is cache
VM	**Run – Overlap** We use VM as Guest OS unaware of frozen time, which result in over-reporting.	**Swapped + Zipped** Exclude as it's transparent to Guest OS
	Ready + Co Stop + Swap Wait + Other Wait Include as it would have been a consumption has there been no contention.	
	Hyper Threading | Turbo Boost Exclude as they are performance, not capacity.	
	System Not included as it's transparent to Guest OS.	

What rules to follow then sizing the Guest OS?

CPU Sizing

What metrics should be excluded when sizing Guest OS? What metrics should be included?

Having the correct inputs increase the accuracy of the prediction.

Exclusion

Hyper-Threading	The Guest OS is unaware of HT. Windows/Linux is still running, regardless of speed and throughput. When Windows/Linux vCPU happens to run on a thread that is sharing a core with another thread, the OS will simply run with lower efficiency. It experiences 37.5% drops in computing power. For example, instead of running on a 3 GHz, it feels like it's running on 1.875 GHz chip. The VM CPU Demand and VM CPU Usage metrics are not suitable as their values are affected by CPU Frequency and HT.
CPU Frequency	Same reason as above. The only exception here is the initial sizing, when the VM is not yet created. The application team may request 32 vCPU at 3 GHz. If what you have is 2 GHz, you need to provide more vCPU.

CPU idle time	Guest OS CPU will be idle for a while when waiting for ESXi to execute IO. However, while making the IO subsystem faster will result in higher CPU utilization, that's a separate scope.
CPU Context Switch	3 reasons: • There is no translation into CPU size. • It is not something you can control. • A high context switch could be caused by too many vCPU or IO. Guest OS is simply balancing among its vCPUs.
Hypervisor overhead	Reason is they are not used by the Guest OS. MKS, VMX, System. While it's part of Demand, it's not a demand coming from within the Guest. The VM CPU Used, Demand, Usage counter include system time at VM level, hence they are not appropriate.

Inclusion

Co-stop & Ready	The Guest OS actually wants to run. Had there been no blockage, the CPU would have been utilized. Adding/reducing CPU does not change the value of these waits, as this represents a bottleneck somewhere else. However, it does say that this is what the CPU needs, and we need to reflect that. We need not consider CPU Limit as it's already accounted for. Guest OS number will be inaccurate because there is "no data", due to its time being frozen.
Other Wait	Guest OS becomes idle as CPU is waiting for RAM or IO (disk or network). So this is the same case with Ready and Co-stop.
Swap Wait	
Overlap	The Guest OS actually wants to run, but it's interrupted by the kernel. Note that this is already a part of CPU Run, so mathematically is not required if we use CPU Run counter.
Guest OS CPU Run Queue	This is the primary counter tracking if Windows or Linux is unable to cope with the demand.

Formula

Based on all the above, the formula to size the Guest OS is:

```
Guest OS CPU Needed = Configured vCPU – Idle + CPU Run Queue factor
```

The result is in the number of vCPU. It is not in % or GHz. We are sizing the Guest OS, not the VM.

We're including all the time the vCPU cannot run (Ready, Costop, Swap Wait, Other Wait) as the Guest OS would have wanted to run.

Guest OS CPU Run Queue metric needs some conversion before it can be used. Let's take an example:

- VM has 8 vCPU.
- CPU Run Queue = 28 for the entire VM.
- Using Run Queue = 3 as the threshold per vCPU, the VM has 24 queues before we add vCPU.

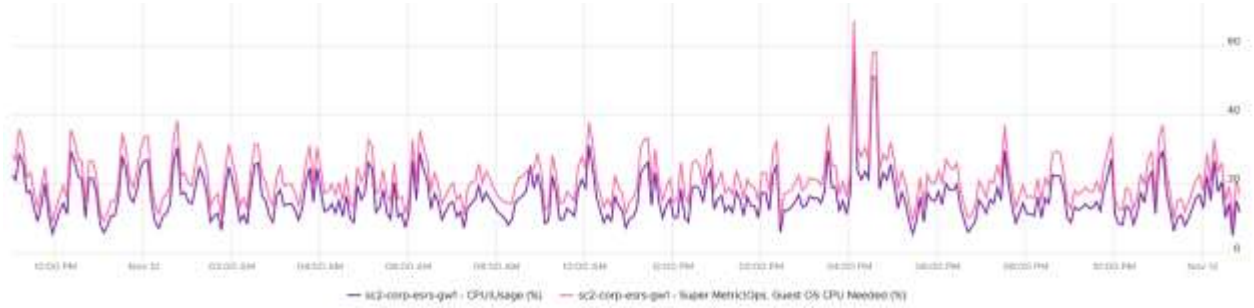
- There is a shortage of 28 – 24 = 4 queues.
- Each additional vCPU can handle 1 process + 3 queues.
- Conclusion: we add 1 vCPU.

Compared with CPU Usage, Guest OS Needed without the CPU run queue factor tends to be within 10% difference. Usage is higher as it includes system time, and turbo boost. Usage would be lower in HT and CPU frequency clocked down case.

Here is an example where Usage is higher.



Here is an example where Usage is lower.



Once we know what the Guest OS needs, we can then calculate the recommended size. This is a projection, taking lots of value. Ideally, the recommendation is NUMA aware. It is applied *after* the sizing is determined. You size, then adjust to account for NUMA. This adjustment depends on the ESXi Host. So it can vary from cluster to cluster, if your vSphere clusters are not identical.

```
Guest OS Recommended Size (vCPU) = round up NUMA (projection (Guest OS Needed (vCPU))
```

For basic NUMA compliant, use 1 socket many cores until you exceed the socket boundary. That means you use 2 vCore 1 vSocket instead of 2 vSockets with 1 vCore each.

With the release of Windows 2008, switching the Hardware Abstraction Layer (HAL) was handled automatically by the OS, and with the release of 64-bit Windows, there is no concept of a separate HAL for uniprocessor and multi-processor machines. That means one vCPU is a valid configuration and you shouldn't be making two vCPU as the minimum.

You should use the smallest NUMA node size across the entire cluster, if you have mixed ESXi with different NUMA node sizes in the cluster. For example, a 12-vCPU VM should be 2 socket x 6 cores and not 1 socket x 12 core as that fits better on both the dual socket 10 core and dual socket 12 core hosts. Take note that the amount of memory on the host and VM could change that recommendation, so this recommendation assumes memory is not a limiting factor in your scenario.

Notice the number is in vCPU, not GHz, not %. Reason is the adjustment is done at a whole vCPU. In fact in most case, it should be an even number, as odd numbers don't work in NUMA when you cross the size of a CPU socket.

Note that when you change the VM configuration, application setting may need to change. This is especially on applications that manage its own memory (e.g. database and JVM), and schedule fixed number of threads.

You can enable Hot Add on VM, but take note of impact on NUMA.

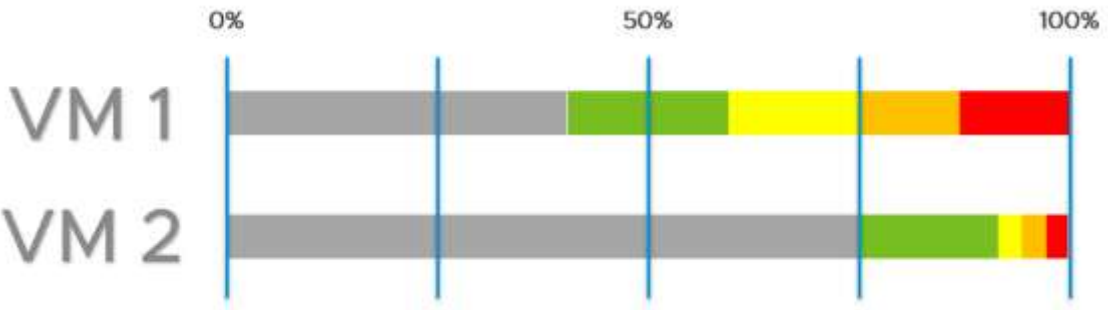
Reference: rightsizing by Brandon Gordon.

Reclaimable CPU

Memory Sizing

Accuracy of Guest OS memory has been a debate for a long time in virtualization world. Take a look at the following utilization diagram. It has two bars, showing memory utilization of Windows/Linux. They use different set of thresholds.

Which one should you use for memory?



My recommendation is no 2.

The reason is memory is a form of cache. It stays even though it's not actively used.

When you spend your money on infrastructure, you want to maximize its use, ideally at 100%. After all, you pay for the whole box. In the case of memory, it even makes sense to use the whole hardware as the very purpose of memory is just a cache for disk.

The green range is where you want the utilization to fall. Below the green threshold lies a grey zone, symbolizing wastage. The company is wasting money if the utilization falls below 50%. So what lies beneath the green zone is not an even greener zone; it is a wastage zone. On the other hand, higher than 75% opens the risk that performance may be impacted. Hence I put a yellow, orange and red threshold. The green zone is actually a relatively **narrow** band.

In general, applications tend to work on a portion of its Working Set at any given time. The process is not touching all its memory all the time. As a result, the rest becomes cache. This is why it's fine to have active + cache beyond 95%. If your ESXi is showing 99%, do not panic. In fact, ESXi will wait until it touches 99.1% before it triggers ballooning process. Windows and Linux are doing this too. The modern-day OS is doing its job caching all those pages for you. So you want to keep the Free pages low.

Include cache	Guest OS uses RAM as cache. If you size the OS based on what it actually uses, it will have neither cache nor free memory. It will start paging out to make room for Cache and Free, which can cause performance problems. As a result, the name of this proposed counter should not be called Demand as it contains more than unmet demand. It is what the OS needs to operate without heavy paging.

	Hence the counter name to use is Needed Memory, not Memory Demand. The challenge here is how much cache do you want to include?
Exclude page file	Including the pagefile will result in sizing that is too conservative as Windows and Linux already has cache even in their In Use counter. Guest OS uses virtual RAM and physical RAM together. They page-in proactively, prefetching pages when there is no real demand due to memory mapped files. This makes determining unmet demand impossible. A page vault does not distinguish between real need versus proactive need.
Exclude balloon	It results in more usage inside the Guest OS, if it comes from the free page.
Don't fallback to VM metric	Since we are sizing the Guest OS, we use Guest OS only. No falling back to VM as it's inaccurate.
Exclude latency	RAM contention measures latency, hence not applicable. We're measuring the disk space, not latency. Space, not Speed. Utilization, not Performance.

Unlike CPU, there are more difference between Windows and Linux when it comes to memory.

For VCF Operations specific implementation, review this post by Brandon Gordon.

VM Footprint

What rules to follow then sizing the footprint of the VM to the underlying SDDC?

CPU Sizing

Include Hyper-Threading	When a VM runs on a thread that has a peer thread running, it's getting less CPU cycle.
Include CPU Frequency	It impacts the footprint. For example, moving a VM to cluster with lower frequency may require more vCPU
Include contention	The VM actually wants to run, but blocked by hypervisor. This means Overlap needs to be added as Used does not include it.
Include VM overhead	They are not insignificant in cases such as Fault Tolerant.
Exclude Guest OS queue	It's transparent to the VM

Based on all the above, the formula to size the VM is:

```
VM CPU Needed = (Used + Overlap + Co-stop + Ready + Other Wait) + System / 20000
```

You express in GHz (utilization model) and vCPU (allocation model).

The GHz is especially important when you need to migrate into another ESXi with different clock speed. To convert into GHz, we multiply the number by the nominal, static clock speed.

Enhance the sizing by considering CPU generation and speed. Take note this can introduce a new problem if not done properly. An application may perform poorly after the reduction in vCPU if it works better with many slow threads vs a few fast threads.

Memory Sizing

Since the goal is to calculate the total footprint, you need to include all the pages associated to the VM.

```
VM Memory Needed = Consumed + Overhead + Swapped + Compressed + Ballooned
```

The effect of Transparent Page Sharing should be included as that likely persist when you vMotion the VM. The challenge is it's not possible to separate intra-VM sharing and inter-VM sharing.

Memory contention is not included as that measures speed, not space.

Provider Objects

Part 3 Chapter 7

In a private cloud architecture, the provider layer is the shared infrastructure. It consists of objects such as vSphere Cluster, ESXi host, datastore, distributed virtual switch, and vSAN.

Overview

A cluster is operationally a collection of ESXi hosts. As a result, the basic counters of CPU, memory, disk and network are basically the sum of the member host.

vSphere Cluster

Think of vSphere Cluster as the smallest logical building block. From operations management, it's basically a single computer. Think of a huge and complex machine.

vSphere Cluster is much more than just a group of ESXi hosts sharing a common network and storage. What makes a cluster more complex than the sum of its hosts is the various cluster-level features and configuration.

Let's start by looking its 2 most basic features:

Features	Impact	Impacts
HA	Capacity	The various options of HA complicate usable capacity calculation.
	Availability	HA results in 2 metrics: actual availability and operational availability. HA event requires VM availability to be verified as application dependency could be affected. The order of booting needs to be kept up to date. HA event needs to be reported and investigated. This typically requires log analysis to find the root cause.
	Configuration	ESXi hosts in the cluster should have identical hardware & software configuration. Customers typically have multiple clusters, and need them to be consistently configured.
	Inventory	Actual needs to match plan. Not only the amount, but also the movement and their status.
DRS	Performance	Degradation of vMotion stunned time could give a clue to overall ESXi performance. vMotion may impact latency-sensitive application. Rate of vMotion should be measured against expectation.
	Configuration	Various DRS settings such as automation level should match plan and standard. VM-level exception can get buried in large environment. Customers typically have multiple clusters, and need them to be consistently configured.

You can see that the above complicate operations, especially in a very large environment with hundreds of clusters. If you add these features on top, you further increase complexity of your operations.

Features	Impact	Impacts
Affinity	Configuration	The settings of affinity and anti-affinity should match plan. In large environment with hundreds of clusters this can get buried and hence overlooked.
Resource Pool	Capacity	Shares, Limit, Reservation done at resource pool level need to be compatible with those at its children VM. Resource Pool should not be peer of VM.
	Performance	
	Configuration	Complication from cascading resource pools. Need to ensure VMs are not siblings of resource pool
DPM	Capacity	DPM impacts capacity as it changes total capacity.
	Performance	DPM is only considering the ESXi utilization metrics. It does not check the VM contention metric.
	Configuration	DPM settings need to match plan.

The above cover the standard vSphere cluster. There are 2 other variants, which take the operational complexity higher.

Features	Impact	Impacts
Stretched Cluster	Configuration	The configuration of each site needs to be checked so VMs always accessed local storage
	Capacity	The utilization of the 2 physical sites may be intentionally unbalanced, because one acts as primary site while the other as DR site.
	Performance	Horse-shoe traffic between VMs on the same site. Traffic ping pong between VMs on different sites.
	Availability	The whole purpose of a stretched cluster is they protect one another. This shall be tested at least once a year.
vSAN Cluster		vSAN impacts all aspects of operations management. It impacts Day 0, Day 1, and Day 2.

In addition, there are complication simply because there are multiple members in the cluster. For example, is cluster utilization simply the average of all its hosts? What if there is imbalanced? It will get buried if the cluster has many hosts.

While a cluster focuses on compute, it is where VM runs and consumes network and storage. This means network and storage counters must be considered as appropriate. If you're using vSAN, then it's mandatory.

Base Metrics

vSphere Client only displays basic set of metrics. They are grouped into 4, as shown in the following screenshot:



Chart Metrics
CPU
Cluster services
Memory
Virtual machine operations

Select counters for this chart:

Counters	Rollups	Units	Internal Name	Stat Type
Total	Average	MHz	totalmhz	Rate
Usage	Average	%	usage	Rate
Usage in MHz	Average	MHz	usagemhz	Rate

For each of the group, there is basic set of metrics. Here it is for memory:

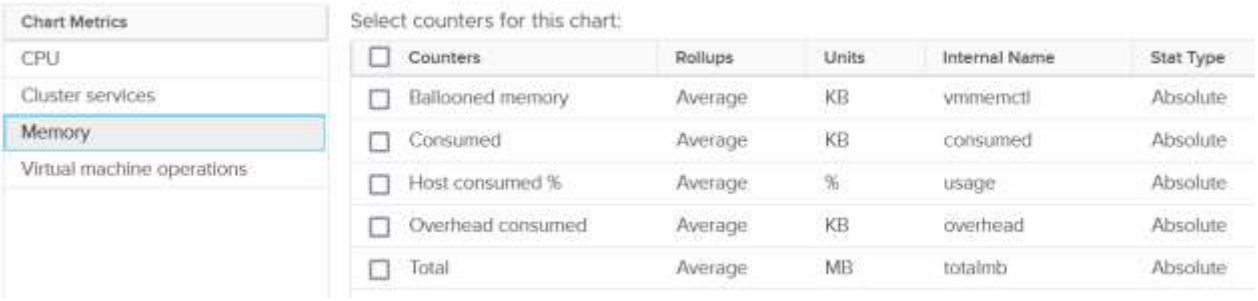


Chart Metrics
CPU
Cluster services
Memory
Virtual machine operations

Select counters for this chart:

Counters	Rollups	Units	Internal Name	Stat Type
Ballooned memory	Average	KB	vmmemctl	Absolute
Consumed	Average	KB	consumed	Absolute
Host consumed %	Average	%	usage	Absolute
Overhead consumed	Average	KB	overhead	Absolute
Total	Average	MB	totalmb	Absolute

The group Cluster Services only provides 3 metrics:



Select counters for this chart:

Counters	Rollups	Units	Stat Type	Description
Current failover level	Latest	num	Absolute	vSphere HA number of failures that can be tolerated
Effective CPU resources	Average	MHz	Rate	Total available CPU resources of all hosts within a cluster
Effective memory resources	Average	MB	Absolute	Total amount of machine memory of all hosts in the cluster that is available for us

VM Operations

vSphere Cluster, being the main object where VM runs, has a set of event metrics. They count the number of times an event, such as a VM gets deleted, happens. This provides insight into the dynamics of the environment.

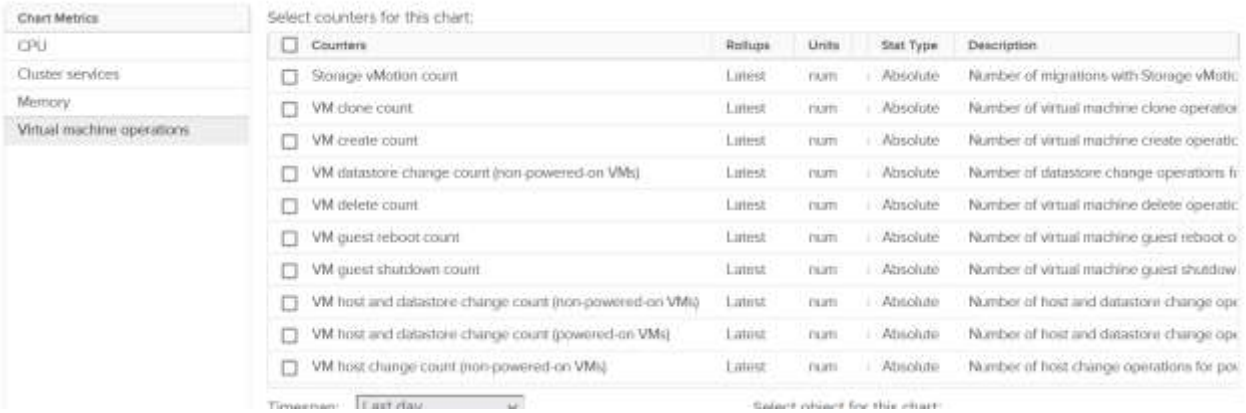


Chart Metrics
CPU
Cluster services
Memory
Virtual machine operations

Select counters for this chart:

Counters	Rollups	Units	Stat Type	Description
Storage vMotion count	Latest	num	Absolute	Number of migrations with Storage vMotic
VM clone count	Latest	num	Absolute	Number of virtual machine clone operatio
VM create count	Latest	num	Absolute	Number of virtual machine create operatic
VM datastore change count (non-powered-on VMs)	Latest	num	Absolute	Number of datastore change operations fi
VM delete count	Latest	num	Absolute	Number of virtual machine delete operatic
VM guest reboot count	Latest	num	Absolute	Number of virtual machine guest reboot o
VM guest shutdown count	Latest	num	Absolute	Number of virtual machine guest shutdow
VM host and datastore change count (non-powered-on VMs)	Latest	num	Absolute	Number of host and datastore change ope
VM host and datastore change count (powered-on VMs)	Latest	num	Absolute	Number of host and datastore change ope
VM host change count (non-powered-on VMs)	Latest	num	Absolute	Number of host change operations for pov

Take note that the metric is accumulative. So it starts since the day the cluster was created. VCF Operations converts into rate, and also make them available at higher level objects (Data Center, vCenter and vSphere World).

Category	Metric Name	Description
Change of State	VM guest reboot count	Only a reboot. The underlying VM is not powered off.
	VM guest shutdown count	I think this triggers VM Power Off too.
	VM standby guest count	My guess this also power off the VM
	VM power off count	I think this is direct, abrupt power off. It does not include proper shut down from Guest OS.
	VM power on count	
	VM reset count	Power cycle, different to Guest OS restart as the VM is momentarily powered off.
	VM suspend count	Deeper than Guest OS Standby. Is this like hibernate in Windows?
Change of Inventory	VM create count	All creation, be it from template, direct, or cloning. So this is the total amount.
	VM clone count	Creation via cloning only.
	VM template deploy count	Counted separately to separate those VMs not deployed from template.
	VM reconfigure count	Log Insight tracks the actual changes.
	VM register count	Add into vSphere inventory
	VM unregister count	Take note the VM file can still exist in datastore and LUN
	VM delete count	All deletion, be it API or UI.
Change of Location	vMotion count	Change of ESXi host only
	Storage Motion count	Change of datastore only.
	VM host and datastore change count	Both change in one event. Powered-on VMs only
	VM datastore change count	Only for powered-off VMs
	VM host and datastore change count	Only for powered-off VMs
	VM host change count	Only for powered-off VMs

You certainly have some expectation on the dynamics of your environment. Does the reality match your expectation?

In production environment, these numbers should be low. Some numbers such as shutdown should also match the change request and happens during the green zone. Some exceptions apply, such as your VDI design includes scheduled reboot on the weekend.

Imbalance

Imbalance is the root cause behind why contention happens despite overall utilization has not reached 100%.

There are 3 types of imbalances in a vSphere Cluster.

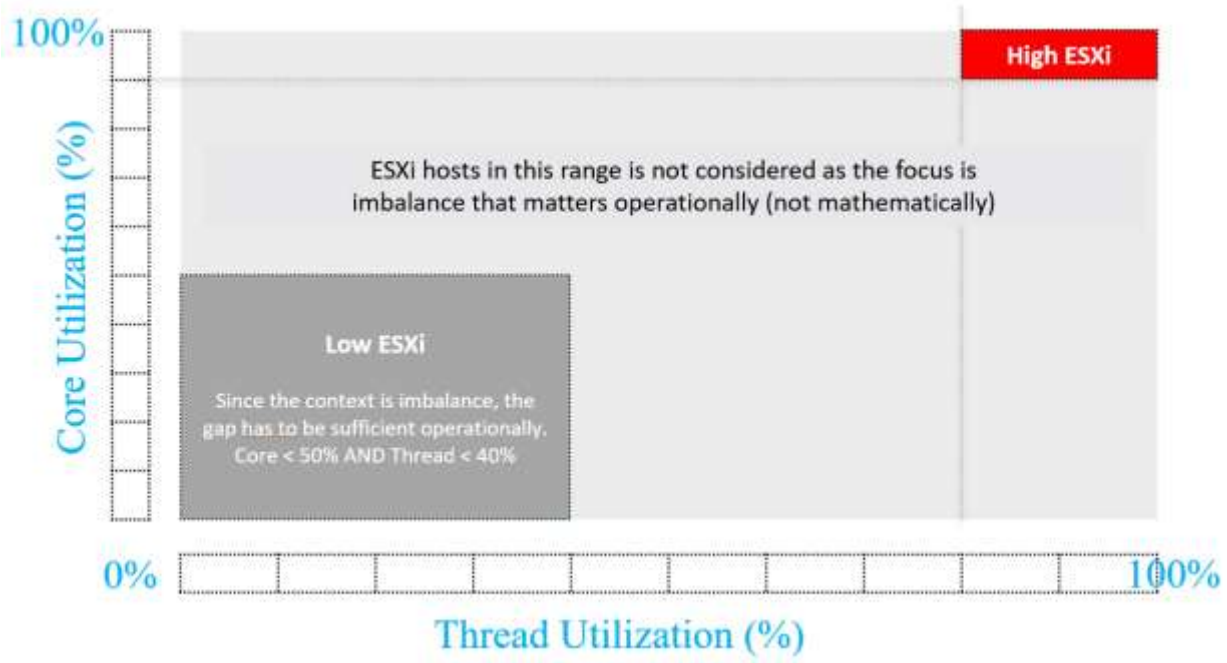
Type of Imbalance	Analysis
Utilization imbalance	There are 2 types here: • Inter Host imbalance • Intra Host imbalance
Overcommit imbalance	Running VM vCPU : ESXi Usable Thread. A high overcommit by itself is fine if the utilization is now. CPU is riskier than memory due to more volatile nature.
Monster VM imbalance	Concentration of monster VMs in a host increase risk of contention.
	The threshold of the monster VM depends on 2 factors: • CPU NUMA architecture. • Hypervisor overhead. The more services you run, the bigger the overhead. If a VM vCPU > 45% of the ESXi count of threads, 2 of these unlikely can run at the same time as vSAN + NSX + VMKernel needs CPU too.

Inter-Host Imbalance

You want clusters to be well utilized but balanced across all hosts in the cluster.

- High Utilisation → Performance risk.
- Low Utilization → Wastage capacity.

The cluster is imbalanced when there are High ESXi hosts and Low ESXi hosts.

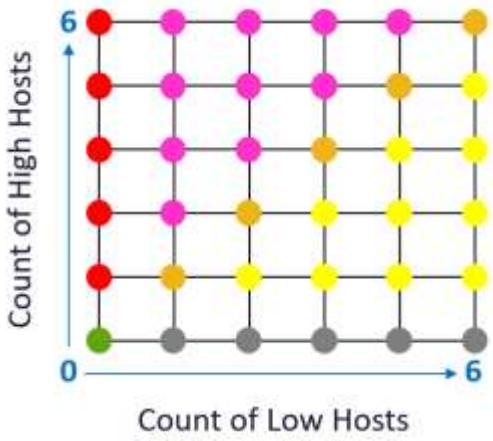


Focus is imbalance at high utilization. Imbalance can happen at low utilization. While that's mathematically true, it's not operationally important.

High ESXi	An ESXi utilization is considered high from day-to-day operations when it reaches this: ESXi Core Util > 90% AND Thread Util > 80%.
	For Core, use 90% and not 100% as the threshold. 100% could be too late if it's sustained at 100% for 300 seconds. NUMA prevents all cores from being used evenly. Some cores are struggling while others are idle
	For Thread, use 80% and not something lower. If we use something lower, we end up including situation where there are still many idle threads. That's not a high utilization issue
Low ESXi	The definition is: Core Utilization < 50% AND Thread Utilzation < 40%

The formula is basically High Hosts / Low Hosts.

Let's visualize using a cluster of 6 hosts as an example:

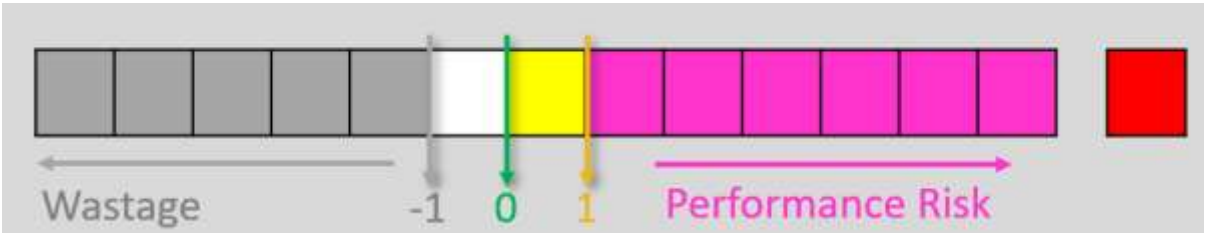


Color	Meaning	
Red	Worst	1 or more High Host, but no Low Host to migrate VM to. May need to move to another cluster.
Pink	Bad	There are more High Hosts than Low Host. There may not be enough capacity to balance
Orange	Bad	The number of High Hosts = Low Hosts. Hopefully, there is enough capacity to balance
Yellow	Warning	The number of High Hosts < Low Hosts. Hopefully, there is more than enough capacity to balance
Green	Good	Ideal. This is what you want. A utilized cluster yet balanced.
Grey	Wastage	1 or more Low Host, but not high host. Cluster is under utilized.

Quantification Strategy

The formula uses a relative number, so we can compare across clusters of different sizes.

Looking at the values, we can see that each color falls within some specific ranges, that is intentionally designed not to overlap.



We can now translate the color into numbers.

Color	Possible Values	
Red	Worst	100, 200, 300, 400, etc. Value is bumped up to avoid overlapping with pink, which can go
Pink	Bad	1 < Value < 63 Not a whole number. It can be decimal.
Orange	Bad	1.
Yellow	Warning	0 < Value < 1
Green	Good	0
Grey	Wastage	-1, -2, -3, …., 64 It's a whole number, not decimal

Limitation

Summarizing always results in the loss of details.

For example, we don't know if it's 1 High Host over 2 Low Host, or 4 hosts over 8 hosts as both returns 0.5.

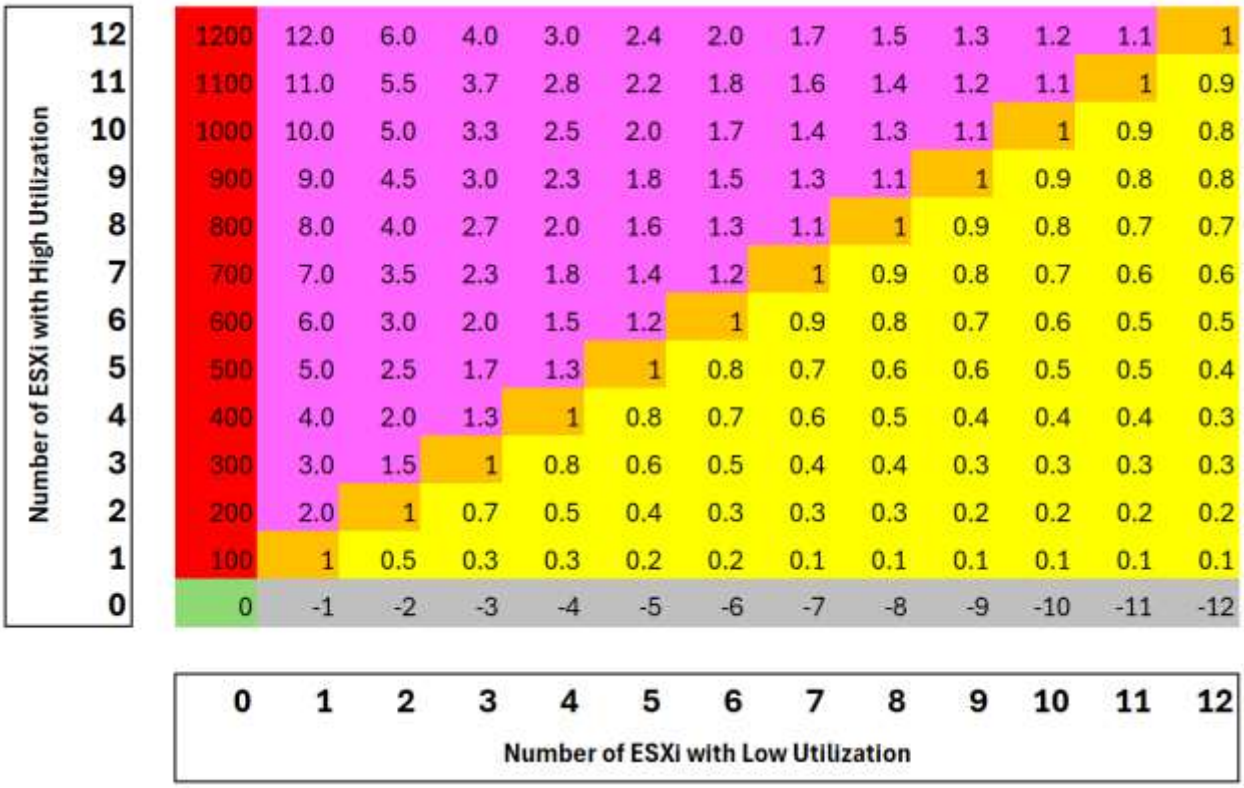
Alternatively, we can concanetate the numbers. So we get 1.2 and 4.8.

- Advantage: we know 1.2 is different to 4.8
- Disadvantage: the sort across many numbers will be less meaningful as the values are no longer relative.

Example

Let's use a 12-node cluster as an example.

The following shows the complete permutation of the 169 possible Cluster Imbalance values.



Number of ESXi with High Utilization													
12	1200	12.0	6.0	4.0	3.0	2.4	2.0	1.7	1.5	1.3	1.2	1.1	1
11	1100	11.0	5.5	3.7	2.8	2.2	1.8	1.6	1.4	1.2	1.1	1	0.9
10	1000	10.0	5.0	3.3	2.5	2.0	1.7	1.4	1.3	1.1	1	0.9	0.8
9	900	9.0	4.5	3.0	2.3	1.8	1.5	1.3	1.1	1	0.9	0.8	0.8
8	800	8.0	4.0	2.7	2.0	1.6	1.3	1.1	1	0.9	0.8	0.7	0.7
7	700	7.0	3.5	2.3	1.8	1.4	1.2	1	0.9	0.8	0.7	0.6	0.6
6	600	6.0	3.0	2.0	1.5	1.2	1	0.9	0.8	0.7	0.6	0.5	0.5
5	500	5.0	2.5	1.7	1.3	1	0.8	0.7	0.6	0.6	0.5	0.5	0.4
4	400	4.0	2.0	1.3	1	0.8	0.7	0.6	0.5	0.4	0.4	0.4	0.3
3	300	3.0	1.5	1	0.8	0.6	0.5	0.4	0.4	0.3	0.3	0.3	0.3
2	200	2.0	1	0.7	0.5	0.4	0.3	0.3	0.3	0.2	0.2	0.2	0.2
1	100	1	0.5	0.3	0.3	0.2	0.2	0.1	0.1	0.1	0.1	0.1	0.1
0	0	-1	-2	-3	-4	-5	-6	-7	-8	-9	-10	-11	-12
	0	1	2	3	4	5	6	7	8	9	10	11	12

Number of ESXi with Low Utilization

Performance

Operationally, you manage at cluster level, not at Resource Pool, ESXi host, or data center level. It's the sweet spot for starting your monitoring and troubleshooting. As usual, we start with the contention metric, followed by the utilization metric.

By definition, the metrics are *average* numbers. So be careful as there can be VM that has issue but obscured in the cluster wide average. Even the so-called total or summation is mathematically an average. For example, the Total CPU Wait counter is the sum of all ESXi CPU Wait metrics, which in turn is the sum of all the VMs. At the end you get a large number, which you need to normalize and convert into average. Since you divide it against the cluster total, you get average.

Contention vs Consumption

There is a common misconception that you cannot have performance issue when cluster has low utilization. We introduced that problem in Part 1 Chapter 1 of the book.

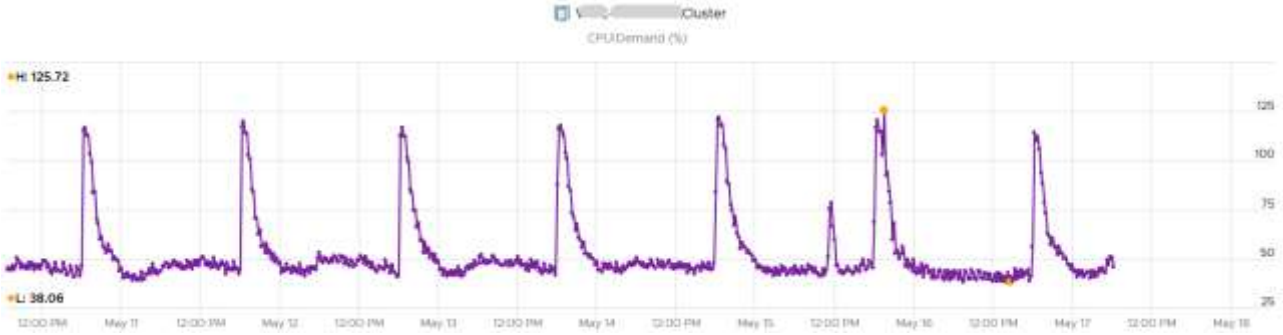
Is there corelation between cluster utilization and cluster contention?

I'll show 2 opposite examples.

Example showing Correlation

If every VM is given the same treatment by the cluster, then yes.

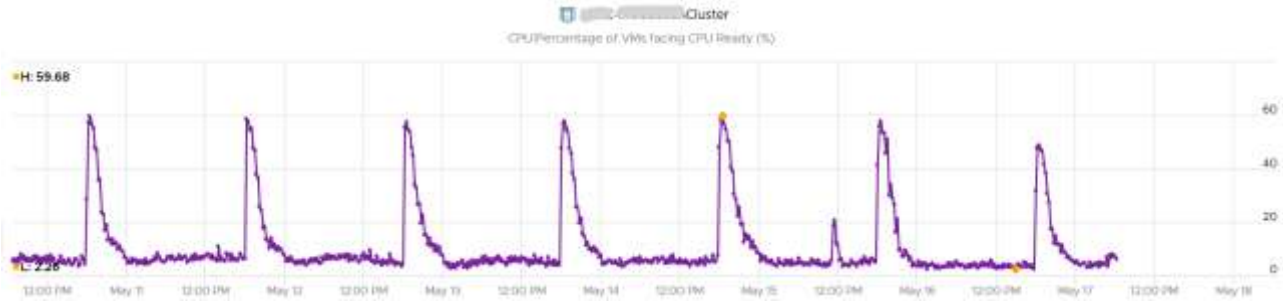
Here is a cluster experiencing regular high utilization in the last 7 days. You can clearly see the peak. The cluster has 14 ESXi Hosts.



A logical question here would be what's the impact on VM performance? Are they getting the CPU they asked? The cluster has 550 running VM.

This is where the contention metrics come in. One tracks the depth of the problem, the other the breadth of the problem.

The counter Percentage of VMs facing CPU Ready > 1% shows a nearly identical pattern. We can see that a big percentage of the VM population is affected.



The second counter tracks the depth, giving the absolute worst CPU Ready value experienced by any VM in the cluster.

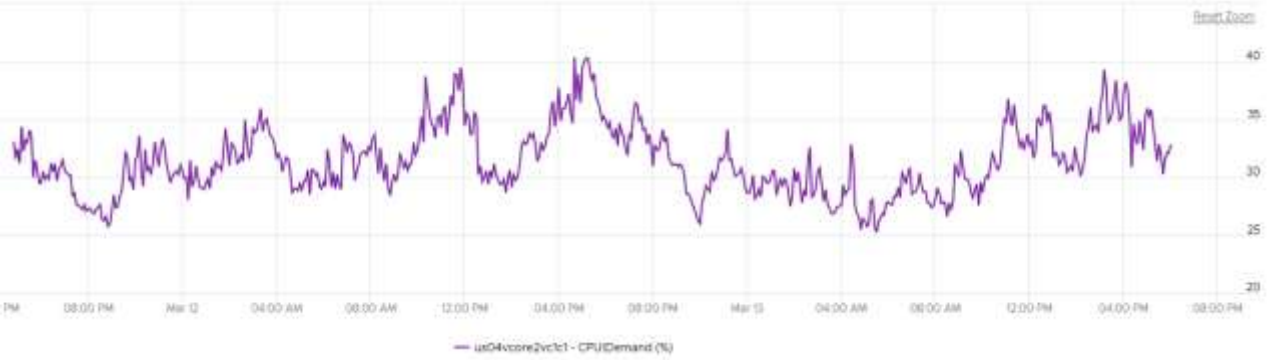


Example showing no Correlation

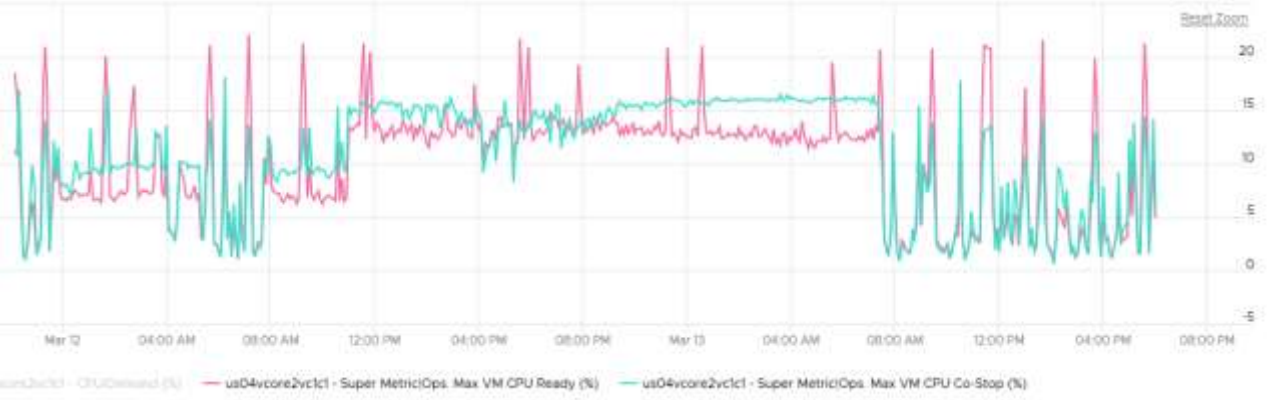
Performance is unmet demand. VM 007 can face very high contention when all other VMs on the same cluster face no contention.

It is possible for VMs in the cluster to suffer from poor performance, while the cluster utilization is low. One main reason is cluster utilization looks at the provider layer (ESXi), while performance looks at individual consumer (VM).

The following cluster has 32 ESXi supporting 2357 VM. The average demand across the cluster is <40%. Since it has 32 ESXi and 2357 VM, we can retire 8 ESXi or add 1K VM.



And yet the VMs in the clusters are facing contention. Both VM CPU Ready and CPU Co-stop are high.

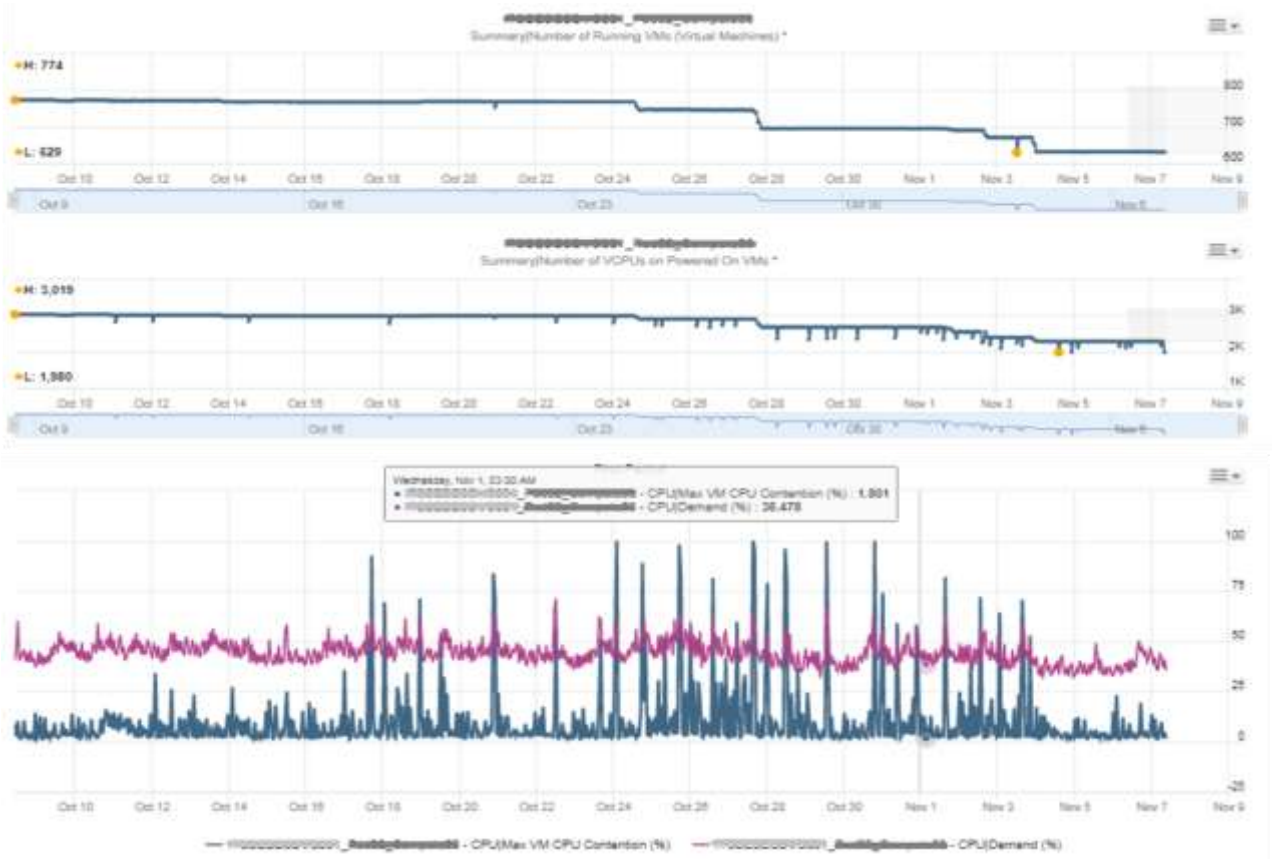


Let me take another example, where you can see the corelation between cluster utilization and VM contention in the cluster. My apology that the picture is not sharp. You can see the cluster has 774 running VM at the start. One month later it has dropped to 629, a drop of 145 VM or 19%. The second line chart reveals the number of running vCPU dropped from 3019 to 1980, a whopping 1039 vCPU or 34%. That indicates the big VMs were moved out.

This cluster was running mission critical VMs. What's going on?! What caused the mass evacuation.

Notice the mass evacuation happened multiple times, so it's not accidental.

Looking at the last chart. It has 2 line. Maroon showing utilization, blue showing contention. Can you figure out what happened?



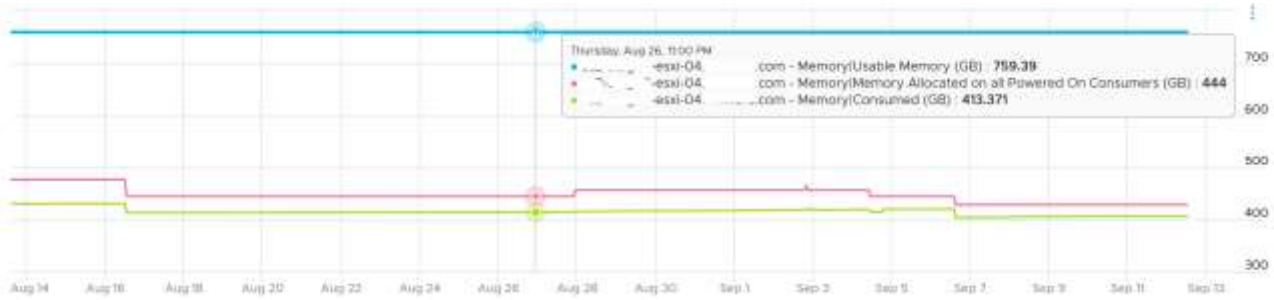
The cluster utilization was hovering around 50%. In that entire month, it barely moved. This cluster was probably 16 nodes, so 50% utilization means you can easily take out a few ESXi hosts actually.

The Max VM CPU Contention told a different story. Notice it spiked well above 75%. That impacted at least 1 VM. There were multiple spikes, leading to multiple complaints, and eventually infrastructure was forced to evacuate the cluster to fix the performance problem. Notice the counter dropped gradually in November, despite utilization remains fairly stable.

Example for Memory

We covered 2 examples for CPU. What about memory, since it's a form storage. It's just a disk space basically, so can VM experience contention when Cluster consume metric is not high?

I'd zoom into ESXi, so it's easier to see. What do you deduce from this ESXi? This chart shows 1 month worth of data.



It has 759 GB of usable memory. All the powered on VM has 444 GB configured, out of which only 413 GB is mapped to physical DIMM. So there is plenty of memory left.

To confirm that it has plenty of memory, let's plot Balloon. What do you expect?

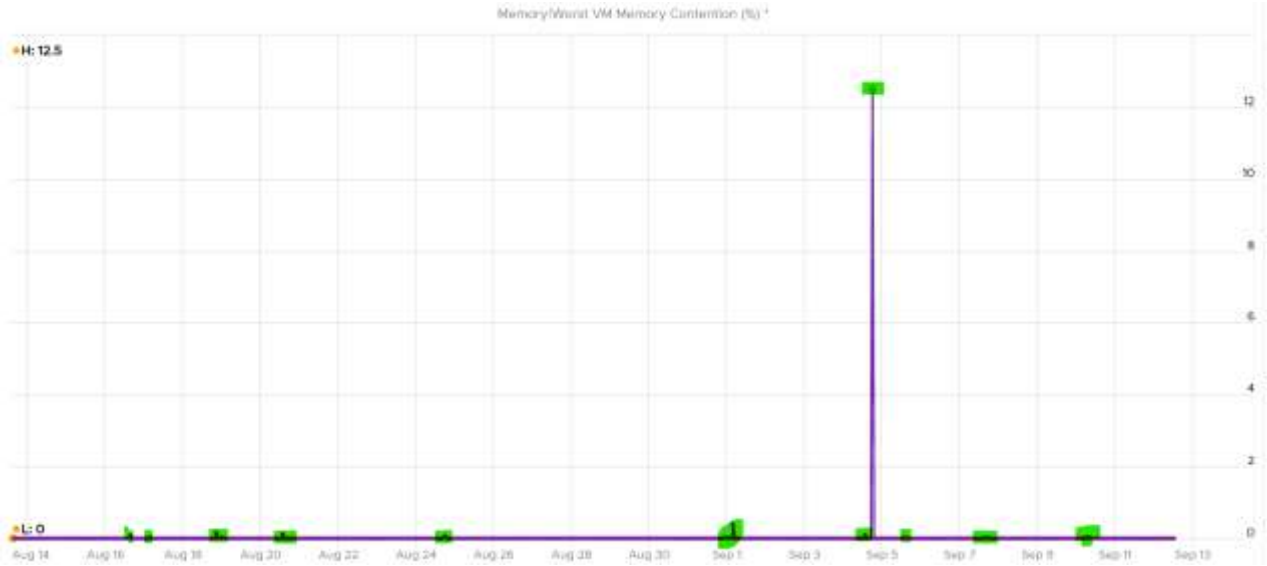


There is no ballooning. ESXi was under no memory pressure whatsoever.

So that's the situation at provider level. How about consumer level?

VCF Operations has a metric that tracks the highest memory contention experienced by any VM in the host. This is a good leading indicator as all it takes is 1 VM, it matters not which VM.

As we can see here, there is a problem.



Can you explain why?

A VM experiences contention when the page is not in the DIMM. It was compressed or swapped out. Checking the compressed metric, it reveals that pages had to be brought it. Notice the swap metric lagged a bit, which makes sense.



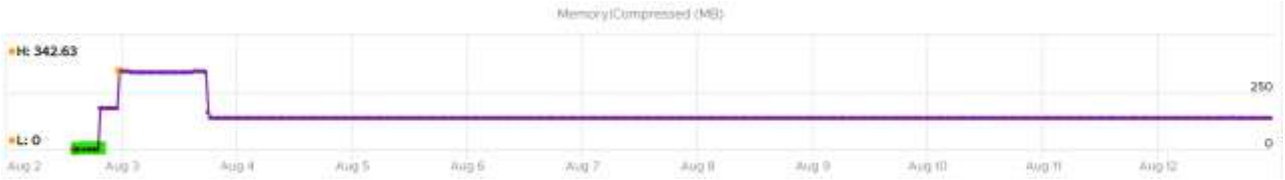
I am not able to explain the earlier dropped, the one in red circle. If you can drop me a note.

Let's complete by plotting Swapped. I'm plotting all the way to the beginning of tracking.

It's all 0. What happened?

That means all the pages could be compressed, so ESXi decided to compress instead of putting them into swapped file.

Now that we know it's due to compression, we know the contention on 5 September was caused by compression. When was that page compressed, no one knows. Plotting back, the compression started around 2 August.



The compression was only 342 MB. Not even 0.1% of consumed memory. But if you are unlucky, it was the active VM that got hit, as in the case here.

The past is harder to debug, as we lack the ability to travel back in time and see the environment as it was. My guess here is the VM had limit, be it indirectly via resource pool or directly.

Cluster Performance (%)

We've covered in the VM chapter how we quantify the KPI of a single VM. How would you represent all the VMs in the cluster? Do you simply average the VM KPI (%)?

The answer is no. A cluster has a different purpose to a VM, so we need to see it from cluster point of view. For examples:

- Contention inside a VM (this means Windows or Linux) is not that relevant to the cluster performance.

- ESXi physical network is relevant to the cluster performance, but not to the VM performance.

A cluster is more of a group of ESXi hosts serving a VM.

	Metrics	Green	Yellow	Orange	Red
How Deep?	Worst Peak vCPU Ready among all VMs	0 – 4%	4 – 8	8 - 16%	16 - 32%
	Worst Peak vCPU Co-Stop among all VMs	0 – 2%	2 – 4	4 – 8	8 - 16%
	Worst Peak Memory Latency among all VMs	0 – 1%	1 – 2	2 – 4	4 – 8%
	Worst Peak Read/Write Disk Latency all VM Virtual Disks	0 – 40 ms	40 – 80	80 – 120 ms	120 – 160 ms
How Broad?	90th Percentile Peak vCPU Ready among VMs	0 – 2%	2 – 4	4 – 8	8 - 16%
	90th Percentile Peak vCPU Co-Stop among VMs	0 – 1%	1 – 2	2 – 4	4 – 8%
	90th Percentile Memory Latency among VMs	0% - 0.5%	0.5 – 1%	1 - 2%	2 - 4%
	90th Percentile R/W Latency among VM Virtual Disks	0 – 20 ms	20 – 40	40 – 60 ms	60 – 80 ms
ESXi	Highest (swapped + zipped) memory	0% - 0.25%	0.25 – 0.5%	0.5 - 1%	1 - 2%
	Highest ESXi (error + dropped) network packets	0%	< 0.2%	0.2 – 0.4%	0.4 – 0.8%

The metrics are grouped into 2:

- Breadth
- Depth.

We cover why we need both dimensions in Part 1 Chapter 2 Performance Management.

The metrics are using the 20-second peak so it does not miss short bursts. The limitation of this metric is outlier. A large cluster with thousands of VMs can easily have 1 VM having contention due to limit. One way to minimize the false positive is to set a higher threshold.

For the breadth, we use 90th percentile instead of average. Average tends to be too late. On the other than, 95th percentile suffers from cluster imbalance.

In a cluster with many hosts, there can be imbalance. It is not possible to aggregate at the host level first. This 2-level aggregation can result in suboptimal number. The drilling down to specific host can be facilitated with the host level metric.

Metrics

vMotion is included as it does impact the VM performance (although the end users may not notice in most cases) and it's a leading indicator that the cluster is struggling to serve the load hence it has to shuffle the VMs around.

Take a look at this cluster. It has 488 running VMs on 16 ESXi host. Notice the percentage of VM being vMotion jumped to 5.3%, as 26 VMs were vMotion.



What do you think will happen to the VM CPU Ready and CPU Co-stop?



They rose. Since only 5% was impacted, the rise will be minimal.

The threshold should reflect reality. For examples:

- While the impact on VMs is the same with Ready, using the same range for CPU Co-stop, CPU Overlap and CPU Other Wait will elevate the KPI score, as practically these 3 have lower score.
- Dropped packet and error packets are very rare. Instead of summing them up, which will result in an average, I took the worst among ESXi host. Since many ESXi sports 25 Gb NIC, I set the threshold to be very low. On the other hand, I did not set green = 0, so the KPI does tolerate some issue.
- Ballooning does not actually impact performance. That's why it's given 4x the threshold of zipped + swapped. I may change to 6x, but will have to consider large cluster. A cluster with 20 TB of RAM means 4% ballooned translates into 800 GB.
- Ballooned, swapped, and zipped are given higher threshold as they could have happened in the past. These metrics are "sticky". They also may not impact performance. But since many clusters sport >4 TB of RAM, I need to balance the absolute size.

Metrics Not Used

The following metrics are not included:

- "Percentage of VM population experience CPU Ready > 1%" and "Percentage of VM population experience Memory Contention > 0%" as they measure the same purpose with the average.
- VM utilization.
  They are irrelevant. Use the metrics at ESXi level instead.
- ESXi Consumed.
  The higher the consumed, the better the performance. We use balloon, swapped and compressed metrics instead. Since they do not directly impact performance, we put as secondary KPI.
- ESXi CPU Utilization.
  They do not actually impact performance. If you include them, take the highest among ESXi instead of cluster wide average, as average is too late.

- VM DRS Score.
  The focus is on VM demand, which is not identical to performance (read: contention).
  For example: 23 VMs are running on a host. 15 VMs with low memory demand (8643MB to 66312MB) have score as 68. 8 VMs with high memory demand (132052MB to 264859MB) have the low score (< 25). The memory over-provision rate is 117%.
  Cluster DRS score = It is a weighted average of the VM DRS Scores of all the powered on VMs in the cluster.
- Disk latency.
  Storage troubleshooting differs to compute troubleshooting. The exception here is Hyperconverged Infrastructure, as the compute and storage are integrated.
- Percentage of VMs facing Disk Latency.
  It should be part of datastore KPI, not the cluster. The reason is if there is a population problem, you troubleshoot the datastore not the cluster(s) mounting it.
- CPU Other Wait.
  There is a false positive. I've seen it hence decided to exclude.
- CPU Overlap.
  I find their values to be very low that they might mask out other problems.

Future enhancements:

- There are metrics that are only available in esxtop. They are not available in vCenter REST API, so they are not retrievable. Example is Local:Remote memory ratio for VM.
- 2 level roll up. This means that the cluster metrics take from its hosts instead VMs. In this way, every host is represented. Need to model the difference scenarios before we decide this solution meets the requirement better.

Implementation

The above is implemented using super metrics.

```
sum ([
   ({This Resource: Super Metrics|vSphere Cluster CPU Ready}                    as Ready
   ({This Resource: Super Metrics|vSphere Cluster CPU Co-Stop}                  as coStop
   ({This Resource: Super Metrics|vSphere Cluster Memory Contention}            as MemLatc
   ({This Resource: Super Metrics|vSphere Cluster Worst VM CPU Ready}           as mReady
   ({This Resource: Super Metrics|vSphere Cluster Worst VM CPU Co-Stop}         as mCoStop
   ({This Resource: Super Metrics|vSphere Cluster Worst VM Memory Contention}   as mMemLatc
   ({This Resource: Super Metrics|vSphere Cluster Worst ESXi Bad Network Packets} as BadNet
   ({This Resource: Super Metrics|vSphere Cluster vMotions}                     as vMotion
   ({This Resource: Super Metrics|vSphere Cluster Memory Stressed}              as MemStress
   ({This Resource: Super Metrics|vSphere Cluster Memory Ballooned}             as Balloon
])
```

All the super metrics are fairly simple. They are simply taking the average or the maximum of either ESXi or VM in the cluster.

Something a little trickier is Ballooned (%). The formula is below.

```
( This Resource: mem|vmmemctl_average )
/
(( This Resource: mem|vmmemctl_average )+( This Resource: mem|consumed_average )-( This Resource: mem|sysUsage_average ))
*100
```

Why did I exclude kernel usage, but include ballooned?

Answer is in the Memory chapter.

Troubleshooting

At any given moment, a running VM always resides on an ESXi Host. Due to DRS and HA, it's easier to monitor at cluster level. Since a cluster can have hundreds of VMs, you need consolidated metrics that can represent the experience of all the running VMs in the cluster. VCF Operations provides the following metrics:

Problem	Optimize or Remediate
Worst VM CPU Ready. Worst VM RAM Contention	These are the highest value among all the running VMs. It shows you the depth of the problem. Check how many VMs are affected to see pattern.
% VMs experienced CPU Ready % VMs experienced RAM Contention	Since it's impacting many VMs, the problem is likely not at VM setting, but at ESXi or cluster level. Some possibilities: • There is 1 or more sibling VM to Resource Pool, causing VMs in the pool to suffer. • Imbalance shares. 1 or more VM has relatively large shares. • High ESXi utilization. However, Cluster utilization is unlikely to be high, due to HA and Buffer. So check for imbalance among hosts, which is common in large clusters.
	Imbalance could be due to: • DRS settings • VM – Host affinity, or VM – VM affinity • Resource Pool • Cluster inconsistent configuration (e.g. storage or network not available to all hosts)
vSAN CPU Ready	It's not something you can change, unless via ESXi advanced settings to modify the kernel scheduler behaviour?
ESXi Error Packets	Check the vmnic driver and firmware

Service Level Metrics

These metrics require the understanding of how SLA, SLI and KPI differ. They are covered in-depth in Part 1 of VMware Operations Transformation book. Their implementation require VCF Operations super metrics and custom dashboard.

Cluster SLA

How to roll up the VM SLA into total SLA for the whole environment? Your CIO likely wants to see this number over time.

Calculating SLA per vSphere cluster also makes management easier. You know which cluster to attend to. The problem is SLA is a lagging indicator. It is based on the last 30 days or the last month.

Cluster SLA is derived from the VM SLA. It is simply the percentage of VMs that fail the SLA. How bad each VM fails the SLA, or how comfortable it exceeds the SLA, is *irrelevant* at this stage. At the cluster level, you care about pass/fail first.

That means the Cluster SLA is not the average of its VM SLA. Doing an average can be too late unless your SLA is 100%.

Once you know how many VM fails, you want to know who the VMs are and troubleshoot if there is a common reason.

Cluster SLI

SLA is a 30-day counter. You can't wait that long before you do something. This is where SLI comes in. It's an indicator, and not mentioned in the SLA contract.

Let's take an example of a cluster with 500 VM. Each VM consumes 4 IaaS resource (CPU, Memory, Disk, Network). It must pass all else it's counted as 1 SLI fails.

The Cluster SLI (%) is simply the percentage of VM that fails the SLI. As a recap, this is the single threshold we use for all classes of service:

IaaS	VM Counter	Threshold
CPU	Ready	2.5%
RAM	Contention	1%
Disk	Latency	10 ms
Network	TX Dropped Packet	0%

It's a normalized average of the VM SLI, taking into account the actual SLI failures. That means it will give a lower number if the VMs are experiencing worse SLI individually. 1 VM experiencing 4 SLI failure will result in the same value as 4 VM experiencing 1 SLI value each.

The formula is

```
100 -
(
  ( Sum([VM]Performance|Number of KPIs Breached) + Sum([Pod]Performance|Number of KPIs Breached )
)
  /
  ( Summary|Number of Running VMs + Summary|Number of Pods )    * 100 / 4
)
```

Capacity

Now that you've reviewed the raw metrics, let's apply them into capacity management.

vSphere Cluster capacity is often misunderstood, as there are multiple considerations.

- On the supply side, you have total capacity and usable capacity. Both have nuances.
- On the demand side, you have utilization, reservation, allocation, and unmet demand.
- The kernel impacts both the supply side and demand side. Be careful of double counting!
- Lastly, CPU, memory, storage, and network have different natures.

Key Metrics

The essential metrics of capacity are

- Total Capacity
- Usable Capacity

From the first principle, the total capacity and usable capacity should not be a variable as it makes capacity management impractical. Your 100% should always be a constant so you have a stable anchor. This makes cost accounting less debatable too.

Total Capacity

To define total capacity, first determine whether its value is fixed or not.

If it's not fixed, the next question is whether the change is Scale Up or Scale Out.

- Scale Up happens on a single object. Example is the just in time capacity such as storage LUN, where the size can be increased on the fly or upon request.

- Scale Out happens on a cluster of objects. The cluster members or nodes are typically identical in size. Examples are K8 cluster, K8 Workload, Horizon VDI Pool, and VMC vSphere Cluster. Scale Out is more popular than scale out as it's cheaper. It does not work when the data can't be partitioned.

Total capacity becomes dynamic only if you *intentionally* change the numbers. Some examples:

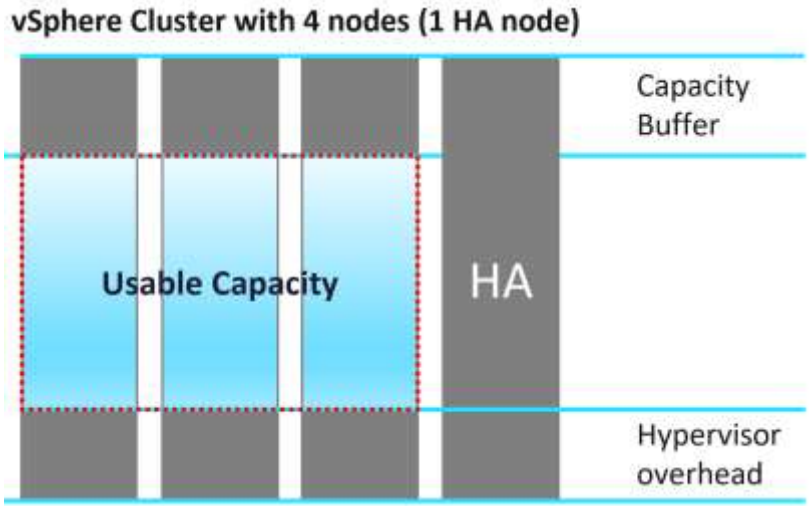
- Horizontal auto-scaling solution such as VMware on AWS
- vSphere Distributed Power Management.

Usable Capacity

Usable Capacity is an imaginary number that you get after deducting total capacity with the portions that capacity team decide to exclude. The number is non-real as actual utilization can exceed it.

Usable Capacity has to be a stable number. Having a volatile value over time makes capacity management hard. The cost of complexity is not worth the accuracy.

Let's use vSphere Cluster as example.



In general, there are 3 components that need to be excluded to form usable capacity.

```
Usable Capacity = Total capacity – Availability protection - Overhead – Buffer
```

Let's look at each:

Availability protection	This covers both local availability (HA) and disaster recovery. For hardware, this means the part that is added to cater for unavailability period. Common examples are RAID in disk, hot spare in storage array, vSphere HA in vSphere Cluster. Many hardware deployments come in a pair (e.g. network switches) because one of the nodes is for availability, not capacity. In vSphere Cluster, you typically design with at least 1 host as spare, so you can perform maintenance, upgrade without service degradation. While this host is actively participating in reality, you exclude this in your usable capacity. In Kubernetes, there is no such thing as HA as the node itself is transient.

Overhead	Overhead is part of the system architecture and can't be avoided. In the case of ESXi, this is the hypervisor. This portion is not available for "consumer" or business workload. Examples: • VMkernel in ESXi • Kubernetes supervisor cluster. • Load of the OS in K8 Node.

Let's apply the above concept into vSphere:

ESXi Usable Capacity	Usable Capacity = Total Capacity – Hypervisor.
	Hypervisor = VMkernel + vSAN + NSX + vSphere Replication
	So what amount should we put for the hypervisor? It turns out that it's not an easy answer. We will dive in later on.
	Extracting the hypervisor portion as a separate value has a bonus as it can be used use cases involving 2 different ESXi hosts, such as migration from non VCF cluster to a cluster running both vSAN and NSX
Cluster Usable Capacity	Cluster is not a matter of adding the ESXi. There are 2 cluster-level settings: • Buffer • High Availability
	Be careful in aggregating as overhead and availability overlaps. Reason is your HA node contains overhead too

Applicability

The formula for Usable Capacity depends on the model.

Model	HA Host	Hypervisor	Buffer
Allocation	Included	Included	Included
Utilization	Included	Cannot be included as values are too volatile for capacity management	Included
Reservation	Included		Included

Buffer

Buffer is a business decision and it's optional.

It's calculated based on capacity after HA, and not on the total capacity.

Peak utilization is often used as the reason to have buffer. The actual reason behind is performance avoidance. Since it's about avoiding the contention from happening, the contention counters cannot be used. Take for example, a large cluster (could be vSphere, Kubernetes, or Horizon) where imbalance can happen among the cluster nodes. You have witnessed that it happens at say 95% utilization. In order to avoid that, you set buffer at 10%, effectively setting the usable capacity at 90%. This in theory gives you 5% buffer, as the last 5% can be measured using contention counter.

Disaster Recovery is a valid use case for buffer. Some examples are:

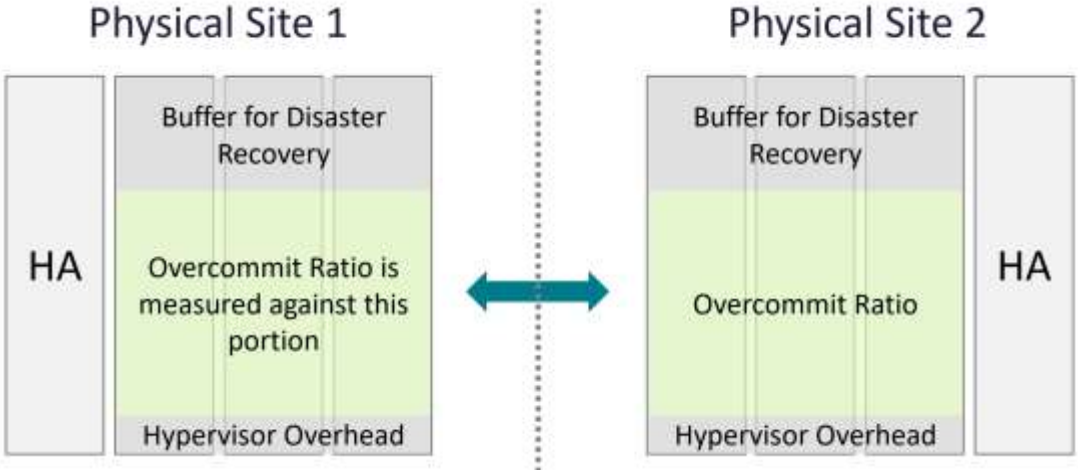
- A pair of vSphere Clusters protecting each other in a DR pair.
- A production cluster and DR cluster. The DR cluster typically run test and development workload, which will be powered off in the event of DR drill or actual DR.
- A stretched cluster.

Let's elaborate the first example

- Say you design a pair of 10 node cluster to protect each other. Each has 9+1 set up for HA to give you room to do the usual cluster maintenance and upgrade.
- You want to limit the utilization of each site at 50% of the 9 hosts. This translates into 45% utilization on each of the 10 hosts, if there is no imbalance.
- To monitor the above, you set 50% buffer on the capacity after HA. That means the actual formula is (Total – HA) x Buffer.
- Now, since the VMs typically run well below their configured size, you use allocation model and cap it at say 3:1.
- To complicate matters, say 60% of the workload can be turned off during DR. In this case, you allow utilization to reach 160%.

Allocation

Overcommit Ratio does not include buffer. The following diagram makes it clear.



Overcommit ratio looks simple on the surface. As we dive deeper, it's not as straight forward. For examples:

- CPU hyperthreading.
  This can be turned on and off. Should the ratio change accordingly? Do we consider it since there are only 2 threads per core? Latest CPU has >100 cores but each has only single threads.
- Datastore. Do you consider the space used by snapshot since overcommit is based on allocation? How about the availability protection created by vSAN?

To answer the question, remember that allocation model does not consider utilization. Since allocation is about allocating to consumer, the provider part should be excluded too.

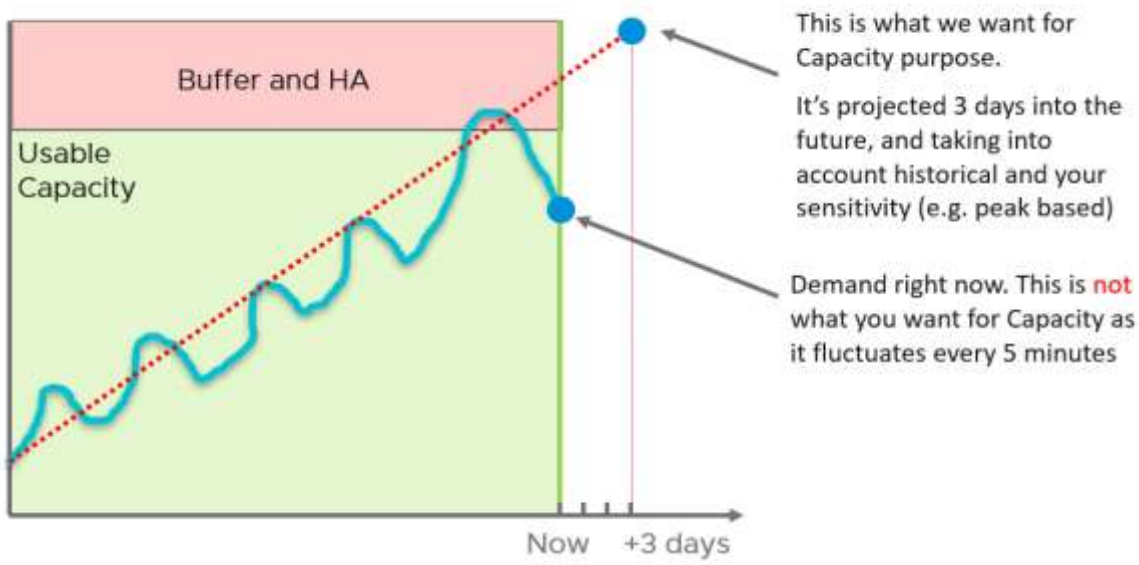
CPU	Include hyper threading. The reason is a VM vCPU maps to a thread, not core.
	Yes, hyperthreading impacts performance, but overcommit is about capacity. Also, you can simply change the number. Instead of saying 8:1 to a core, you say 4:1 to a thread, with HT enabled.
	To mitigate the performance impact, simply get a CPU that is 60% faster. Example: • CPU speed is 1 GHz. • When HT is enabled, each thread gets 625 MHz. • To get back to 1 GHz per thread, you need 1.6 GHz.
	Take note a lot of education is needed. When your tenants buy 4 GHz CPU from you, they expect 4 GHz, not 2.5 GHz (62.5% throughput). Since you enable HT, one way to simplify the message is to say the CPU speed is 2.5 GHz.
Disk	For datastore, only consider the VM virtual disks. Exclude other VM disk space such as snapshot and memory swap. If you have a lot of snapshots and high memory overcommit, this can result in higher actual disk space consumption.
	If the datastore is vSAN, exclude the vSAN availability protection (Failure To Tolerate). The downside of this is the overcommit ratio may fail to serve its purpose, which is to give early warning before actual utilization happens. So set a lower number, matching to the FTT setting. For example, if your FTT doubles the disk space consumption, then your overcommit ratio has to be halved.
Network	For physical switch, exclude the inter-switch link. For ESXi, only include the physical ports used by the VM.

Projected Metrics

At the heart of capacity is projecting the historical consumption into the future.

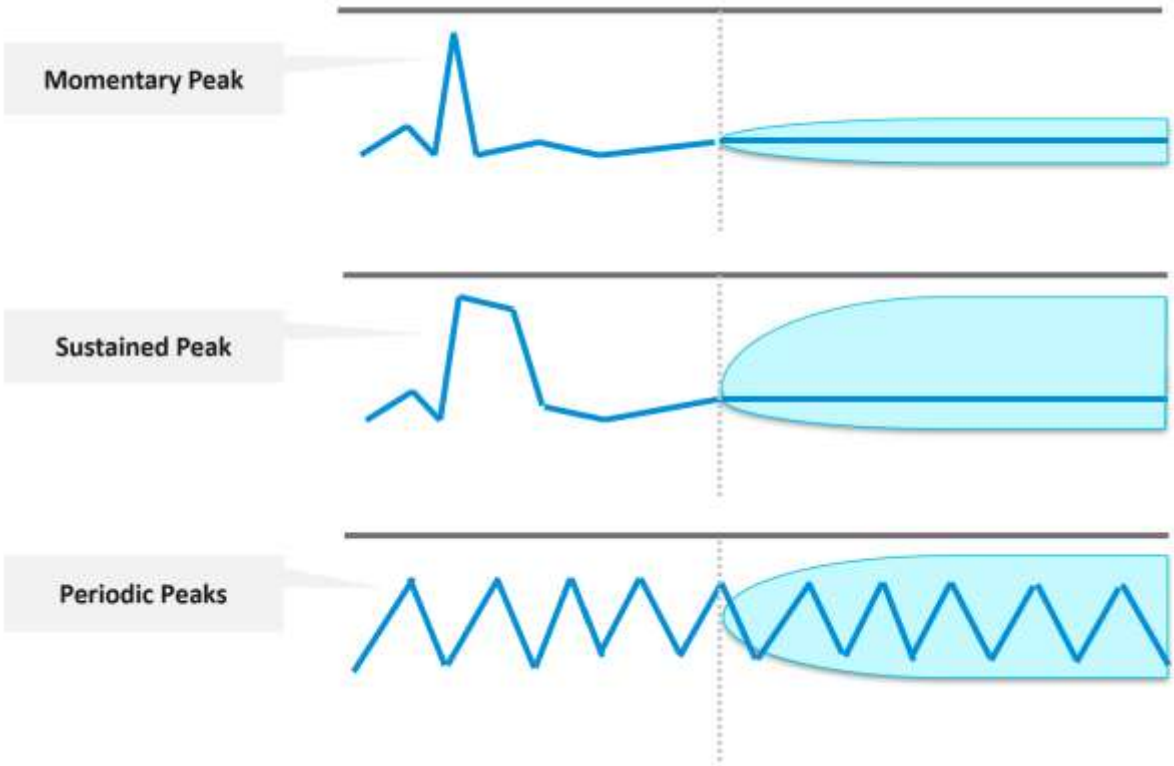
Projection

If there is clear trend, a projection is superior to a simple percentile calculation.



The accuracy of the prediction depends on the amount of data and the length of the cycle. If the data is limited and the pattern that matches your business cycles has not developed, the projection would not meet your expectations.

A workload with quarter end peak will naturally need at least 6 months for it to be accurate. If there is enough data, VCF Operations will consider 6+ months' worth of data. While it gives extra weight to recent data, if there is a sudden but short-lasting change, it may not be enough to impact the projection.

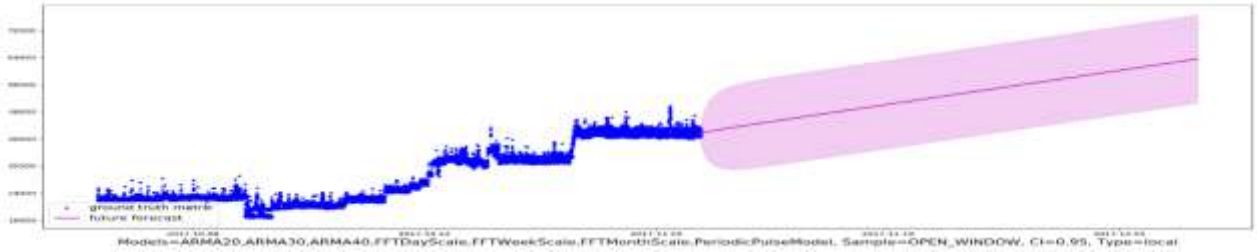


Momentary peaks that are short lived and one-off should not impact capacity planning so the impact may not be noticeable in the projection.

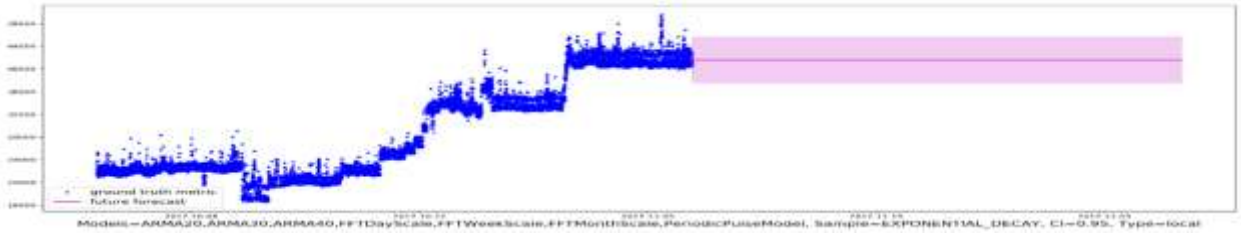
Sustained peaks last for a longer time and do impact projections. If the peak is not periodic, the impact on the projection lessens over time due to exponential decay. Data is exponentially weighted based on how far back in time they are, giving recent data points more important than older ones.

Periodic peaks exhibit cyclical patterns or waves, such as hourly, daily, weekly, and last day of the month. There can be multiple overlapping cyclical patterns, which will also be detected. While you should not make capacity decision based on just a few days of data, you do need the 5-minute granularity as input. A 5-minute peak that gets repeated every hour should be considered.

Exponential decay is important as newer data is more relevant, but it might make the projection visually odd. The following projection looks "make sense", because your eyes see the whole period and give equal weightage to all data points.



If you give higher weightage to newer data, it can potentially look like this.



Workaround

If you do not have 3 months and just need an overall sizing, consider using the 97th percentile value. Why 97th percentile? It's based on standard deviation principle. Two Standard Deviation away from the midpoint equals to 95%, and 3 Standard Deviation = 99.7%. 97th percentile hence provides a good balance between 2 SD and 3 SD. By and large, it captures just the right amount of peak and outlier.

Capacity Remaining

Capacity Remaining (%) metric is a complex metric as it depends on the time. Let me use an analogy as it's easier when the demand has end date

You operate a restaurant business. It has 100 seats.

It's 4 pm in the afternoon, and you have plenty of seats as it's not a busy hour. But you're fully booked for dinner.

What is your Capacity Remaining (%)?

The answer is it depends on what time.

What is your Time Remaining (Days)?

The answer depends on your projection.
If you only reject a small percentage of customers and the additional potential revenue is not worth the capital expansion, your Time Remaining is forever.
If you reject enough customers and foresee demand will grow, you take the risk of adding more capacity. In this case, your time remaining is 0 days.

The Capacity Remaining (%) metric is a projected value, 3 days into the future, hence it might differ with currently used capacity. The 3-days is hardcoded, not something you can change. No, it does not and should not care about the future demand, even though they are already committed.

As it is a future value, there is confidence band. You can choose between aggressive (based on the upper limit of the band) and conservative (based on the actual trajectory).

Note that the value of will be set to 0 if during the given collection cycle the demand breaches the usable capacity. This is because at that moment there is really no capacity. This can cause fluctuating value of Capacity Remaining metric if the load regularly touches the usable capacity threshold.

Take note that CPU Capacity Remaining (%) and Memory Capacity Remaining (%) appear in the policy as enabled but cannot be used. That's an internal metric which should have been hidden.

Time Remaining

Capacity Remaining (%) and Time Remaining (days) metrics need to be reviewed together, because the ideal situation is the object (VM, cluster, etc.) is low capacity remaining yet high time remaining.

It measures the number of days before capacity runs out. For conservative projection, this measures the time required from now to when the upper confidence interval of Long Term Forecast intersects/crosses Usable Capacity. The projection is up to 1 year, with time remaining above 1 year is simply shown as 1 year.

Formula wise, it's based on the utilization metric. It's *not* a projection of the Capacity Remaining metric. But should it be? Let me know your thought!

Hypervisor

Why Hypervisor?	Why not use the word kernel or VMkernel?
	The hypervisor is more than kernel. There are user-level or application that is runs on top of the kernel.
	The word kernel is often mistaken with VMkernel. VMkernel does not include vSAN & NSX as they are not traditionally considered part of kernel. vSAN for example has processes parked under /opt resource group
Capacity or Performance?	Why do I put hypervisor under capacity, and not performance?
	Because operationally, the metrics impact capacity management. Since the hypervisor gets the highest priority, you do not monitor the metrics from performance viewpoint. If you need to see the ready time for each of the kernel process, see esxtop.

Kernel does not have allocation as it's an OS process.

The hypervisor has 3 types of metrics:

- Reservation
- Limit
- Utilization.

Which one do you use?

- Utilization is not feasible as it changes by the seconds. Just like total capacity cannot be volatile, the same goes with usable capacity.
- Limit does not even make sense as certain features of hypervisor impacts all VM, hence should take higher priority.
- Reservation tends to be too low if you run vSAN and NSX, and too high if you only run ESXi. It also fluctuates over time, giving you unstable usable capacity.

The last option is to manually include a static value when calculating the usable capacity. This means we need to know the amount.

Metric Type

ESXi scheduler uses share, limit and reservation to manage its worlds. Broadly speaking, there are 2 types of worlds:

- VM
- Non VM

You will see 3 types of metrics in the vCenter UI:

Type	Analysis
Utilization	This is the actual, visible, consumption. It can be lower than reservation, but not higher than allocation.
	Since you've already paid for the hardware, you want to drive ESXi utilization as high as possible so long there is no contention. Since the hypervisor has higher priority than VM, we can safely assume we can use VM contention as the proxy for overall contention (assuming manual VM Limit is not set). The ESXi utilization metric considers both the hypervisor and VM. There is no need to separate the hypervisor in this case. The only time we need to separate is when we're migrating the VMs into another architecture.
Reservation	For the hypervisor processes, the maximum amount is taken care of by allocation, while the minimum amount is by reservation. This is a safety mechanism to ensure the hypervisor can still run when all the VMs want 100% resource. Processes that run at hypervisor level does not get its reserved memory up front. It's granted on demand. CPU, being an instruction in nature, does not use the reserved amount unless it needs to run. If you plot in vSphere Client UI, you will see the value of utilization can be lower than reservation.
Allocation	For VM, allocation is useful as there is overcommit between virtual and physical. For the non VM, it is not useful since there is no overcommit because there is no virtual part. You notice that some hypervisor processes have no limit. If you plot them in vSphere Client UI, you will find their limits are either blank or 0.

The above 3 values vary over time. Why is it hard to determine the size of the above 3 values up front?

Taking from page 258 of Frank Denneman and Niels Hagoort's book, with some changes:

- Some services have static values (allocation and reservation) regardless of the host configuration. Ok, this is the easy part.
- Some services have relative values. It scales with the memory configuration of the host. Ok, that means you need to know the percentage for each.
- Some services have relative values that are tied to the number of active VMs. Ok, that means you need to know how many VMs are active.
- Some services consume more when they do more work. Example is storage and networking stack.
- Some services consume more depending on the configuration. For example, vSAN consumes more when you turn on dedupe and compression.

Since an ESXi host has many services, it is impossible to predict the overall values of the above 3 metrics.

Grouping

All the processes that run in the hypervisor belong to one these 5 top-level resource groups[20]:

System	host/system resource pool for low-level hypervisor services and drivers. You will find world such as minfree, kernel, helper, fault tolerant, vmotion, storage vmotion, vmk API mod, idle, and drivers. Doing multiple vMotion simulaneously will increase the consumption of vmotion resource. The data plane portion of vSAN is reported here, although there is no separate counter for it.
VIM	VIM = virtual infrastructure manager. vmvisor = hypervisor. This include NSX, and vSAN management plane. host/vim resource pool for host management process such as HA (aam), vCenter agent vpxa, hostd, VIM user (the group for DCUI, shell, SSH, Tools), authd, tmp, envoy, GPU Manager, ESX tokend, healthd
User	host/user resource pool All the running VMs are children of the User resource pool. This includes the VM overhead as it's part of the VM. There is no breakdown for this pool. The only metric is host/user. vSphere Client UI does not display the CPU or memory reservation metrics.
Opt	Mostly vSAN. You will see it as opt/vsan. An example of process will be vsan/vsanperfsvc for the performance monitoring. Added in vSphere 8.0.1
IO Filter	host/iofilter resource pool The IO Filter processes are grouped here. The generic framework allows 3rd party partner software to intercept and process network and storage IO. More about it at vSphere manual. Just search for "About I/O Filters". If you are unsure, read this by Ken Werneburg. Note: vSphere Client UI does not display the CPU or memory reservation metrics.

In the older version of vCenter, you could see the structure. The dialog box is no longer available in the present vCenter UI. I've made the screenshot smaller as the details has changed, so this is just to show the idea.

[20] The structure is deep. To know more about how ESXi resource pool group structure, I recommend these talks by Valentin Bondzio. Specifically, minute 18:10 on his VMware Explore Barcelona 2023 session.



Relative Comparison

You will notice major differences in the way the resource groups consume resources.

	CPU	Memory
System	Surprisingly low. It can be well below 1 GHz.	Relatively high. It's ~20 – 30 GB depending on the ESXi
VIM	Relatively high. It's around 4 – 12 GHz depending on the ESXi.	Surprisingly low. It could be even 0 GB.

```
vmgid   name                  pname        amin   amax   minLimit units    shares
-------------------------------------------------------------------------------

0       host                  none         3200   3300   3200     pct      4096000

1       |- system             host         110    -1     -1       pct      500
2       |- vim                host         342    -1     -1       pct      500
3       |- iofilters          host         0      -1     -1       pct      1000
4       |- user               host         0      -1     -1       pct      9000
5       |- opt                host         0      -1     -1       pct      500


-------------------------------------------------------------------------------

-------------------------------------------------------------------------------
     gid name           parGid   nChild        min        max   minLimit     shares
-------------------------------------------------------------------------------

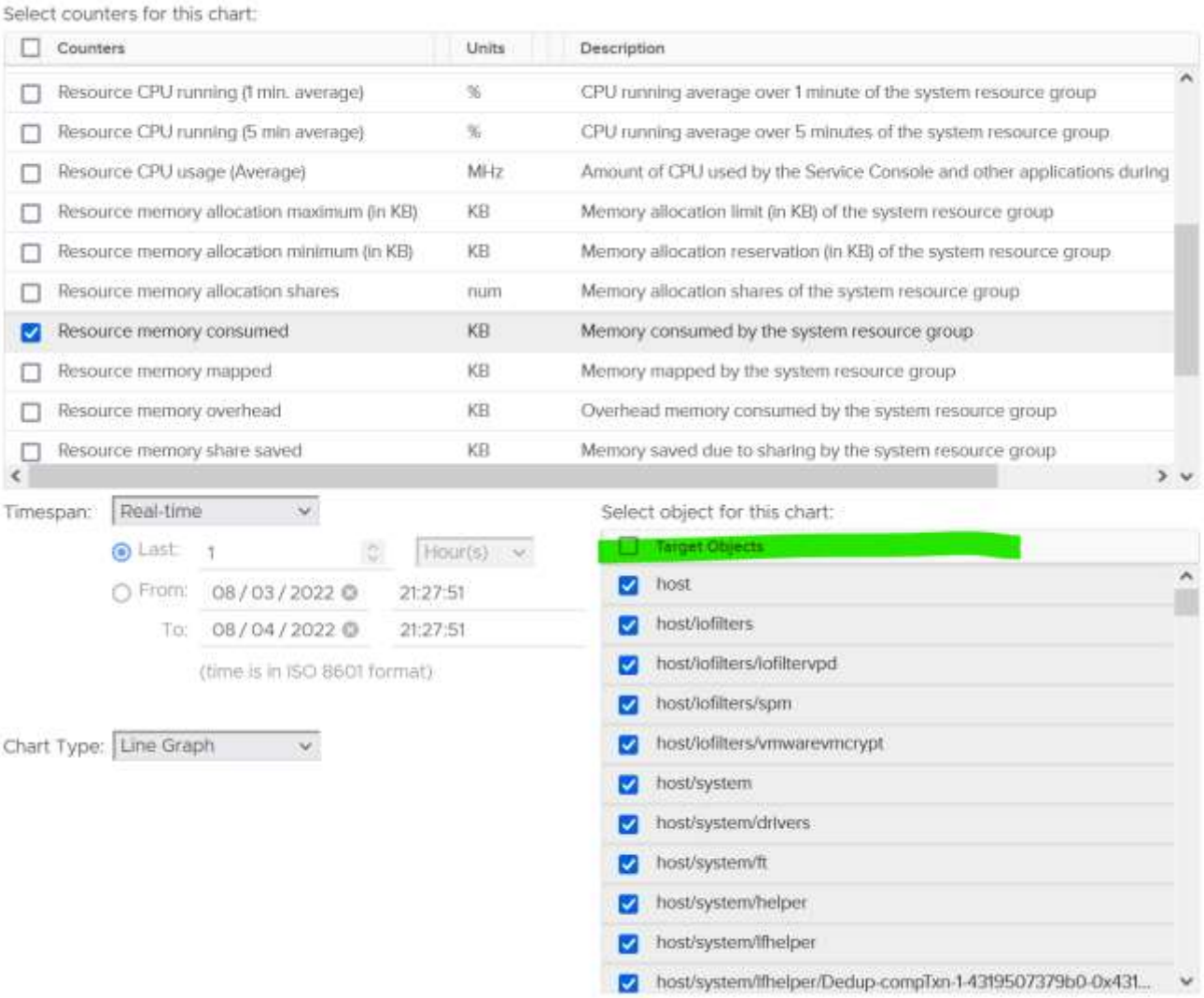
       0 host               -1        5     391428     391428     391428 8589934591

       1   system             0       15      14720         -1         -1        500
       2   vim                0        3          0         -1         -1        500
       3   iofilters          0       11          0         -1         -1        500
       4   user               0        2          0         -1         -1       9000
       5   opt                0        1          0         -1         -1        500
```

Metrics

In the vSphere Client UI, you will see the list of resource grouping in the Target Objects section in the performance chart.

I've highlighted them in the following screenshot:



To see the kernel consumption, select only these 3 from the list above:

- host/iofilters
- host/system
- host/vim.

The rest of the items are part of them, so no need to plot them. More importantly, they are fairly small, well below 0.5 GHz. The following screenshot shows their highest 20-second average in the last 1 hour.

Object	Maximum	Units	Measurement	Rollup
host/system	3,484	MHz	Resource CPU usage	Average
host/vim/vmvisor	3,121	MHz	Resource CPU usage	Average
host/vim/vmvisor/sealstage	1,677	MHz	Resource CPU usage	Average
host/vmvisor/[redacted]	388	MHz	Resource CPU usage	Average
host/vim/vmvisor/infravisor	245	MHz	Resource CPU usage	Average
host/vim/vmvisor/[redacted]	186	MHz	Resource CPU usage	Average
host/vim/vmvisor/infravisor/cmds	160	MHz	Resource CPU usage	Average
host/vim/vmvisor/backup.sh	128	MHz	Resource CPU usage	Average
host/vim/vmvisor/logging	102	MHz	Resource CPU usage	Average
host/vim/vmvisor/[redacted]	89	MHz	Resource CPU usage	Average
host/vim/tmp	83	MHz	Resource CPU usage	Average
host/vim/vmvisor/envoy	82	MHz	Resource CPU usage	Average
host/system/fdhelper	74	MHz	Resource CPU usage	Average
host/vim/vmvisor/[redacted]-plugins	64	MHz	Resource CPU usage	Average
host/vim/vmvisor/vmkmemstats.sh	50	MHz	Resource CPU usage	Average
host/vim/vmvisor/systemStorage	32	MHz	Resource CPU usage	Average
host/vim/vmvisor/[redacted]	23	MHz	Resource CPU usage	Average
host/vim/vmvisor/auto-backup.sh	19	MHz	Resource CPU usage	Average
host/vim/vmvisor/hostd-tmp	16	MHz	Resource CPU usage	Average
host/vim/vmvisor/crond	16	MHz	Resource CPU usage	Average
host/vim/vmvisor/[redacted]	13	MHz	Resource CPU usage	Average
host/vim/vmvisor/[redacted]	12	MHz	Resource CPU usage	Average
host/vim/vmvisor/apiForwarder	12	MHz	Resource CPU usage	Average
host/vim/vmvisor/watchdog	9	MHz	Resource CPU usage	Average
host/vim/vmvisor/healthd	9	MHz	Resource CPU usage	Average
host/vim/vmvisor/hostd-probe/stats/probe	8	MHz	Resource CPU usage	Average
host/vim/vmvisor/hostd-probe	8	MHz	Resource CPU usage	Average
host/vim/vmvisor/hostd-probe/stats	8	MHz	Resource CPU usage	Average
host/vim/vmvisor/[redacted]	8	MHz	Resource CPU usage	Average
host/vim/vmvisor/lacpd	6	MHz	Resource CPU usage	Average
host/vim/vmuser	6	MHz	Resource CPU usage	Average
host/vim/vmvisor/hostdCgiServer	6	MHz	Resource CPU usage	Average
host/vim/vmvisor/vobd	4	MHz	Resource CPU usage	Average
host/vim/vmvisor/nscd	4	MHz	Resource CPU usage	Average
host/vim/vmvisor/nfcpproxy	4	MHz	Resource CPU usage	Average
host/vim/vmvisor/sandboxd	4	MHz	Resource CPU usage	Average
host/vim/vmvisor/fpt	3	MHz	Resource CPU usage	Average
host/vim/vmvisor/auditd	3	MHz	Resource CPU usage	Average
host/vim/vmvisor/vmkdevd	2	MHz	Resource CPU usage	Average

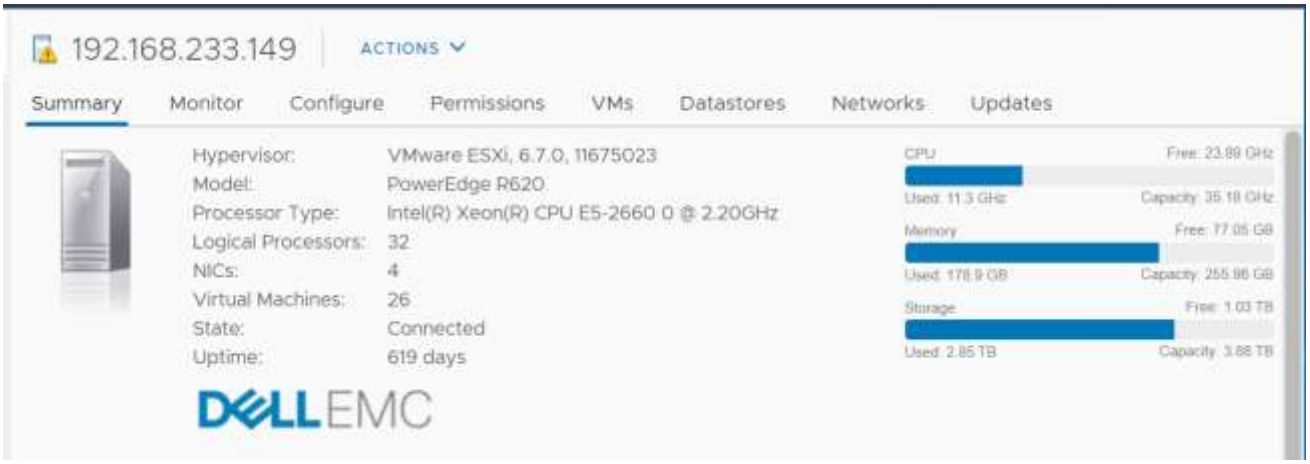
To see their total, plot their values in vCenter by stacking up their values, as shown below.



CPU

When you buy a CPU, what *exactly* is the capacity that you actually get?

To recap, this is what vSphere uses for ESXi.



vSphere simply takes the base frequency x number of cores.

- It does not include turbo boost
- It does not include hyper threading.

The above is great for mission critical, where you need to be conservative and performance takes priority. For the rest of the workload, you can actually squeeze more. However, you need to set expectation as as the CPU speed depends on the model you buy.

I recommend you optimize the above answer. You can get more while keeping the trade off low. How?

Let's answer with a simple example. You have 2 ESXi servers:

ESXi 01	ESXi 02
Cores: 20	Cores: 10
Threads: 40	Threads: 20
Speed: • Nominal: 1 GHz • All Cores Turbo: 1.25 GHz	Speed: • Nominal: 2 GHz • All Cores Turbo: 2.5 GHz

Using the model provided by vCenter, what's the total capacity of each server?

Answer:

- ESXi 1 capacity = 20 cores x 1 GHz = 20 GHz.
- ESXi 2 capacity = 10 cores x 2 GHz = 20 GHz.

The above is a good answer, but can we improve it?

On ESXi 2, VM will run 2x faster, but you can only run half as many VMs. If you run the same number of vCPU as you do on ESXi 1, the VMs on ESXi 2 will compete and incur ~50% CPU Ready time. Workload performance likely becomes unpredictable. CPU context switch will be very high.

That means ESXi 2 has 2x the performance, but 0.5x the capacity. The 200% performance only happens when you run at 50% capacity of ESXi 1. When you load ESXi 2 with 1x the capacity of ESXi 1, its performance could drop below 1x of ESXi 1.

The above shows the imperfect correlation between performance and capacity. This is why you cannot use a single number to measure both. Capacity should not include “the speed of the run”.

	ESXi 01	ESXi 02
Capacity (The Space)	40 threads	20 threads

Not what you expect?

Okay, let’s dive in.

The CPU capacity is in thread, not in Hertz.

Capacity does not consider performance or speed. It simply looks at the part of the CPU where a VM can run. Since a thread can run in parallel with partner thread in a core, it is as simple as counting the physical threads.

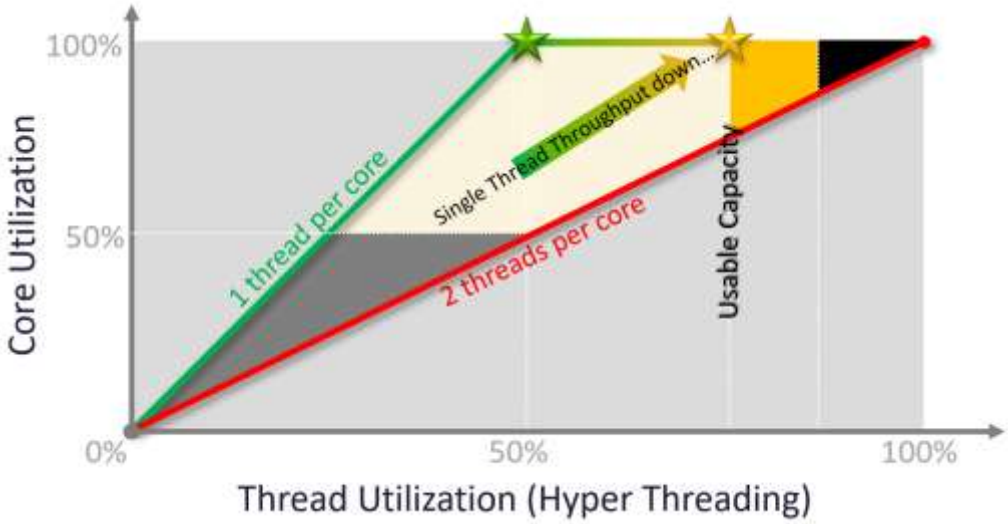
ESXi 01 can run 40 vCPU worth of VMs concurrently. By that definition, that means you do not overcommit when you run 40 vCPU, if we set aside hypervisor overhead for now. This is true as the VMs do not experience CPU Ready. Sure, they will run slower but that’s a performance, and not capacity question. The effect would be the same as having a slower hardware. Capacity is not performance. Think of capacity as space, while performance as speed.

Using highway analogy, the number of lanes is fixed, but the allowed speed typically vary depends on the segment of the highway.

BTW, this is consistent with AWS. It counts the threads, not physical core. AWS market it as no overcommit. Yes, they use allocation model and not utilization model.

Metric	Allocation Model	Demand Model
Total Capacity	Total physical threads in the box	Core utilization and thread utilization. Do not use CPU Cycles (GHz).
Hypervisor Overhead	No of physical threads you manually assigned	Not applicable, as it’s included in total ESXi counters
Consumption	Sum of all running VM vCPU	Core utilization and thread utilization. Usage (GHz) tends to over report.
	Performance is not applicable.	Ready + CoStop. Swap Wait and Other Wait are not CPU related.

Utilization



All cores are fully utilized. Before this point, you have idle core.
Aim to utilize all cores. Use the 2nd thread as buffer.

Drive utilization towards all core utilized and most threads utilized.
The throughput per vCPU goes down as more threads and cores are running.

Your Target. 100% core utilized, and you hit the Usable Capacity that you set.

Ideal combination. No core is running >1 thread.
Until Core Utilization = 50%, you want to be in this line.

Worst combination. Each cores run 2 threads.
Each vCPU gets 37.5% hyperthreading throughput reduction.

Any point in here means ≥ 1 cores are running 2 threads, yet ≥ 1 core is idle.
Imbalance. Performance issue, not Capacity issue.

This combination does not exist.
A thread has to run on a core. 3 threads can't run on a core.

Buffer for Capacity or Disaster Recovery (for cluster participating in DR site).
Occasionally it's expected to be here due to momentary spike. Ensure contention is low.

Reserved for Cluster HA.
Do not cross this else availability issue creates performance issue.

Half the cores are not even running.
Big wastage. Each VM becomes more expensive as a result.

Hyper Threading

What should we do with HT?

I recommend enabling it, but set your customers expectation on the CPU speed.

Note that HT technology may change in the future. New Intel Xeon no longer has HT, but uses small core and big core instead. Future Intel may bring it back. AMD still use it.

CPU Cycles

Do not express an ESXi capacity in MHz, as the total "capacity" becomes volatile.

- If you enable hyper threading, the total capacity only goes up by 1.25x. However, the speed reduction experienced by VM is significant. It's 37.5% slower.
- All Cores Turbo brings up the total capacity. This number varies per CPU model.

The usage of GHz as the unit complicates calculation as it's mixing performance and capacity.

Consumption

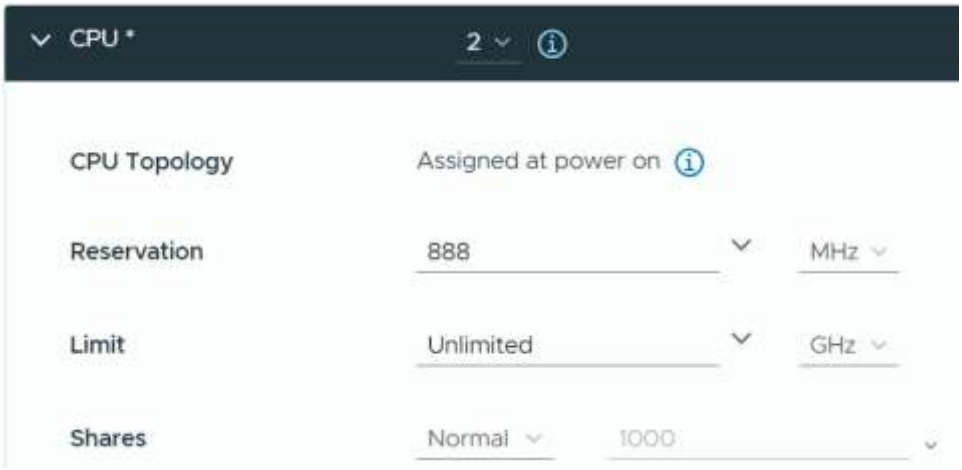
For allocation-based model, the consumption is simply the configured vCPU for all the running VM.

For demand-based model, the consumption is the maximum of CPU Usage and CPU Reservation for all the running VMs.

Do not include VM CPU contention, but make sure performance is tracked explicitly.

Reservation

Take note that allocation is done in vCPU, but reservation is done in Hertz. When you vMotion a VM to faster ESXi, let me know if the reservation also increases accordingly. If not, you need to adjust manually.



Hypervisor

In the planning stage, we need a single number for usable capacity.

In the monitoring stage, we should be mindful that our estimate may be too aggressive or conservative. This is why tracking contention is paramount.

Recommended Value

What number do I recommend?

Based on the profiling documented in the kernel section later on, I'd use the following at 2.5 GHz clock speed:

- 12 threads if you use NSX and vSAN.
- 4 threads if you use ESXi only.

NSX EDP adds 2-4 cores as it regularly polls the network card.

vSphere Replication and HCX need to be sized separately.

vSAN File Services needs 2 vCPU as it's a VM. Set reservation.

CPU Metrics

I've given recommendations on the number to provide as part of the planning process. Now let's dive into how the numbers are derived.

The following screenshot shows the CPU counter names used by vSphere Client UI. What do you notice?

Counters	Rollups	Units	Description
Resource CPU active (1 min average)	Latest	%	CPU active average over 1 minute of the system resource group
Resource CPU active (5 min average)	Latest	%	CPU active average over 5 minutes of the system resource group
Resource CPU allocation maximum (in MHz)	Latest	MHz	CPU allocation limit (in MHz) of the system resource group
Resource CPU allocation minimum (in MHz)	Latest	MHz	CPU allocation reservation (in MHz) of the system resource group
Resource CPU allocation shares	Latest	num	CPU allocation shares of the system resource group
Resource CPU maximum limited (1 min)	Latest	%	CPU maximum limited over 1 minute of the system resource group
Resource CPU maximum limited (5 min)	Latest	%	CPU maximum limited over 5 minutes of the system resource group
Resource CPU running (1 min. average)	Latest	%	CPU running average over 1 minute of the system resource group
Resource CPU running (5 min average)	Latest	%	CPU running average over 5 minutes of the system resource group
Resource CPU usage ({rollupType})	Average	MHz	Amount of CPU used by the Service Console and other applications

Yes, the roll up of the counter.

In general, when you take the latest value of something, you tend to get a much higher value than averaging the entire period.

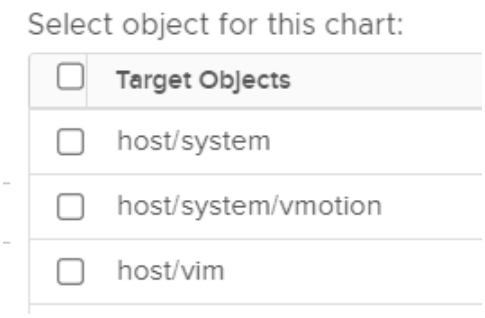
Utilization

There are 3 counters provided to track the actual utilization.

- Usage
- Running
- Active

Usage is what you should use as it has the 4 resource groups and their sub pools.

Running and Active counters only has these 3 objects, hence they are less useful. You lose host/user, host/opt so you won't get complete picture.



Plus, Active uses "latest" as its rollup.

If you still need to know about Active and Running, reach out to me and happy to share more details.

Usage

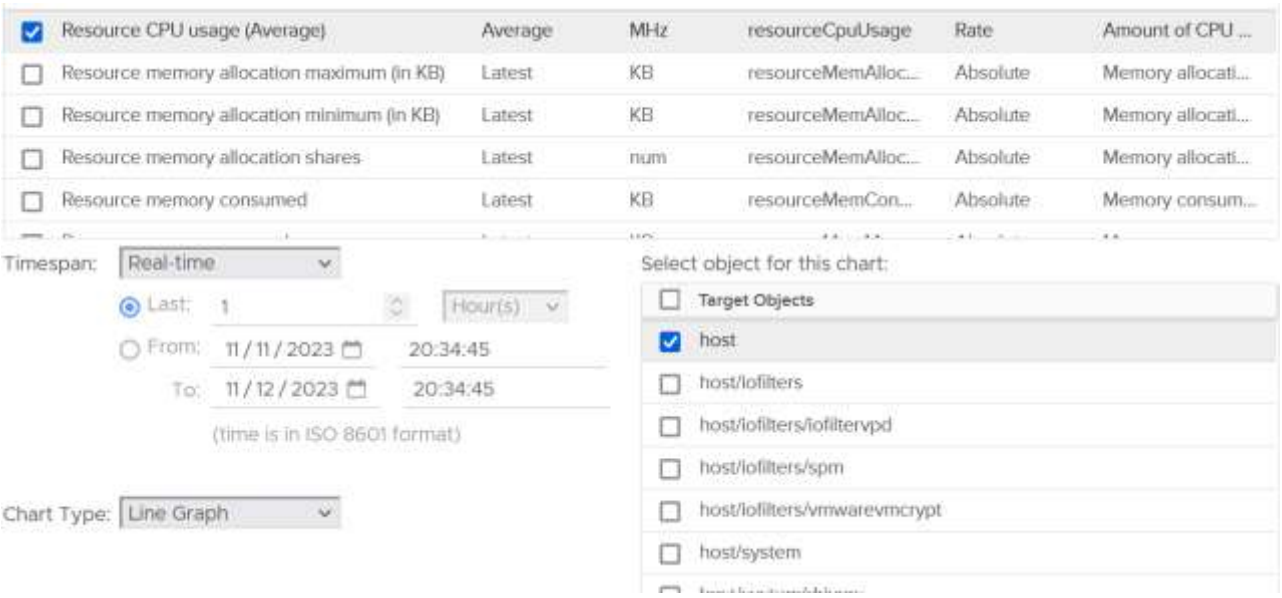
Now that we know which counters to use, what do you expect the values of the 4 groups?

Here is a sample from ~400 ESXi hosts, where I sort the top 7 from highest System usage.

Host	system ↓	vim	iofilters	user	Usage	Demand
29.29 GHz	3.66 GHz	0.13 GHz	0.0006 GHz	24.86 GHz	29.3 GHz	32.32 GHz
11.92 GHz	2.46 GHz	0.41 GHz	0.0007 GHz	8.9 GHz	11.92 GHz	16.84 GHz
9.28 GHz	2.36 GHz	0.42 GHz	0.0008 GHz	6.36 GHz	9.27 GHz	15.39 GHz
8.97 GHz	2.35 GHz	0.47 GHz	0.0009 GHz	5.93 GHz	8.97 GHz	14.91 GHz
9.46 GHz	2.3 GHz	0.4 GHz	0.0009 GHz	6.6 GHz	9.46 GHz	15.6 GHz
9.05 GHz	2.29 GHz	0.48 GHz	0.0009 GHz	6.2 GHz	9.05 GHz	15.15 GHz
10.48 GHz	2.28 GHz	0.41 GHz	0.0007 GHz	7.59 GHz	10.48 GHz	15.5 GHz
12.21 GHz	**0.69 GHz**	**0.32 GHz**	**0.0008 GHz**	**11.18 GHz**	**12.21 GHz**	**19.86 GHz**
91.65 GHz	**3.66 GHz**	**5.98 GHz**	**0.0031 GHz**	**91.22 GHz**	**91.65 GHz**	**95.53 GHz**

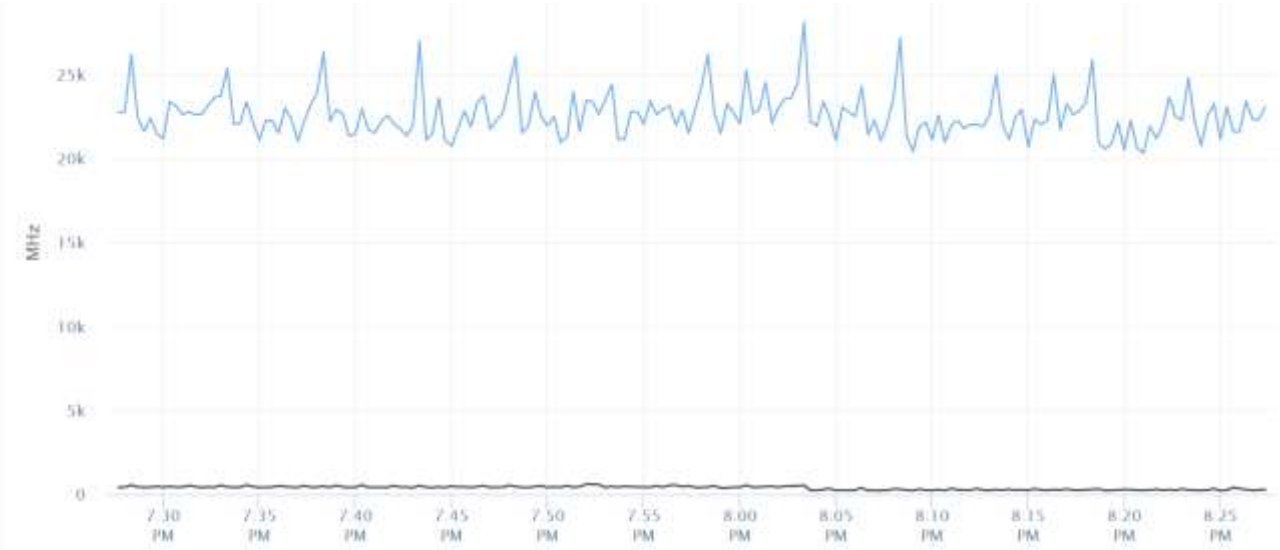
The bottom two rows show the summary. The first summary is the average among all the hosts, while the last row is the highest value.

Usage maps to the ESXi CPU Usage metrics under CPU group.



The value at host matches the value of CPU Usage. This means the metric CPU \ Usage (MHz) is the same with System \ Resource CPU Usage (Average) (MHz).

As the value contains VM metrics, the value is much higher than the kernel. You can see the host/system is far lower.



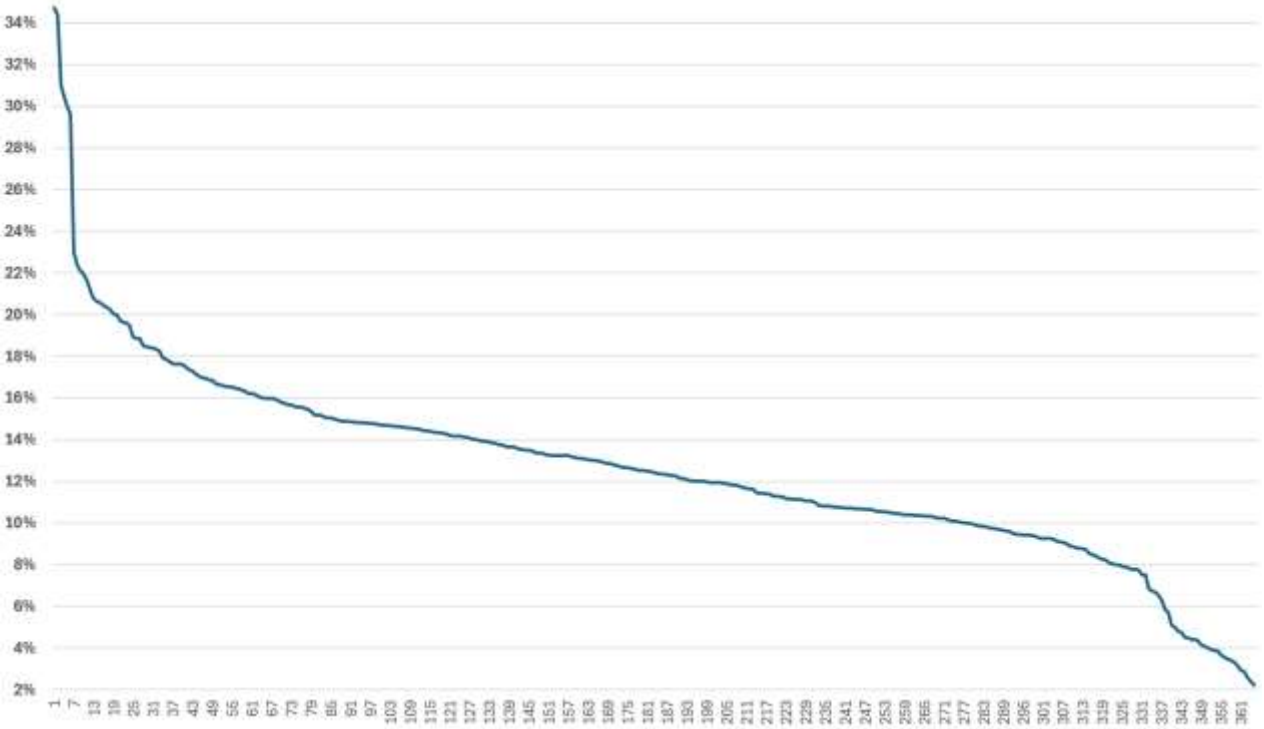
Performance Chart Legend

Key	Object	Measurement	Rollup	Units	Latest	Maximum	Minimum	Average
	host	Resource CPU usage (Average)	Average	MHz	23,097	28,129	20,331	22,611.729
	host/system	Resource CPU usage (Average)	Average	MHz	287	598	213	377.528

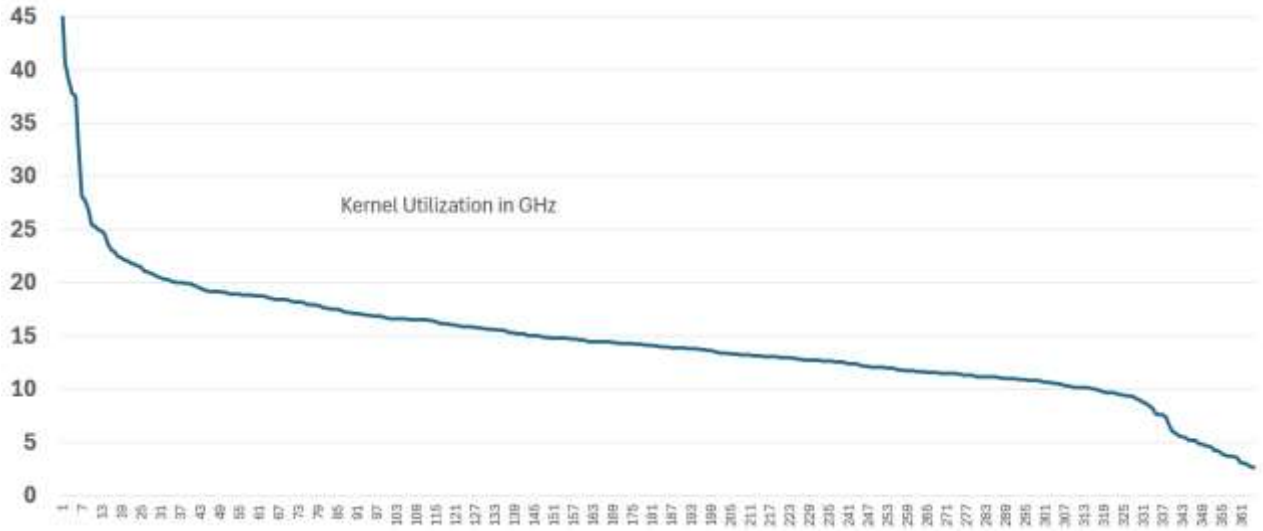
Real World Samples

I plotted 364 ESXi hosts running production workload. All of them are doing at least 100 GHz and are running vSAN and NSX. For vSAN, they are a mixed of OSA and ESA architecture.

The line below shows the kernel relative to the total CPU Usage.

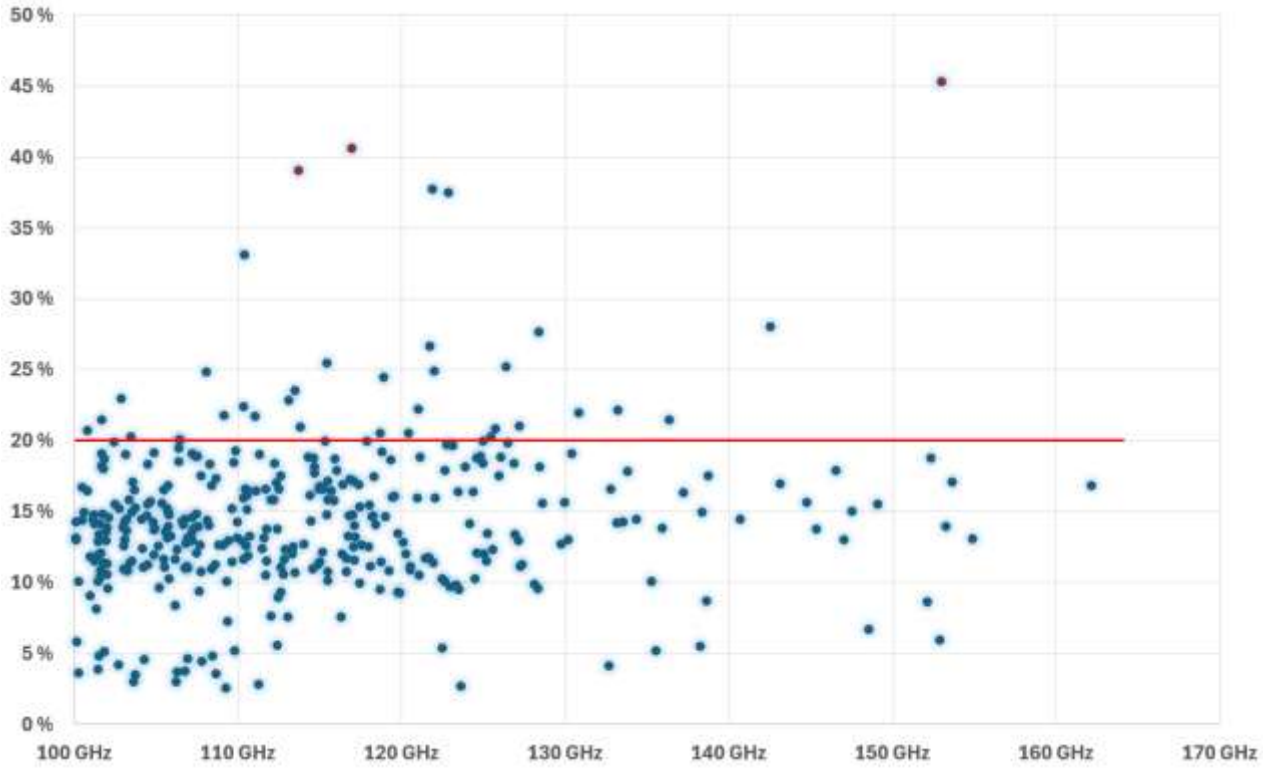


In terms of absolute utilization, the actual utilization has a wide range. This is despite all these ESXi were running at least 100 GHz.

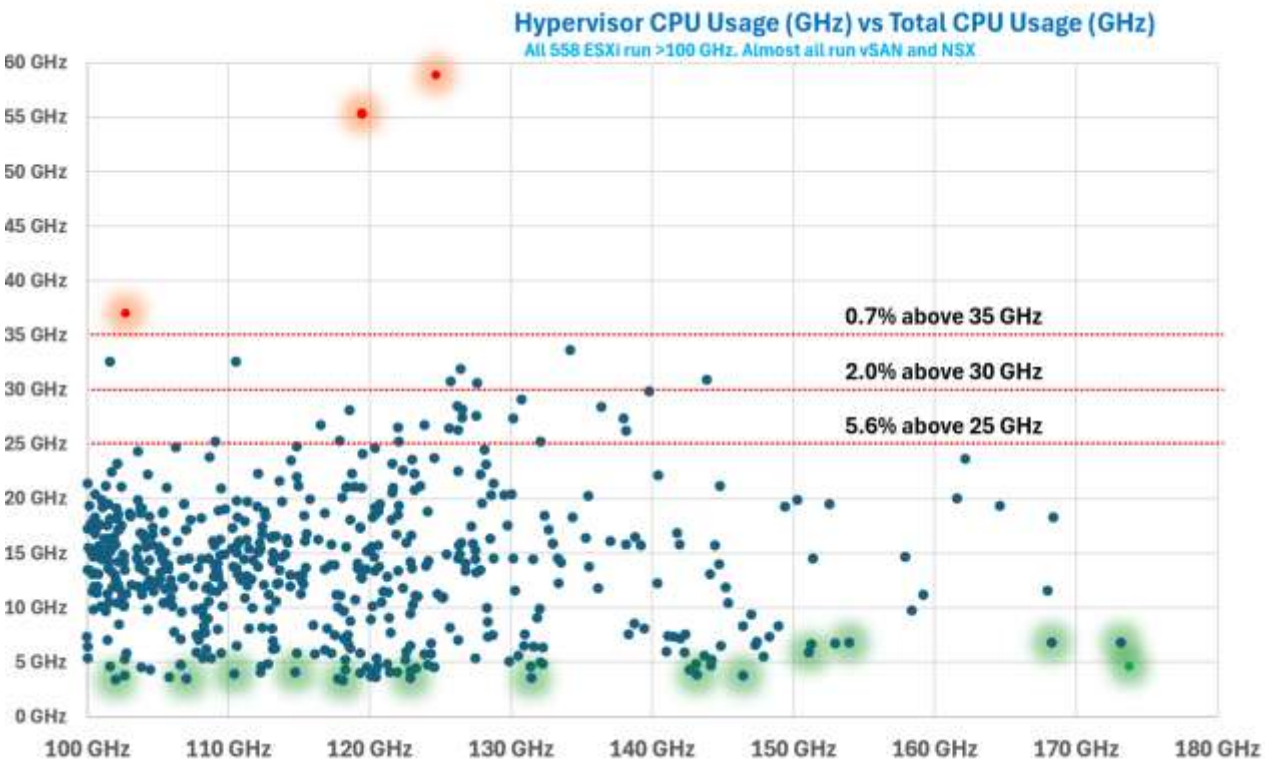


Take note there is no perfect correlation between kernel utilization and VM utilization. This is especially true when the kernel has NSX and vSAN. All these 364 ESXi were running vSAN (mixed of OSA and ESA) and NSX.

The following chart shows that a great majority were below 10%. There is no strong correlation between the relative overhead and the absolute overhead.



Another measurement, taken at a different time. This time there were 557 ESXi with CPU Usage > 100 GHz, with 2 of them clocking > 170 GHz.

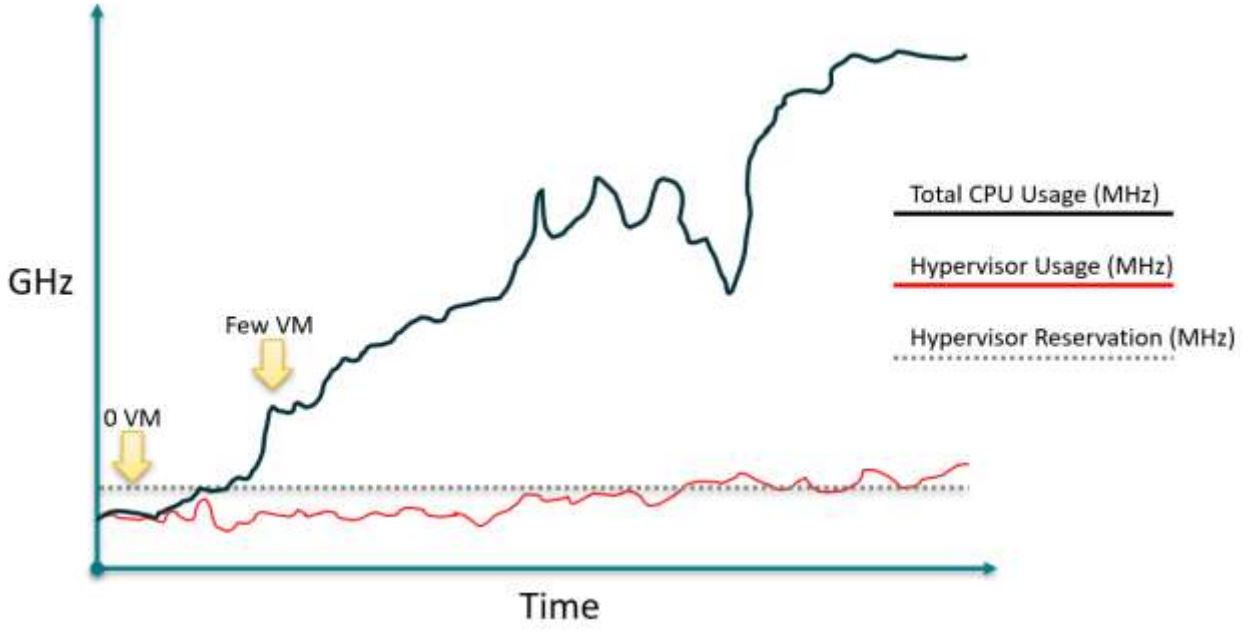


There were 2 outliers at > 40 GHz, highlighted in orange. The hypervisor overhead remains steady at 100 GHz vs > 150 GHz. I drew a red line at 25 GHz to show that majority of the numbers are below this.

Plot the values across all your ESXi hosts. If you take enough hosts, you will notice the values vary. The following chart shows 558 ESXi hosts. Almost all are running both vSAN and NSX. They are all running at least 100 GHz. What do you notice?

Yes, there is hardly any correlation between total CPU Usage and hypervisor CPU Usage.

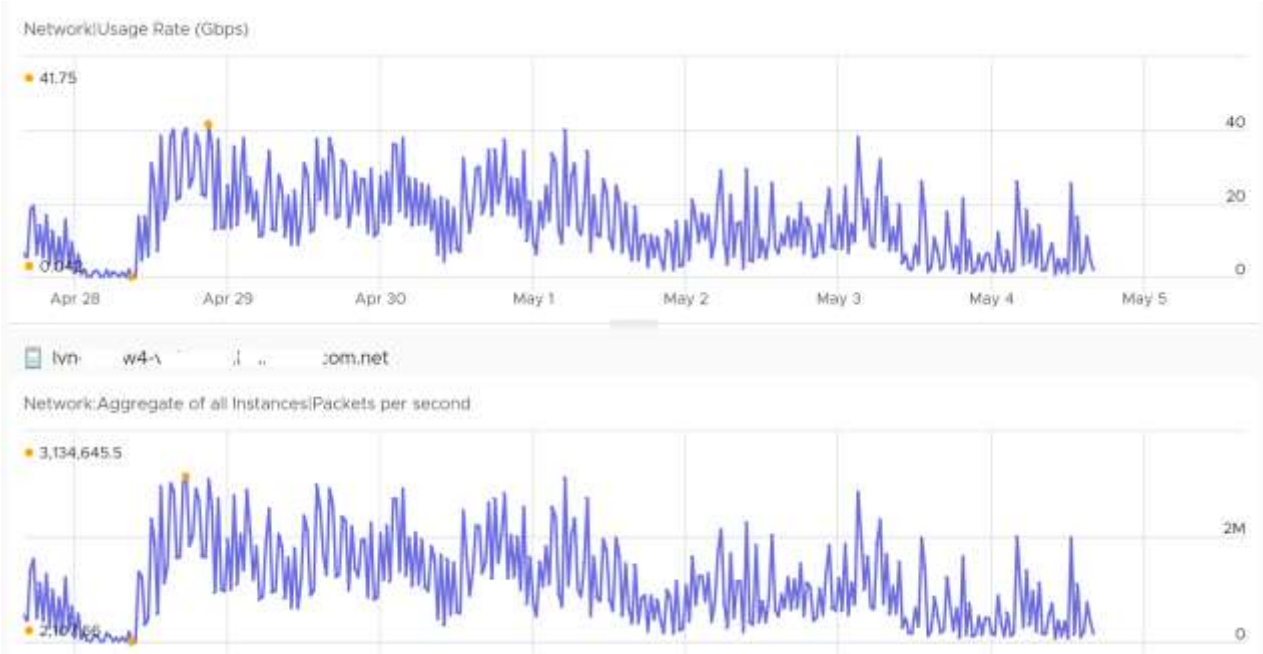
I drew the following illustration to show the lack of predictable relationship between hypervisor CPU reservation, hypervisor CPU usage and total CPU usage.



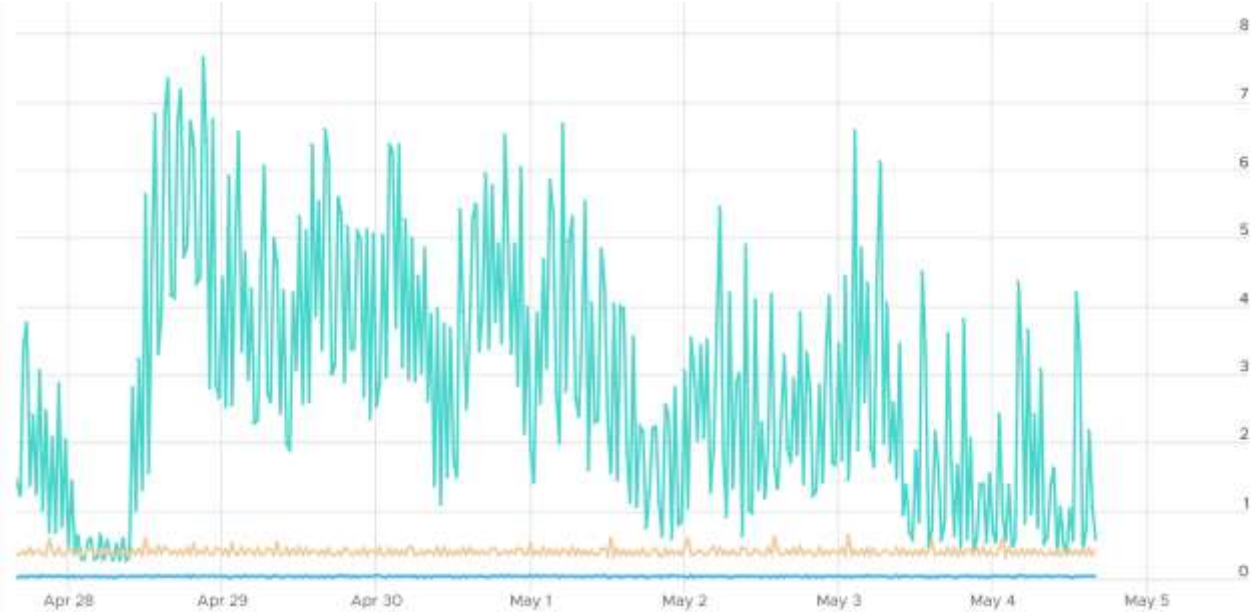
Network Impact

What's the kernel overhead to do network packet processing?

The following ESXi was doing > 40 Gigabit per second multiple times. It was processing > 3 million packets.



Hardly any impact on the kernel. The kernel was less than 8 GHz.



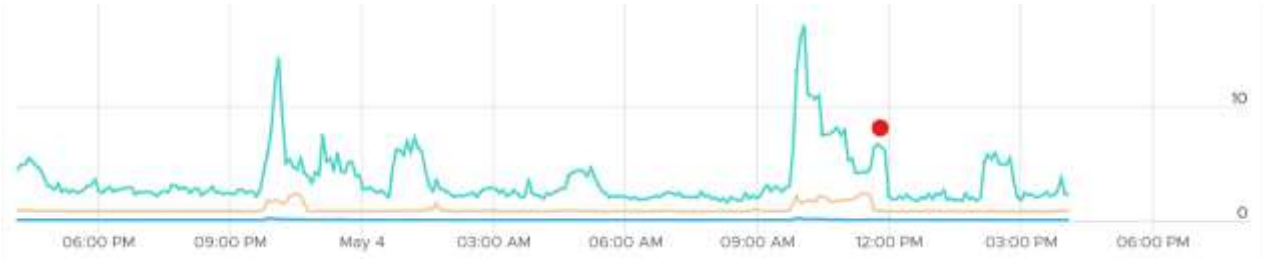
Storage Impact

Storage IO processing can require more kernel if the IOPS and throughput are high. The following ESXi hit > 200K IOPS two times.



You can see a corresponding spike in the kernel. It went above 10 GHz.

The red dot is because of network.



Reservation

Utilization is relatively more volatile or dynamic, while reservation is logically more stable. The following screenshot shows CPU Usage fluctuates every 20 seconds, while reservation remains perfectly constant. Expect Usage to be higher reservation at high utilization.



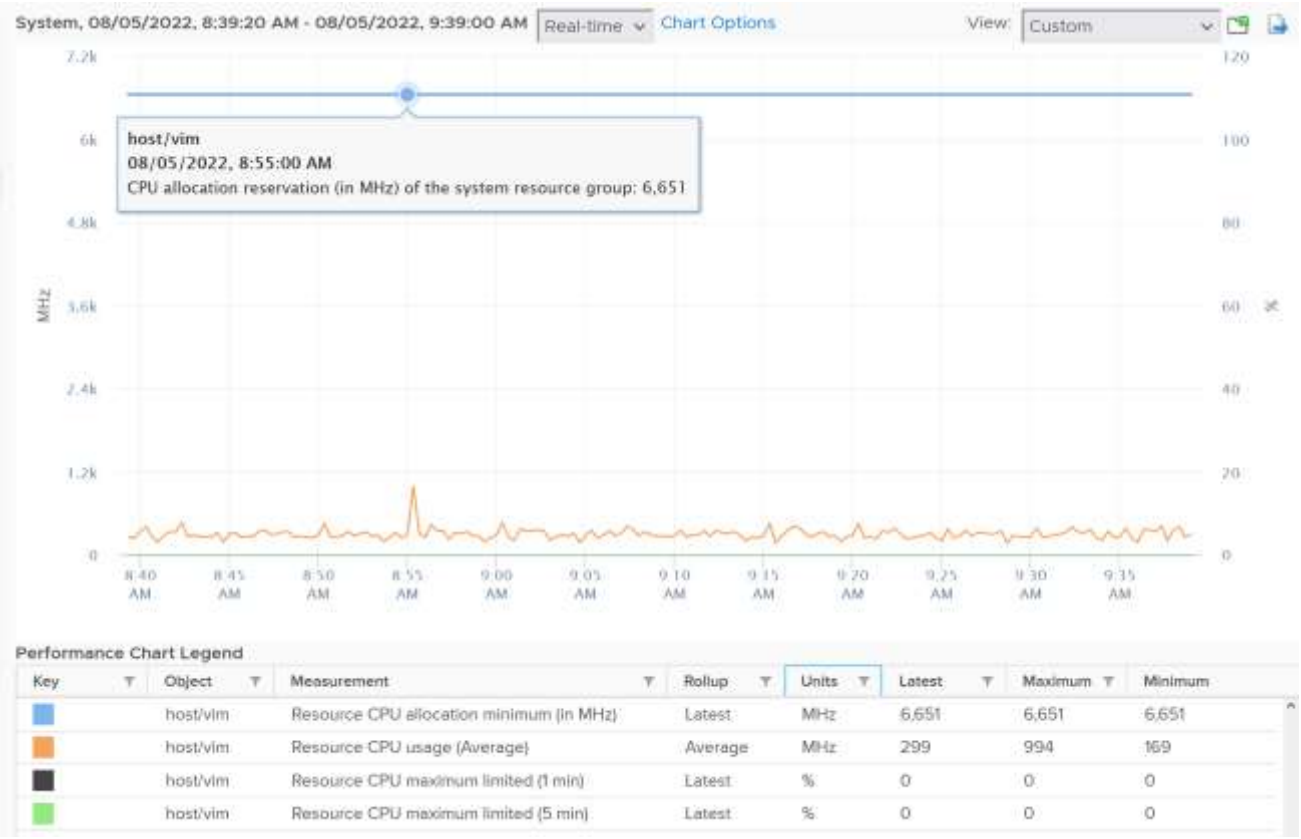
Performance Chart Legend

Key	Object	Measurement	Rollup	Units	Latest	Maximum	Minimum
	host/system	Resource CPU usage (Average)	Average	MHz	213	294	115
	host/system	Resource CPU allocation minimum (in MHz)	Latest	MHz	220	220	220
	host/system	Resource CPU maximum limited (1 min)	Latest	%	0	0	0
	host/system	Resource CPU maximum limited (5 min)	Latest	%	0	0	0

Notice the maximum limited value is perfectly flat. That's what you want as kernel processes should not have a limit.

The above is for host/system. The reservation is surprisingly low.

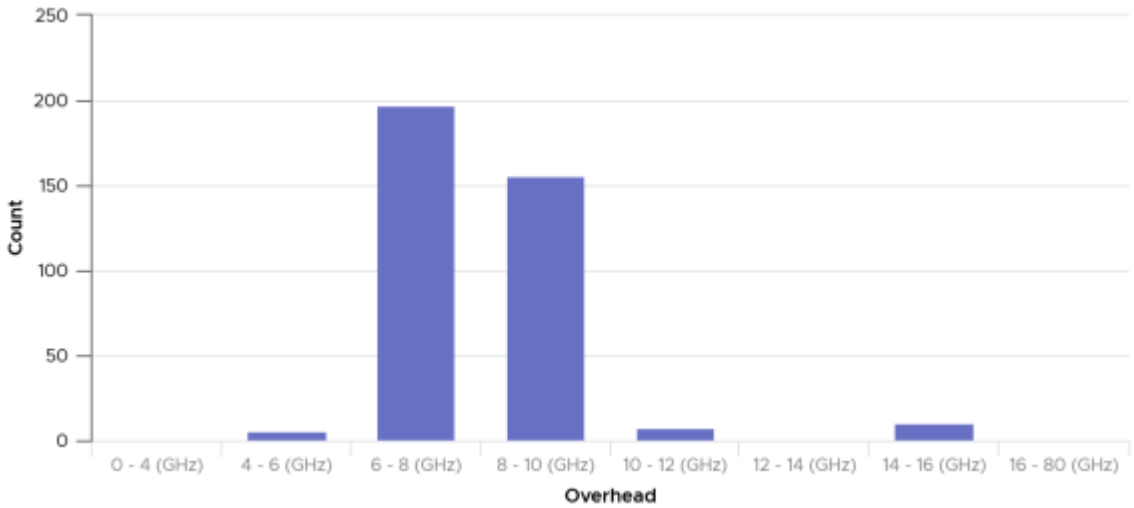
Now let's look at host/vim. What do you notice from the following screenshot?



Surprisingly the reservation is not low. It's around 6.6 GHz.

Real World Samples

The above is from 1 ESXi. We need to plot for many to get a better understanding. The following diagram shows the distribution of the kernel overhead based on a sample of almost 400 ESXi in production environment.



By far the majority of the values lie in 6 – 10 GHz.

Their values tend to be stable over days, although from time to time I see fluctuating metrics, which is reasonable as there are multiple factors impacting the reservation.

The following chart shows both the fluctuating pattern and steady pattern (most common). They are from 2 ESXi hosts.



Memory

Memory is simpler than CPU as there is only "space" dimension. There is no "speed".

Memory is more complex than CPU as Guest OS and VM are 2 different realms. None is perfect as an input.

Capacity Metrics

There are

Metric	Allocation Model	Demand Model
Total Capacity	Total physical memory in the box. This is the same for either model	
Hypervisor Overhead	No of GB you manually assigned	Not applicable, as it's included in total ESXi counters
Consumption	Sum of all running VM configured RAM.	ESXi Consumed
	Performance is not applicable.	ESXi Swapped + Zipped + Guest OS Ballooned.

Hypervisor Overhead

In the planning stage, we need a single number for usable capacity.

In the monitoring stage, we should be mindful that our estimate may be too aggressive or conservative. This is why tracking contention is paramount.

What number do I recommend?

Based on the profiling documented in the kernel section later on, I'd say:

- 64 GB if you use NSX and vSAN
- ~20 GB if you use ESXi only. I don't have real world numbers to back this up as the environment I have is NSX and vSAN.

Demand Metric

Unlike allocation, demand is tricky as different layers in virtualization has their own perspective. ESXi applies multiple memory management techniques, which makes it harder to determine the total demand:

- TPS results in less actual usage.
- Balloon means ESXi is under memory pressure, or the VM hit a limit.
- Compress means the pages are still in DIMM, albeit occupying less space. How much less depends on the zipped result and if the remaining page is fully used or not.
- Swapped and compressed share the same input. When a page cannot be compressed, it got swapped.
- Host cache.
- Memory tiering such as Intel Optane.
- We exclude VM overhead as it's negligible.

Because of the above, it is better not to mix metrics from Guest OS and VM.

```
ESXi Demand = Kernel Consumed + Sum of (Running VM Demand)
     where VM Demand = Min (Limit, Consumed + Ballooned + Zipped + Swapped)
```

Limitation of VM counters:

- Consumed metric is mostly inactive pages. So adding ballooned, zipped, swapped will make it even more conservative.
- The Guest OS counter is more accurate as it's closer to application. It tends to be smaller. However, Guest OS is unaware of ESXi memory management techniques.

Hypervisor Metrics

The following screenshot shows the counter names used by vSphere Client UI

Resource memory allocation maximum (in KB)	KB	Memory allocation limit (in KB) of the system resource group
Resource memory allocation minimum (in KB)	KB	Memory allocation reservation (in KB) of the system resource group
Resource memory allocation shares	num	Memory allocation shares of the system resource group
Resource memory consumed	KB	Memory consumed by the system resource group
Resource memory mapped	KB	Memory mapped by the system resource group
Resource memory overhead	KB	Overhead memory consumed by the system resource group
Resource memory share saved	KB	Memory saved due to sharing by the system resource group
Resource memory shared	KB	Memory shared by the system resource group
Resource memory swapped	KB	Memory swapped out by the system resource group
Resource memory touched	KB	Memory touched by the system resource group
Resource memory zero	KB	Zero filled memory used by the system resource group

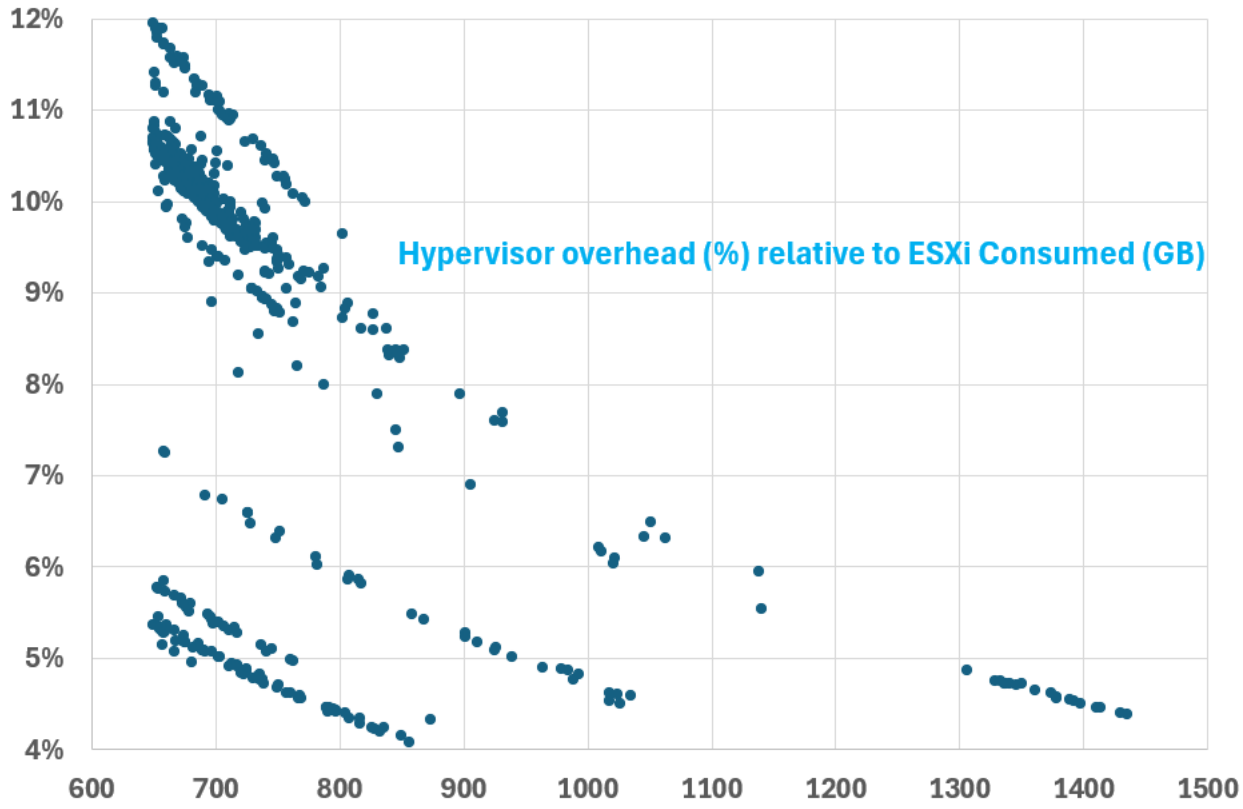
Unlike CPU, the Rollups column values are all Latest. This makes sense as memory is measure storage space. You want to know the last value, not the average over collection period.

The Stat Types column values are all Absolute.

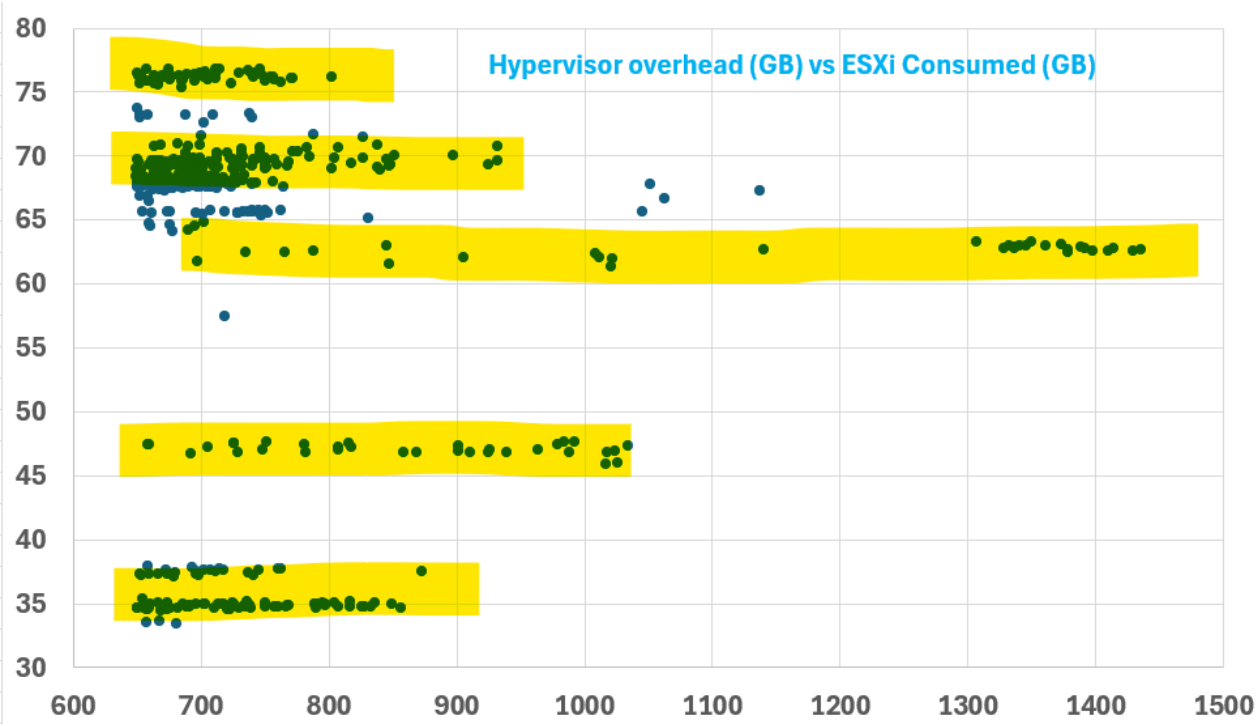
Allocation maximum	As per CPU, this is the limit.
Allocation minimum	As per CPU, this is reservation.
Shares	Relative shares of each the kernel world. This is the kernel internal metric, not something vSphere Administrator should change
Consumed	The actual consumption. Just like CPU, this can be lower than the reservation. The host/vim world has no reservation.
Mapped	I'm unsure what mapped means. Regardless, there seems to be no use case for customer operations. The rest of the metrics are fairly similar with the associated metric at VM and ESXi level.
Overhead	
Share Saved	
Shared	
Swapped	
Touched	
Zero	The entire block contains just a series of 0.

Utilization

I plotted 607 production ESXi running vSAN and NSX. The hosts have Consumed memory between 650 GB and 1450 GB. As expected, the kernel overhead decreases relatively as total memory grow.



The number dropped to well below 10% once Consumed passed 800 GB. This means that the absolute amount plateau at a certain level. We can validate that by plotting the absolute utilization.



Interestingly, there are levels. From the preceding chart, you can see there are 5 groups of similar number range. I think it's because of vSAN configuration.

Reservation

The metric name is Memory \ ESX System Usage (KB).

It is a raw counter from vCenter. Just in case you're wondering, the name ESX System Usage is a legacy name.



Performance Chart Legend

Key	Object	Measurement	Rollup	Units	Latest	Maximum	Minimum
	host/system	Resource memory allocation minimum (in KB)	Latest	KB	26,510,500	26,514,472	26,481,736
	host/system	Resource memory consumed	Latest	KB	22,263,120	22,266,196	22,227,640
	host/vim	Resource memory allocation minimum (in KB)	Latest	KB	0	0	0
	host/vim	Resource memory consumed	Latest	KB	703,344	715,136	694,828

The following is an ESXi 6.7 U3 host with 1.5 TB of memory. Notice the kernel values remains constant over a long period. The number of running VM eventually dropped to 0. While the Granted counter drops to 1.5 GB (not sure what it is since there is no running VM), the kernel did not drop. This makes sense as they are reservation and not the actual usage.



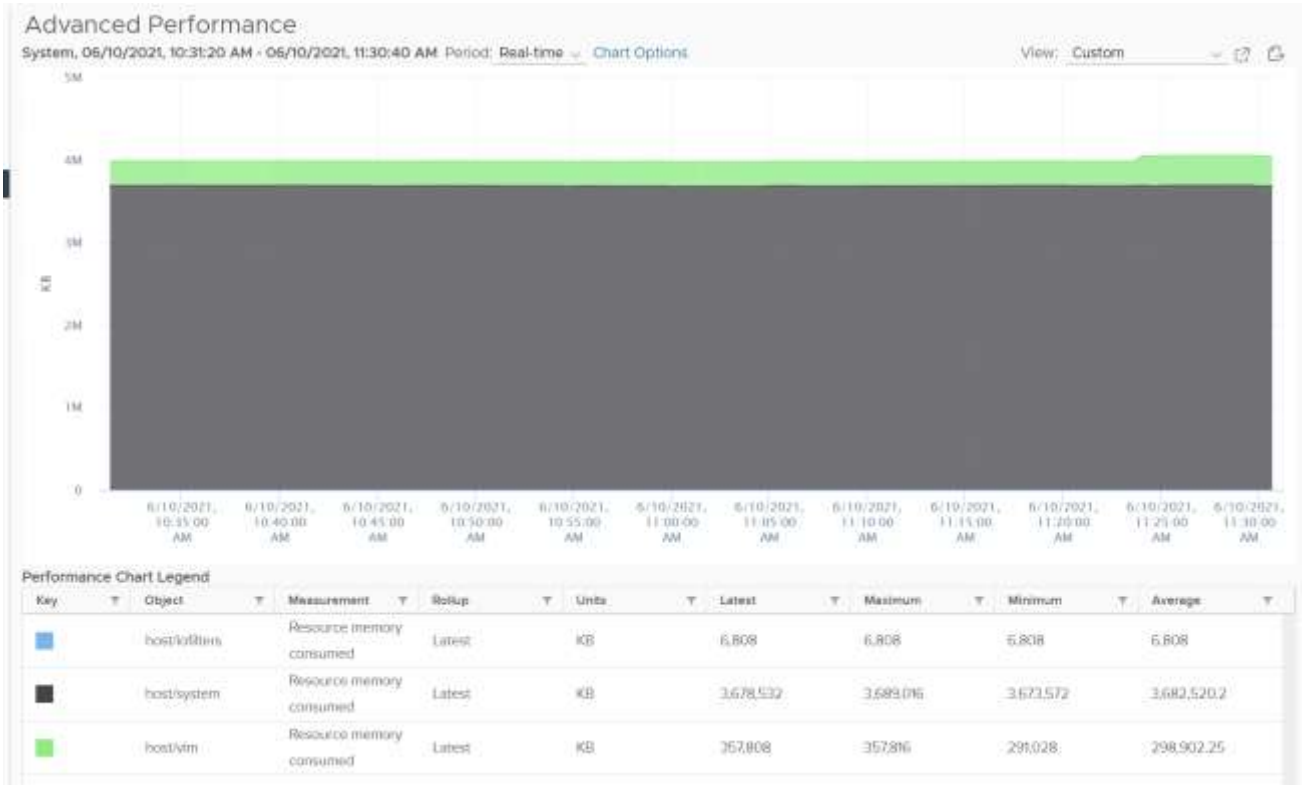
Based on a sample of 500+ ESXi hosts, the range varies from 6 GB to 88 GB. In an ultra large ESXi with 12 TB RAM running vSAN and NSX, the reservation went up to 300 GB.

Utilization vs Reservation

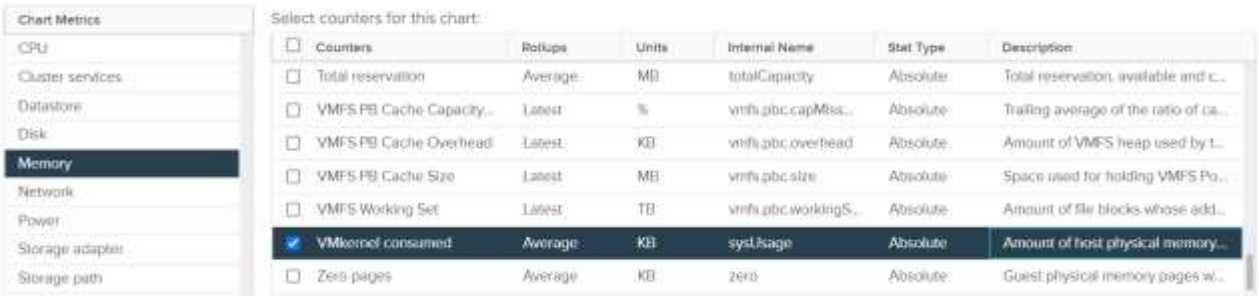
Logically, utilization does not always correspond to the reserved amount. The following chart shows the reservation remains steady when the utilization drops by 90%, from 40 GB to single digit.



To see the actual usage, choose the metric Resource Memory Consumed metric from vSphere Client. Stack them, and you see something like this. The system part typically dwarfs the other 2 resources.



Do not take the value from Memory \ VMkernel consumed counter. That's only the system resource. You can verify by plotting this and compare against host/system resource. You will get identical charts.



This value is for vSphere kernel modules. It does not include vSAN.

Storage

Used > Allocated

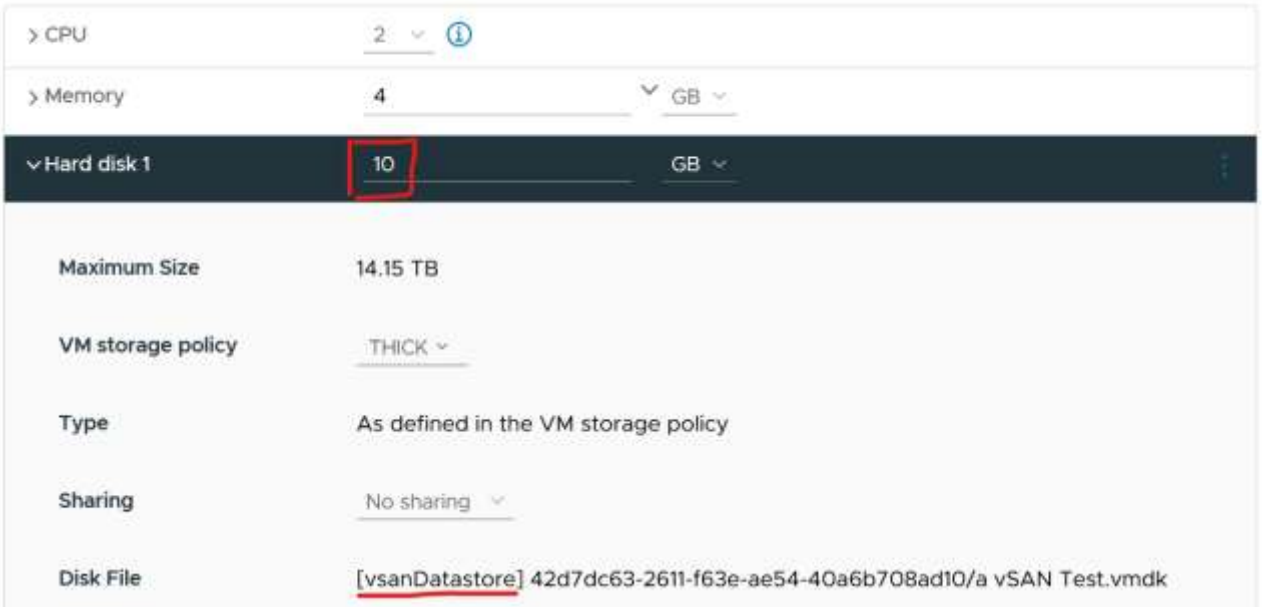
Can you use more than what you're allocated?

That sounds illogical, doesn't it?

Well, it can happen when "other consumption" comes into play.

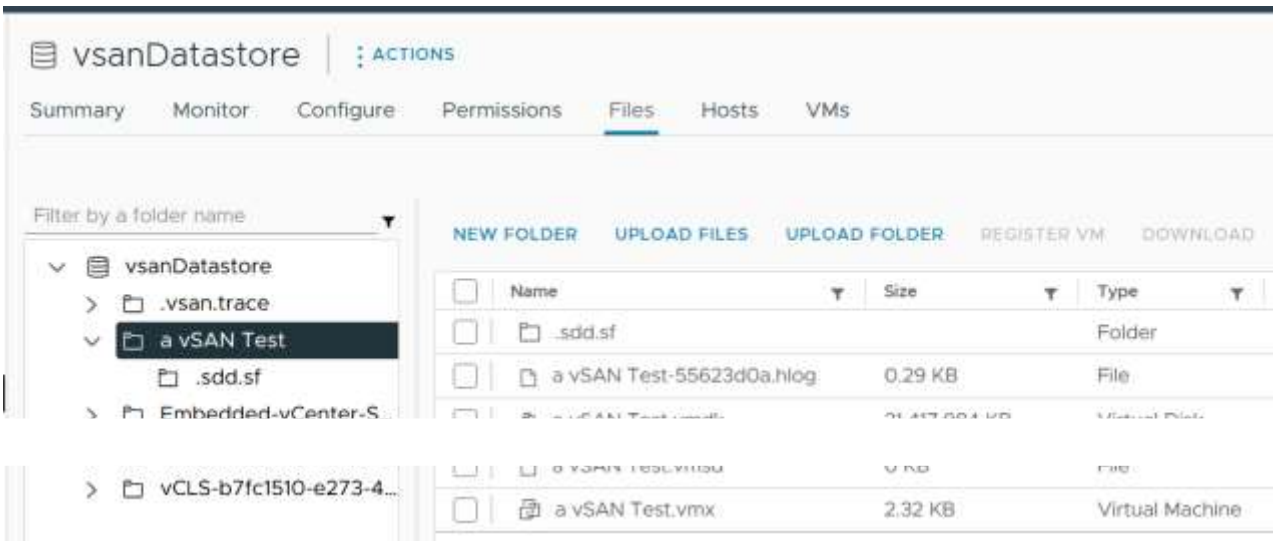
For example, software-defined storage such as vSAN delivers the protection at software layer, not hardware layer.

The following screenshot shows a VM configured with 10 GB hard disk. That means the guest OS is allocated with 10 GB.



It's thick provisioned as specified by vSAN policy.

Guess how much disk space it actually consumes at the VMFS layer?



You're right. It is 20 GB.

Implementation

Aria Operations metrics

Memory \ Total Capacity (KB)	The capacity as seen by the kernel, which is essentially the physical size.
Memory \ Utilization (KB)	Sum of demand from all running VM (see below) + ESXi kernel reservation. Demand is the maximum of VM reservation and Guest OS needed memory + total page-in in the collection cycle (default is 5 minutes). Page in = page in rate x memory block size. If Guest OS is missing, it falls back to consumed. The amount also includes the VM memory overhead.
Memory \ Workload (%)	Utilization / Total Capacity. Likely this is usable.
Memory \ Memory Allocated on all Powered On Consumers	Sum of all running VM configured memory. This is used in allocation model.

At the vSphere Cluster level, here are the metrics:

Cluster Configuration \vSphere HA \ HA Memory Failover (%)	Cluster HA failover for memory.
Memory|Demand|Usable Capacity after HA and Buffer (GB)	Total Capacity minus HA above and buffer (not shown as property)
Memory|ESX System Usage (GB)	Kernel reservation
Memory \ Utilization (KB)	Sum of all ESXi

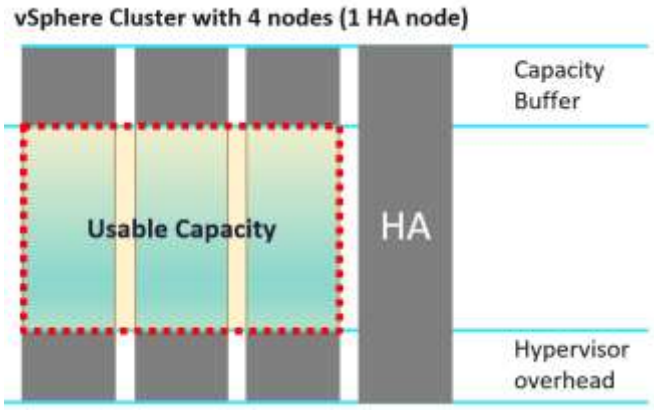
Memory|Demand|Workload (%)	Utilization / Usable
Memory \ Memory Allocated on all Powered On Consumers	Sum of all ESXi
Memory|Workload (%)	Normalized average of all ESXi?

Cluster Capacity

Cluster capacity is more complex than ESXi capacity due to the following cluster-level property

Total Capacity	Unlike ESXi, this could be dynamic due to reasons such as maintenance mode and DPM. Hybrid cloud such as VM sports on-demand host that is added dynamically. Dynamic cluster size increases complexity significantly. As a best practice, avoid removing hosts from the cluster if the cluster has < 5 ESXi hosts as your availability overhead becomes higher.
Buffer	For most cases, this is 10% for CPU and 0% memory. For stretched cluster, this is 50% for CPU and memory. For DR, this depends on the DR workload.
HA	This impacts usable capacity. For example, if it's 9+1, then cluster average utilization at 100% means each host is averaging 90%.
Stretched Cluster	The 2 sites have their own capacity calculation, yet they impact each other.
Host-VM Affinity	The group of hosts have their own capacity, operating like a subcluster.
Resource Pool	Each pool has their own capacity.
DR	A cluster may participate in disaster recovery by providing destination during DR dry run and actual. This is why you need to specify buffer, so that usable capacity reflect this rarely happens workload. BTW, the buffer default value is 0% in VCF Operations.

Total vs Usable



Let's take an example.

Assuming 10 hosts in a cluster, with N+1 HA setting, and Buffer is set to 0%.

Usable Capacity is 9 hosts, so 9 is the 100% operationally.

From here, if a host is out, the calculation depends on what actually caused it. There are 3 different scenarios:

	Intentional?	Desired?	Impact
vSphere DPM	Yes	Yes	Total Capacity
Maintenance Mode	Yes	No	Usable Capacity
HA happen	No	No!	Usable Capacity

Intentional means it's something you knowingly execute. In the case of vSphere DPM, it's also something you want to happen. In the case of Maintenance Mode, you intentionally do it but it's not something you want. So the 2 have different impact. vSphere DPM does not impact your HA as you still want HA even though you take out host(s). The length of DPM can be as long as there is no request for extra host. The length of maintenance mode should be as short as possible, hence the name maintenance.

HA events is an outage. It is obviously not something desired.

Undesired event impacts usable capacity and not total capacity.

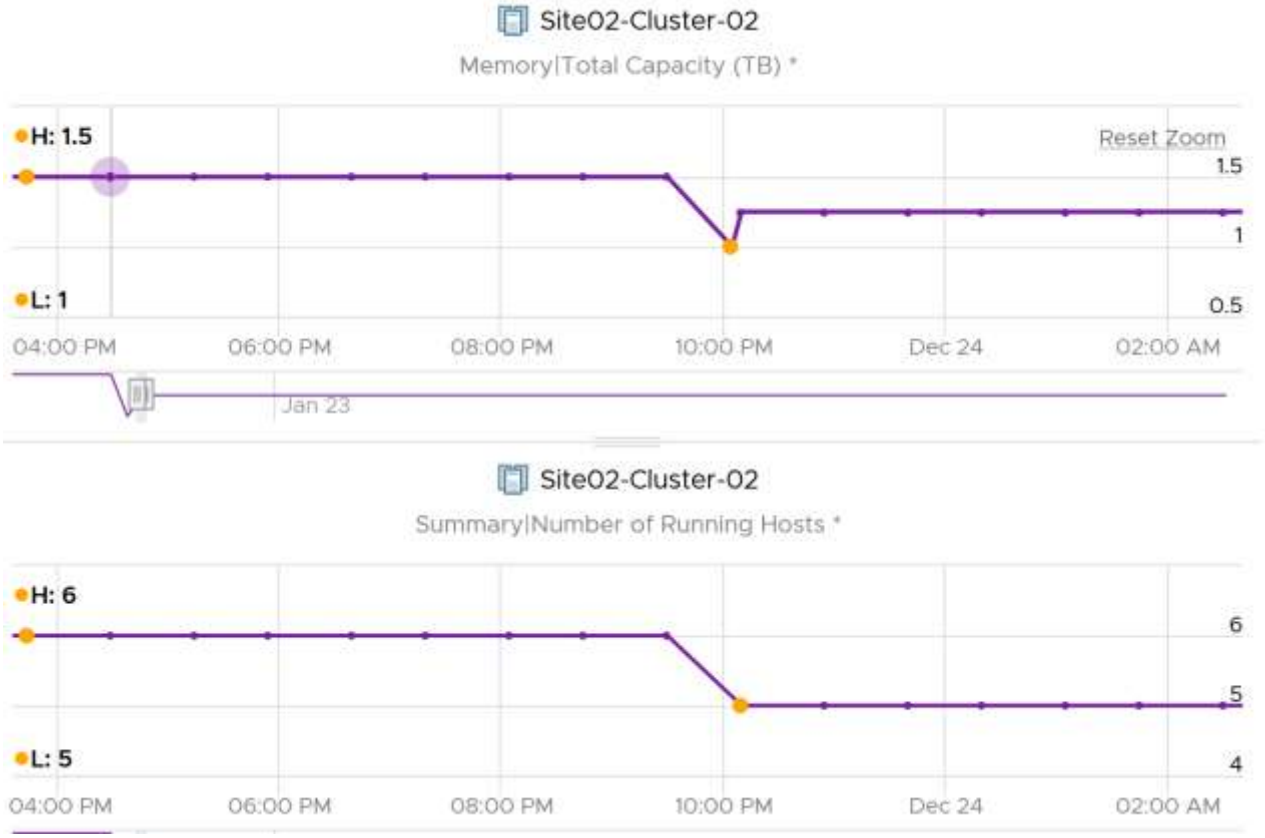
	DPM Event	Maintenance Mode	HA Event
Total Capacity	9	10	10
Usable Capacity	8	9	
Actual Availability	9/9 = 100%	9 / 10 = 90%	
Operational Availability	9 / 8 = 100% (capped)	9 / 9 = 100%	

The actual availability drops to reflect reality. The operational availability remains at 100% due to N+1 HA design.

For completeness, let's follow with a 2nd host out:

	DPM Event	Maintenance Mode	HA Event
Total Capacity	8	10	10
Usable Capacity	7	8	
Actual Availability	8 / 8 = 100%	8 / 10 = 80%	
Operational Availability	8 / 7 = 100%	8 / 9 = 89%	

BTW, the metric Total Capacity only counts those ESXi hosts that are connected to vCenter. If a host is connection state = disconnected, its value becomes blank, so the Total Capacity is affected.



Other Metrics

Cluster Risk (%)

As you want to minimize being reactive, you need a leading indicator. The risk (%) metric quantifies the potential capacity issue and performance issue.

	Metrics	Green	Yellow	Orange	Red
Utilization	Highest ESXi CPU Thread Utilization	0 - 60	60 - 80	80 - 90	90 - 95
	Highest ESXi Memory Ballooned (%)	0 – 1	1 – 1.25	1.25 - 2	2 – 4X
	Highest ESXi CPU Imbalance	0 – 80	80 - 90	90 - 95	95 - 100
	Highest ESXi Network Throughput (Gbps)	0 – 1%	1 – 2	2 – 4	4 – 8%
	Highest ESXi Disk IOPS	0 – 25K	25K – 50K	50K – 100K	100K – 150K
Allocation	Net CPU overcommit ratio	0 – 1	1 – 2	2 – 3	3 - 4
	Net Memory overcommit ratio	0 – 1	1 – 1.25x	1.25 – 1.5x	1.5 – 2x
Monster VM	Ratio of CPU Monster VM / ESXi Hosts	0 – 1x	1 – 2x	2 – 4x	4 – 8x
	Ratio of Memory Monster VM / ESXi Hosts	0 – 1x	1 – 2x	2 – 4x	4 – 8x
Cluster	Inter-hosts CPU Imbalance	> 0	1	> 1	≥ 100
	Percentage of vMotion	0 – 1%	1 – 2	2 – 4	4 – 8%

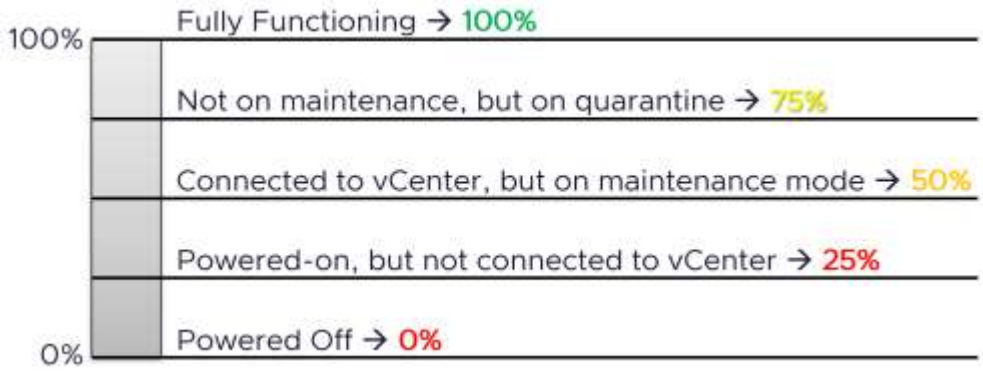
For utilization, it uses the highest among the ESXi hosts in the cluster. This complements the average value, which is used to measure the actual capacity consumption. If the value is high, but the cluster-wide utilization is not high, you have a risk of contention due to imbalance.

The allocation is the net value. It is based on usable capacity.

The presence of large VM introduces performance risk.

Availability

The availability of a complex system such as an ESXi host is not a simple binary. There is degradation, which is important to distinguish to help manage in large farm.



Implementation using VCF Operations super metric:

```
sum
([
   ({ This Resource: sys|poweredOn } ) ,
   (count({ This Resource: runtime|connectionState  , where = ($value contains 'connected' )})       > 0 ? 1 : 0 ) ,
   (count({ This Resource: runtime|maintenanceState , where = ($value contains 'notInMaintenance' )}) > 0 ? 1 : 0 ) ,
   (count({ This Resource: runtime|quarantineState  , where = ($value contains 'notInQuarantine' )}) > 0 ? 1 : 0 )
]) * 25
```

What are the limitations of the above?

Yes, it does not recognise the sub-degradation in a fully functioning ESXi. VMs running on an ESXi are unlikely to experience any performance degradation as the hardware has redundancy on network card, storage HBA, local disks, fan, and power supply. ESXi can also lose the iLO network as it's a separate network.

esxtop

Now that we have covered many of the metrics, the esxtop output would be easier to understand. This documentation is not about how to use esxtop, but about what the metrics mean and their relevance in operations management.

Overview

While the manual uses the term Guest, esxtop does not actually have any Guest OS metrics. Distinguish between Guest OS and VM as they are 2 different realms.

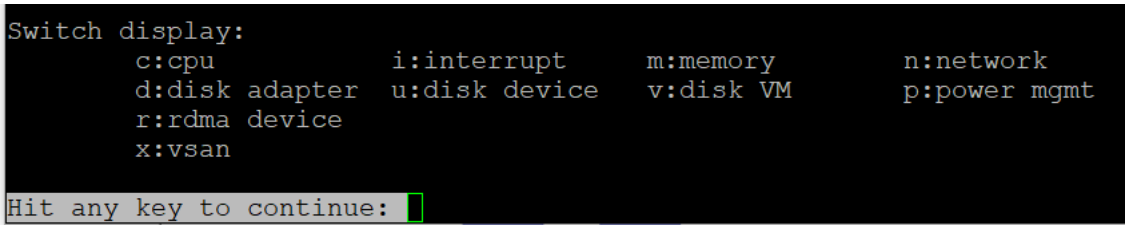
The view from a VM (consumer) and the view from ESXi (provider) are of opposite nature. vCPU is a construct of a VM, while core and thread are constructs seen by ESXi. You get to see both VM level and ESXi level objects at the same time. It is confusing for newbie, but convenient for power user, and if you're looking at esxtop, you are a power user 😊

The nature of esxtop means it is excellent for performance troubleshooting, especially real time and live situation where you know the specific ESXi Host. The tool is not so suitable for capacity management, where you need to look at long term (often weeks or months). As a result, I cover the contention metrics first, followed by consumption.

I have not had the need to use some of the metrics, hence I don't have much guidance on them. If you do, let's collaborate.

Grouping

The esxtop screen groups the metrics into 10 screen panels, as shown below:



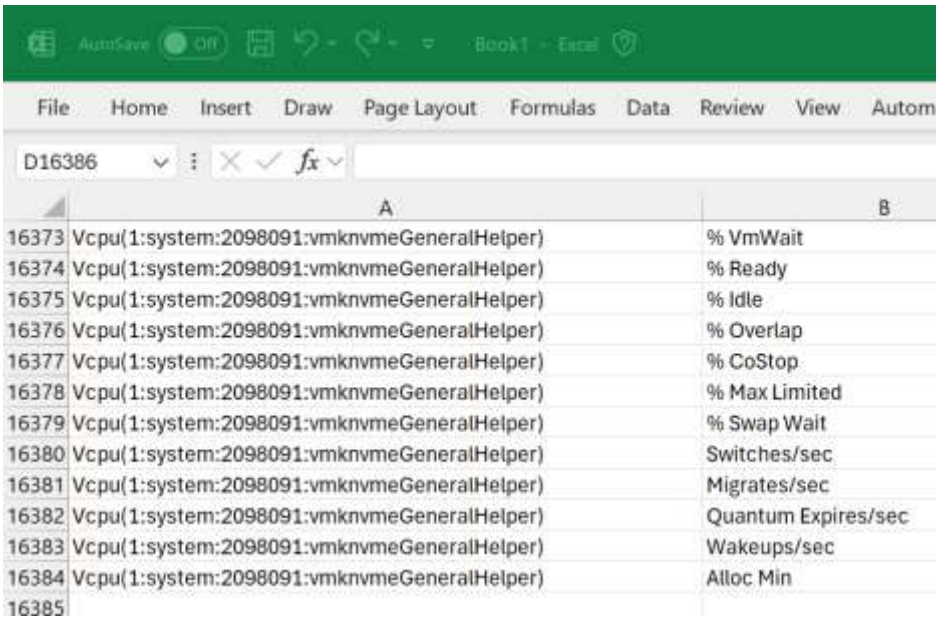
There are relationships among *some* of the 10 panels, but they are not obvious as the UI simply presents them as a list. To facilitate understanding of the metrics, we need to group them differently.

So instead of documenting the 10 panels, I'd group the panels into 4.

Group	Consumer	Provider	Remarks
CPU	Yes	Sort of	The CPU panel has a 4 line summary that provides the provider's viewpoint. I moved Power Management panel here as it only covers CPU. It does not cover memory, disk, network and other parts of the

			box (e.g. fan, motherboard). It complements the CPU panel as it covers the provider's viewpoint. Take note that it does not show at socket level. And if you enable HT, it does not show at core level. I moved interrupt panel here as it's about CPU.
Memory	1 shared panel for both		Provider and Consumer are shown in 1 panel. The panel has a summary at the top, which cover the provider's viewpoint
Storage	Yes	Almost	The Disk VM panel covers from consumer's viewpoint. The Disk Adapter panel and Disk Device panel cover from provider's viewpoint, and are best to be analyzed together. BTW, notice the Path panel is missing. I moved vSAN panel here as all the metrics are disk metrics. There is no vSAN network and CPU counter, but you can see them in the respective network and CPU panel.
Network	1 shared panel for both		Provider and Consumer are shown in 1 panel I moved RDMA device here as it's about network card

Export



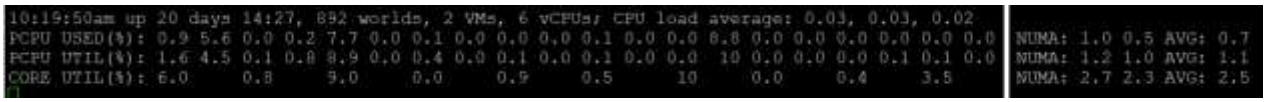
Avoid exporting to CSV file. If you need to do it, limit to specific metrics and keep the time short. If you collect everything, you end up with a large file (easily > 100 MB) with >10K metrics. The following shows 16384 metrics being collected.

CPU

The CPU panel consists of 2 parts:

- Summary
- Detail. It shows a table.

Here is the summary section. It has 4 lines.

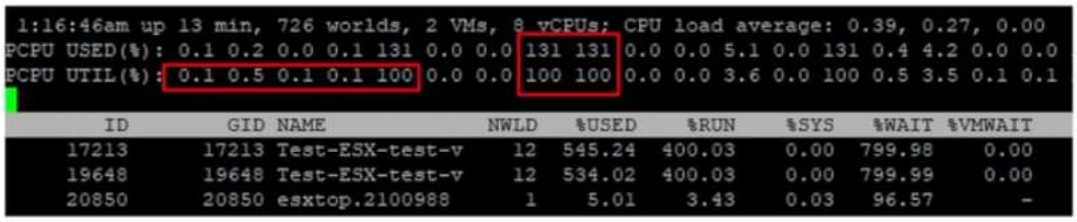


The first line shows the summary of the physical load average in the last 1 minute, 5 minute and 15 minutes, respectively.

The next 3 lines covers Used (%), Utilization (%) and Core Utilization (%). The reason why I swapped the order in the book is Used (%) is built upon Utilization, and it's a more complex counter. You can see in the following screenshot[21] that Used (%) hit 131% while Util (%) maxed at 100%.

Note that their values are in percentage, meaning you need to know what they use for 100%.

Example: ESXi host with 2 active VMs



If you guess that Used (%) and Utilization (%) eventually map into vSphere Client metrics Usage (%) and Utilization (%), respectively, you are right. However, you need to know how they map.

PCPU means a physical, hardware execution context. That means it is a physical core if CPU SMT is disabled, or a physical thread inside a core if SMT is enabled. It does not mean CPU socket. A single socket with 10 cores and 20 threads will have 20 PCPU metrics.

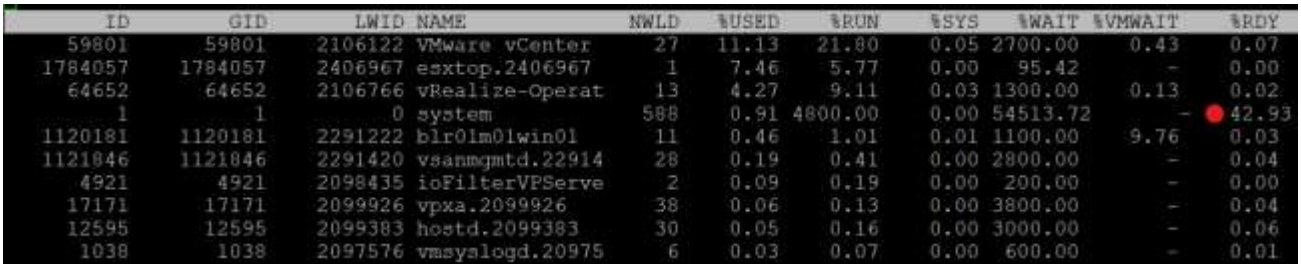
The white vertical line shows where I cut the screenshot, as the text became too small and unreadable if I were to include all of them. Anyway, it's just repeating for each CPU physical thread.

At the end of each 3 lines (after the white line in preceding screenshot), there are NUMA information. It shows the average value across each NUMA node (hence there are 2 numbers as my ESXi has 2 NUMA nodes). The number after **AVG** is the whole box, system wide average. The per NUMA node metric values are useful to easily identify if a particular NUMA node is overloaded.

The detail section takes a consumer view. It is different to the physical view above.

Take a look at the panel below. Notice something interesting?

[16] Source: VMworld presentation HCP2583 by Richard Lu and Qasim Ali



ID	GID	LWID	NAME	NWLD	%USED	%RUN	%SYS	%WAIT	%VMWAIT	%RDY
59801	59801	2106122	VMware vCenter	27	11.13	21.80	0.05	2700.00	0.43	0.07
1784057	1784057	2406967	esxtop.2406967	1	7.46	5.77	0.00	95.42	-	0.00
64652	64652	2106766	vRealize-Operat	13	4.27	9.11	0.03	1300.00	0.13	0.02
1	1	0	system	588	0.91	4800.00	0.00	54513.72	-	42.93
1120181	1120181	2291222	blr0lm01win01	11	0.46	1.01	0.01	1100.00	9.76	0.03
1121846	1121846	2291420	vsanmgmtd.22914	28	0.19	0.41	0.00	2800.00	-	0.04
4921	4921	2098435	ioFilterVPServe	2	0.09	0.19	0.00	200.00	-	0.00
17171	17171	2099926	vpxa.2099926	38	0.06	0.13	0.00	3800.00	-	0.04
12595	12595	2099383	hostd.2099383	30	0.05	0.16	0.00	3000.00	-	0.06
1038	1038	2097576	vmsyslogd.20975	6	0.03	0.07	0.00	600.00	-	0.01

It mixes VM and non VM processes in a single table. The non-VM also has Ready time. What it does not have is VM Wait, which is expected.

If you want to only show VMs, just type the capital letter **V**.

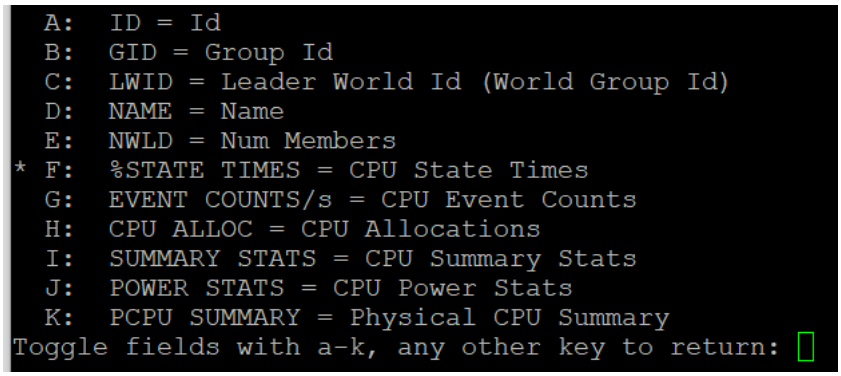
- Name based filtering allows regular expression based filtering for groups and worlds.
- Type the capital letter **G** to only show groups that match given string. This is useful when a host has large number of VMs and you want to focus on a single or set of interesting VMs.
- Once a group is expanded you can type the small letter **g** to show only the worlds that match the given string. This is useful when running a VM with many vCPUs and you want to focus on specific worlds like storage worlds or network worlds.

If you want to see all, how to tell which ones are VM? I use %VMWAIT column. This tracks the various waits that VM world gets, so it does not apply to non VM.

Notice the red dot in the picture. Why the Ready time is so high for system process?

Because this group includes the idle thread. Expand the GID and you will see Idle listed.

There are many columns, as shown below. The most useful one is the %State Times, which you get by pressing **F**.



```
  A:  ID = Id
  B:  GID = Group Id
  C:  LWID = Leader World Id (World Group Id)
  D:  NAME = Name
  E:  NWLD = Num Members
* F:  %STATE TIMES = CPU State Times
  G:  EVENT COUNTS/s = CPU Event Counts
  H:  CPU ALLOC = CPU Allocations
  I:  SUMMARY STATS = CPU Summary Stats
  J:  POWER STATS = CPU Power Stats
  K:  PCPU SUMMARY = Physical CPU Summary
Toggle fields with a-k, any other key to return:
```

The rest of the information are relatively static or do not require sub-20 second granularity.

You know that only Utilization (%) and Used (%) exist at the thread level because they are the only one you see at, as shown below.

CPU State

We covered earlier in the CPU Metric that there are only 4 states. But esxtop shows a lot more metrics.

NAME	%USED	%RUN	%SYS	%WAIT	%VMWAIT	%RDY	%IDLE	%OVRLP	%CSTP	%MLMTD	%SWPWT
VMware vCenter	23.82	35.91	0.06	2655.50	0.00	0.29	363.38	0.08	0.00	0.00	0.00
esxtop.2328582	6.75	4.28	0.00	95.41	-	0.00	0.00	0.00	0.00	0.00	0.00
vRealize-Operat	4.72	9.03	0.03	1286.71	0.21	0.30	190.57	0.03	0.00	0.00	0.00
system	1.80	4722.03	0.00	53618.23	-	99.68	0.00	1.25	0.00	0.00	0.00
blr01m01win01	0.47	1.06	0.01	1094.96	12.90	0.12	185.40	0.01	0.00	0.00	0.00
hostd.2099383	0.30	0.68	0.00	2989.43	-	0.12	0.00	0.00	0.00	0.00	0.00

So what does it mean? How come there are more than 4 states?

The answer is below. Some of these metrics are included in the other metrics.



Review the metrics below, starting with %USED.

Which one does *not* actually belong to a CPU state, meaning it's not something you mix with the rest?

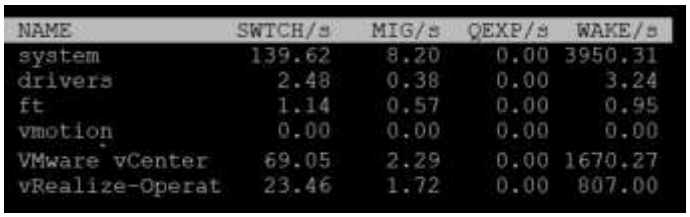
GID	NAME	NWLD	%USED	%RUN	%SYS	%WAIT	%VMWAIT	%RDY	%IDLE	%OVRLP	%CSTP	%MLMTD	%SWPWT
59801	VMware vCenter	27	10.04	19.06	0.05	2666.43	0.56	0.05	370.74	0.04	0.00	0.00	0.00
1076099	esxtop.2283469	1	5.35	4.06	0.00	95.41	-	0.00	0.00	0.00	0.00	0.00	0.00
64652	vRealize-Operat	13	3.47	7.57	0.02	1285.33	0.49	0.15	191.44	0.02	0.00	0.00	0.00
1	system	588	1.67	4737.42	0.00	53689.83	-	35.12	0.00	1.09	0.00	0.00	0.00
12595	hostd.2099383	30	0.16	0.35	0.00	2982.99	-	0.06	0.00	0.00	0.00	0.00	0.00
17171	vpxa.2099926	38	0.12	0.27	0.00	3779.10	-	0.05	0.00	0.00	0.00	0.00	0.00
1038	vmsyslogd.20975	6	0.02	0.04	0.00	596.66	-	0.00	0.00	0.00	0.00	0.00	0.00
1075991	sshd.2283455	1	0.01	0.03	0.00	99.44	-	0.00	0.00	0.00	0.00	0.00	0.00
24737	hostd-probe.210	8	0.01	0.03	0.00	795.67	-	0.02	0.00	0.00	0.00	0.00	0.00
9374	hostdCgiServer.	12	0.01	0.03	0.00	1193.28	-	0.01	0.00	0.00	0.00	0.00	0.00

That's right, it's %USED.

%USED	It should be excluded from this panel as it is influenced by power management and hyperthreading. We explained the reason why in CPU Metric chapter. That's why it's necessary to review the VM CPU states before reading each esxtop metric.
%RUN	Run is covered in-depth under VM CPU Metrics.
%SYS	System time is covered in-depth under VM CPU Metrics.
%WAIT	The wait counter and its components are covered in-depth under VM CPU Metrics. VMWAIT includes SWPWT. VCF Operations does not show VM Wait and uses a new counter that excludes Swap Wait. The reason is the remediation action is different. You're welcome.
%VMWAIT	
%SWPWT	
%IDLE	
%RDY	Ready is covered in-depth under VM CPU Metrics. As discussed in the CPU scheduling, each vCPU has its own ready time. In the case of esxtop, the metric is simply summed up, so it can go >100% in theory.
%CSTP	Co-Stop is covered in-depth under VM CPU Metrics. This is also 100% per vCPU.
%OVRLP	Overlap is covered in-depth under VM CPU Metrics.
%MLMTD	MLMTD is Max Limited, not some Multi-Level Marketing scam 😊. It measures the time

	the VM was halted due to manual limit, as opposed to the kernel has no CPU resource. This is more of an event as you should not be setting limit in the first place.

CPU Event Count

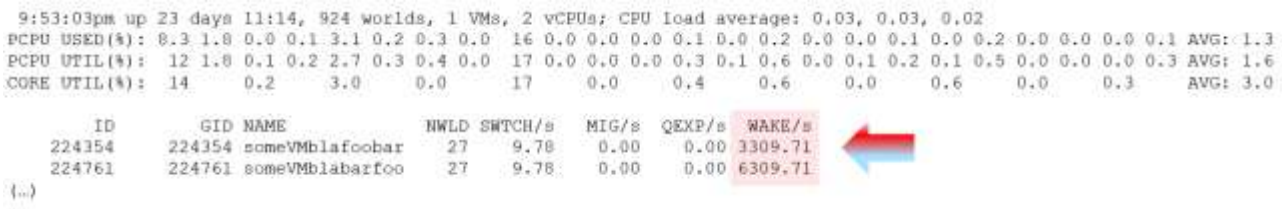


NAME	SWTCH/s	MIG/s	QEXP/s	WAKE/s
system	139.62	8.20	0.00	3950.31
drivers	2.48	0.38	0.00	3.24
ft	1.14	0.57	0.00	0.95
vmotion	0.00	0.00	0.00	0.00
VMware vCenter	69.05	2.29	0.00	1670.27
vRealize-Operat	23.46	1.72	0.00	807.00

SWTCH/s	Number of world switches per second, the lower the better. I guess this number correlates with the overcommit ratio, the number of VM and how busy they are. What number will be a good threshold and why?
MIG/s	Number of NUMA and core migrations per second. It will be interesting to compare 2 VM, where 1 is the size of a single socket, and the other is just a bit larger. Would the larger one experience a lot more switches?
WAKE/s	Number of time the world wakeups per second. A world wakes up when its state is changes from WAIT to READY. A high number can impact performance.

The metric QEXP/s (Quantum Expirations per second) has been deprecated from ESXi 6.5 in an effort to improve vCPU switch time.

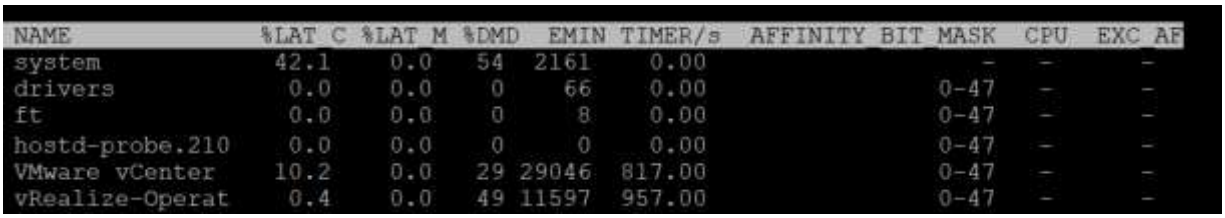
In rare case where the application has a lot of micro bursts, CPU Ready can be relatively higher to its CPU Run. This is due to the CPU scheduling cost. While each scheduling is negligible, having too many of them may register on the counter. If you suspect that, check esxtop, as shown below:



```
 9:53:03pm up 23 days 11:14, 924 worlds, 1 VMs, 2 vCPUs; CPU load average: 0.03, 0.03, 0.02
PCPU USED(%): 8.3 1.8 0.0 0.1 3.1 0.2 0.3 0.0  16 0.0 0.0 0.0 0.1 0.0 0.2 0.0 0.0 0.1 0.0 0.2 0.0 0.0 0.0 0.1 AVG: 1.3
PCPU UTIL(%):  12 1.8 0.1 0.2 2.7 0.3 0.4 0.0  17 0.0 0.0 0.0 0.3 0.1 0.6 0.0 0.1 0.2 0.1 0.5 0.0 0.0 0.0 0.3 AVG: 1.6
CORE UTIL(%):  14     0.2     3.0     0.0     17     0.0     0.4     0.6     0.0     0.6     0.0     0.3     AVG: 3.0

      ID        GID NAME             NWLD SWTCH/s   MIG/s  QEXP/s  WAKE/s
  224354     224354 someVMblafoobar    27    9.78    0.00    0.00 3309.71
  224761     224761 someVMblabarfoo    27    9.78    0.00    0.00 6309.71
(...)
```

Summary Stats



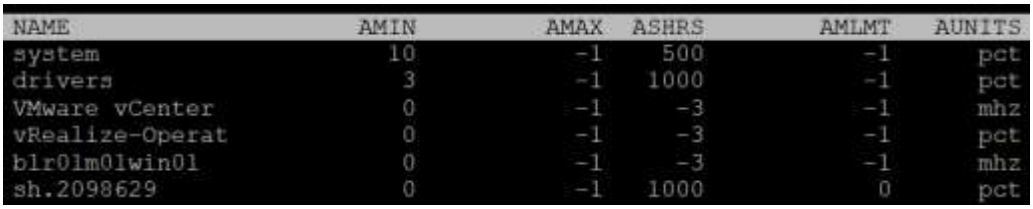
NAME	%LAT C	%LAT M	%DMD	EMIN	TIMER/s	AFFINITY BIT MASK	CPU	EXC	AF
system	42.1	0.0	54	2161	0.00	-	-	-	
drivers	0.0	0.0	0	66	0.00	0-47	-	-	
ft	0.0	0.0	0	8	0.00	0-47	-	-	
hostd-probe.210	0.0	0.0	0	0	0.00	0-47	-	-	
VMware vCenter	10.2	0.0	29	29046	817.00	0-47	-	-	
vRealize-Operat	0.4	0.0	49	11597	957.00	0-47	-	-	

Other than the first 3 (which I'm unsure why they are duplicated here as they are shown in the CPU State already), the other metrics do not exist in vSphere Client UI and VCF Operations.

%LAT_C	This is covered in-depth in CPU Contention
%LAT_M	This is covered in-depth in Memory contention
%DMD	This is covered in-depth in CPU Demand
EMIN	This is the minimum amount of CPU in MHz that the world will get when there is not enough for everyone.
TIMER/s	Timer rate for this world
AFFINITY BIT MASK	Bit mask showing the current scheduling affinity for the world. Not set for Latency Sensitive = High VMs
CPU	The physical or logical processor on which the world was running when esxtop obtained this information. BTW, I'm not 100% sure as this is always blank for me
EXC_AF	Yes means the VM has exclusive affinity. This happens when you enabled the Latency Sensitivity setting. Use this feature very carefully.

The column HTQ is no longer shown in ESXi 7.0. In earlier release, this indicates whether the world is quarantined or not. 'N' means no and 'Y' means yes.

CPU Allocation

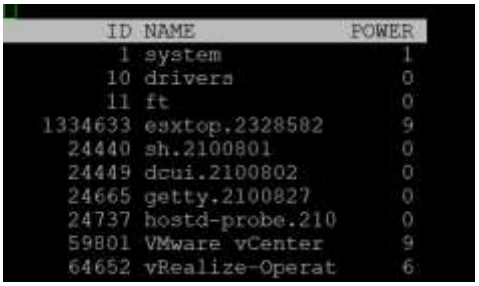


NAME	AMIN	AMAX	ASHRS	AMLMT	AUNITS
system	10	-1	500	-1	pct
drivers	3	-1	1000	-1	pct
VMware vCenter	0	-1	-3	-1	mhz
vRealize-Operat	0	-1	-3	-1	pct
blr01m01win01	0	-1	-3	-1	mhz
sh.2098629	0	-1	1000	0	pct

AMIN	Allocation Minimum. Basically, the reservation
AMAX	Allocation Maximum. Basically, the limit.
ASHRS	Allocation shares
AMLMT	Max Limited. I'm unsure if this is when it's applied or not.
AUNITS	Units. For VM, this is in MHz. For the kernel module, this is in percentage.

Power Stats

This complements the power management panel as it lists per VM and kernel module, while the power panel lists per ESXi physical treads (logical CPU).

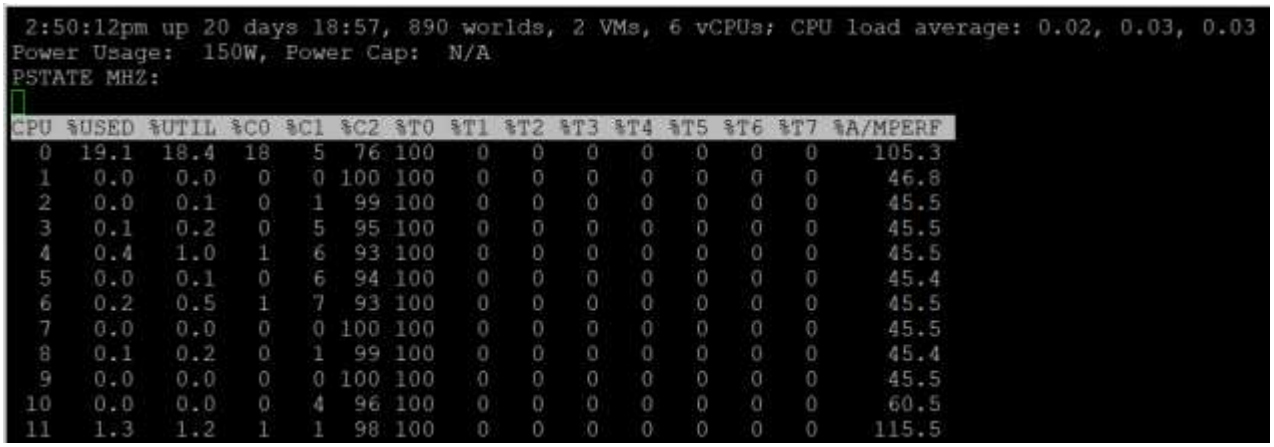


ID	NAME	POWER
1	system	1
10	drivers	0
11	ft	0
1334633	esxtop.2328582	9
24440	sh.2100801	0
24449	dcui.2100802	0
24665	getty.2100827	0
24737	hostd-probe.210	0
59801	VMware vCenter	9
64652	vRealize-Operat	6

POWER	Current CPU Power consumption in Watts. So it does not include memory, disk, etc.

Power Consumption

Power management is given its own panel. This measures the power consumption of each physical thread. If you disable hyper-threading, then it measures at physical core.



```
 2:50:12pm up 20 days 18:57, 890 worlds, 2 VMs, 6 vCPUs; CPU load average: 0.02, 0.03, 0.03
Power Usage:  150W, Power Cap:  N/A
PSTATE MHZ:
```

CPU	%USED	%UTIL	%C0	%C1	%C2	%T0	%T1	%T2	%T3	%T4	%T5	%T6	%T7	%A/MPERF
0	19.1	18.4	18	5	76	100	0	0	0	0	0	0	0	105.3
1	0.0	0.0	0	0	100	100	0	0	0	0	0	0	0	46.8
2	0.0	0.1	0	1	99	100	0	0	0	0	0	0	0	45.5
3	0.1	0.2	0	5	95	100	0	0	0	0	0	0	0	45.5
4	0.4	1.0	1	6	93	100	0	0	0	0	0	0	0	45.5
5	0.0	0.1	0	6	94	100	0	0	0	0	0	0	0	45.4
6	0.2	0.5	1	7	93	100	0	0	0	0	0	0	0	45.5
7	0.0	0.0	0	0	100	100	0	0	0	0	0	0	0	45.5
8	0.1	0.2	0	1	99	100	0	0	0	0	0	0	0	45.4
9	0.0	0.0	0	0	100	100	0	0	0	0	0	0	0	45.5
10	0.0	0.0	0	4	96	100	0	0	0	0	0	0	0	60.5
11	1.3	1.2	1	1	98	100	0	0	0	0	0	0	0	115.5

The Power Usage line tracks the current total power usage (in Watts). Compare this with what the hardware specification. Power Cap shows the limit applied. You only do this hard limit when there is insufficient power supply from the rack.

The PSTATE MHZ line tracks the CPU clock frequency for each state.

Now let's go into the table. It lists all the physical core (or thread if you enable HT). Note it does not group them by socket.

%USED	Used (%) metric is covered in-depth in ESXi CPU metric sub-chapter.
%UTIL	This is Thread Utilization (%), not Core Utilization (%).
%CState %TState	Percentage of time spent in a C-State, P-State and T-State. Power management is covered in System Architecture sub-chapter.
%A/Mperf	Actual / Measured Performance, expressed in percentage. The word measured in this case means the nominal or static value. So a value above 100% means Turbo,

	while a value below 100% means power saving kicked in. If this number is not what you are expecting, check the power policy settings in BIOS and ESXi. Notice this is not on a vCPU. This means you do not know the boost per VM. This counter is only applicable when the core is on %C0 state. In the preceding example, ignore the values from CPU 1 – CPU 11.

The following screenshot shows ESXi with 14 P-States, where P0 is represented as 2401 MHz. Each row is a physical thread as HT is enabled.

See PCPU 10 and 11 (they share core 6). What do you notice?

```
10:15:23am up 2 days 53 min, 674 worlds, 1 VMs, 2 vCPUs; CPU load average: 0.10, 0.09, 0.03
Power Usage:  147W, Power Cap:  N/A
PSTATE MHZ: 2401 2400 2300 2200 2100 2000 1900 1800 1700 1600 1500 1400 1300 1200

CPU %USED %UTIL %C0 %C1 %C2 %P0 %P1 %P2 %P3 %P4 %P5 %P6 %P7 %P8 %P9 %P10 %P11 %P12 %P13 %A/MPERF
  (...)
  4   0.3   0.5   0  11  88  95   0   0   0   0   0   0   0   0   0    0    0    0    4    95.2
  5   0.0   0.1   0   3  97   9   0   0   0   0   0   0   0   0   1    0    0    0   91    77.8
  6   0.1   0.1   0   7  93   0   0   0   0   0   0   0   0   0   0    0    0    0  100   105.5
  7   0.5   0.7   1   1  99 100   0   0   0   0   0   0   0   0   0    0    0    0    0   117.1
  8   2.5   2.4   2  16  81  17   0   0   0   0   0   0   0   0   0    0    0    0   83   103.9
  9   0.1   0.3   0   1  98   6   0   0   0   0   0   0   0   0   0    0    0    0   94    59.7
 10  63.0 100.0 100   0   0 100   0   0   0   0   0   0   0   0   0    0    0    0    0   130.2
 11  62.9 100.0 100   0   0 100   0   0   0   0   0   0   0   0   0    0    0    0    0   130.1
 12   3.1   3.1   3  12  85  85   0   0   0   0   0   0   0   0   0    0    0    0   15   102.4
 13   0.3   0.5   0  17  83  12   0   0   0   0   0   0   0   0   0    0    0    0   88    79.8
 14   0.4   0.5   1  16  84  43   0   0   0   0   0   0   0   0   0    0    0    0   57    94.1
 15   0.1   0.3   0   2  97 100   0   0   0   0   0   0   0   0   0    0    0    0    0   73.0
 16   3.7   3.1   3   4  93   5   0   0   0   0   0   0   0   0   0    0    0    0   95   126.0
 17   0.0   0.1   0   5  95   3   0   0   0   0   0   0   0   0   0    0    0    0   97    50.8
 18   0.4   0.7   1   9  90   7   0   0   0   0   0   0   0   0   0    0    1    0   92    54.8
 19   1.4   1.4   1  14  85  22   0   0   0   0   0   0   0   0   0    0    0    0   78   103.3
  (...)
```

Utilization (%) shows 100% for both. This means both threads run, hence competing.

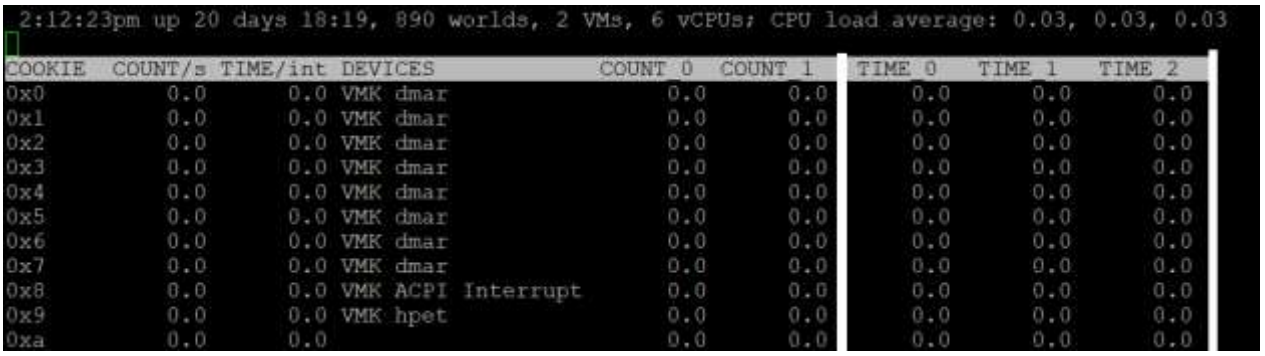
The core is in Turbo Boost. The %A/MPERF shows frequency increase of 30% above nominal. The core is in C0 state and P0 state. This counter was introduced in ESXi 6.5. It is not available via vSphere Client UI.

Why is Used (%) for PCPU 10 and 11 are showing 63.0% and 62.9%?

Unlike Utilization (%) which adds up to 200%, Used (%) adds up to 100%. So each thread maxes out at 50%. But Used (%) considers frequency scaling. Since there is a turbo boost at 130%, you get 50% x 130% = 65%. Pretty close to the numbers shown there.

Interrupt

This panel captures the interrupt vectors. In the following screenshot, I've added 2 vertical white lines to show where I cropped the screenshot. It's showing the value of each CPU thread, so the column became too wide.



COUNT/s	Total number of interrupts per second. This value is cumulative of the count for every CPU.
COUNT_x	Count 0, Count 1, etc. Interrupts per second on CPU x. My guess is CPU 0 is the first thread in the first core in the first socket.
TIME/int	Average processing time per interrupt (in microseconds). It will be interesting to profile this for each type of interrupt.
TIME_x	Time 0, Time 2, etc. Average processing time per interrupt on CPU x (in microseconds).
DEVICES	Devices that use the interrupt vector. If the interrupt vector is not enabled for the device, its name is enclosed in angle brackets (< and >).

To see the list of devices, issue the command at ESXi console: sched-stats -t sys-service-stats. You will get something like this:

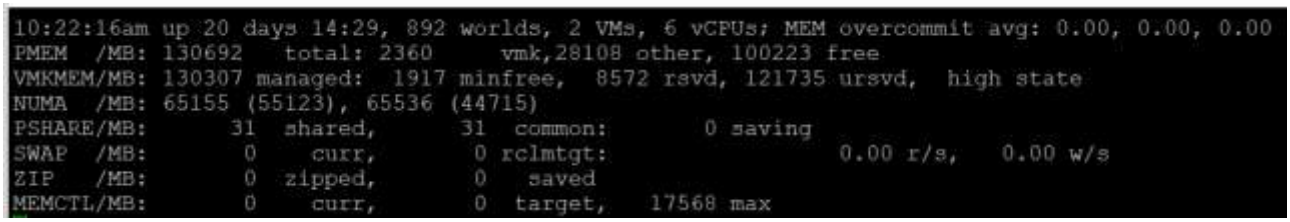
```
service      count       time maxElapsed maxService   name
     32   98973493    171.267       0.000       0.000   VMK-lsi_msgpt3_0
     33   93243036    153.993       0.000       0.000   VMK-lsi_msgpt3_0
     34 1783955246   1841.025       0.000       0.000   VMK-igbn-rxq0
     36          4      0.000       0.000       0.000   VMK-Event
     37  167025903    418.733       0.000       0.000   VMK-xhci0-intr
     51  242318260    792.014       0.000       0.000   VMK-0000:19:00.1-TxRx-0
     60   21281764     80.125       0.000       0.000   VMK-vmw_ahci_00003b000
    244     176227      0.090       0.000       0.000   VMK-timer-ipi
    245    1250405      0.163       0.000       0.000   VMK-monitor
    246 1868139923    340.709       0.000       0.000   VMK-resched
    248  414047027    189.255       0.000       0.000   VMK-tlb
   4096 3193917027   1321.416       0.000       0.000   0_2nd-level-intr-handler
   4097  304258696    193.711       0.000       0.000   1_smpcall
   4099        246      0.003       0.000       0.000   3_VOB-Wakeup
   4100   35706272      6.186       0.000       0.000   4_TimerBH
```

```
4101  399313616  10339.744      0.000      0.000   5_fastSlab
4104     859208      7.851      0.000      0.000   8_logEvent
4105  109560008    158.914      0.000      0.000   9_netTxComp
4106         26      0.197      0.196      0.196   10_keyboard
4107         56      0.001      0.000      0.000   11_SMIEnableCountPCPU-bh
4165  365305096   2433.530      0.001      0.001   TCPIPRX
4167   54024607     55.359      0.000      0.000   SCSI
4171   54520415    124.983      0.000      0.000   START-PATH-CMDS
4173   55109136    254.927      0.000      0.000   COMPL.-ADAPTER-CMD
4174   55102189     85.804      0.000      0.000   START-ADAPTER-CMDS
4180 5254928064  13877.461      0.001      0.001   Netpoll
```

BTW, some services maybe combined and reported under VMK-timer. For example, IOChain from vSphere Distributed Switch does not appear on its own.

Memory

The top part of the screen provides summary at ESXi level. They are handy in seeing overall picture, before diving into each VM or the kernel modules.



MEM overcommit avg	Average memory overcommit level in the last 1-minute, 5-minute, and 15-minute, respectively. Calculation is done with Exponentially Weighted Moving Average. Memory overcommit is the ratio of total requested memory and the "managed memory" minus 1. According to this, the kernel computes the total requested memory as a sum of the following components: 1. VM configured memory (or memory limit setting if set), 2. the user world memory, 3. the reserved overhead memory. If the ratio is > 1, it means that total requested VM memory is more than the physical memory available. This is fine, because ballooning and page sharing allows memory overcommit. I'm puzzled why we mix allocation and utilization. No 1 and no 3 make sense, but what exactly is no 2? My recommendation is you simply take the configured VM memory and ignore everything else. While it's less accurate, since the purpose is capacity and not performance, it's more than good enough and it's easier to explain to management. There is no need to get other details.
PMEM	Physical Memory.

	Total = vmk + Other + Free Total is what is reported by BIOS. vmk is ESXi the kernel consumption. This includes kernel code section, kernel data and heap, and other the kernel management memory. Other is memory consumed by VM and non VM (user-level process that runs directly on the kernel)
VMKMEM	The kernel memory. The following metrics are shown: • Managed. The memory space that ESXi manage. Typically this is slightly smaller than the total physical memory, as it does not contain all the components of vmk metric. It can be allocated to VM, non VM user world, or the kernel itself. • Minfree. The minimum amount of machine memory that the kernel would like to keep free. The kernel needs to keep some amount of free memory for critical uses. Note that minfree is included in Free memory, but the value tends to be negligible. • Reserved. The sum of the reservation setting of the groups + the overhead reservation of the groups + minfree. I think by group it means the world or resource pool. • Unreserved. It is the memory available for reservation. I have not found a practical use case for the above 4 metrics. If you do, let me know! State is the memory state. You want this to be on high state.
NUMA	In the preceding screenshot, there are 2 NUMA nodes. For each node there are 2 metrics: the total amount and the free amount. Note that the sum of all NUMA nodes will again be slightly smaller than total, for the same reason why the kernel managed is less than total. If you enable Cluster-on-Die feature in Intel Xeon, you will see 2x the number of nodes. For details, see this by Frank Denneman.
PSHARE	**shared**: the amount of VM physical memory that is being shared. **common**: the amount of machine memory that is common across Worlds. **saving**: the amount of machine memory that is saved due to page-sharing.
SWAP	Swapped counter is covered under VM memory. What “cannot” be zipped is swapped. What you see on this line is sum of all the VMs. The metric rclmtgt shows the target size in MB that ESXi aims to swap.
ZIP	Zipped counter is covered under VM memory. What you see on this line is sum of all the VMs.
MEMCTL	Memory Control, also known as ballooning, is covered here under VM memory. What you see on this line is sum of all the VMs.

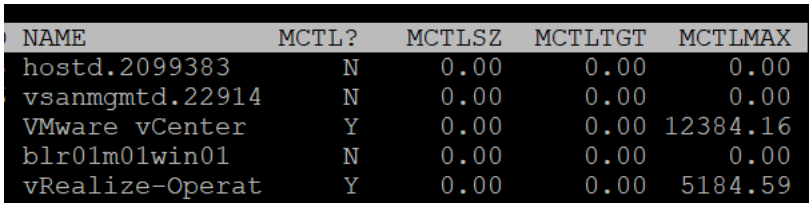
There are a lot of metrics in many panels. It’s easier to understand if we group them functionally.

Contention

As usual, we start with the contention-type of metrics.

Balloon

I start with Balloon as this is the first level of warning. Technically, this is not a contention. Operationally, you want to start watching as Balloon only happens at 99% utilization. So it's high considering you have HA enabled in the cluster.



NAME	MCTL?	MCTLSZ	MCTLTGT	MCTLMAX
hostd.2099383	N	0.00	0.00	0.00
vsanmgmtd.22914	N	0.00	0.00	0.00
VMware vCenter	Y	0.00	0.00	12384.16
blr01m01win01	N	0.00	0.00	0.00
vRealize-Operat	Y	0.00	0.00	5184.59

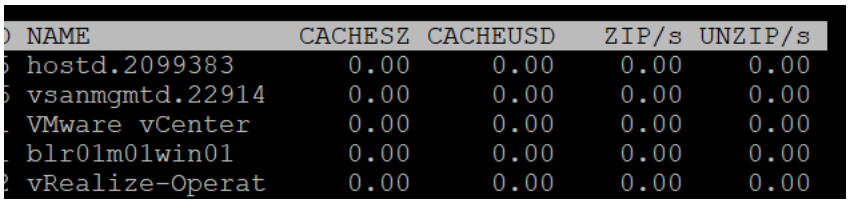
MCTL?	'Y' means the line is a VM, as the kernel processes is not subjected to ballooning.
MCTLSZ (MB)	Memory Control Size is the present size of memory control (balloon driver). If larger than 0 hosts is forcing VMs to inflate balloon driver to reclaim memory as host is overcommitted
MCTLTGT (MB)	Amount of physical memory the ESXi system attempts to reclaim from the resource pool or VM by way of ballooning. If this is not 0 that means the VM can experience ballooning.
MCTLMAX (MB)	Maximum amount of physical memory the ESXi system can reclaim from the resource pool or VM by way of ballooning. This maximum depends on the type of Guest OS.

Compressed & Swapped

I think that Swap and Compressed should be shown together as what can't be compressed is swapped.

Why am I showing Compressed first?

Because it's faster than swapped.



NAME	CACHESZ	CACHEUSD	ZIP/s	UNZIP/s
hostd.2099383	0.00	0.00	0.00	0.00
vsanmgmtd.22914	0.00	0.00	0.00	0.00
VMware vCenter	0.00	0.00	0.00	0.00
blr01m01win01	0.00	0.00	0.00	0.00
vRealize-Operat	0.00	0.00	0.00	0.00

CACHESZ (MB)	Compression memory cache size.
CACHEUSD (MB)	Used compression memory cache
ZIP/s (MB/s)	The rate at which memory pages are being zipped. Once zipped, it's not immediately available for the VM. This is a capacity problem. Your ESXi needs more RAM. If the pages being zipped is unused, the VMs will not experience memory contention.

	Keep this number 0. See Capacity chapter for details.
UNZIP/s (MB/s)	The rate at which memory pages are being unzipped so it can be used by VM. This is a performance problem. The pages are being asked. The VM CPU is waiting for the data. If you check the VM memory contention counter, it will not be 0%. Make sure that number is within your SLA or KPI.

Swapped

NAME	SWCUR	SWTGT	SWR/s	SWW/s	LLSWR/s	LLSWW/s
dcui.2100802	0.00	0.00	0.00	0.00	0.00	0.00
getty.2100827	0.00	0.00	0.00	0.00	0.00	0.00
hostd-probe.210	0.00	0.00	0.00	0.00	0.00	0.00
VMware vCenter	0.00	0.00	0.00	0.00	0.00	0.00
ioFilterVPServe	0.00	0.00	0.00	0.00	0.00	0.00
sh.2098515	0.00	0.00	0.00	0.00	0.00	0.00

SWCUR (MB)	Swapped Current is the present size of memory on swapped. It typically contains inactive pages.
SWTGT (MB)	The target size the ESXi host expects the swap usage by the resource pool or VM to be. This is an estimate.
SWR/s (MB) SWW/s (MB)	Swapped Read per second and Swapped Write per second. The amount of memory in megabyte that is being brought back to memory or being moved to disk
LLSWR/s (MB) LLSWW/s (MB)	These are similar to SWR/s but is about host cache instead of disk. It is the rate at which memory is read from the host cache. The reads and writes are attributed to the VMM group only, so they are not displayed for VM. LL stands for Low Latency as host cache is meant to be faster (lower latency) than physical disk. Memory to host cache can be written from both the physical DIMM and disk. So the counter LLSWW/s covers all these sources, and not just from physical DIMM.

NUMA



Logically, this statistic is applicable only on NUMA systems.

NAME	NWLD	NHN	NMIG	NRMEM	NLMEM	N%L	GST_ND0	OVD_ND0	GST_ND1	OVD_ND1
dcui.2100802	2	-	-	-	-	-	-	-	-	-
getty.2100827	1	-	-	-	-	-	-	-	-	-
hostd-probe.210	8	-	-	-	-	-	-	-	-	-
VMware vCenter	27	1	16	0.00	19456.00	100	0.00	19.87	19456.00	51.26
rhttpproxy.2099	21	-	-	-	-	-	-	-	-	-
vRealize-Operat	13	0	16	0.00	8192.00	100	8192.00	24.12	0.00	8.75

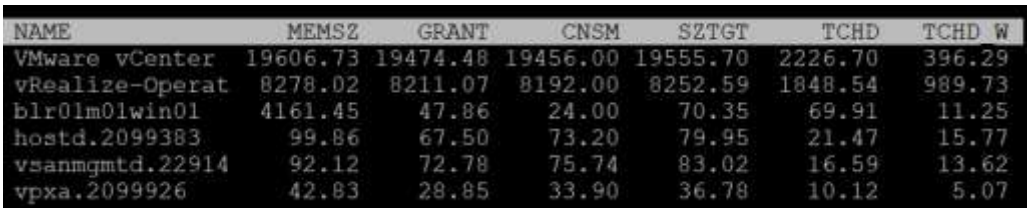
NHN	The count of NUMA Home Node for the resource pool or VM. If the VM has no home node, a dash (-) appears. You want to see the number 1. If you see the number 0/1 or 2, that means the VM is split into multiple nodes, which could impact performance. I notice when the number is 0/1, the NMIG column shows 0.
	When you enable CPU Hot Add, esxtop will report multiple home nodes as NUMA is disabled. It also does not distinguish remote and local memory as memory is interleaved. For more information, see this by Frank.
NMIG	Number of NUMA migrations. It gets reset upon VM power cycle, meaning this counter is accumulative. Be careful as you could be looking at past data. Use Log Insight to plot the event over time. Migration is costly as all pages need to be remapped. Local memory starts at 0% again and grows overtime. Copying memory pages across NUMA boundaries cost memory bandwidth.
NRMEM (MB)	Current amount of remote memory allocated to the VM or resource pool. Ideally this amount is 0 or a tiny percentage. You decrease the chance by decreasing the VM configured RAM. A VM whose configured memory is larger than the ESXi RAM attached to a single CPU socket have higher chance of having remote memory.
N%L	Current percentage of memory allocated to the VM or resource pool that is local. Anything less than 100% is not ideal.
GST_NDx (MB)	Guest memory allocated for a resource pool on NUMA node x, where GST_ND0 means the first node. The following screenshot shows the VMware vCenter VM runs on node 2 while the vRealize-Operat VM runs on node 1. 
OVD_NDx (MB)	VMM overhead memory allocated for a resource pool on NUMA node x, where x starts with 0 for the first node.

NAME	N%L	GST_ND0	OVD_ND0	GST_ND1
esxtop.3141796	-	-	-	-
VMware vCenter	100	0.00	19.87	19456.00
sensord.2346563	-	-	-	-
vRealize-Operat	100	8192.00	24.00	0.00
vmware-usbarbit	-	-	-	-
blr01m01win01	100	24.00	10.25	0.00

Consumption

I group metrics such as consumed, granted, and overhead under consumption as they measure how much the VM or the kernel module consumes.

Consumed

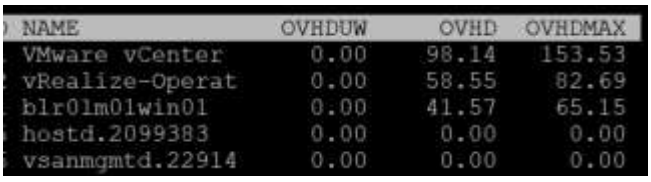


NAME	MEMSZ	GRANT	CNSM	SZTGT	TCHD	TCHD_W
VMware vCenter	19606.73	19474.48	19456.00	19555.70	2226.70	396.29
vRealize-Operat	8278.02	8211.07	8192.00	8252.59	1848.54	989.73
blr01m01win01	4161.45	47.86	24.00	70.35	69.91	11.25
hostd.2099383	99.86	67.50	73.20	79.95	21.47	15.77
vsanmgmtd.22914	92.12	72.78	75.74	83.02	16.59	13.62
vpxa.2099926	42.83	28.85	33.90	36.78	10.12	5.07

MEMSZ (MB)	Amount of physical memory allocated to a resource pool or VM. The values are the same for the VMM and VMX groups. MEMSZ = GRANT + MCTLSZ + SWCUR + "never touched" I'm unsure where the compressed page goes. It's still occupying space but 50% or 25%.
GRANT (MB)	Do not confuse it with Consumed 😊
CNSM	Yup, this is that legendary Consumed metric.
SZTGT (MB)	Size Target in MB. Amount of machine memory the ESXi kernel wants to allocate to a resource pool or VM. The values are the same for the VMM and VMX groups.
TCHD (MB)	The size of touched pages in MB Working set estimate for the resource pool or VM. The values are the same for the VMM and VMX groups.
TCHD_W	As per above, but only for the write operations. A relatively much lower value compared to TCHD means the activities are mostly read.

Overhead

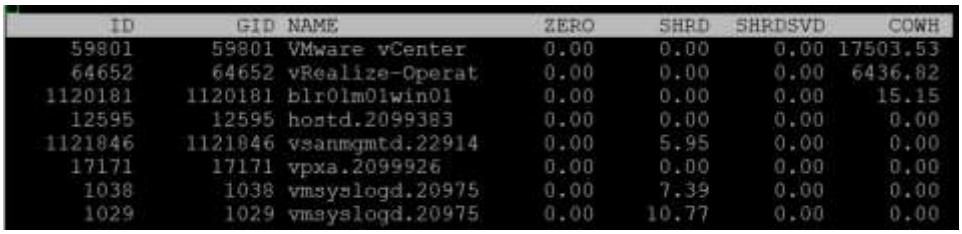
I find overhead is a small amount that is practically negligible, considering ESXi nowadays sports a large amount of RAM. Let me know the use case where you find otherwise.



NAME	OVHDUW	OVHD	OVHDMAX
VMware vCenter	0.00	98.14	153.53
vRealize-Operat	0.00	58.55	82.69
blr01m01win01	0.00	41.57	65.15
hostd.2099383	0.00	0.00	0.00
vsanmgmtd.22914	0.00	0.00	0.00

OVHD (MB)	Current space overhead for resource pool.
OVHDMAX (MB)	Maximum space overhead that might be incurred by resource pool or VM.
OVHDUW (MB)	Current space overhead for a user world. It is intended for VMware use only.

Shared

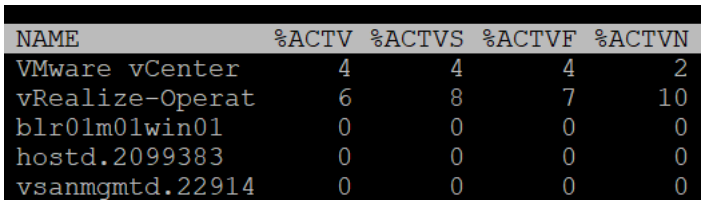


ID	GID	NAME	ZERO	SHRD	SHRDSVD	COWH
59801	59801	VMware vCenter	0.00	0.00	0.00	17503.53
64652	64652	vRealize-Operat	0.00	0.00	0.00	6436.82
1120181	1120181	blr01m01win01	0.00	0.00	0.00	15.15
12595	12595	hostd.2099383	0.00	0.00	0.00	0.00
1121846	1121846	vsanmgmtd.22914	0.00	5.95	0.00	0.00
17171	17171	vpxa.2099926	0.00	0.00	0.00	0.00
1038	1038	vmsyslogd.20975	0.00	7.39	0.00	0.00
1029	1029	vmsyslogd.20975	0.00	10.77	0.00	0.00

ZERO (MB)	Resource pool or VM physical pages that are zeroed.
SHRD (MB)	Total amount that is shared.
SHRDSVD (MB)	Machine pages that are saved because of sharing. Notice this counter does not exist in vSphere Client UI.
COWH (MB)	Copy on Write Hint. An estimate of the amount of Guest OS pages for TPS purpose.

Active

The manual uses the word Guest to refer to VM. I distinguish between VM and Guest. Guest is an OS, while a VM is just a collection of processes. Guest has its own memory management that is completely invisible to the hypervisor.



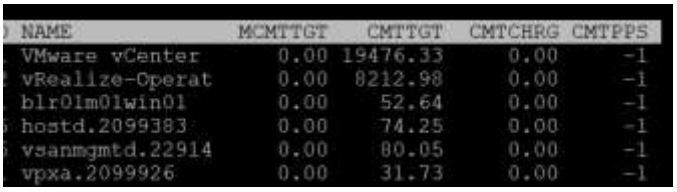
NAME	%ACTV	%ACTVS	%ACTVF	%ACTVN
VMware vCenter	4	4	4	2
vRealize-Operat	6	8	7	10
blr01m01win01	0	0	0	0
hostd.2099383	0	0	0	0
vsanmgmtd.22914	0	0	0	0

%ACTV	Active is covered in-depth in Active metric
%ACTVS	Percentage Active Slow and Percentage Active Fast.
%ACTVF	Slow is the slow moving average, taking longer period. Longer is more accurate. I don't have a use case for the fast moving average.
%ACTVN	Percentage Active Next. It predicts of what %ACTVF will be at next sample estimation. It is intended for VMware use only.

Committed

Committed page means the page has been reserved for that process. Commit is a counter for utilization but it's not really used, especially for VM.

Note: none of these metrics exist in vSphere Client and VCF Operations, as they are meant for internal use.



NAME	MCMTTGT	CMTTGT	CMTCHRG	CMTPPS
VMware vCenter	0.00	19476.33	0.00	-1
vRealize-Operat	0.00	8212.98	0.00	-1
blr01m01win01	0.00	52.64	0.00	-1
hostd.2099383	0.00	74.25	0.00	-1
vsanmgmtd.22914	0.00	80.05	0.00	-1
vpxa.2099926	0.00	31.73	0.00	-1

MCMTTGT	Minimum Commit Target in MB. I think this value is not 0 when there is reservation, but I'm not sure.
CMTTGT	Commit Target in MB.
CMTCHRG	Commit Charged in MB. I think this is the actual committed page.
CMTPPS	Commit Pages Per Share in MB

Allocation & Reservation

NAME	NWLD	AMIN	AMAX	ASHRS	AMLMT	AUNITS
net-lacp.209801	3	0	-1	-3	-1	kb
vpxa.2099926	38	0	-1	-3	-1	kb
dhclient-uw.209	1	0	-1	-3	-1	kb
busybox.2098035	1	0	-1	-3	-1	kb
rhttpproxy.2099	21	0	-1	-3	-1	kb
busybox.2098124	1	0	-1	-3	-1	kb
init.2097526	1	0	-1	-3	-1	kb
vmsyslogd.20975	1	0	-1	-3	-1	kb

AMIN	Allocation minimum. This is the term esxtop uses for memory reservation for this resource pool or VM. A value of 0 means no reservation, which is what you should set for most VM. Reservation for the kernel modules should be left as it is.
AMAX	Allocation maximum. This is the term esxtop uses for memory limit for this resource pool or VM. A value of -1 means Unlimited. Limit for the kernel modules should be left as it is.
AMLMT	Limit. You should expect the value -1, means no limit assigned. I'm not sure how this differs to AMAX.
ASHRS	Memory shares for this resource pool or VM.
AUNITS	This is just displaying the units of allocations counters

Checkpoint

Checkpoint is required in snapshot or VM suspension. You can convert a VM checkpoint into a core dump file, to debug the Guest OS and applications.

CPTRD (MB)	Checkpoint Read. Amount of data read from checkpoint file. A large amount can impact the VM performance.

CPTTGT (MB)	Checkpoint Target. The target size of checkpoint file that the kernel is aiming for. I'm unsure why it needs to have a target, unless this is just an estimate of the final size and not a limit.

Storage

The Storage monitoring sports 3 panels:

- VM
- Adapter
- Device

We covered in Part 2 that an ESXi host has adapter, path and devices. I'm unsure why esxtop does not have a panel for path. It would be convenient to check dead path or inactive path as the value will be all 0. If your design is active/active, it can be useful to compare if their throughput is not lopsided.

Datastore is also missing. While VMFS can be covered with Device (if you do 1:1 mapping and not using extent), NFS is not covered.

On the other hand, esxtop does provide metrics that vSphere Client does not. I will highlight those.

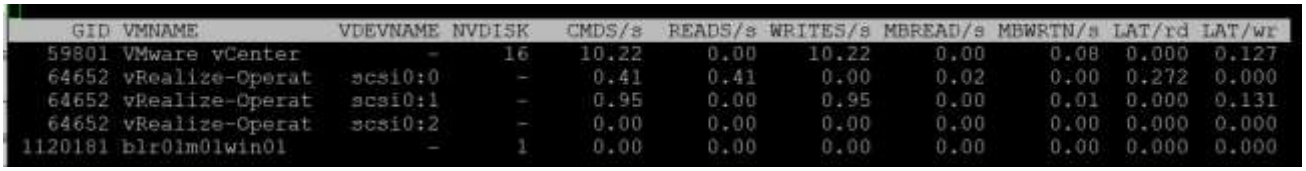
ESXi uses adapter to connect to device. As a result, their main contention and utilization metrics are largely similar. I've put them side by side here, and highlight the similar metric *groups* with vertical green bar. I highlighted the word group, as the group name may be identical, but the actual metrics within the group differ.

Adapter metrics groups	Device metrics groups
	ID = Path/World/Partition Id NUM = Num of Objects SHARES = Shares BLKSZ = Block Size (bytes)
PATH = Path Name NPATHS = Num Paths	
QSTATS = Queue Stats IOSTATS = I/O Stats RESVSTATS = Reserve Stats	QSTATS = Queue Stats IOSTATS = I/O Stats RESVSTATS = Reserve Stats
LATSTATS/cmd = Overall Latency Stats (ms) LATSTATS/rd = Read Latency Stats (ms) LATSTATS/wr = Write Latency Stats (ms)	LATSTATS/cmd = Overall Latency Stats (ms) LATSTATS/rd = Read Latency Stats (ms) LATSTATS/wr = Write Latency Stats (ms)
ERRSTATS/s = Error Stats	ERRSTATS/s = Error Stats
PAESTATS/s = PAE Stats SPLTSTATS/s = SPLIT Stats	PAESTATS/s = PAE Stats SPLTSTATS/s = SPLIT Stats
	VAAISTATS= VAAI Stats VAAILATSTATS/cmd = VAAI Latency Stats (ms)

VM

We begin with VM as that's the most important one. It complements vSphere Client by providing unmap and IO Filter metrics.

You can see at VM level, or virtual disk level. In the following screenshot, I've expanded one of the VM. The VM shown as vRealize-Operat has 3 virtual devices.

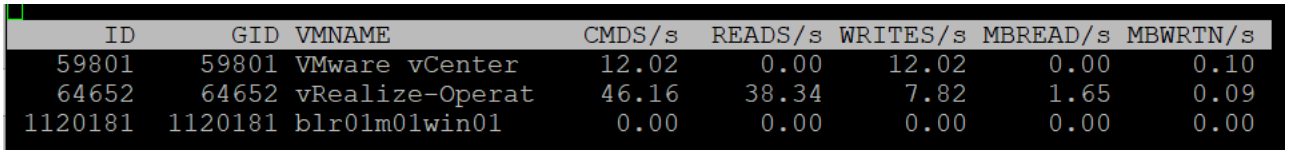


GID	VMNAME	VDEVNAME	NVDISK	CMDS/s	READS/s	WRITES/s	MBREAD/s	MBWRTN/s	LAT/rd	LAT/wr
59801	VMware vCenter	-	16	10.22	0.00	10.22	0.00	0.08	0.000	0.127
64652	vRealize-Operat	scsi0:0	-	0.41	0.41	0.00	0.02	0.00	0.272	0.000
64652	vRealize-Operat	scsi0:1	-	0.95	0.00	0.95	0.00	0.01	0.000	0.131
64652	vRealize-Operat	scsi0:2	-	0.00	0.00	0.00	0.00	0.00	0.000	0.000
1120181	blr01m01win01	-	1	0.00	0.00	0.00	0.00	0.00	0.000	0.000

Contention

LAT/rd	Average latency (in milliseconds) per read.
LAT/wr	Average latency (in milliseconds) per write.

Consumption

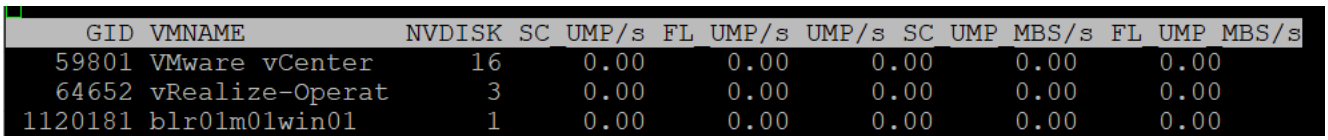


ID	GID	VMNAME	CMDS/s	READS/s	WRITES/s	MBREAD/s	MBWRTN/s
59801	59801	VMware vCenter	12.02	0.00	12.02	0.00	0.10
64652	64652	vRealize-Operat	46.16	38.34	7.82	1.65	0.09
1120181	1120181	blr01m01win01	0.00	0.00	0.00	0.00	0.00

CMDS/s	Count of disk IO commands issued per second. This is basically IOPS. Both the Read IOPS and Write IOPS are provided.
READS/s	
WRITES/s	
MBREAD/s	Total disk amount transferred per second in MB. This is basically throughput. Both the read throughput and write throughput are provided.
MBWRTN/s	

Unmap

It has unmap statistics. This can be useful that there is no such information at vSphere Client. In the UI, you can only see at ESXi level.



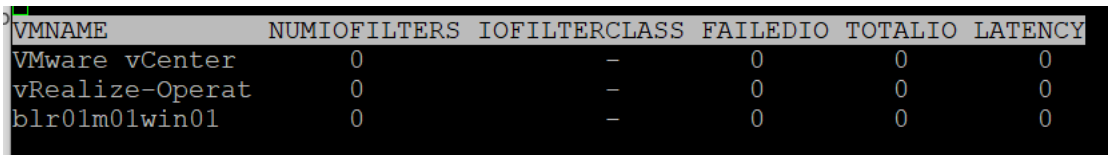
GID	VMNAME	NVDISK	SC_UMP/s	FL_UMP/s	UMP/s	SC_UMP_MBS/s	FL_UMP_MBS/s
59801	VMware vCenter	16	0.00	0.00	0.00	0.00	0.00
64652	vRealize-Operat	3	0.00	0.00	0.00	0.00	0.00
1120181	blr01m01win01	1	0.00	0.00	0.00	0.00	0.00

SC_UMP/s	Successful, Failed and Total Unmaps per second.
FL_UMP/s	Unmap can fail for a variety of reason. One example that was addressed in vSphere 6.7 Patch ESXi670-202008001 and documented in in KB is Guest OS does not refresh unmap granularities and keep sending unmap based on older value. Eventually limit is reached and the operation fail.
UMP/s	
SC_UMP_MBS/s	As above, but in MB/second.
FL_UMP_MBS/s	

IO Filter

I/O Filter in ESXi enable the kernel to manipulate the IO sent by Guest OS before processing it. This obviously opens up many use cases, such as replication, caching, Quality of Service, encryption.

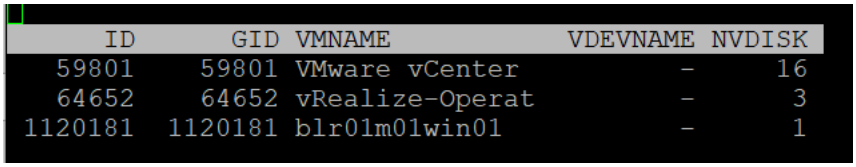
There is no such metric at vSphere Client. You will not find IO Filter metrics at both VM object and ESXi object.



VMNAME	NUMIOFILTERS	IOFILTERCLASS	FAILEDIO	TOTALIO	LATENCY
VMware vCenter	0	-	0	0	0
vRealize-Operat	0	-	0	0	0
blr01m01win01	0	-	0	0	0

NUMIOFILTERS	Number of IO Filters
IOFILTERCLASS	Type of IO Filter Class
FAILEDIO	I think Failed IO should be 0 at all times.
TOTALIO	
LATENCY	I'm unsure if this latency measures the additional overhead introduced by IO Filter, or the total latency as seen by the VM.

Configuration



ID	GID	VMNAME	VDEVNAME	NVDISK
59801	59801	VMware vCenter	-	16
64652	64652	vRealize-Operat	-	3
1120181	1120181	blr01m01win01	-	1

ID	Resource pool ID or VSCSI ID of VSCSI device.
GID	Resource pool ID.
VMNAME	Name of the resource pool.
VSCSINAME	Name of the VSCSI device.
NDK	Number of VSCSI devices

Disk Adapter

ESXi uses adapter to connect to device, so let's begin with adapter, then device.

The panel has a lot of metrics and properties, so let's group them for ease of understanding.

Errors

Since you check availability before performance, let's check the errors first. This type of problem is best monitored as accumulation within the reporting period as any value other than 0 should be investigated.

BTW, none of these metrics are available at vSphere Client UI.

FCMDS/s	Number of failed commands issued per second. How does this differ to Reset and Aborted?
FREAD/s	Number of failed read commands issued per second.
FWRITE/s	Number of failed write commands issued per second.
FMBRD/s	Megabytes of failed read operations per second.
FMBWR/s	Megabytes of failed write operations per second.
CONS/s	Number of SCSI reservation conflicts per second. This number should stay 0?
FRESV/s	Number of failed SCSI reservations per second, if the conflict can't be solved timely.
RESV/s	Number of SCSI reservations per second. This number should stay within the limit, but how to know what the limit is?
ABRTS/s	Number of commands cancelled per second.
RESETS/s	Count of disk commands reset per second.

Queue

For storage, the queue gives insight into performance problem. It's an important counter so I was hoping there will be more, such as the actual queue.

AQLEN	Current queue depth of the storage adapter. This is the maximum number of kernel active commands that the adapter driver is configured to support. This counter is not available in vSphere Client UI

Contention

You expect to get 4 sets (Device, Kernel, Guest, Queue). For each set, you expect read, write, and total. 12 metrics, and that's exactly what you got below.

DAVG/cmd	Average latency per command in milliseconds.
KAVG/cmd	It's an average number, not the last number in the reporting period. If you have 1000 IOPS, that means 5K IOPS over the 5 second reporting period.
GAVG/cmd	It's a weighted average between read and write. If the IO commands are mostly read, then high latency from write could be masked out.
QAVG/cmd	
DAVG/rd	Average read latency per read operation in milliseconds. The same set of metrics as above, except it only counts the reads.
KAVG/rd	It's useful to see read and write separately as the numbers tend to be different. More importantly, the remediation action is different.
GAVG/rd	
QAVG/rd	
DAVG/wr	The same set of metrics as above, except it only counts the writes.
KAVG/wr	
AVG/wr	

QAVG/wr	

Consumption

Now that we get the more important metrics (errors, queue, and contention) done, you then check utilization counter. In this way you have better context.

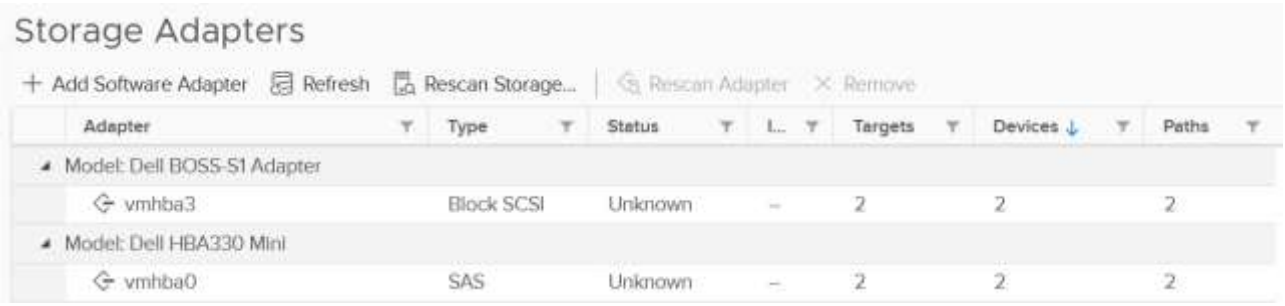
ACTV	The definition is “Number of commands that are currently active”. I don’t know how it differs to IOPS as I’m unsure what the word “active” exactly mean here.
CMDS/s READS/s WRITES/s	I combine these 3 metrics as they are basically IOPS. Total IOPS, read IOPS and write IOPS.
MBREAD/s MBWRTN/s	I combine them as they measure throughput. Interestingly, there is no total throughput metric, but you can simply sum them up. Read the string MBWRTN as MB Written.

PAE and Split

PAECMD/s	PAE Command per second and PAE Copy per second.
PAECP/s	PAE (Physical Address Extension) no longer applicable in 64-bit and modern drivers/firmware/OS, as the size is big enough. Copy operations here refer to the kernel copies the data from high region (beyond what the adapter can reach) to low region. This statistic applies to only paths.
SPLTCMD/s	Split Commands per second. Disk IO commands with large block size have to be split by the kernel. This can impact the performance as experiences by the Guest OS.
SPLTCP/s	Number of split copies per second. A higher number means lower performance

Configuration

The panel provides basic configuration. I use vSphere Client as it provides a lot more information, and I can take action on them. The following is just some of the settings available.



Compare the above with what esxtop provides, which is the following:

```
4:40:19pm up 105 days 20:47, 933 worlds, 3 VMs,

ADAPTR PATH                    NPTH
vmhba0 vmhba0:C0:T0:L0            -
vmhba0 vmhba0:C0:T2:L0            -
vmhba1 -                          0
vmhba2 -                          0
vmhba3 vmhba3:C0:T0:L0            -
vmhba3 vmhba3:C0:T2:L0            -
```

NPTH	Number of path. This should match your design. An adapter typically has more than 1 path, which is why I said it would be awesome to have a panel for path

Disk Device

The device panel has a lot of metrics and properties, so let's group them for ease of understanding.

Errors

I'm always interested in errors first, before I check for contention and utilization.

ABRTS/s	Number of commands cancelled per second. Expect this to be 0 at all times.
RESETS/s	Number of commands reset per second. Expect this to be 0 at all times.

Queue

You've seen that there is only 1 counter for queue in Disk Adapter. How many do you expect for Disk Device?

Interestingly, there are 6 metrics for queue, as shown below.

```
DEVICE                                  DQLEN WQLEN ACTV QUED %USD  LOAD
eui.0050430000000000                       31     -    0    0    0  0.00
naa.500056b323e140fd                      254     -    0    0    0  0.00
naa.5002538b00145ad0                      254     -    0    0    0  0.00
t10.ATA_____SSDSCKKB240G8R___________      31     -    0    0    0  0.00
```

LOAD	The formula is (active commands + ESXi kernel queued commands) / queue depth. If LOAD > 1, check the value of the QUED counter.
QUED	Number of commands in the kernel that are currently queued. You want this to be as low as possible, well below the queue depth.
%USD	USD (%) = ACTV / QLEN For world stats, QLEN is WQLEN. For LUN (aka device) stats, QLEN is DQLEN. Percentage of the queue depth used by ESXi kernel active commands. So this does not include the queued command? Does it mean that if this number is

	not 100%, then there is nothing in the queue, as queue should only develop when it's 100% used? Obviously when Used = 100% it means the queue is full. That will introduce outstanding IO, which in turn will increase latency
DQLEN WQLEN	I combine this together as a device can have 1 or more world, and there is a per-device maximum. DQLEN is the device configured queue length. The corresponding counter for adapter is called AQLEN WQLEN is the world queue depth. The manual states "This is the maximum number of ESXi kernel active commands that the world is allowed to have". So it does not look like the queue at present. So we show maximum value for each world, and the present value for each device.
ACTV	The definition is "Number of commands that are currently active". I think this means the IO in flight. This is worth profiling and I expect it to be small most of the time.

Contention

See Disk Adapter as both sport the same 12 metrics.

Consumption

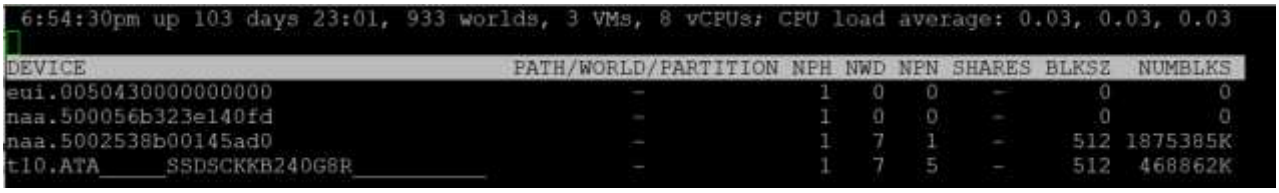
See Disk Adapter as both sport the same 5 metrics.

PAE and Split

See Disk Adapter as both sport the same 4 metrics.

Configuration

As you can expect, esxtop provides minimal configuration information. They are shown below.



Path/World/Partition

They are grouped as 1 column, and you can only see one at a time.

By default, none of them is shown. To bring up one of them, type the corresponding code. In the following screenshot, I've type the letter **e**, which them prompted me to enter one of the device.

```
 7:42:30pm up 103 days 23:49, 933 worlds, 3 VMs, 8 vCPUs;
Device to expand/rollup (device): naa.5002538b00145ad0
DEVICE                                  PATH/WORLD/PARTITIO
eui.0050430000000000                                    -
naa.500056b323e140fd                                    -
naa.5002538b00145ad0                                    -
t10.ATA_____SSDSCKKB240G8R___________                   -
```

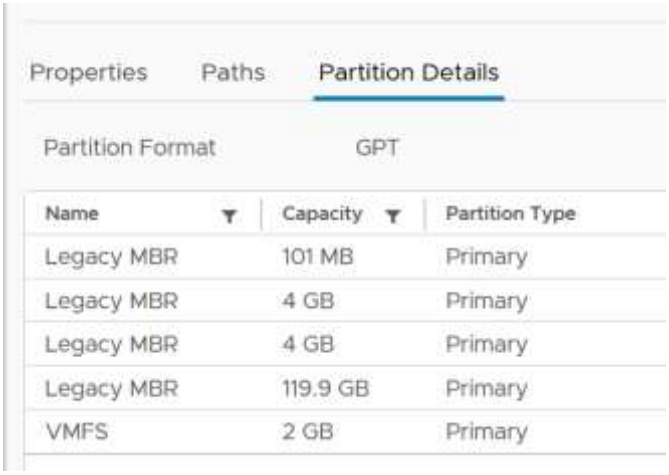
Path is obviously the path name, such as vmhba0:C0:T0:L0.

A disk device can have >1 world, which I'm unsure why. You can see each world ID, and you get the statistics per world.

```
 7:36:13pm up 103 days 23:43, 933 worlds, 3 VMs, 8 vCPUs; CPU load aver

DEVICE                                  PATH/WORLD/PARTITION NPH NWD NPN
eui.0050430000000000                                       -   1   0   0
naa.500056b323e140fd                                       -   1   0   0
naa.5002538b00145ad0                                 2097209   1   6   1
naa.5002538b00145ad0                                 2099383   1   6   1
naa.5002538b00145ad0                                 2099698   1   6   1
naa.5002538b00145ad0                                 2106123   1   6   1
naa.5002538b00145ad0                                 2106767   1   6   1
naa.5002538b00145ad0                                 2291223   1   6   1
t10.ATA_____SSDSCKKB240G8R___________                      -   1   7   5
```

Partition shows the partition ID. Typically this is a simple number, such as 1 for the first partition. vSphere Client provides the following, which is more details yet easier.



Others

Let's cover the rest of the metrics.

NPH	Number of paths. This should not be 1 as that means a single point of failure.
NWD	Number of worlds. If you know the significance of this in troubleshooting, let me know.
NPN	Number of partitions. Expect this to be 1 for VMFS
SHARES	Number of shares. This statistic is applicable only to worlds. This is interesting, as that means each world can have their own share? Where do we set them then?

BLKSZ	Block size in bytes. I prefer to call this sector format. International Disk Drive Equipment and Materials Association (IDEMA) increased the sector size from 512 bytes to 4096 bytes (4 KB). This is important, and you want them to be in 4K (Advanced Format) or at least 512e (e stands for emulation). Microsoft provides additional information here.
NUMBLKS	Number of blocks of the device. Multiply this with the block size and you get the total capacity. In vSphere UI, you get the capacity, which I think it's more relevant.

For configuration, I use vSphere Client as it provides a lot more information, and I can take action on them. The following is just some of the settings available.



VAAI

VMware vSphere Storage APIs - Array Integration (VAAI) offloads storage processing to the array, hence improving performance or reducing overhead. This is obviously vendor-dependant. There is no VAAI counter at adapter level or path level, as the implementation is at back-end array.

The VAAI has a lot of metrics. There are essentially 2 types of metrics: non latency and latency metrics.

As with metrics, check for contention type of metrics first. There are metrics that track failed operations, such as CLONE_F, ATSF and ZERO_F.

In this book, I'm grouping them by function as it's easier to understand.

I saw this note from VMware vSphere Storage APIs – Array Integration (VAAI) document by Cormac Hogan, which I think it's worth mentioning. Because the nature of VAAI as an offloads, you will see higher latency value of KAVG metric. Other latency metrics are not affected, so there is no issue unless there are other symptoms present.

At this moment, I have not found the need to document them further. So what you get here is mostly from the KB article above. Andreas Lesslhumer also has useful information in this blog article. Other references are this blog by Cormac and this this KB article.

Extended Copy

Hardware Accelerated Move (the SCSI opcode for XCOPY is 0x83)

Clone_RD	RD stands for reader.
Clone_WR	The number of CLONE commands successfully completed where this device was a source.
Clone_F	WR stands for writer. The number of CLONE commands successfully completed where this device was a destination The number of failed CLONE commands

LCLONE_RD	The same set of 3 metrics, except for Linked Clone.
LCLONE_WR	
LCLONE_F	
MBC_RD/s	MBC = megabytes of clone data.
MBC_WR/s	RD/s is read per second, and WR/s is written per second
AVAG/suc	The average clone latency per successful command
AVAG/f	The average clone latency per failed command

Atomic Test & Set

Hardware Accelerated Locking on Single Extent Datastore or on Multi Extent Datastore (SCSI code 0x89).

ATS	The number of Atomic Test & Set (ATS) commands successfully completed
ATSF	The number of ATS commands failed. Expect this to be 0?
AAVG/suc	The Average ATS latency per successful command
AAVG/f	The Average ATS latency per failed command

Write Same

Hardware Accelerated disk space initialization by writing 0s on all the blocks for faster future operations. The SCSI code for WRITE SAME operations is 0x93 or 0x41.

ZERO	The number of ZERO commands successfully completed
ZERO_F	The number of ZERO commands failed
MBZERO/s	The megabytes zeroed per second
ZAVG/suc	The average zero latency per successful command
ZAVG/f	The average zero latency per failed command

Unmapped

Unmapped block deletion (SCSI code 0x42). We discussed unmapped block (TRIM) in earlier chapter.

DELETE	The number of successful DELETE commands
DELETE_F	The number of failed UNMAP commands, this value should be 0
MBDEL/s (MB/s)	The rate at which the DELETE command getting processed. Measured in Megabytes per second

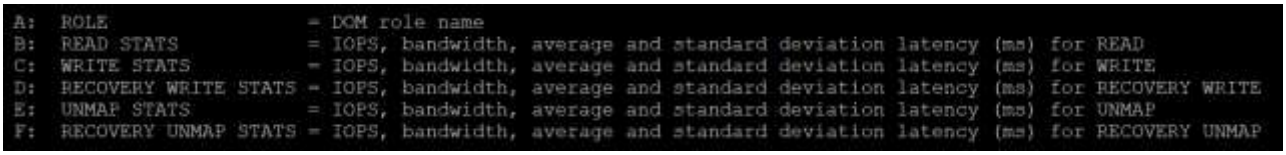
Others

RESSPACE	Reservation Space.
RESSPACE_F	The number of commands which were successful while doing space reservation for a VMDK file in thick Provisioning format. RESSPACE_F captures the failure.

EXTSTATS	Extended Statistics
EXTSTATS_F	The number of commands which were successful in reporting extended statistics of a clone after the cloning process had been completed. EXTSTATS_F captures the failure
CAVG/suc	The average clone latency per successful command. Unit is millisecond per clone.
CAVG/f	CAVG/f captures the failures.
LCAVG/suc	As per above, but for Linked Clone.
LCAVG/f	
RAVG/suc	The average latency (in ms) per successful VAAI Space Reservation command.
RAVG/f	RAVG/f captures the failures
ESAVG/suc	As per above, but for Extended Statistics
ESAVG/f	

vSAN

I group the vSAN panel under Disk as esxtop only covers storage related information. There is no network or compute (vSAN kernel modules).



The panel provides visibility into 5 types of IO operations:

- Read
- Write
- Recovery Write
- Unmap
- Recovery Unmap

For each, it provides the IOPS, bandwidth, average latency (ms) and standard deviation latency (ms). Take note that some use MB, while others use GB.

ROLE	The Distributed Object Manager (DOM) role of that component, such as client, owner, and component manager.
READS/s MBREAD/s AVGLAT SDLAT	Reads/second is the number of reads operations. This is IOPS. MBReads/s is read throughput in Megabytes/second. AvgLat is the average latency. Standard deviation of latency, when above 10ms latency.
WRITES/s MBWRITE/s AVGLAT	Same set of metrics, like above, but for write

SDLAT	
RECOWR/s MBRECOWR/s AVGLAT SDLAT	Same set of metrics, like above, but for Recovery Write. Recovery covers component rebuild task (e.g. from disk failure). Read the string MBRECOWR as MB Reco Wr.
UNMAPS/s GBUNMAP/s AVGLAT SDLAT	Same set of metrics, like above, but for unmap operations. I think this number should be within your expectation, as excessive unmap can impact performance. GBUNMAP/s = Unmapped rates in Gigabytes/second Read the string GBUNMAP as GB Unmap
RECOUN/s GBRECOUN/s AVGLAT SDLAT	Same set of metrics, but for Recovery Unmap operations. Read the string GBRECOUN as GB Reco Un. RecoUn/s is the number of recovery unmapped operations per second. GBRecoUn/s is the amount of disk space in GB/second by Recovery Unmapped.

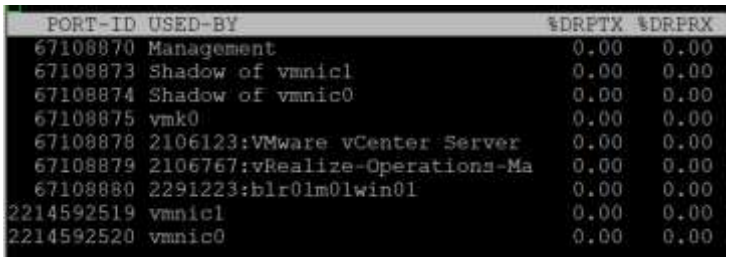
Network

Take note that the network panels mix the virtual and physical networks.

Focus on the virtual network first as that's closer to the VM and kernel.

Contention

As usual, we check contention first. There are no network latency and packet retransmit metrics.



PORT-ID	USED-BY	%DRPTX	%DRPRX
67108870	Management	0.00	0.00
67108873	Shadow of vmnic1	0.00	0.00
67108874	Shadow of vmnic0	0.00	0.00
67108875	vmk0	0.00	0.00
67108878	2106123:VMware vCenter Server	0.00	0.00
67108879	2106767:vRealize-Operations-Ma	0.00	0.00
67108880	2291223:blr01m01win01	0.00	0.00
2214592519	vmnic1	0.00	0.00
2214592520	vmnic0	0.00	0.00

%DRPTX	Percentage of Dropped Packet.
%DRPRX	Expressed in percentage, which makes it easier as you expect this not to exceed 0.x%. In dedicated network such as vSAN and vMotion, this should be flat 0% non-stop for every single ESXi. Transmit and Receive have different nature. A high drop in transmit means your physical NIC card or uplink switch is unable to cope. A high drop in receive means your ESXi or VM may not have enough CPU to process the packet, or the ring buffer size is too small. esxtop show dropped packets: The screen output show dropped receive packets at the virtual switch port. They are actually dropped between the virtual switch and

	the guest OS driver. The dropped packets can be reduced by increasing the Rx buffers for the virtual network driver.

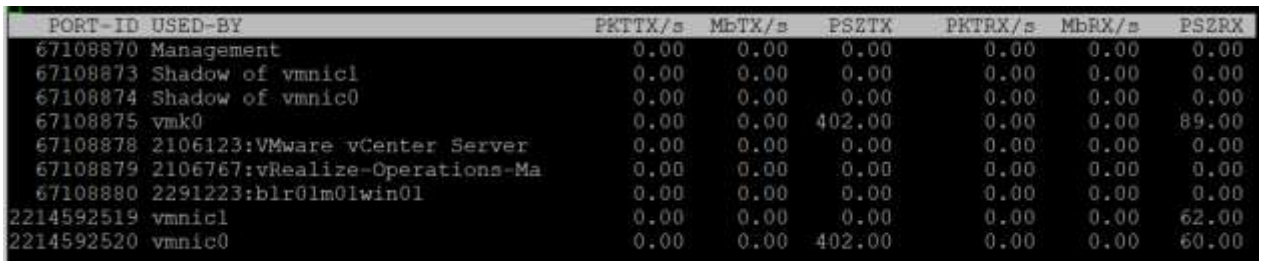
Consumption

As usual, check the non-unicast packets first and make sure they match the expectation at that time.

Non-Unicast Packets

PKTTXMUL/s	Number of multicast packets transmitted or received per second.
PKTRXMUL/s	Read the string PKTTXMUL as Pkt Tx Mul, which is Packet TX Multicast. Same with PKTRXMUL.
PKTTXBRD/s	Number of broadcast packets transmitted or received per second.
PKTRXBRD/s	Read the string PKTTXBRD as Pkt Tx Brd, which is Packet TX Broadcast. Same with PKTRXBRD

All Packets



PORT-ID	USED-BY	PKTTX/s	MbTX/s	PSZTX	PKTRX/s	MbRX/s	PSZRX
67108870	Management	0.00	0.00	0.00	0.00	0.00	0.00
67108873	Shadow of vmnic1	0.00	0.00	0.00	0.00	0.00	0.00
67108874	Shadow of vmnic0	0.00	0.00	0.00	0.00	0.00	0.00
67108875	vmk0	0.00	0.00	402.00	0.00	0.00	89.00
67108878	2106123:VMware vCenter Server	0.00	0.00	0.00	0.00	0.00	0.00
67108879	2106767:vRealize-Operations-Ma	0.00	0.00	0.00	0.00	0.00	0.00
67108880	2291223:blr01m01win01	0.00	0.00	0.00	0.00	0.00	0.00
2214592519	vmnic1	0.00	0.00	0.00	0.00	0.00	62.00
2214592520	vmnic0	0.00	0.00	402.00	0.00	0.00	60.00

PKTTX/s	This is the total packets, so it includes multicast packet and broadcast packet.
PKTRX/s	Multicast packet and broadcast packet are listed separately. This is handy as they are supposed to low most of the time.
MbTX/s	This is measured in bit, unlike vCenter Client UI which shows in byte.
MbRX/s	Packet length is typically measured in bytes. A standard packet is 1500 bytes, so a 10 Gb NIC would theoretically max out at 833,333 packets on each direction. Compare this with your ESXi physical network card.
PSZTX	This is convenient. If you see a number far lower than 1500, it's worth discussing with network team.
PSZRX	

There is another metric **ACTN/s**, which is the number of actions per second. The actions here are the kernel actions. It is an internal counter, not relevant to day to day operations.

Configuration

This panel mixes physical and virtual. For virtual, it shows both the kernel network and VM network. I find it easier to use the information in vSphere Client.

```
   PORT-ID UPLINK UP SPEED FDUPLX USED-BY                                TEAM-PNIC DNAME
  67108870      N  -     -      - Management                                 n/a vSwitch0
  67108873      N  -     -      - Shadow of vmnic1                           n/a vSwitch0
  67108874      N  -     -      - Shadow of vmnic0                           n/a vSwitch0
  67108875      N  -     -      - vmk0                                    vmnic0 vSwitch0
  67108878      N  -     -      - 2106123:VMware vCenter Server           vmnic1 vSwitch0
  67108879      N  -     -      - 2106767:vRealize-Operations-Ma          vmnic1 vSwitch0
  67108880      N  -     -      - 2291223:blr01m01win01                   vmnic0 vSwitch0
2214592519      Y  Y 10000      Y vmnic1                                       - vSwitch0
2214592520      Y  Y 10000      Y vmnic0                                       - vSwitch0
```

PORT-ID	Virtual network device port ID.
UPLINK	'Y' means that the corresponding port is an uplink. 'N' means it is not. The physical NIC cards (vmnic0, vmnic1, etc.) serve as the uplink
UP	'Y' means that the corresponding link is up. 'N' means it is not.
SPEED	Link speed in Megabits per second.
FDUPLX	'Y' means the corresponding link is operating at full duplex. 'N' means it is not, which is a problem.
USED-BY	Virtual network device port user.
DNAME	Virtual network device name.

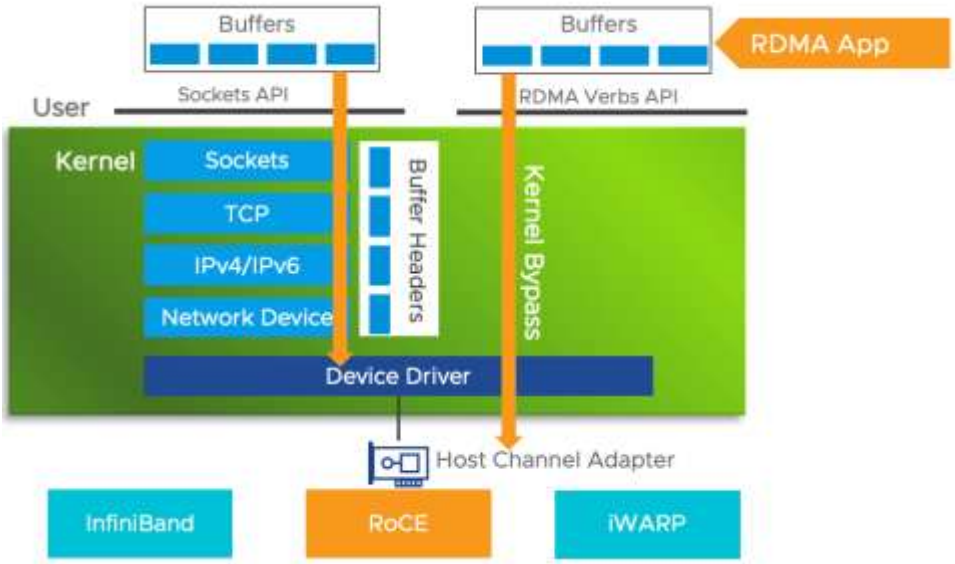
The metric DTYP (Virtual network device type, where H means Hub and S means switch) does not seem to be available anymore.

vSphere Client separates the components. You can see the virtual switches, the kernel network and physical cards. The level of details is more comprehensive.



RDMA Device

Remote Direct Memory Access (RDMA) enable direct access to the physical network card, bypassing the OS overhead. The following screenshot, taken from here, shows 2 types of access from application (that lives inside a VM. The VMs are not shown).



Usage

Since it's about network, you get both the TX (transmit or sent) and RX (received or incoming).

For contention, there is only packet dropped. There is no packet retransmit or latency. The metrics are:

%PKTDTX	Percentage of packet dropped relative to number of packets sent.
%PKTDRX	

For utilization, you get them in both amount of data, and number of packets. Both are important metrics. There is no breakdown on the type of packets (broadcast, multicast, unicast).

PKTTX/s	Packets per second. Check the limit for packet per second in your specific card.
PKTRX/s	
MbTX/s	Network throughput in Megabit/second.
MbRX/s	

There is no packet size. This can be handy to determine if they are much smaller or larger than you expect. For example, if you expect jumbo frame but the reality is much smaller.

These metrics are not available in vSphere Client UI, so you need to use esxtop to get the visibility. Just in case you're wondering where I got the following screenshot from, they are courtesy of Shoby Cherian and Aditya Kiran Pentyala.

```
6:55:55pm up 10 min, 669 worlds, 0 VMs, 0 vCPUs; CPU load average: 0.00, 0.01, 0.00
```

NAME	TEAM-PNIC	PKTTX/s	MbTX/s	PKTRX/s	MbRX/s	QP	MR
vmrdma0	vmnic2	44820.60	25.30	1951328.36	16286.88	2	4
vmrdma1	vmnic3	4876380.02	40700.99	111792.39	63.12	2	4

You also get the queue usage information.

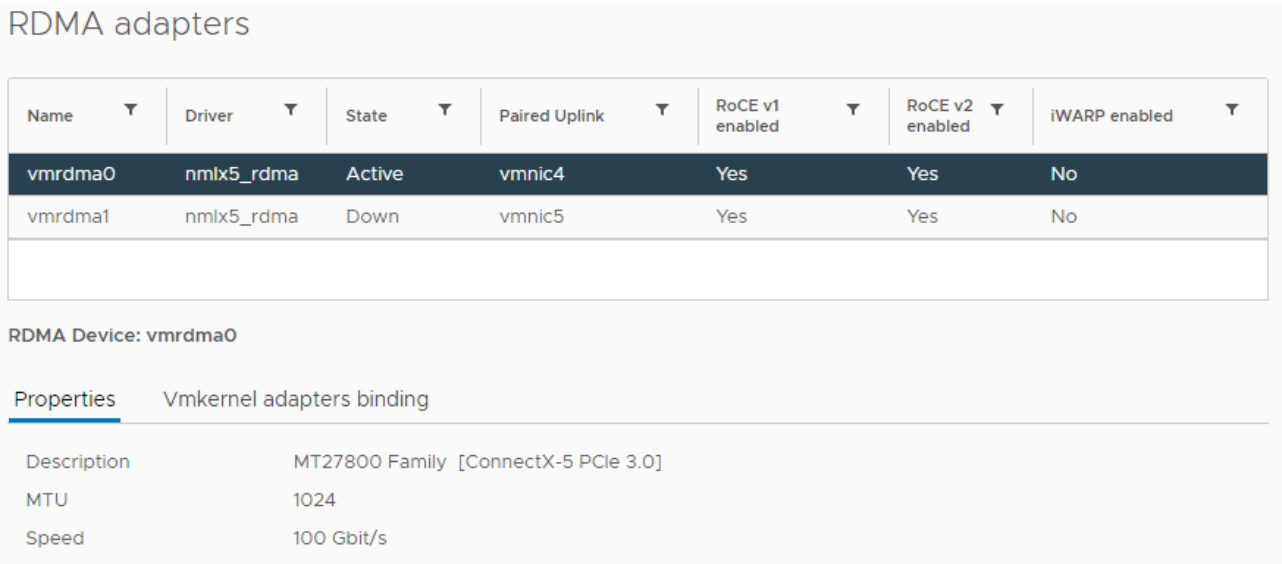
QP	Number of Queue Pairs Allocated and Completion Queue Pairs Allocated.

CQ	RDMA uses these queues for communication.
SRQ	Number of Shared Receive Queues Allocated I think this is required in virtualization as the physical NIC card can be shared.
MR	Memory Regions Allocated. Check that this is inline with your expectation.

For more reading on RDMA, I found this academic paper, title "Understanding the concepts and mechanisms of RDMA" useful.

Configuration

vSphere Client provides the following information. You get the first 4 columns in esxtop.



The information you get in esxtop covers the first 4 columns in the preceding screenshot. They are:

NAME	Name of the device
DRIVER	Name of the driver
STATE	Active or down
TEAM-PNIC	The physical Network Interface Card that the RDMA adapter is paired with.

MS Windows

Part 3 Chapter 8

Microsoft Windows and Linux have different techniques for resource management. They also have different metrics names and accounting strategy.

Introduction

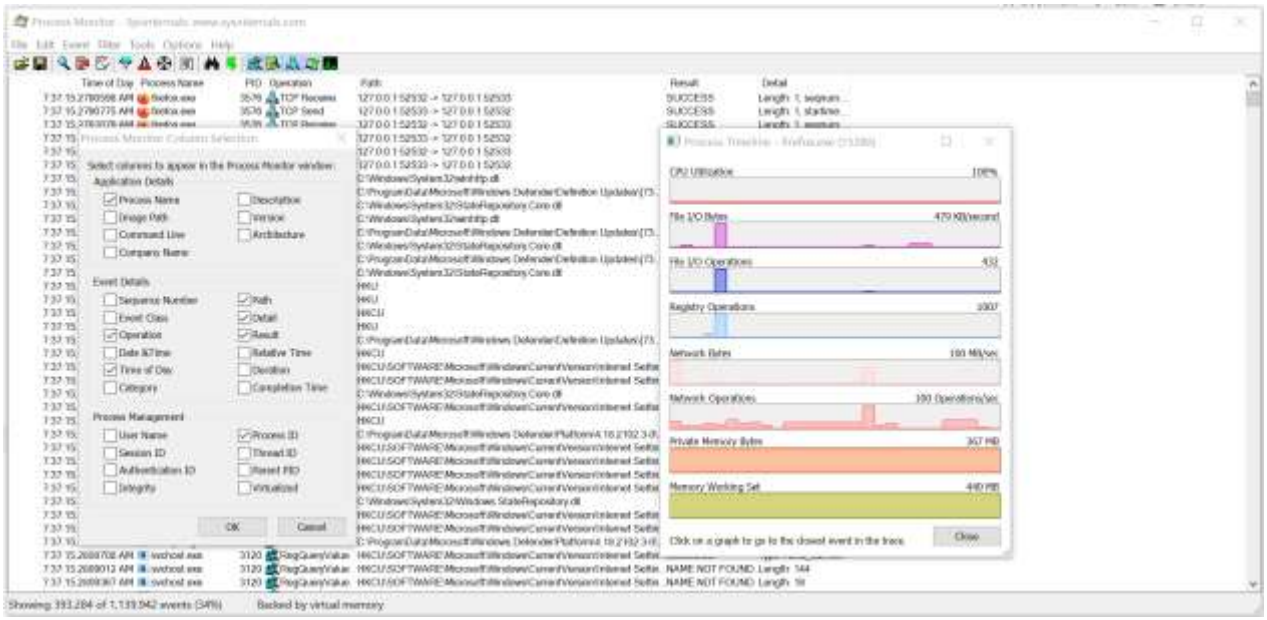
There are 3 tools in Windows:

- Performance Monitor. This is the oldest one, and is superceded by the next two.
- Task Manager
- Resource Monitor

They use different names. It is also possible that they use different formula.

One major difference between Guest OS and VM is that Windows/Linux runs a lot of threads and processes. The problem is there is minimal observability on individual thread or process. For example, there is no CPU queue metric, memory page fault, network dropped packet, and disk latency at process level.

The following shows Windows Sysinternal[22], a great tool for Windows troubleshooting. As you can see, they are just utilization metrics. There is no contention metric.

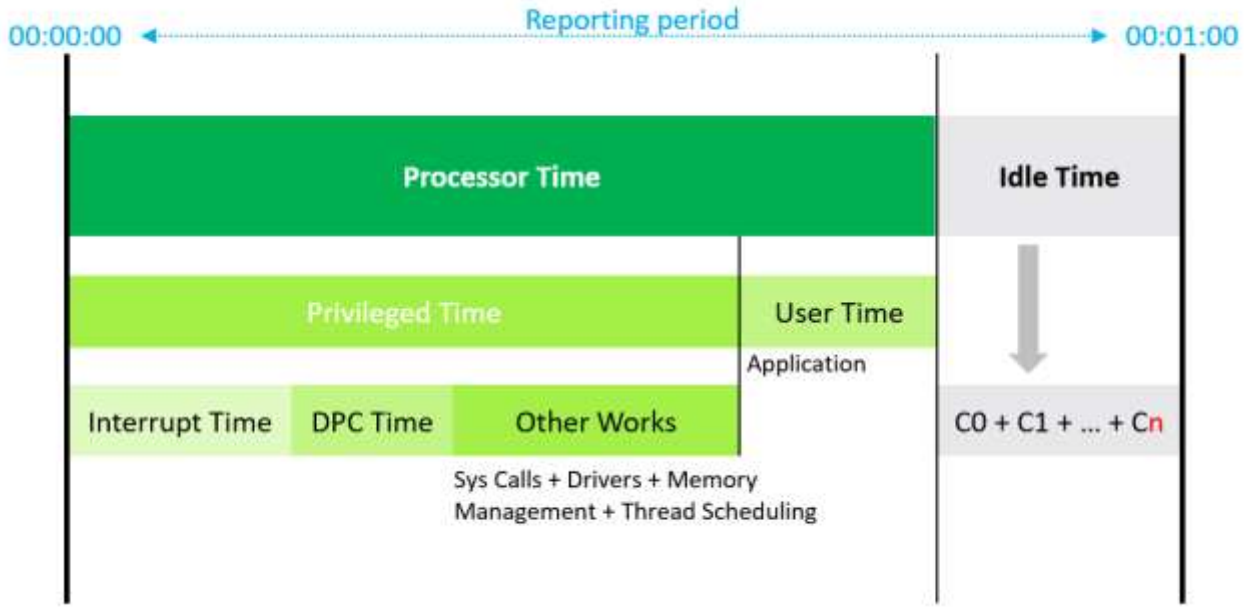


[22] Your anti-walware may classify it as a threat even though it is obviously not a malware. Malware uses psexec, a component of sysinternal, to execute commands on remote machines. Some of the executables also behave in such a way that anti-malware software classifies them as malicious.

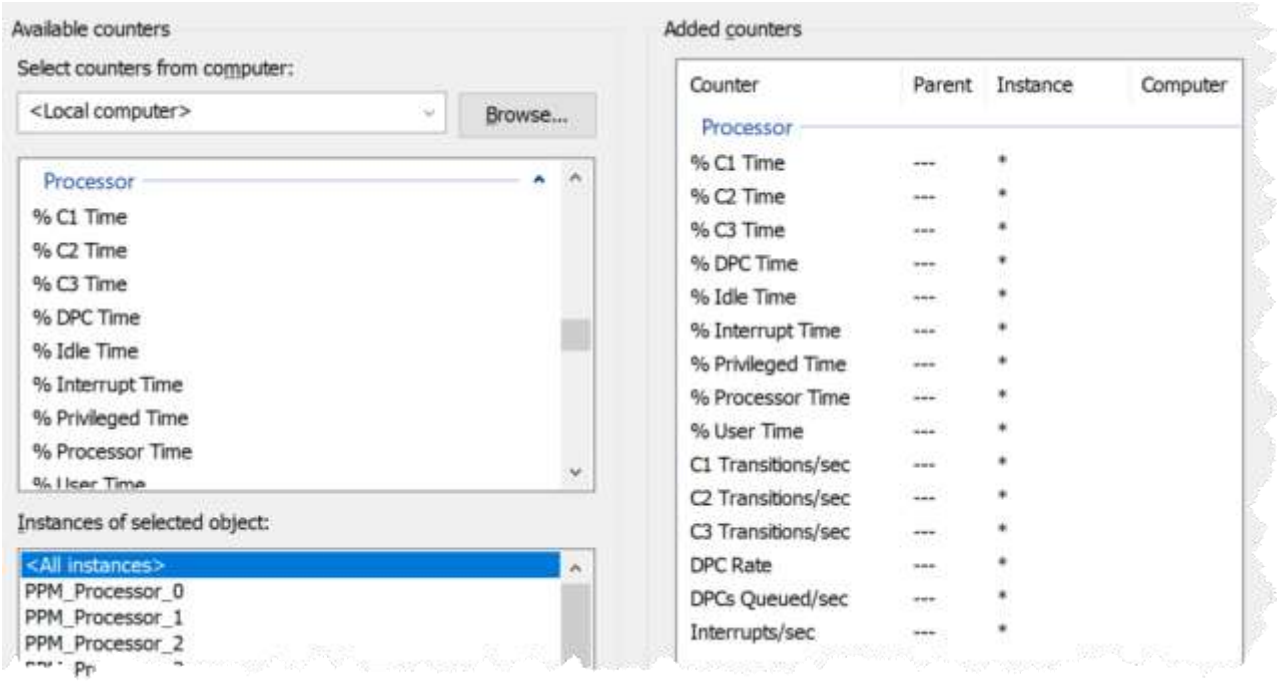
CPU

Performance Monitor is still the main tool for Microsoft Windows, despite the fact it has not been enhanced for decades. Go to docs.microsoft.com and browse for Windows Server. It took me to this article, which cover PerfMon. Many explanations on metrics at https://learn.microsoft.com/ are still based on end-of-life Windows.

PerfMon groups the counters under Processor group. However, it places the Processor Queue Length and Context Switches metrics under the System group. The System group covers system wide metrics, not just CPU.



The following screenshot shows the counters under Processor group.



The counters can be grouped into 2:

- Time-based
- Non time-based.

PerfMon UI provides a description, which I use as a reference below.

% Processor Time % Idle Time	These 2 metrics add up to 100%
Priority Time	It is a subset of % Processor Time — specifically the portion of busy CPU time consumed by threads running at elevated scheduling priorities
% User Time % Privileged Time	These 2 metrics add up to 100%. A program's process can switch between user mode and kernel mode (when executing system service call). This does not incur CPU context switch as it's the same thread. As a result, I'm not seeing the use case of knowing the split between kernel mode and user mode. Reference: Windows
% DPC Time	Deferred Procedure Calls (DPC). According to this, this counter is a part of the Privileged Time (%) because DPCs are executed in privileged mode. They are counted separately and are not a component of the interrupt counters.
% Interrupt Time	Interrupt means the processor was interrupted from executing normal thread. This can happen for a variety of reasons, such as system clock, incoming network packets, mouse and keyboard activity. Interrupt can happen on regular basis, not just ad hoc. For example, the system clock does it every 10 milliseconds in the background.
	A high interrupt value can impact performance.
% C1 Time % C2 Time % C3 Time	Based on this April 2004 article, Windows can operate in 4 different power levels. The C0 is the highest, while C3 consumes the least amount of power. C1 + C2 + C3 may not add to Idle Time. If you set dynamic power management, expect the lower power to be registering higher value during idle period. Reference: here.

The non time-based counters are:

C1 Transitions/sec C2 Transitions/sec C3 Transitions/sec	The amount of time on each power level xdoes not tell the full picture. You also need to know how frequent you enter and exit that level. These 3 metrics track the number of transitions into the respective level. For example, a high number on all 3 counters mean Windows is fluctuating greatly, resulting in inconsistent speed.
DPC Rate	This is an input to the above, as the above is calculated as the delta of 2 rates, divided over sampling period.
Interrupts/sec	As above, but for interrupts.

System Calls/second:

- It is not a CPU metric but is commonly used next to the queue and context-switch counters.
- A call is a request to the kernel to do work on behalf. It occurs because the program cannot perform a task itself because it involves protected resources (like hardware or memory belonging to other processes). This instruction forces the CPU to switch from Ring 3 (User Mode) to Ring 0 (Kernel Mode).
- It measures the number of times a software application transitions from User Mode to Kernel Mode to request a service from the kernel.
- If System Calls/sec is high and Privileged Time (%) is also high (e.g., >30%), the CPU is spending too much time on OS management rather than running your actual application code.
- Values consistently exceeding 20,000 per processor core are generally considered high and warrant investigation. In most cases, it's below 2000 per core.
- This metric is not collected by Tools.

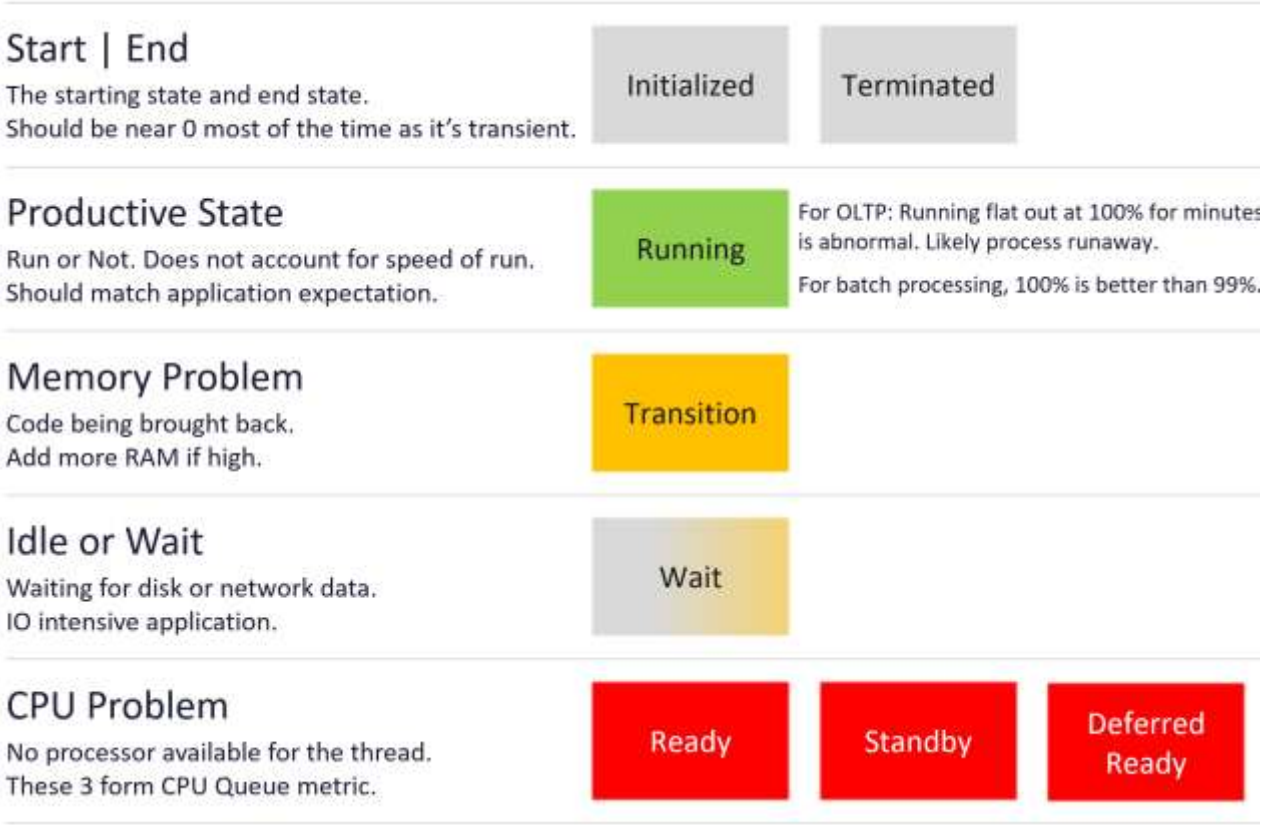
Thread State

In Windows, an application consists of ≥ 1 process. A process consists of ≥ threads. Thread is what is scheduled in the CPU.

It's easier to understand the CPU scheduling when you think in terms of 2 distinct objects:

- Processors
- Threads.

The following information is taken from the famous Windows Internal book, 7th edition and from this Microsoft link. A thread is always in one of these 8 possible states.



You may see a higher or lower number than the number of processors as the observability tools counts the threads, not the processors. Windows PerfMon or WMI code cycles through each thread and get the state. It collects the state at that particular moment. In the 100 milliseconds or so it takes to finish counting thousands of threads, the "Standby" slots on the CPUs could have been emptied and refilled hundreds of times.

CPU Issue

This means there is not enough CPU, or there is the performance issue is with CPU.

Ready	The thread is waiting to use a processor because all processors are busy running. The thread is prepared to run on the next available processor. When looking for a thread to execute, the CPU dispatcher considers only the threads in this state.
	It also counts when a thread is waiting to be in-swapped after completing a wait state.
	It also counts when a thread was interrupted by higher priority thread, as preemption moves the thread into Ready state. However, it puts the thread at the front of the queue, so it can complete its remaining time slice.
	The number should be near 0 for most of the time. However, it can be high if you have many high-priority threads constantly "bumping" lower-priority threads. Even though work is technically getting done, the lower-priority threads are "stacking up" in the Ready state, waiting for their chance to finish their interrupted tasks.
Standby	The thread has been selected by Windows as the next thread to run on this processor.

	Only one thread can be in the standby state for each processor on the system. A Windows OS with 10 processors will have up to 10 threads. When the correct conditions exist, the dispatcher performs a context switch to this thread. Naturally, just because you're next to be run, does not mean you get to run. A higher priority, typically IO interrupt, can be the thread that got the CPU time.
Deferred Ready	When a thread is ready to run (e.g., an I/O operation finishes), the kernel needs to pick a processor for it. Instead of immediately locking that target processor's Ready queue, the kernel puts the thread into a general Deferred Ready list. So the thread has ben selected to run on a specific processor but have not actually started running there. This non-processor specific state exists so that the kernel can minimize the amount of time the per-processor lock on the scheduling database is held. The lock would slow down the CPU currently doing work.
	In Performance Monitor and WMI (Win32_PerfRawData_PerfProc_Thread), the label is typically "Unknown." However, in the actual Windows Kernel logic, it actually means Deferred Ready.

Non-CPU Issue

CPU is much faster than memory, disk or network. These 2 states capture times where CPU is waiting for data.

Idle or Wait	"Waiting" is the natural state for 99% of threads. At any given moment, Windows might have 3,000 threads, but 2,950 of them are "Waiting" for user interaction or for a background update to ping a server.
	Possible reasons: • A thread voluntarily waits for an object to synchronize its execution • The OS waits on the thread's behalf (such as to resolve a paging I/O) • An environment subsystem can direct the thread to suspend itself.
	When the thread's wait ends, depending on its priority, the thread either begins running immediately or is moved back to the ready state.
	The performance issue could be disk or network.
Transition	The thread is waiting as its kernel stack was paged out of memory. After its kernel stack is brought back into memory, the thread enters the ready. You should notice a corresponding increase in Memory Page Faults counter.
	This indicate the performance issue is with memory, not CPU. Increase memory to the application.

Other States

Initialized	A state that indicates the thread is being created, hence it cannot be run yet.

	The value at this state should be near 0 most of the time. A high value indicates a problem. Check if the application is spawning threads excessively. If not, check why a quick process is taking a long time.
Terminated	A state that indicates the thread has finished executing and has exited. This is not a performance problem if it ends as per planned.
Running	A state that indicates the thread is currently using a processor. This is the productive state, meaning work is being done, hence it's the only state in green.
	The value should be close to the sum of infrastructure team sizing and application team sizing. Infrastructure team size based on Windows or Linux, plus the standard software such as VMware Tools and Security Agents. This is fairly standardized across all virtual machines. Application team size based on their application workload. It can be profiled by before versus after of the application loading.

Run Queue

Number of threads in the processor queue. Unlike Linux, Windows excludes the threads that are running (being executed).

Let's take a VM configured with 8 vCPUs. The Guest OS sees 8 threads so it will schedule up to 8 parallel processes. If there is more demand, it will have to queue them. This means the queue needs to be accounted for in Guest OS sizing.

Because it reports the queue, this is the *primary* counter to measure Guest OS CPU performance. It tells you if the CPU is struggling to serve the demand or not.

What is a healthy value?

Here is my take:

- 2 - 4 per processor
- 4 – 8 per processor
- > 8 per processor

Windows Performance Monitor UI description is not consistent with MSDN documentation (based on Windows Server 2016 documentation). The description shown in Windows UI is "Processor Queue Length is the number of threads in the processor queue. Unlike the disk counters, this counter shows ready threads only, not threads that are running. There is a single queue for processor time even on computers with multiple processors. Therefore, if a computer has multiple processors, you need to divide this value by the number of processors servicing the workload. A sustained processor queue of less than 10 threads per processor is normally acceptable, dependent of the workload."

MSDN document states that a sustained processor queue of greater than 2 threads generally indicate processor congestion. SQL Server document states 3 as the threshold. Let me know if you have seen other recommendations from Microsoft or Linux.

Windows or Linux utilization may be 100%, but as long as the queue is low, the workload is running as fast as it can. Adding more vCPU will in fact slow down the performance as you have higher chance of context switching.

You should profile your environment, because the number can be high for some VMs. Just look at the numbers I got below. I profiled 3385 Windows VM in the last 1 month. I plotted the highest value.



Name	95P Queue	Max Queue	Net Run at Max (at Guest|Peak vCPU Queu...	Context Switch at Max (at Guest|Pe...	vCPU	Guest OS
IRV...	53.38	58.3	51.78 %	1,380.53	8 vCPUs	Microsoft Windows Server 2008 R2 (64-bit)
WPD...	13.8	20.78	63.44 %	18,897.47	4 vCPUs	Microsoft Windows Server 2019 (64-bit)
saps...	10.21	14.8	32.71 %	4,577.13	1 vCPUs	Microsoft Windows Server 2008 R2 (64-bit)
NT-I...	8.53	14.87	12.91 %	2,574.33	1 vCPUs	Microsoft Windows Server 2019 (64-bit)
Wind...	6.58	7.22	2.06 %	2,895.47	4 vCPUs	Microsoft Windows Server 2012 (64-bit)
Wind...	5.93	6.58	1.51 %	2,916.6	4 vCPUs	Microsoft Windows Server 2012 (64-bit)
contr...	5.23	6.43	97.79 %	23,082.2	2 vCPUs	Microsoft Windows Server 2016 or later (64-bit)
NT-I...	5.17	6.7	56.76 %	8,579.87	2 vCPUs	Microsoft Windows Server 2008 R2 (64-bit)
cho-...	4.9	8.87	72.21 %	53,631.6	4 vCPUs	Microsoft Windows Server 2019 (64-bit)
NTL...	4.23	17.33	74.25 %	2,295.13	2 vCPUs	Microsoft Windows Server 2016 or later (64-bit)
NT-I...	4.13	5.23	9.61 %	2,624	2 vCPUs	Microsoft Windows Server 2008 R2 (64-bit)
tsimc...	3.93	11.6	31 %	1,576.67	1 vCPUs	Microsoft Windows Server 2022 (64-bit)
NT-I...	3.57	4.47	18.68 %	5,028.07	2 vCPUs	Microsoft Windows Server 2008 R2 (64-bit)
NT-I...	3.54	4.6	9.56 %	1,587.93	2 vCPUs	Microsoft Windows Server 2008 R2 (64-bit)
Wind...	2.88	4.95	1.51 %	2,251.33	4 vCPUs	Microsoft Windows Server 2012 (64-bit)
NT-I...	2.56	3.07	5.56 %	5,628.07	6 vCPUs	Microsoft Windows Server 2008 R2 (64-bit)

29 Mar 2026, 15:23 - 5 Apr 2026, 15:23 1 - 50 of 3378 items 1 2 3 4 5 ... 68

Columns explanation:

- 95P means the value at 95th percentile during the last 1 week.
    - I take 95th percentile because there are 30240 data points.
    - While the collection is every 5 minutes, the value is a snapshot taken every 20 seconds. It is the highest among 15 data points. There are 15 data points * 7 days * 24 hours * 12 = 30240 data points.
- Max Queue means the highest in the last 1 week. That means it is the highest of 30240 data points.
- Net Run at Max means the value of CPU Net Run when the value of vCPU Queue is at the highest. Using the first VM as an example, Net Run showed 52% when CPU Queue was at 58%. It's a correlated number.
- Context Switch as Max means the value of Microsoft Windows CPU Context Switch when the value of vCPU Queue is at the highest.
- vCPU is the present configured CPU. This number is generally stable, hence I took the present.

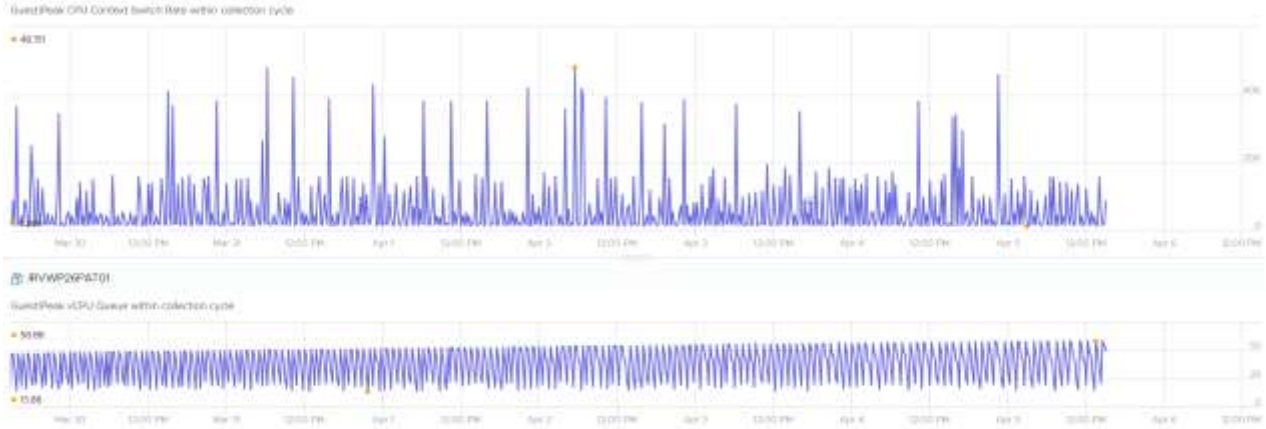
High CPU Queue

Let's drill down to the first VM in the preceding table.

How can an 8 vCPU Windows Server VM have CPU Queue of 53 when only running at 52%?

We expect the number to be around 8 since only half the CPU are running. Basically it should be 0 Ready + 8 Standby + 0 Deferred Ready.

What do you notice from the following chart?



There seems to be no strong correlation between CPU Queue and CPU Context Switch.

The CPU Queue:

- Fluctuates from 14 to 59 constantly. The pattern is repeatable. When I zoomed in, each cycle has a sharp jump within 5 – 10 minutes, then slowly drops over 1 hour.
- Trend upwards over 7 days. It's a gentle trend. This means something that slowly deteriorates caused it. Memory leak?

FYI:

- CPU Ready, CPU Co-Stop and CPU Other Wait are all low. This means NUMA is not an issue.
- Memory contention is flat 0.
- Virtual Disk latency is low most of the time, and did not correspond to the above 3 charts.

Here is CPU consumption metrics:



We can see massive imbalance. While the average CPU Run is 50%, the gap between the busiest vCPU and the least busy is >140%. Seems like the threads congregate in some of the CPU.

I don't have the details of the invidual vCPU. It will be interesting to plot each to see if there is high thread migation.

Let's pick a shorter time window so we can see the chart clearly.

The following chart shows the pattern over 6 hours.



The spike on CPU Context switch does not match the CPU Queue. There is a match to CPU Run, which makes sense. CPU Usage Disparity went down, because the overall CPU Run went up.

Other things to check:

- Is it running a multi-threaded application? Some applications spawn many threads. If yes, it's possible that many threads are waiting for a Software Lock (like a Mutex or Semaphore). The threads in the queue cannot be distributed to those idle cores because they are all waiting for a specific resource held by a thread on a different core. They are marked as "Ready" because they aren't waiting on I/O, but they are effectively stuck in a bottlenecked pipeline.
- If Processor% Privileged Time counter is high but User Time counter is low, the kernel (drivers/interrupts) is hogging the CPU, not the application. We have a driver or kernel-locking.

Low CPU Run

How about when Net Run (%) is low?

The following Microsoft Windows server VM has 2 vCPU only.

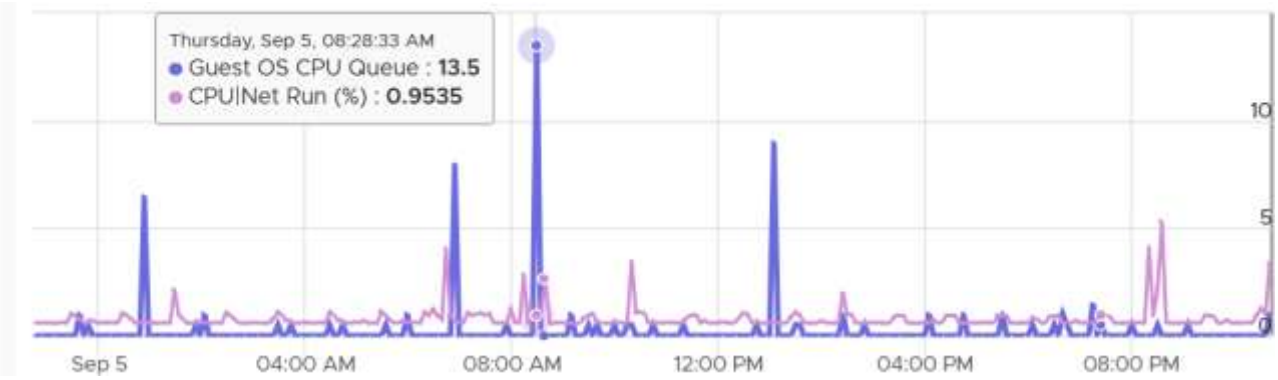


FYI:

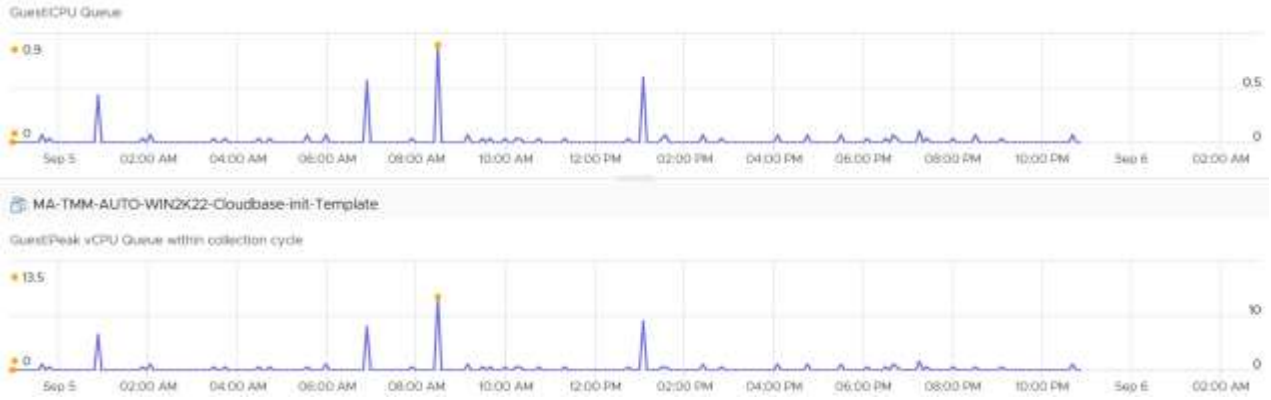
- CPU Ready is below 0.5%. CPU Co-Stop and CPU Other Wait are all low. This is expected as the VM is basically idle.
- Memory contention is flat 0.
- Virtual Disk latency is 1 ms, and did not correspond to the above 3 charts.

Queue and Idle

It is possible that Guest OS shows high CPU queue when it was idling. This is abnormal, indicating the application created a lot of threads. The following shows the application was idling while having high queue.



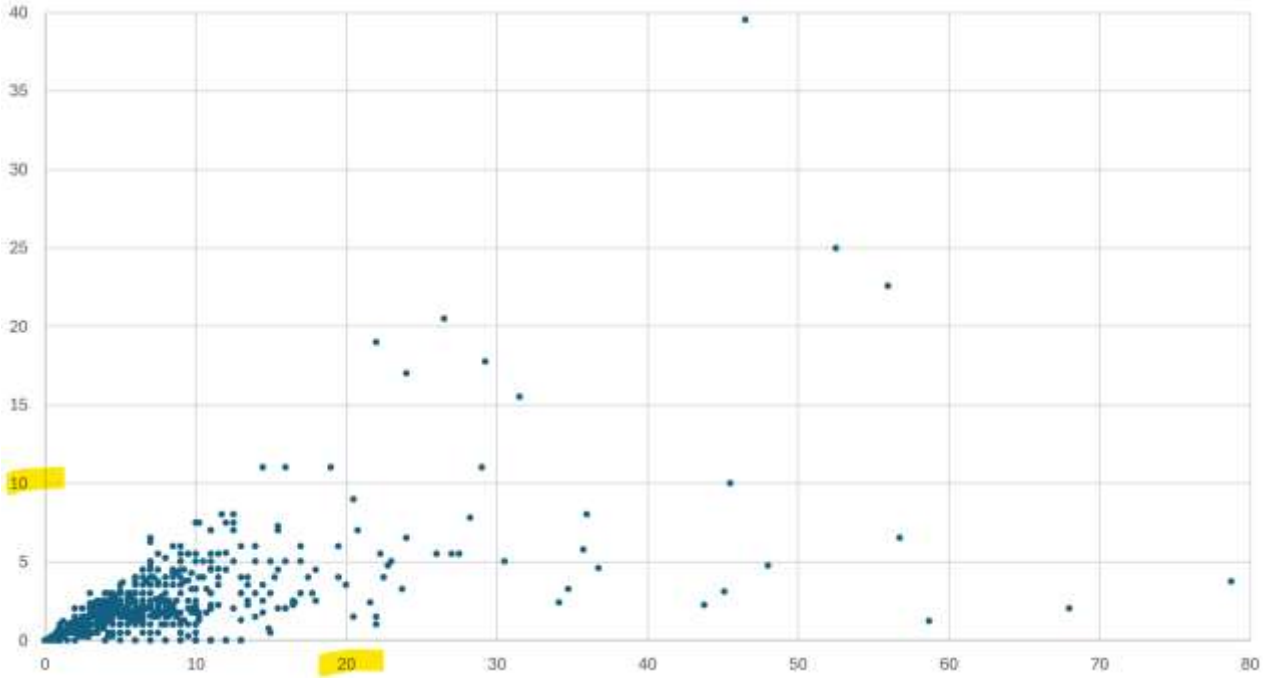
The CPU Run Queue spikes multiple times. It does not match the CPU Usage. It also did not match CPU Context Switch Rate pattern. The spike only last 20 seconds, as the 5-minute average shows identical pattern but much smaller number.



Overall Behaviour

What is the behaviour of CPU Queue?

I profiled 800 VM in the last 1 month. For each VM, I extracted the peak value and the 99th percentile value. From the following scatter chart, you can see that the value at 99th percentile is less than half. Using a spreadsheet, the average value of the 99th percentile is 34% for peak that is ≥ 3.



BTW, the value from Guest OS displays the last observed value only; it is not an average. Windows & Linux do not provide the highest and lowest variants either.

The counter name in Tools is guest.processor.queue. For Windows, it is based on Win32_PerfFormattedData_PerfOS_System = @#ProcessorQueueLength from WMI

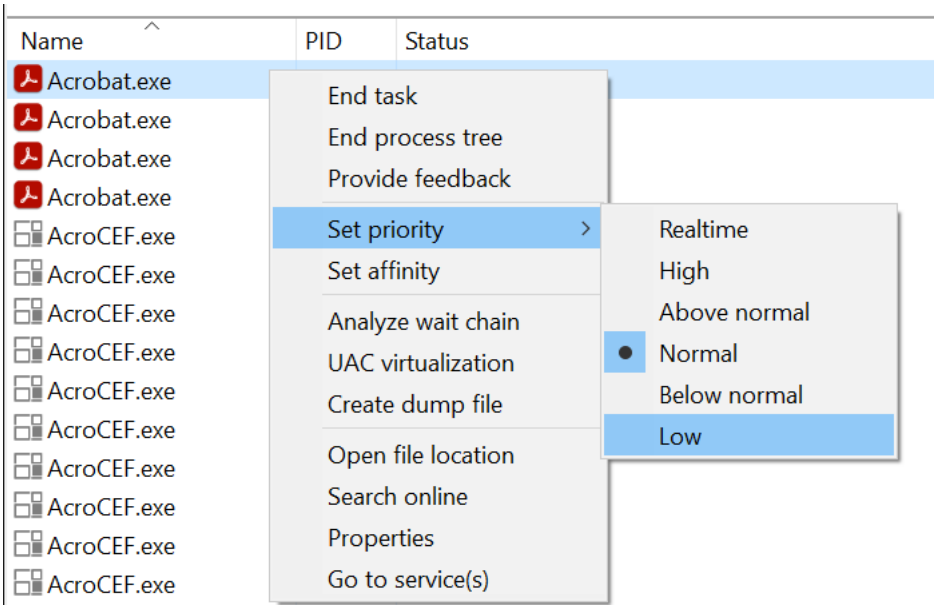
Reference: Windows

I can't find documentation that states if CPU Hyper Threading (HT) technology provides 2x the number of queue length. Logically it should as the threads are at the start of the CPU pipelines, and both threads are interspersed in the core pipeline.

CPU Priority

If a process is often in queue, one possility is it has lower relative priority. Priority is a concept of Windows that ESXi does not have. ESXi uses a fair-share scheduler instead, as it does not have foreground processes.

Windows or Linux provides priority for foreground activities, as that's what the user experience. For Windows, there are 6 levels as shown below. Ensure all your agents are given lower priority and limited CPU resource.



Context Switch

CPU Context Switch costs performance "due to running the task scheduler, TLB flushes, and indirectly due to sharing the CPU cache between multiple tasks". It's important to track this counter and at least know what's an acceptable behaviour for that specific application.

Context switches are considered "expensive" operations, as the CPU can complete many instructions within the time taken to switch context from one process to another. A sustained high number degrade performance noticeably through "cache pollution" — where the CPU has to constantly reload data into L1/L2 caches because the previous thread was swapped out.

For further reading, read this paper.

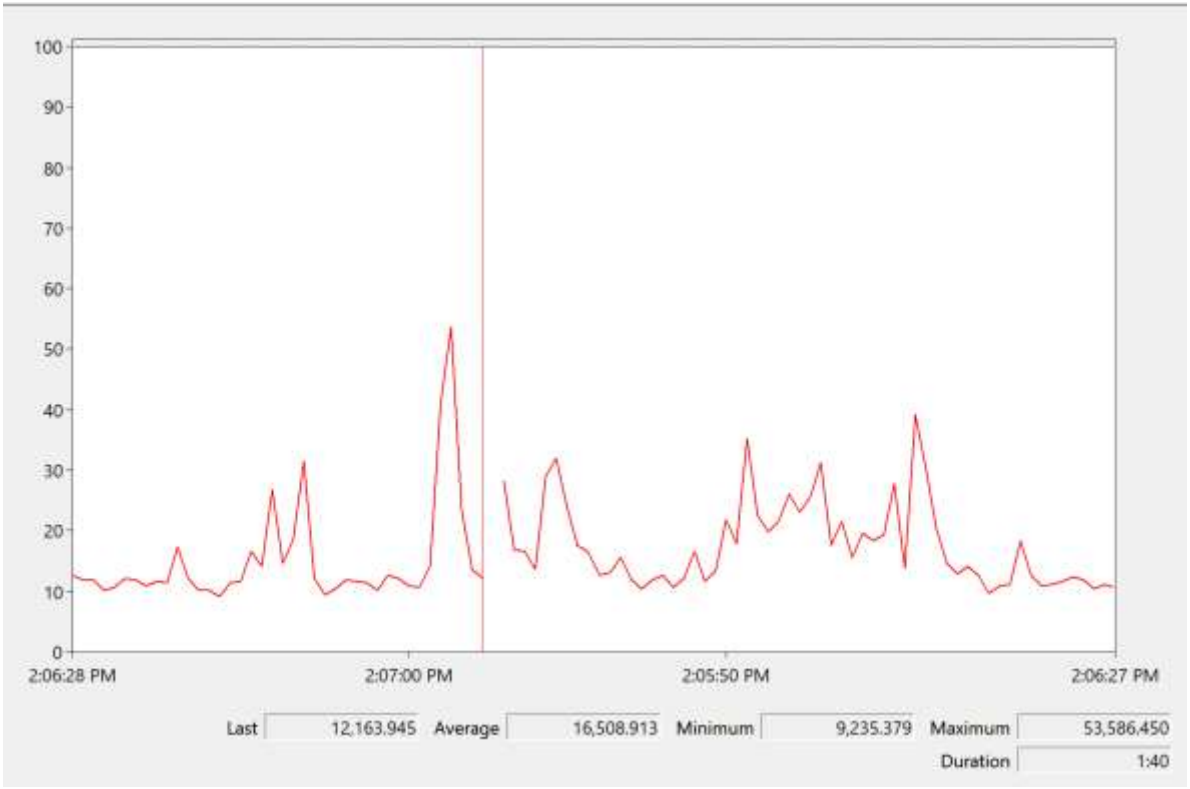
Based on Windows 10 Performance Monitor documentation, context switches/sec is the combined rate at which all processors on the computer are switched from one thread to another. All else being equal, the more the processors, the higher the context switch. Note that thread switches can occur either inside of a single multi-thread process or across processes. A thread switch can be caused either by one thread asking another for information, or by a thread being pre-empted by another, higher priority thread becoming ready to run.

Here is my take:

- 2K – 4K per processor
- 4K – 8K per processor
- > 8K per processor

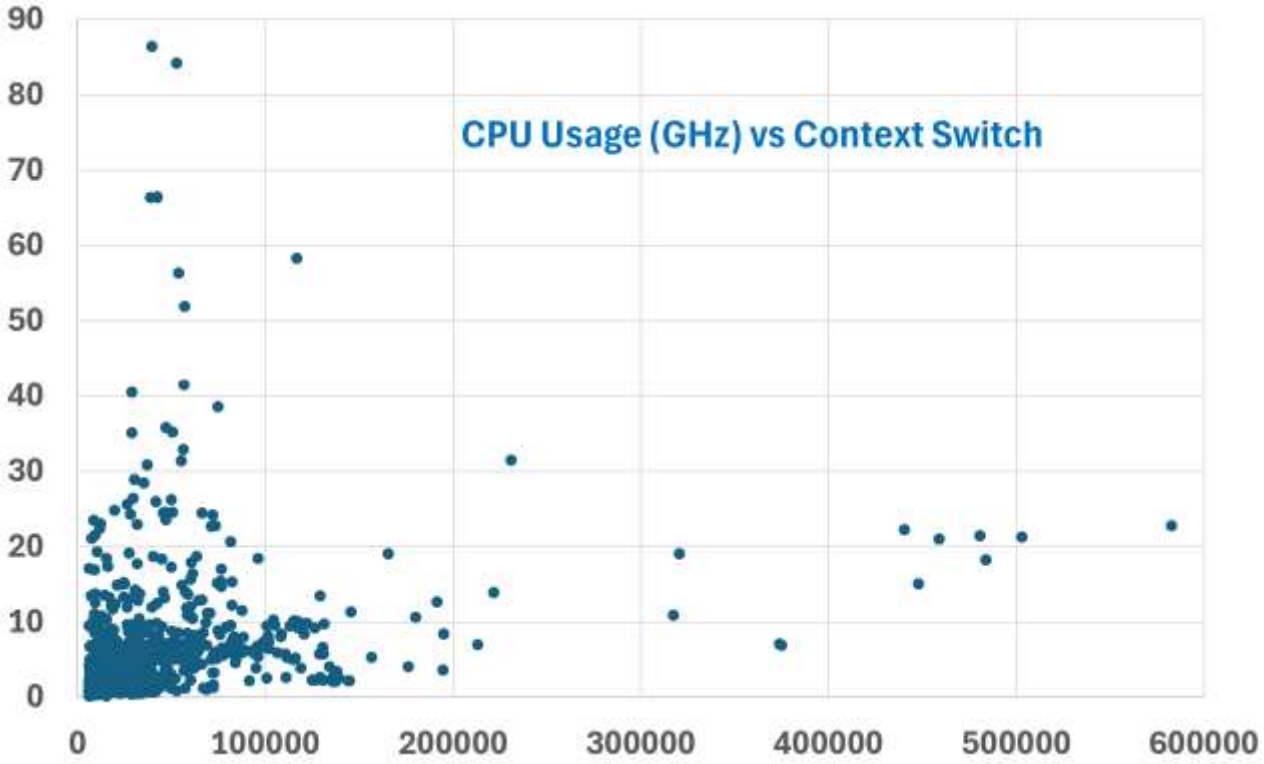
There are context switch metrics on the System and Thread objects. VCF Operations only report the total.

The rate of Windows or Linux switching CPU context per second ranges widely. The following is taken from a Windows 10 desktop with 8 physical threads, which runs around 10% CPU. I observe the value hovers from 10K to 50K.


100
90
80
70
60
50
40
30
20
10
0
2:06:28 PM
2:07:00 PM
2:05:50 PM
2:06:27 PM
Last 12,163.945 Average 16,508.913 Minimum 9,235.379 Maximum 53,586,450
Duration 1:40

Correlation

The value does not correlate well with CPU “utilization”, because not all CPU instructions require context switching. Overall, the higher the utilization the higher the chance of CPU context switch. I plotted 3328 VM on a scatter chart.



The above does not mean there is no correlation. The following chart shows a near perfect corelation. Every time CPU Usage went up, CPU Context Switch also. I should have plotted the disk IO or network IO as IO operations tend to require context switch.



CPU context switch can happen even in a single thread application. The following shows a VDI VM with 4 vCPU. I plotted the CPU Usage Disparity vs CPU Context Switch. You can see the usage disparity went up to 78%, meaning the gap between the busiest vCPU and the idlest vCPU is 78%. This was running a security agent, which is unlikely to be designed to occupy multiple vCPU.



Let's plot the context switch at the same period. There is a spike at the same time, indicating that the agent was busy context switching. Note that it does not always have to be this way. The red dot shows there is no spike in context switch even though the vCPU Usage Disparity went up.



Range Analysis



The values of CPU Context Switch vary widely. For many VM, the values will be in low hundreds. In extreme situation, it can sustain well beyond 10 millions, as shown in the preceding chart. The above VM was not doing heavy IOPS nor high CPU usage. There was no correlation with these 2 metrics either.

Because of above, it's important to profile and establish a normal base line for that specific application. What is healthy for 1 VM may not be healthy for another.

	Worst Value	Value at 95th Percentile	Average Value	Average CPU Queue	Configured CPU
2_old	982,602.75	873,558.3	50,288.08	0.49	4
3_old	942,715	820,889.81	47,231.92	0.34	4
2	587,465	197,595.75	67,727.37	0.12	32
l_old	569,746.75	3,796.15	2,286.84	0.31	4
	444,395.66	431,853.34	54,194.86	0.06	8
es1	439,858.81	145,050.48	38,475.51	0.04	104
	392,009.41	191,951.11	100,839.07	0.34	32
3	377,503.53	191,964.06	101,228.42	0.37	32

You can see from the table that some VM experience prolonged CPU context switch, while others do not. The VM #4 only has a short burst as the value at worst 5th percentile dropped to 3796. Momentary peak of context switch may not cause performance problem so in general it's wiser to take the value somewhere between 95th and 99th percentile.

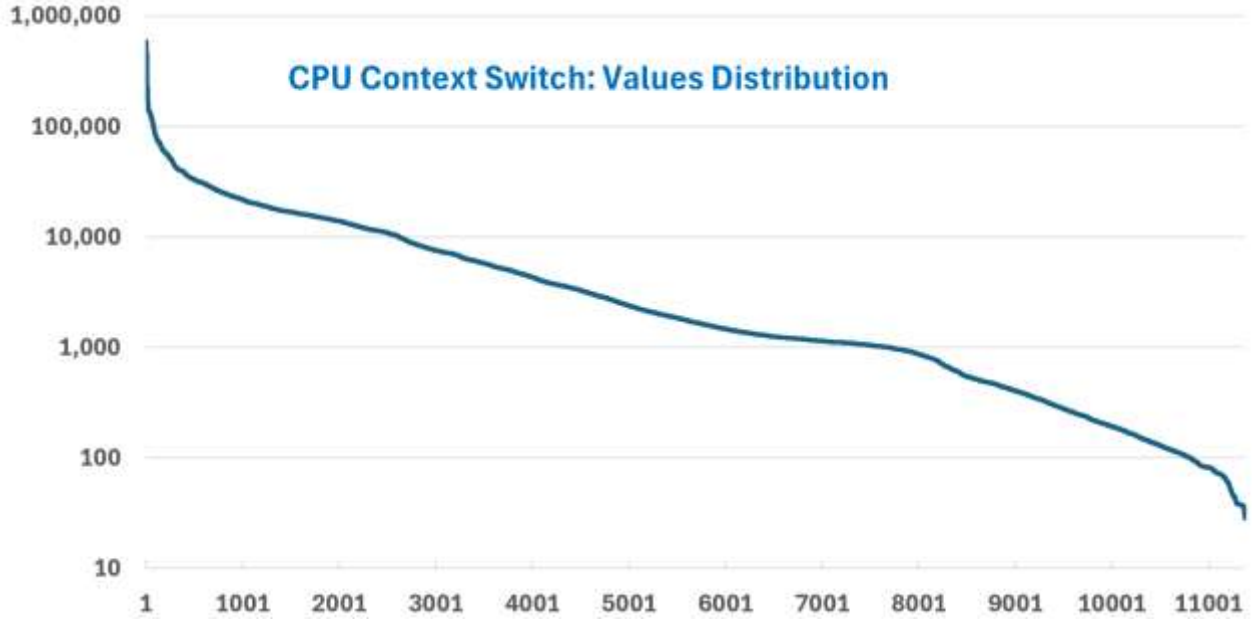
Let's drill down to see the first VM. This CentOS VM sporting only 4 vCPU constantly hit almost 1 million context switches. The pattern match CPU Usage.



Do you add more CPU?

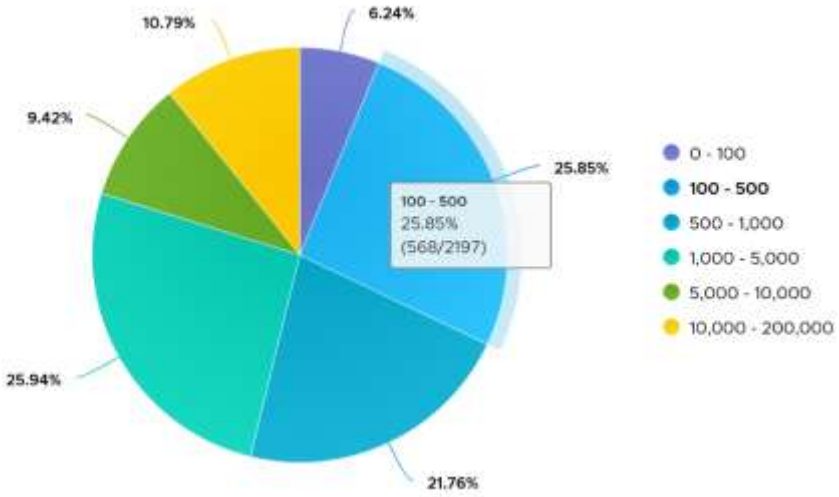
You should not, as the queue remain manageable. Check what the queue like on a more granular reading.

The following distribution chart shows the values from 11372 VMs. I had to use log-10 scale as the values vary wildly.



Majority of Guest OS spends well below 10K. You can see that the values between 0 – 10000 accounts for 80%.

Now that you know the wide distribution, you can use buckets. Adjust the bucket size by grouping all the values above 10K as one bucket, and splitting 0 – 10K bucket into multiple buckets. You can see more than half has < 1000 CPU Context Switch Rate.



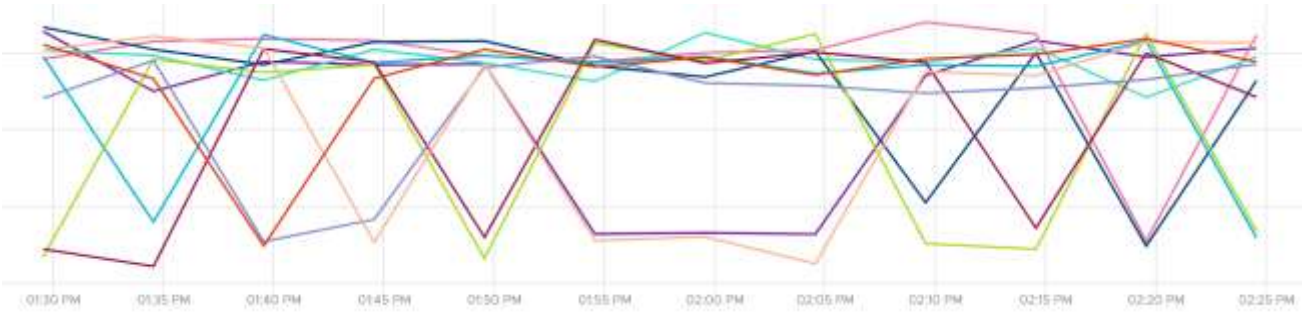
Thread Ping Pong

The following is a Windows Server 2019 DC edition VM with 10 vCPU. It's basically idle, as you can see below.



But if we zoom into each vCPU, they are taking turn to be busy.

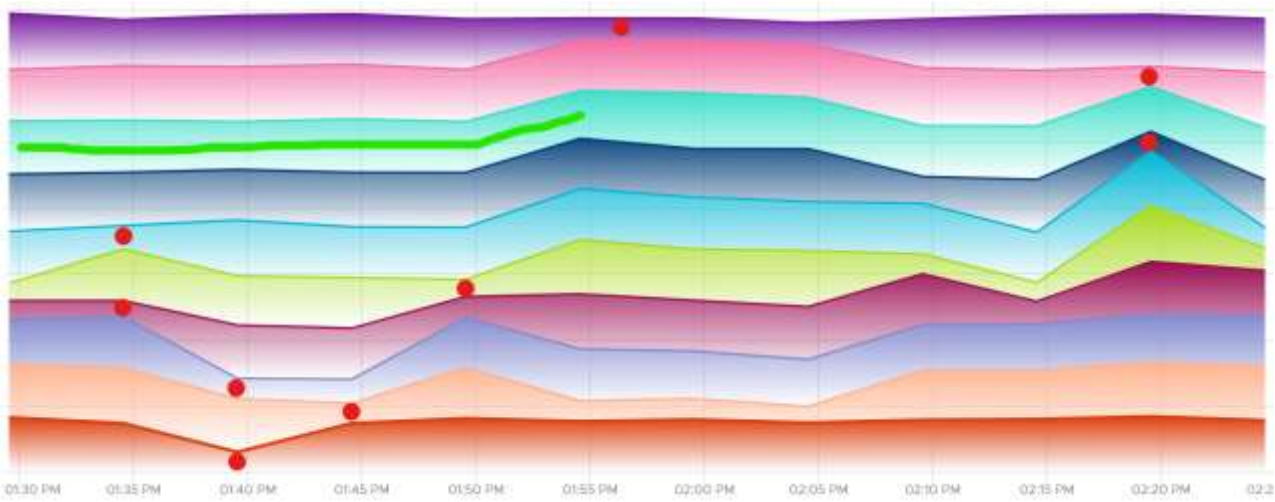
In the span of just 1 hour, the 10 vCPU inside Windows take turn.



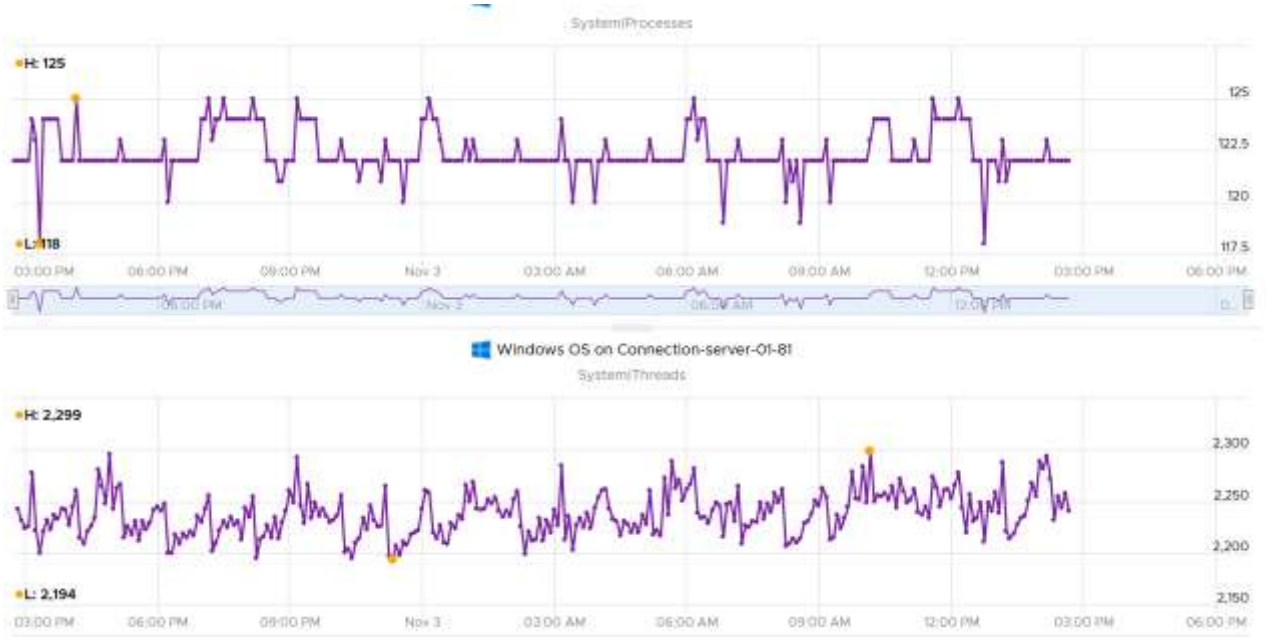
This is a bit illogical. Is this a process ping pong?

It is hard to tell

We can see them clearer if we stack them up. Notice they take turn, except the 3rd one from the top (I drew a green line on it). That one is actually fairly stable.

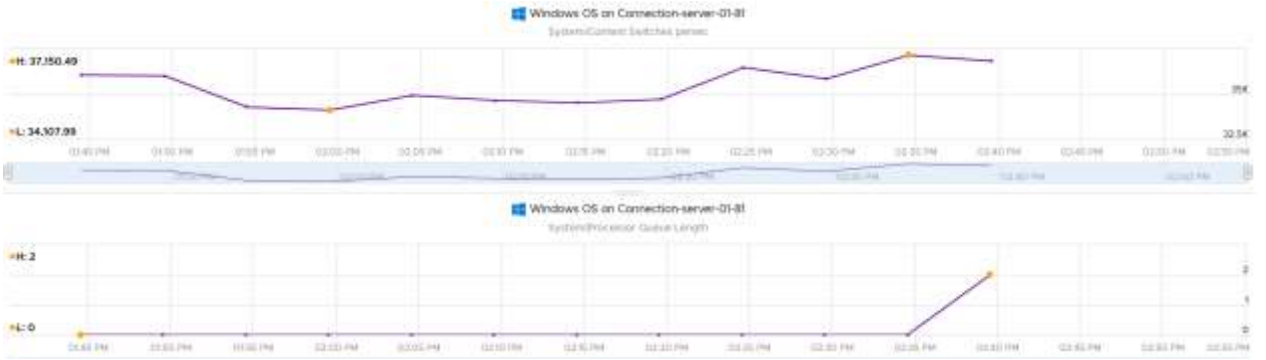


It is running Horizon Connection Server. It has around 118 – 125 processes, but much higher threads counts.



CPU Run Queue is very low, which is expected as the system is basically idle.

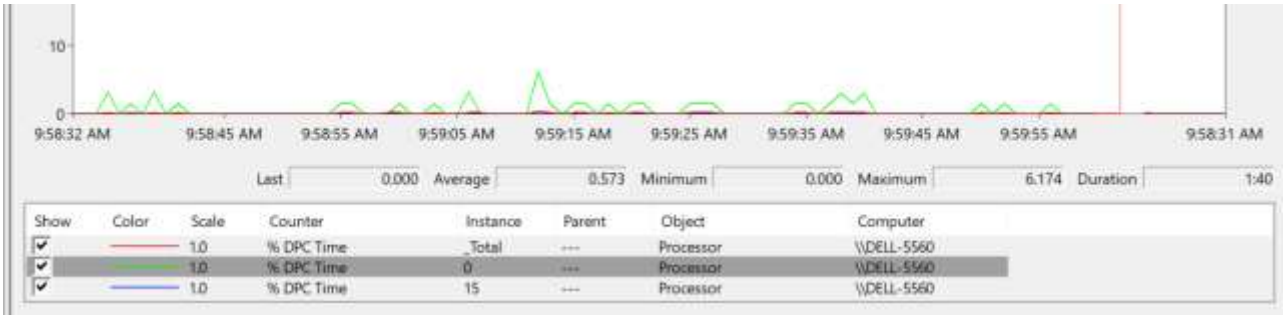
Context switches is fairly steady. The following screenshot shows it hovers between 34K and 37K switches per second. This is expected as it consistently run >2K threads on >100 processes on just 10 CPU. Each CPU does ~3.5K switches per second.



DPC Time

According to System Center wiki, the system calls are deferred as they are lower priority than standard interrupts. A high percentage of deferral means Windows was busy doing higher priority requests.

They can happen even during low CPU utilization if there is issue with driver or application. The following screenshot is taken on Performance Monitor in Windows 11 laptop which was not running high. Notice the DPC time for CPU 0 is consistently higher than CPU 15, indicating imbalance. It did exceed >5% briefly. My Dell laptop has 8 cores 16 threads.



Set the graph scale to 1 for ease of reading, and change the axis scale accordingly.

Runaway Process

What do you see from the CPU charts below?



There are 8 CPU as seen by Windows 10. Hint: look at the total picture, no need to see each in detail. That's why I made the screenshot tiny.

Yes, you're right. CPU0 is running flat. The reason was one of Windows common service went into infinite loop. Ironically, this is the troubleshooting service (Diagnostic Policy Service) itself. So it's chewing up CPU flat out non-stop. But since there are 7 other CPU, Windows overall is responsive. I could still do my work.

A counter that tracks at entire Guest OS level will not capture it. You need to complement it with a counter that tracks the highest among its CPU. If this is flat out all the time, you likely have a runaway process.

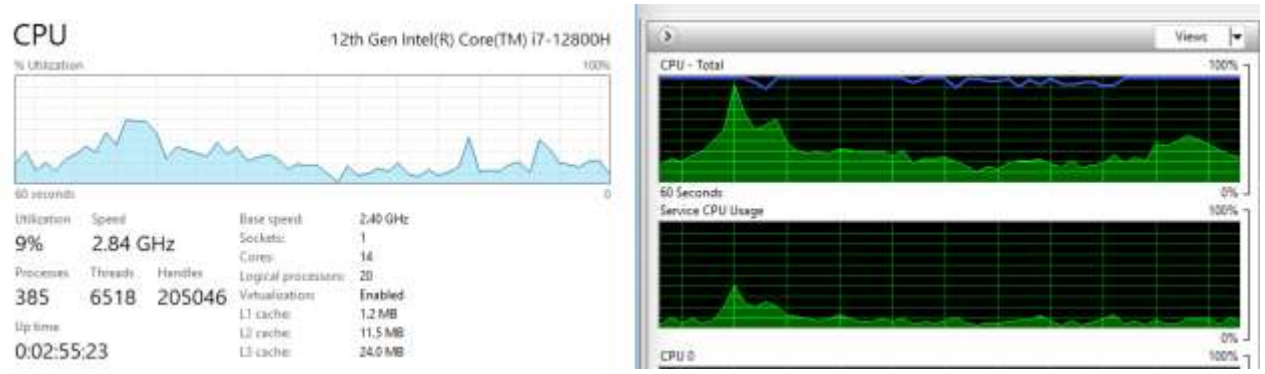
Usage | Utilization | Time

The 3 tools in Windows use different names to measure CPU consumption:

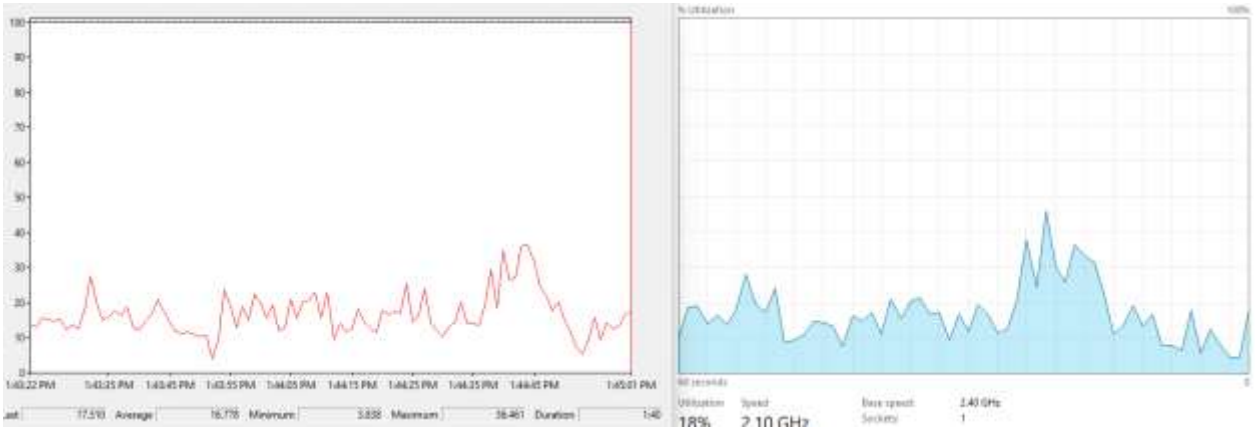
- Performance Monitor uses the name Processor Time.
- Task Manager uses the name CPU Utilization.

- Resource Monitor uses the name CPU Usage.

Looking at the values, they are not measuring the same thing. The following screenshot is a side-by-side comparison between Task Manager and Resource Monitor.



Let's compare Performance Monitor and Task Manager:



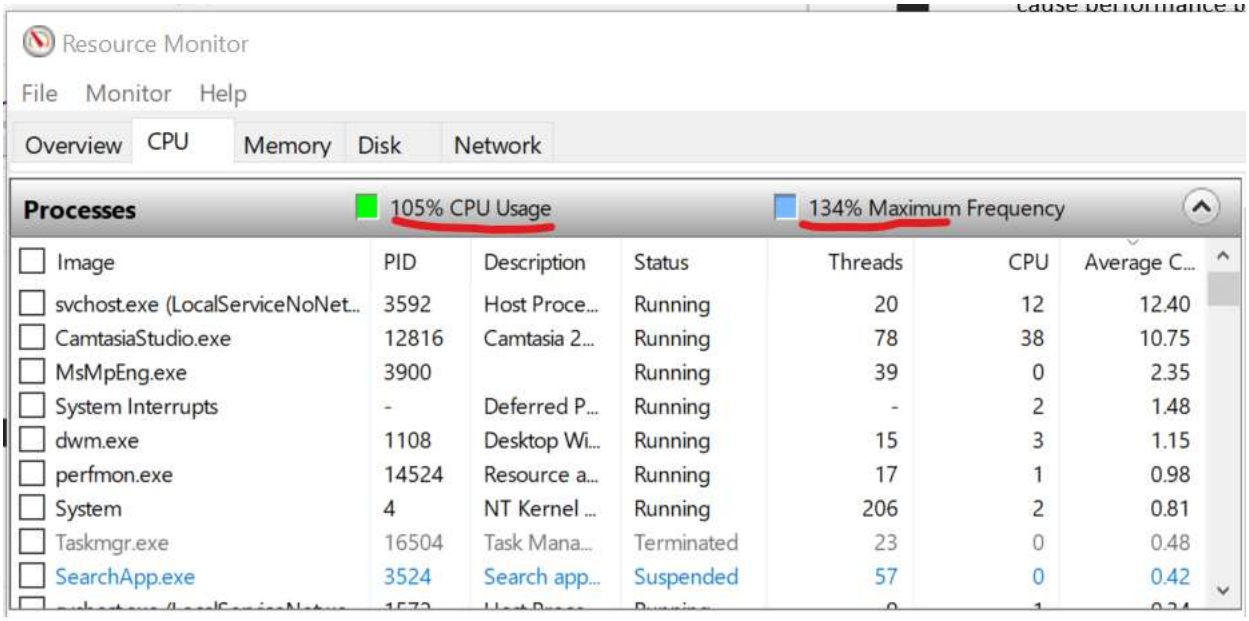
CPU Usage in Windows is not aware of the underlying hypervisor hyper-threading. When Windows run a CPU at 100% flat, that CPU could be competing with another physical thread at ESXi level. In that case, what do you expect the value of VM CPU Usage will be, all else being equal?

62.5%.

Because that's the hyper-threading effect.

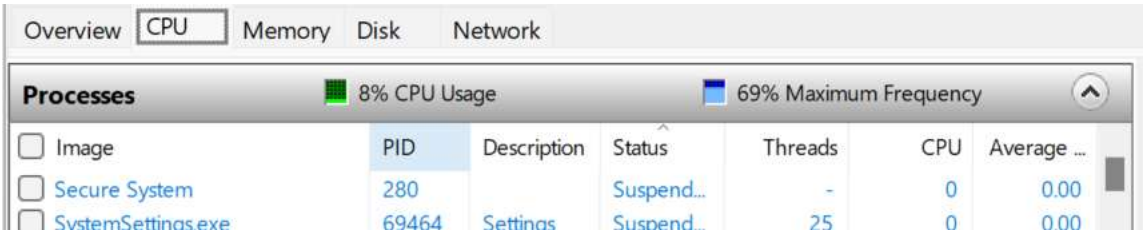
What about VM CPU Demand? It will show 100% .

However, CPU Usage is affected by power management. Windows 8 and later will report CPU usage >100% in Task Manager and Performance Monitor when the CPU Frequency is higher than nominal speed. The reason for the change is the same with what we have covered so far, which is the need to distinguish amount of work being done. More here.



BTW, what does the Maximum Frequency mean?

Let's show an opposite scenario, where CPU Usage (%) is low.



The number means the speed it's running relative to the nominal frequency. During low utilization, Windows will conserve power by lowering the speed.

In the first screenshot, since it's running at 134% of the clock speed, the total time exceeded 105%, meaning there were time it did not run. Had it run all the time, it would report 134% instead.

In the first screenshot, it could be running more than 8% of the time. But since it's only running at 69% of the speed, the CPU productive cycle completed is only 8%. The througput was degraded.

What happens to CPU Usage when VM is experiencing contention? VM Contention = Ready, Co-Stop, Overlap, Other Wait.

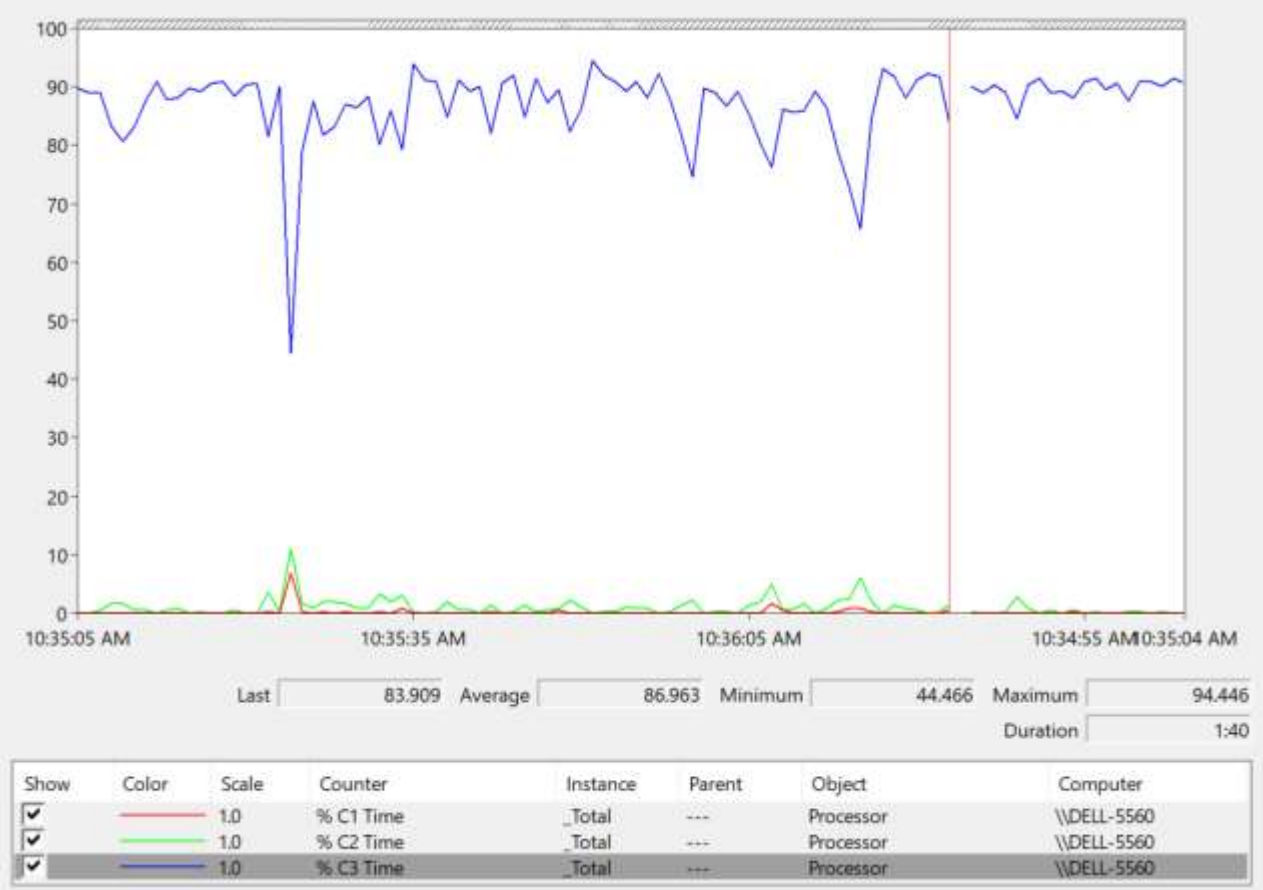
Time basically stops. So there is a gap in the system time of Windows. How does it deal with the gap? Does it ignore the gap, or artificially fills it with best guess values? I'm not sure. If you do let me know.

The above nature of CPU Usage brings an interesting question. Which VM counters can be used when you have no visibility into the Guest? Let's do a comparison:

Metric	Frequency Scaling	Hyperthreading	VM Contention
Guest CPU Usage	Yes	No	No
VM CPU Run	No	Yes	No
VM CPU Usage	Yes	Yes	No
VM CPU Demand	Yes	No	Yes

If there is slowness but utilization is low, it's worth checking if the utilization is coming from lower power state. This is important for application that requires high frequency (as opposed to just lots of light threads).

Windows provides the time the CPU spent on C1, C2 and C3 state. The following is taken from my laptop. Notice a dip when the total of C1 + C2 + C3 < 100%. That's basically the time on C0.



The Idle loop is typically executed on C3. Try plotting the Idle Time (%) and C3 Time (%), and they will be similar.

OS vs Process

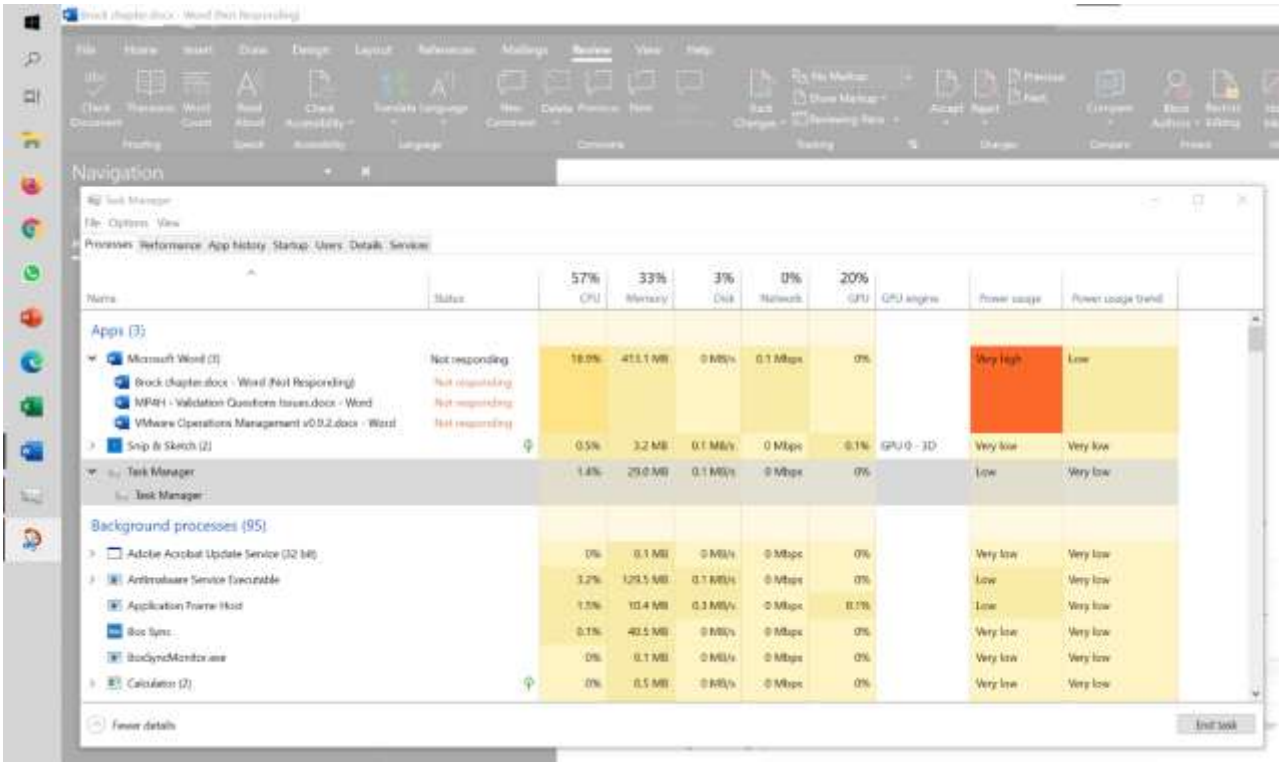
CPU imbalance can happen in large VM.

Review the following chart carefully. It's my physical desktop running Windows 10. The CPU has 1 socket 4 cores 8 threads, so Windows see 8 logical processors. You can see that Microsoft Word is not responding as its window is greyed out. The Task Manager confirms that by showing that none of the 3 documents are responding. Word is also consuming a very high power, as shown in the power usage column.

It became unresponsive because I turned on change tracking on a 500 page document and deleted hundreds of pages. It had to do a lot of processing and it did not like that. Unfortunately I wasn't able to reproduce the issue after that.

At the operating system, Windows is responding well. I was able to close all other applications, and launched Task Manager and Snip programs. I suspect because Word does not consume all CPUs. So if we track at Windows level, we would *not* be aware that there is a problem. This is why process-level monitoring is important if you want to monitor the application. Specific to hang state, we should monitor the state and not simply the CPU consumption.

From the Windows task bar, other than Microsoft Word and Task Manager, there is no other applications running. Can you guess why the CPU utilization at Windows level is higher than the sum of its processes? Why Windows show 57% while Word shows 18.9%?



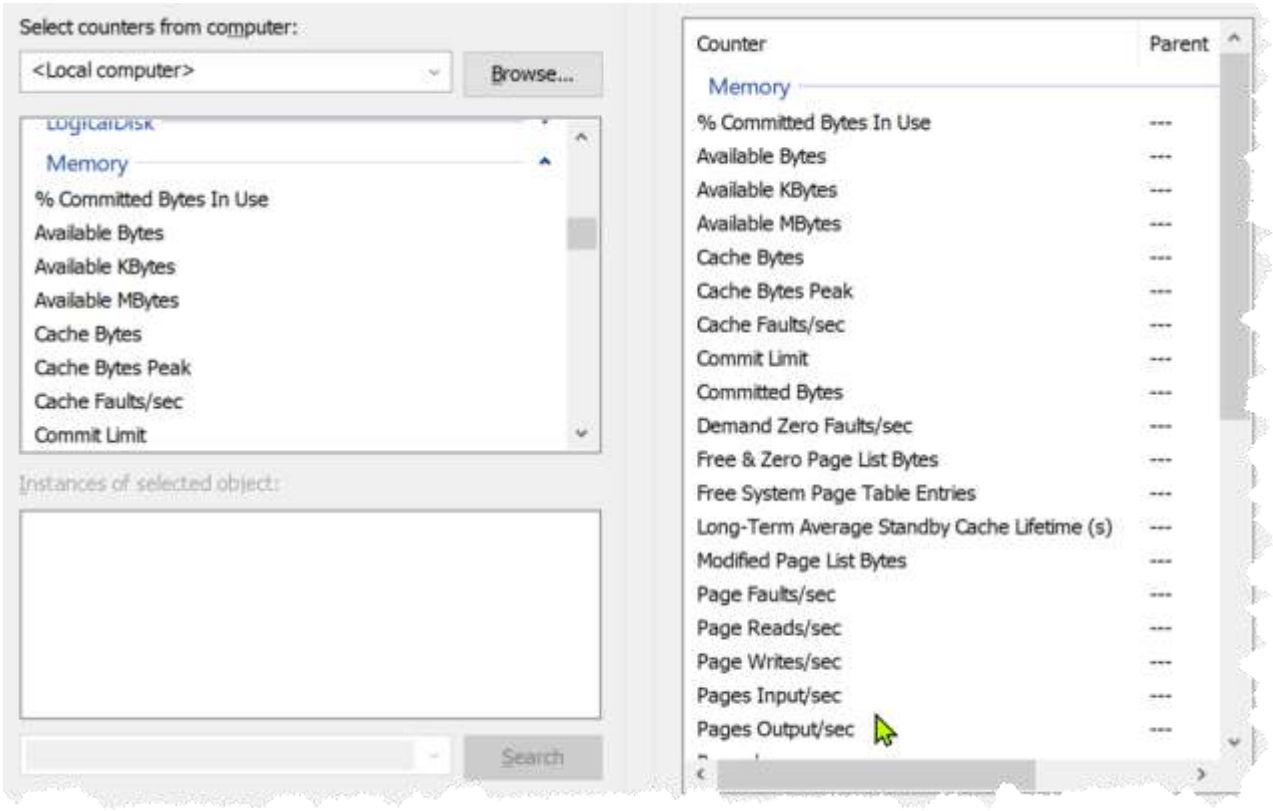
My guess is Turbo Boost. The CPU counter at individual process level does not account for it, while the counter at OS level does.

I left it for 15 minutes and nothing change. So it wasn't that it needed more time to process the changes. I suspect it encountered a CPU lock, so the CPU where Word is running is running at 100%. Since Windows overall only reports 57%, it's important to track the peak among Windows CPU. This is why VCF Operations provides the peak value among the VM vCPU.

Memory

Windows memory management is not something that is well documented. Ed Bott sums it this article by saying "Windows memory management is rocket science". Like what Ed has experienced, there is conflicting information, including the ones from Microsoft. Mark Russinovich, cofounder of Winternals software, explains the situation in this TechNet post.

Windows Performance Monitor provides many metrics, some are shown below.

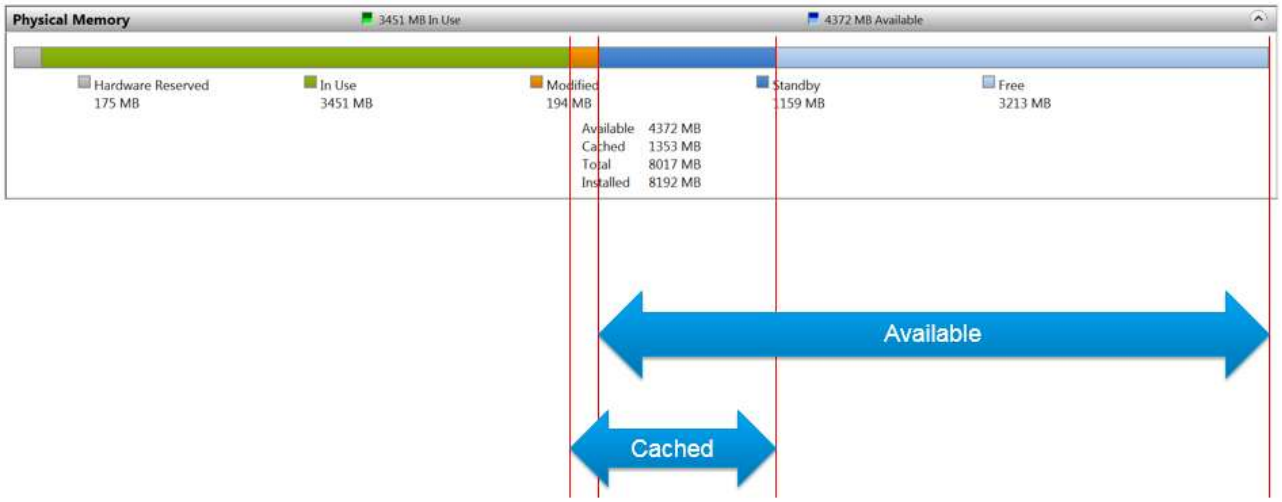


In formula, here is their definition:

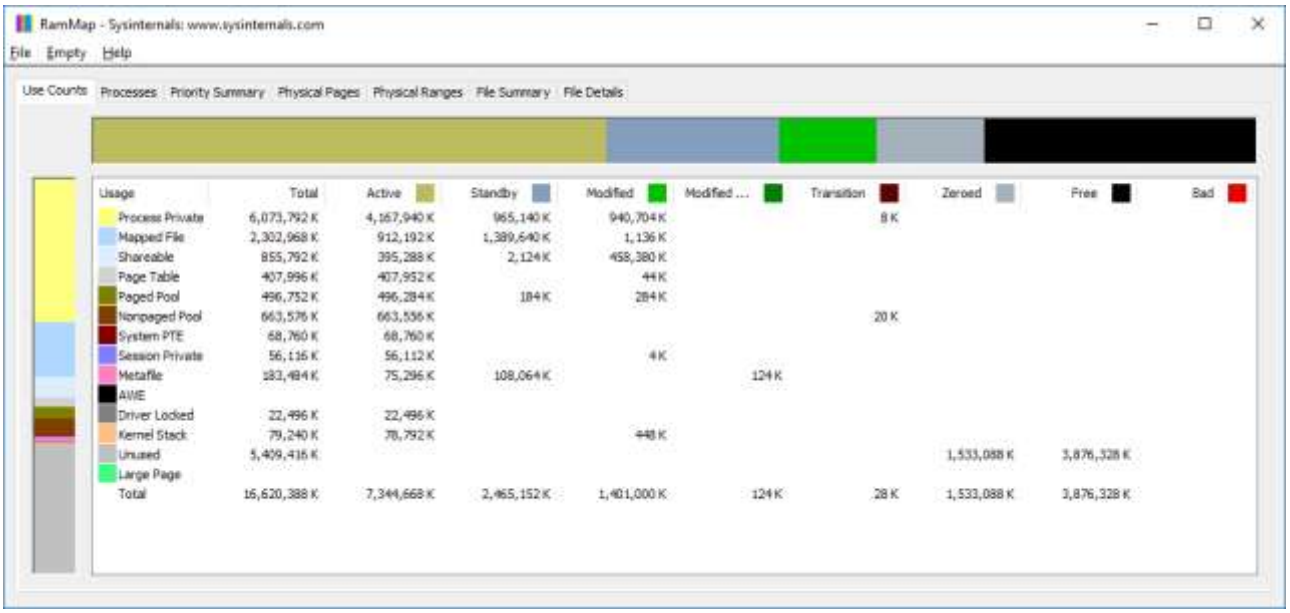
- Cached = Standby + Modified
- Available = Standby + Free

Available means exactly what the word means. It is the amount of physical memory immediately available for use. Immediately means Windows does not need to copy the existing page before it can be reused.

It is easier to visualize it, so here it is:

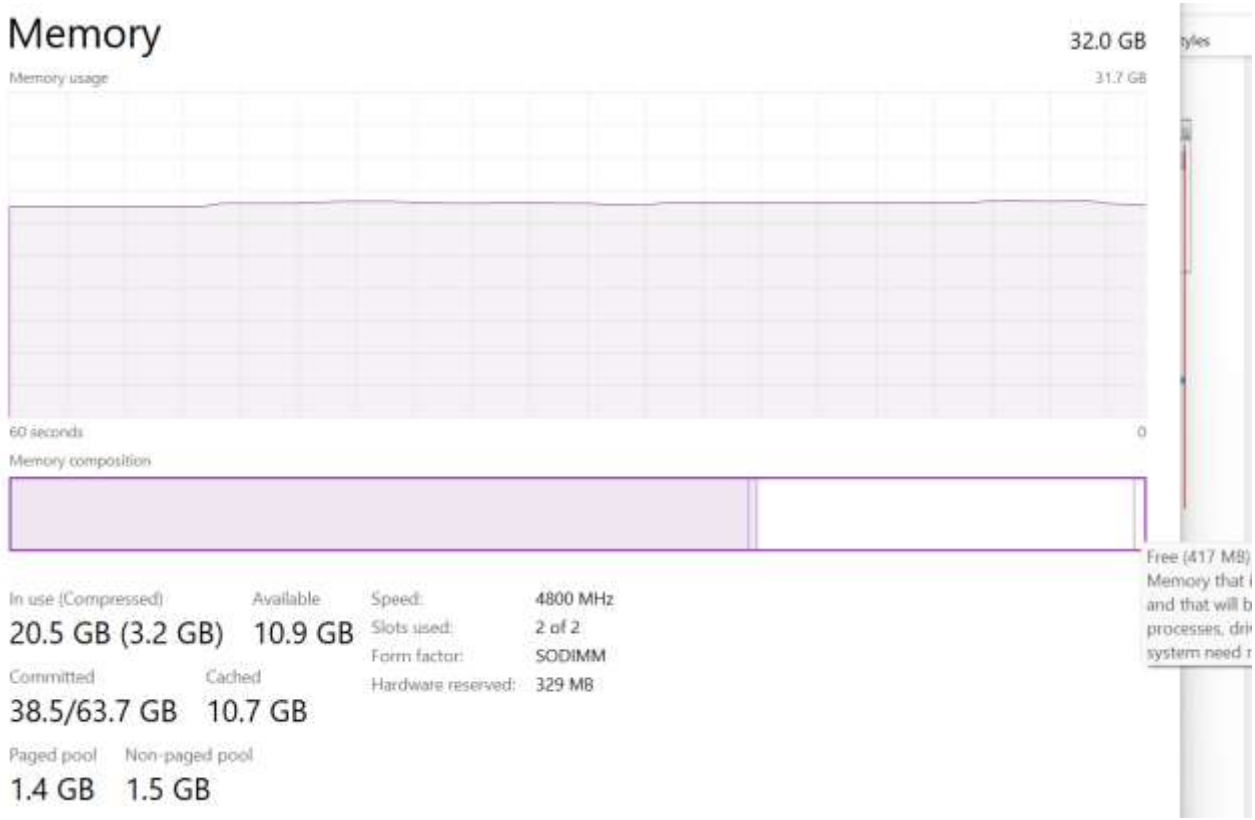


Microsoft SysInternal provides more detail breakdown. In addition to the above, it shows Transition and Zeroed.



In Use

This is the main counter used by Windows, as it's featured prominently in Task Manager.



This is often thought as the minimum that Windows needs to operate. This is not true. If you notice on the preceding screenshot, it has compressed 457 MB of the 6.8 GB In Use pages, indicating they are not actively used. Windows compresses its in-use RAM, even though it has plenty of Free RAM available (8.9 GB available). This is a different behaviour to ESXi, which do not compress unless it's running low on Free.

Look at the chart of Memory Usage above. It's sustaining for the entire 60 seconds. We know this as the amount is too high to sustain for 60 seconds if they are truly active, let alone for hours.

Formula:

```
In use = Total – (Modified + Standby + Free)
```

A problem related to the In Use counter is memory leak. Essentially, the application or process does not release pages that it no longer needs, so over time it accumulates. This is hard to detect as the amount varies by application. The process will continue growing until the OS runs out of memory.

Take note this is a new metric in VCF Operations 8.6. We call it Used Memory. You're welcome.

Modified

Page that was modified but no longer used, hence it's available for other usage but requires to be saved to disk first. It's not counted as part of Available, but counted as part of Cache.

OS does not immediately write all inactive pages to disk, especially if the disk is on power saving mode. It will consolidate these pages and write them in one shot, minimizing IO to the disk. In the case, of SSD disk, it can shorten the life span as SSD has physical limits on the number of writes.

Standby

Windows has 3 levels of standby. As reported by VMware Tools, their names are:

- Standby Core
- Standby Normal
- Standby Reserve

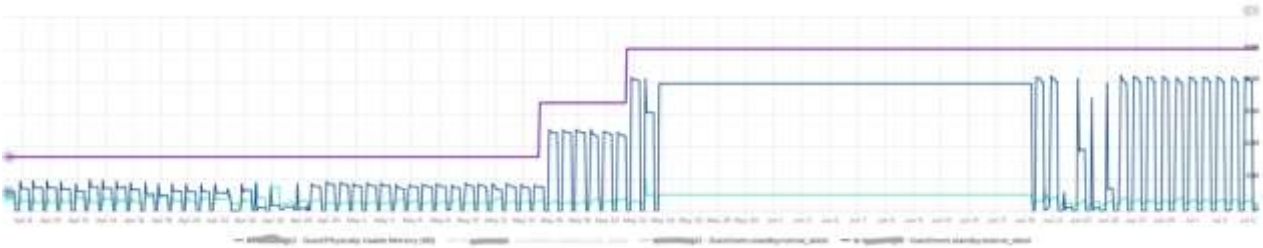
Different applications use the memory differently, resulting in different behaviour of the metrics. As a result, determining what Windows actually uses is difficult.

The Standby Normal counter can be fluctuating wildly, resulting in a wide difference if it's included in rightsizing. The following VM is a Microsoft Exchange 2013 server mailbox utility.



Notice the Standby Normal fluctuates wildly, reaching as high at 90%. The other 2 cache remains constantly negligible. The chart above is based on >26000 samples, so there is plenty of chance for each 3 metrics to fluctuate.

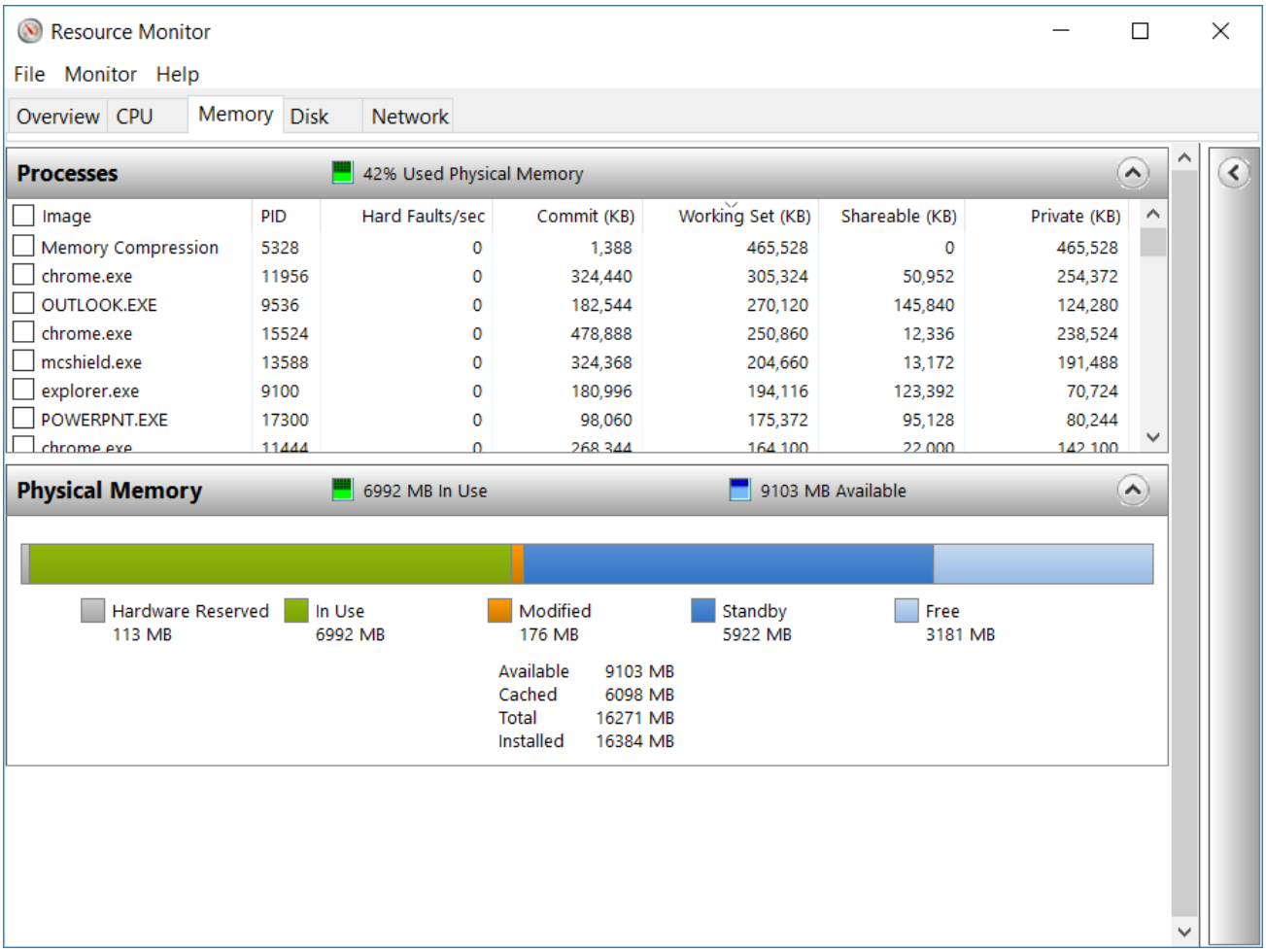
Now let's look at another example. This is a Windows Server 2016. I think it was running Business Intelligence software Tableau.



Notice the VM usable memory was increased 2x in the last 3 months. Standby Normal hardly move, but Standby Reserve took advantage of the increments. It simply went up accordingly, although again it's fluctuating wildly.

Cache

Cache is an integral part of memory management, as the more you cache, the lower your chance of hitting a cache miss. This makes sense. RAM is much faster than Disk, so if you have it, why not use it? Remember when Windows XP introduced pre-fetch, and subsequently Windows SuperFetch? It's a clue that memory management is a complex topic. There are many techniques involved. Unfortunately, this is simplified in the UI. All you see is something like this:



Free

As the name implies, this is a block of pages that is immediately available for usage. This excludes the cached memory. A low free memory does not mean a problem if the Standby value is high. This number can reach below 100 MB, and even touch 0 MB momentarily. It's fine so long there is plenty of cache. I'd generally keep this number > 500 MB for server VM and >100 MB for VDI VM. I set a lower number for VDI because they add up. If you have 10K users, that's 1 TB of RAM.

When Windows or Linux frees up a memory page, it normally just updates its list of free memory; it does not release it. This list is not exposed to the hypervisor, and so the physical page remains claimed by the VM. This is why the Consumed counter in vCenter remains high when the Active counter has long dropped. Because the hypervisor has no visibility into the Guest OS, you may need to deploy an agent to get visibility into your application. You should monitor both at the Guest OS level (for example, Windows and Red Hat) and at the application level (for example, MS SQL Server and Oracle). Check whether there is excessive paging or the Guest OS experiences a hard page fault. For Windows, you can use tools such as pfmon, a page fault monitor.

This is one the 3 major metrics for capacity monitoring. The other 2 metrics are Page-in Rate and Commit Ratio. These 3 are not contention metrics, they are utilization metrics. Bad values can contribute to bad performance, but they can't measure the severity of the performance. Windows and Linux do not have a counter that measures how long or how often a CPU waits for memory.

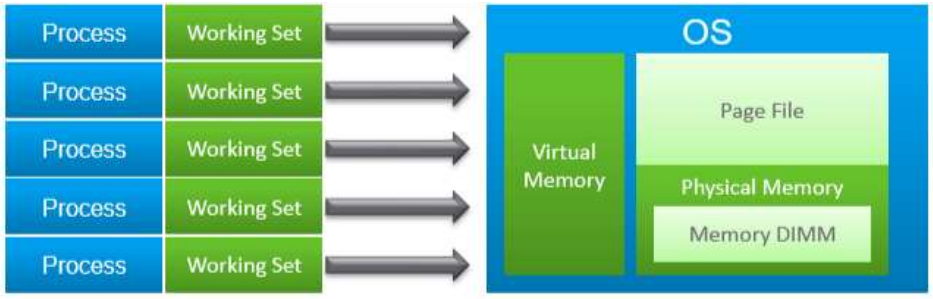
In Windows, this is the Free Memory counter. This excludes the cached memory. If this number drops to a low number, Windows is running out of Free RAM. While that number varies per application and use case, generally keep this number > 500 MB for server VM and >100 MB for VDI VM. The reason you should set a lower number for VDI because they add up quickly. If you have 10K users, that's 1 TB of RAM.

It's okay for this counter to be low, so long other memory metrics are fine. The following table shows VMs with near 0 free memory. Notice none of them are needing more memory. This is the perfect situation as there is no wastage.

Name	Free	Capacity	Needed Memory	Page In	Page Out	GuestIActive
CE...	0.0003 GB	20 GB	8.63 GB	47	0	0.41 GB
e...	0.0012 GB	18 GB	5.71 GB	23,790.2	0	0 GB
wi...	0.0032 GB	16 GB	13.01 GB	0.2	0	0.23 GB
ws...	0.01 GB	12 GB	4.11 GB	13.53	0	0.5 GB
mt...	0.01 GB	24 GB	9.19 GB	3.47	0	0.37 GB
fus...	0.02 GB	8 GB	3.79 GB	0.27	0	0 GB
v...	0.03 GB	8 GB	6.25 GB	0	0	0.23 GB
V...	0.04 GB	24 GB	10.05 GB	0	0	0 GB
sc...	0.04 GB	16 GB	11.32 GB	0.07	0	0 GB
MI...	0.04 GB	8 GB	4.14 GB	0.27	0	0.52 GB
SO...	0.05 GB	8 GB	3.4 GB	0	0	0.21 GB

Page File

Memory paging is an integral part of Guest OS Memory Management. OS begins using it even though it still has plenty of physical memory. It uses both physical memory and virtual memory at the same time. Microsoft recommends that you do not delete or disable the page file. See this for reference.



As shown on the diagram, processes see virtual memory, not physical memory. Guest OS presents this as system API to processes. The virtual memory is backed by the page file and physical memory. Guest OS shields the physical memory and hardware. Paging is an operation of reading/writing from the page file into the physical memory, not from physical disk into the page file.

Let Windows manages the pagefile size. This is the default setting, so you likely have it already. By default, windows sets the pagefile size to the same size with the physical memory. So if the VM has 8 GB of RAM, the pagefile is an 8 GB file. Anything above 8 GB indicates that Windows is under memory pressure.

The VM metric Guest \ Swap Space Remaining tracks the amount of swap space that's free.

The size of Page File is not a perfect indicator of the RAM usage, because they contain pages that are never demanded by the application. Windows does SuperFetch, where it predicts what pages will be used and prefetch them in advance. Some of these pages are never demanded by the application. Couple with the nature that Guest OS treats RAM as cache, including the page file will result in oversized recommendation. Paging rate is more realistic as it only considers the recent time period (300 seconds in VCF Operations case)

A page would be used as cache if it was paged out at some point due to memory pressure and it hasn't been needed since. The OS will reuse that page as cache. That means that at some point the OS was constrained on memory enough to force the page out to happen.

A page that was paged out earlier, has to be brought back first before it can be used. This creates performance issue as the application is waiting longer, as disk is much slower than RAM.

There are 2 types of page operations:

- Page In. This is a potential indicator for performance.
- Page-out. This is a potential indicator for capacity.

While Paging impacts performance, the correlation between the paging metrics and performance varies per application. You can't set a threshold and use it to monitor different applications or VM. The reason is paging is not always used when Guest OS runs out of memory. There are a few reasons why paging may not correlate to memory performance:

- Application binary. The initial loading causes a page-in. Nobody will feel the performance impact as it's not even serving anyone.

- Memory mapped files. This is essentially a file that has a mapping to memory. Processes use this to exchange data. It also allows the process to access a very large file (think of database) without having to load the entire database into memory.
- Proactive pre-fetch. It predicts the usage of memory and pre-emptively reads the page and bring it in. This is no different to disk where the storage array will read subsequent blocks even though it's not being asked. This especially happens when a large application starts. Page-in will go up even though there is no memory pressure (page out is low or 0).
- Windows performs memory capacity optimization in the background. It will move idle processes out into the page file.

If you see both Page-in and Page-out having high value, and the disk queue is also high, there is a good chance it's memory performance issue.

The rate pages that are being brought in and out can reveal memory performance abnormalities. A sudden change, or one that has *sustained over time*, can indicate page faults. Page faults indicate pages aren't readily available and must be brought in. If a page fault occurs too frequently it can impact application performance. While there is no concrete guidance, as it varies by application, you can judge by comparing to its past behaviour and its absolute amount.

Operating Systems typically use 4KB or 2MB page sizes. Larger page size will result in more cache, which translates into more memory required.

The counter %pagefile tracks how much of the pagefile is used, meaning the value 100% indicate the pagefile is fully utilized. While the lower the number the better, there is no universal guidance. If you know, let me know!

Reference: this is an old article as it covers 32 bit Windows. If you find a newer one, kindly let me know.

Guest OS Paging metrics

There are 2 metrics. Page-in and Page-out.

The unit is in number of pages, not MB. It's not possible to convert due to mix use of Large Page (2 MB) and Page (4 KB). A process can have concurrent mixed usage of large and non-large page in Windows. The page size isn't a system-wide setting that all processes use. The same is likely true for Linux Huge Pages.

The page-in rate metric tracks the rate OS brings memory back from disk to DIMM per second. Another word, the rate of reads going through paging/cache system. It includes not just swap file I/O, but cacheable reads as well (so it's double pages/s).

Page Out is the opposite of the above process. It is not as important as Page In. Just because a block of memory is moved to disk that does not mean the application experiences memory problem. In many cases, the page that was moved out is the idle page. Windows does not page out any Large Pages.

The following shows the page out value at 99th percentile in the last 4 months. What do you observe?



There are 3325 VM in the above chart. In the last 4 months, 97% of them have page-out rate of less than 32000 pages, on a 5-minute average basis.

How about the remaining 3%?

Surprisingly, a few of them can be well 500000, indicating there is a wide range. So majority of VMs do not page out, but those that do, they do it excessively.

The block size is likely 4 KB. Some applications like Java and databases use 2 MB pages. Using 8 KB as the average, 10000 pages per second sustained over 5 minutes means 80000 KB x 300 = 24 GB worth of data.

You can profile your environment to see which VMs are experiencing high paging. Create a view with the following 6 columns

- Highest Page-In. Color code it with 1000, 10000, and 100000 as the thresholds. That means red is 10x orange, which in turn is 10x yellow.
- Page-In value at 99th percentile. Same threshold as above.
- Highest Page-Out. Same threshold as above.
- Page-Out value at 99th percentile. Same threshold as above.
- Sum of Page-In
- Sum of Page-Out

Set the dates to the period you are interested, but make it at least 1 week, preferably 3 months. There 2016 data points in a week, so the 99th percentile ignores the highest 20 datapoints.

In the following example, I used 4 months. I listed the top VMs in terms, sorted by the highest page-in. What observation do you see?

Max Page In	Max Page Out	99P Page In	99P Page Out	Total Page In	Total Page Out
5,069,497	485,203.81	29,154.43	4,870.5	134,816,679.69	16,388,061.44
2,669,535.75	133,252.59	803,063.49	30,117.99	142,533,966.84	7,896,754.56
2,354,336.75	64,080.54	57,443.6	2,758.59	28,895,867.75	5,115,804.13
2,049,341.62	6,993	292,200.29	518.6	1,155,861,855.53	1,705,288.52
1,981,613.75	397,136.88	1,340,973.8	53,184.11	492,948,074.08	24,948,522.02
1,942,909.12	242,705.8	1,175,849.13	146,928.51	2,410,523,742.33	1,685,080,697.52
1,905,597.75	474,633.53	481,597.83	22,202.44	504,889,416.49	30,380,398.84
1,735,044	2,018,888.75	133,688.38	83,359.82	486,513,221.79	124,046,075.9
1,687,839.62	56,295.46	422,305.38	5,762.52	1,008,964,853.91	13,363,532.74

For a start, some of those numbers are really high!

They are above 1 millions. Assuming 8K block size, that's 8 GB per second, sustained for 300 seconds.

What else do you notice?

Page-In is higher than Page-Out. I average all the 3K VMs and I got the following result:

102,768.57	24,285.03	36,470.28	4,063.16	48,519,306.68	24,862,411.77

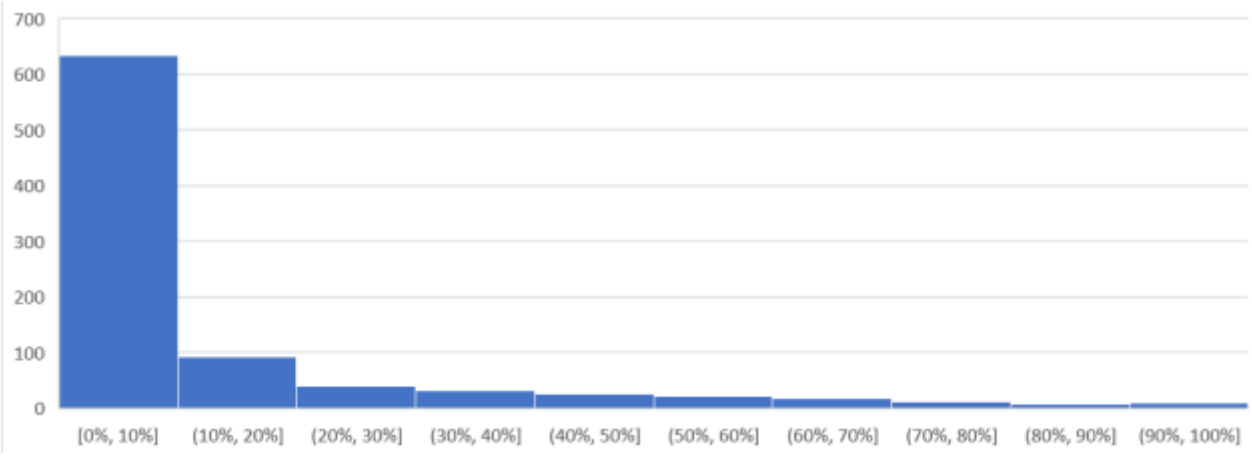
Page-In is 4x higher in the max value. Page-In also sustains longer, while Page-Out drops significantly. At the 99th percentile mark, Page-In is 9x higher. I suspect it is the non-modifiable page, like binary. Since it cannot be modified, it does not need to be paged out. It can simply be discarded and retrieved again from disk if required.

The good news is both do not sustain, so the paging is momentary. The following shows that the value at 99th percentile can drop well below 5x.

Max Page Out ↓	99P Page Out
132,995.14	1,010.02
124,549.87	1,333.71
116,500.87	15,672.75
110,829.6	44,771.87
105,329	21,409.8
104,753.87	40,857.52
102,492.13	22,379.24
100,612.2	34,733.32
99,579.73	435.82

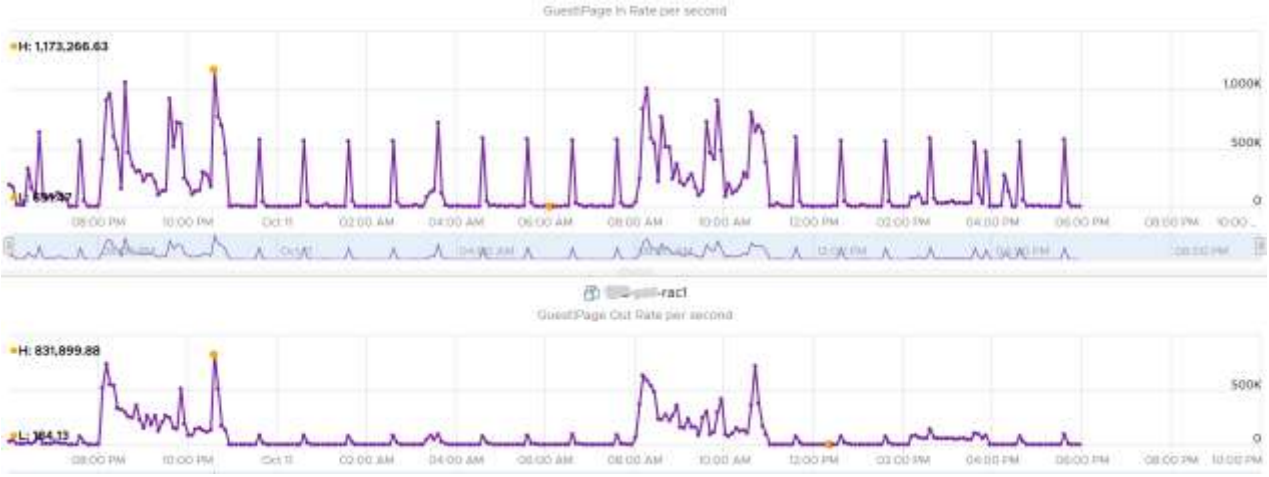
Max Page In ↓	99P Page In
5,069,497	29,154.43
2,669,535.75	803,049.24
2,354,336.75	57,443.6
2,049,341.62	292,200.29
1,981,613.75	1,341,561.23
1,942,909.12	1,175,849.13
1,905,597.75	481,597.83
1,735,044	133,688.38
1,687,839.62	422,305.38

To confirm the above, I downloaded the data so I can determine if the paging is indeed momentarily. Using a spreadsheet, I build a ratio between the 99th percentile value and the maximum value, where 10% means there is a drop of 10x. I plotted around 1000 value and got the following.



As you can see, majority of the paging drops drastically at 99th percentile.

Let's dive into a single VM, so we can see pattern over time. I pick a database, as it does heavy paging. The following is a large Oracle RAC VM. Notice this has a closer ratio between page in and page out, and there is correlation between the two.



Assuming the page size is 4 KB, that means 100,000 pages = 400 MB/sec. Since VCF Operations averages the value over 300 seconds, that means 400 MB x 300 = 120 GB worth of paging in 5 minutes!

Active File Cache Memory

This is the actively in-use subset of the file cache. Unused file cache and non-file backed anonymous buffers (mallocs etc) are not included.

This is the size of the portion of the system file cache which is currently resident and active in physical memory. The System Cache Resident Bytes and Memory \ Cache Bytes metrics are equivalent. Note that this counter displays the last observed value only; it is not an average during the collection period.

For further reading, refer to Windows

Committed

Commit sounds like a guaranteed reservation, which means it's the minimum the process can get.

This tracks the currently committed virtual memory, although not all of them are written to the pagefile yet. It measures the demand, so commit can go up without In Use going up, as Brandon Paddock shares here. If Committed exceeds the available memory, paging activity will increase. This can impact performance.

Commit Limit: Commit Limit is physical RAM + size of the page file. Since the pagefile is normally configured to map the physical RAM, the Commit Limit tends to be 2x. Commit Limit is important as a growing value is an early warning sign. The reason is Windows proactively increases its **pagefile.sys** if it's under memory pressure.

The pagefile is an integral part of Windows total memory, as explained by Mark Russinovich explains here. There is Reserved Memory, and then there is Committed Memory. Some applications like to have its committed memory in 1 long contiguous block, so it reserves a large chunk up front. Databases and JVM belong in this category. This reserved memory does not actually store meaningful application data or executable. Only when the application commits the page that it becomes used. Mark explains that "when a process commits a region of virtual memory, the OS guarantees that it can maintain all the data the process stores in the memory either in physical memory or on disk".

Notice the word **on disk**. Yes, that's where the pagefile.sys comes in. Windows will use either the physical memory or the pagefile.sys.

So how do we track this committed memory?

The metric you need to track is the Committed Byte. The % Committed metric should not hit 80%. Performance drops when it hits 90%, as if this is a hard threshold used by Windows. We disabled the pagefile to verify the impact on Windows. We noticed a visibly slower performance even though Windows 7 showing >1 GB of Free memory. In fact, Windows gave error message, and some applications crashed. If you use a pagefile, you will not hit this limit.

We have covered Free Memory and Committed Memory. Do they always move in tandem? If a memory is committed by Windows, does it mean it's no longer free and available?

The answer is no. Brandon Paddock demonstrated here that you can increase the committed page without increasing the memory usage. He wrote a small program and explained how it's done. The result is Windows committed page is double that of memory usage. The Free Memory & Cached Memory did not change.

Needed Memory

This is *not* a raw counter from Windows or Linux. This is a derived counter provided by VMware Tools to estimate the memory needed to run with minimum swapping. It's a more conservative estimate as it includes some of the cache. It has 5% buffer for spike, based on the general guidance from Microsoft. Below this amount, the Guest OS may swap.

```
= physical memory - Maximum of (0, ( Unneeded  - 5 % of physical ))
  where Unneeded = Free + Reserve Cache + Normal Priority Cache
```

Example: the VM has 10 GB of RAM. So the Physical RAM = 10 GB

So 5% of physical = 0.5 GB

Situation 1: max memory utilization.
Memory Available = 0 GB.
Tools will calculate Needed memory as
= 10 GB - Maximum (0, 0 – 0.5)
= 10 - Maximum (0, -0.5)
= 10 - 0 GB
= 10 GB
Needed memory is the same as it's already maxed.

Situation 2: high memory utilization.
Memory Available = 2 GB.
Tools will calculate Needed memory as
= 10 GB - Maximum (0, 2 – 0.5)
= 10 - Maximum (0, 1.5 GB)
= 10 - 1.5 GB
= 8.5 GB
You actually still have 2 GB here. But Tools adds around 5%

Situation 3: low memory utilization.
Memory Available = 8 GB.
Tools will calculate Needed memory as
= 10 GB - Maximum (0, 8 – 0.5)
= 10 - Maximum (0, 7.5 GB)
= 10 - 7.5 GB
= 2.5 GB
Again, Tools adds around 5%.

We've covered that you need to look at more than 1 metric before you decide to add more memory. I'm afraid it is case by case, as shown in the following table. All these VMs are low on free memory, but other than VM on row no 3, the rest has sufficient memory.

Name	Free	Capacity	Needed Memory	Page In	Page Out	GuestIActive
or...	0.1 GB	16 GB	10.2 GB	650.6	968.27	3.07 GB
ps...	0.1 GB	8 GB	7.39 GB	0	0	0.17 GB
ap...	0.1 GB	12 GB	12 GB	306,841.47	494.53	0.07 GB
AT...	0.11 GB	8 GB	6.2 GB	10,492.6	1,071.4	0.74 GB
wd...	0.11 GB	8 GB	3.42 GB	0	0	0 GB
ps...	0.11 GB	8 GB	7.39 GB	0	0	0.13 GB
vc...	0.11 GB	8 GB	7.39 GB	1.13	0	0.32 GB
BL...	0.11 GB	12 GB	8.19 GB	2	0	0 GB
ps...	0.12 GB	8 GB	2.34 GB	0	0	0 GB
ws...	0.12 GB	32 GB	22.28 GB	2,636.53	0	0 GB
sk...	0.12 GB	24 GB	9.61 GB	68,455.2	5,495.53	7.62 GB
AT...	0.12 GB	8 GB	1.39 GB	0	12.13	3.37 GB
sk...	0.13 GB	24 GB	10.52 GB	5,069.87	3,556.33	7.09 GB

Storage

This is the layer that application team care as it is what is presented to them.

Questions	Description
Configuration	For each partition, need to know name, filesystem type (e.g. NTFS, ext4), network or local, block size. Ideally, we get the mapping between partition and virtual disk.
Capacity	For each partition, need to know the configured space and used space. For free space, we need to know both in absolute (GB) and relative (%). Need to alert before running out of disk space, else the OS crashes. We should not include the networked drive in Guest OS capacity, because the networked drive is typically shared by many. An exception is in VDI use case, where the user personal files is stored on the network.
Reclamation	This can be determined from the free space. Reclamation is tricky as it needs to shrink partition.
Performance	Queue, Latency (read and write), IOPS, Throughput

Disk Queue

With VMware Tools, you get Guest OS visibility into the partitions and disk queue. The first one is critical for capacity, while the second is critical for performance.

This counter tracks the queue inside Linux or Windows storage subsystem. It's not the queue at SCSI driver level, such as LSI Logic or PVSCSI. If this is high then the IO from applications did not reach the underlying OS SCSI driver, let alone the VM. If you are running VMware storage driver, such as PVSCSI, then discuss with VMware Support.



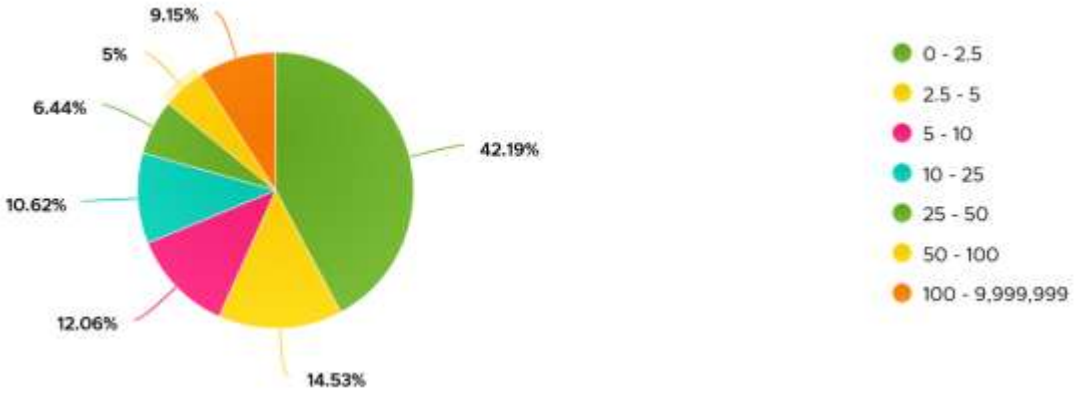
There are actually 2 metrics: One is a point in time and the other is average across the entire collection cycle. Point in time means the snapshot at the collection period. For example, if the collection is every 5 minute, then it's number on the 300th second, not the average of 300 numbers.

Windows documentation said that "Multi-spindle disk devices can have multiple requests active at one time, but other concurrent requests await service. Requests experience delays proportional to the length of the queue minus the number of spindles on the disks. This difference should average < 2 for good performance."

guest.disk.queue	Win32_PerfFormattedData_PerfDisk_PhysicalDisk.Name = \"_Total\"#CurrentDiskQueueLength" from WMI
guest.disk.queueAvg	Win32_PerfFormattedData_PerfDisk_PhysicalDisk.Name = \"_Total\"#AvgDiskQueueLength" from WMI

High disk queue in the guest OS, accompanied by low IOPS at the VM, can indicate that the IO commands are stuck waiting on processing by the OS. There is no concrete guidance regarding these IO commands threshold as it varies for different applications. You should view this in relation to the Outstanding Disk IO at the VM layer.

Based on 3000 production VMs in the last 3 months, the value turn out to be sizeable. Almost 70% of the value is below 10. Around 10% is more than 100 though, which I thought it's rather high.



Strangely, there are values that seem to *off the chart*. I notice this in a few metrics already, including this. Look at the values below. Do they look like a bug in the counter, or severe performance problem?

Name	Guest\Disk Queue
hq...	214,653.22
av...	214,748.16
av...	214,748.19
atl...	214,748.33
atl...	214,748.64
atl...	214,749.12
atl...	214,750.88

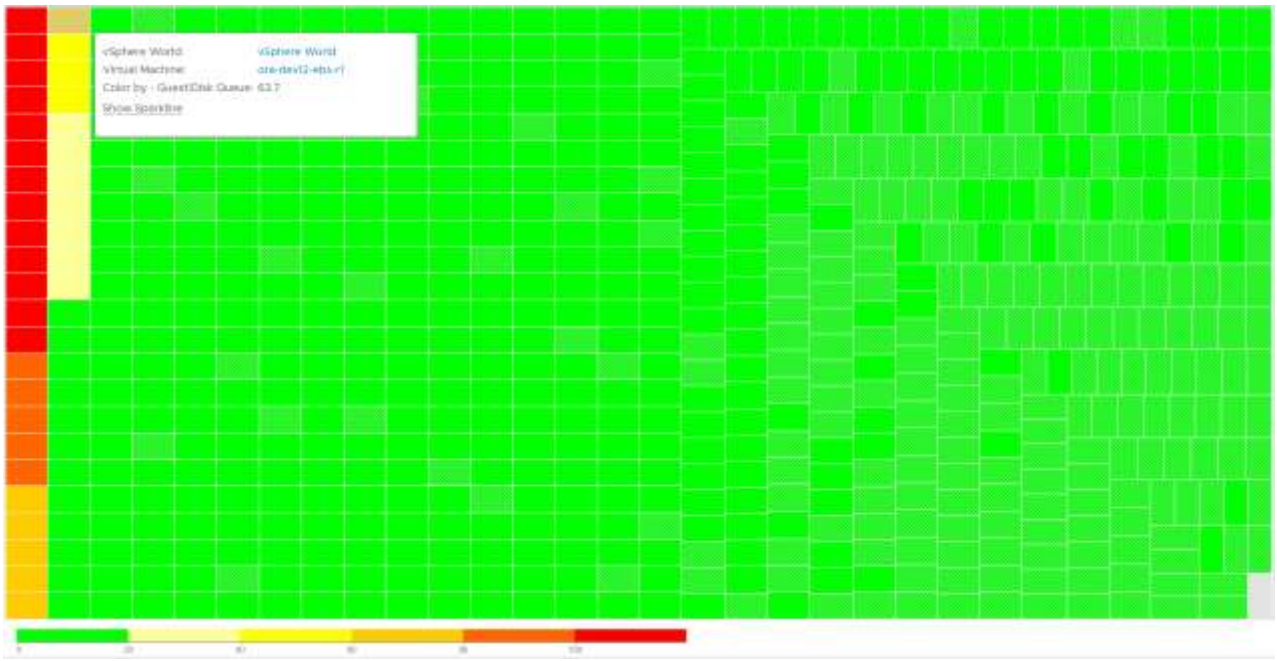
Unfortunately, we can't confirm as we do not have latency counter at Guest OS level, or even better, as application level. I am unsure if the queue is above the latency, meaning the latency counter does not start counting until the IO command is executed.

I plot the values at VM level, which unsurprisingly does not correlate. The VM is tracking IO that has been sent, while Guest OS Disk Queue tracks the one that has not been sent.



The preceding line chart also reveals an interesting pattern, which is disk queue only happens rarely. It's far less frequent than latency.

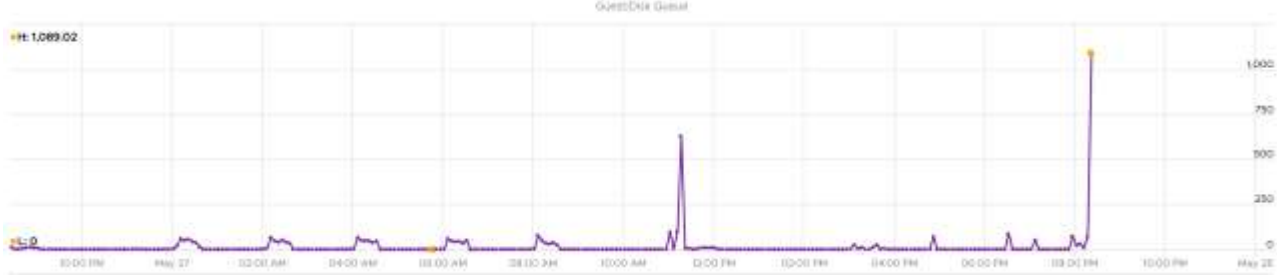
Let's find out more. From the following heat map, you can see there are occurrences where the value is >100.



However, when we compare between current value and maximum value, the value can be drastically different.

Name	Current Queue	Current Outstanding IO	Max Disk Queue ↓	Max Outstanding IO	vDisk
AT...	196.82	0.8 OIOs	5,036.62	4.24 OIOs	5
CO...	0.6	0.13 OIOs	2,411.2	22.64 OIOs	8
hd...	0.0033	0.0017 OIOs	2,132.78	9.23 OIOs	7
hd...	0.0067	0.0018 OIOs	1,809.57	12.82 OIOs	7
vc...	0.5	0.0004 OIOs	1,152.99	0.8 OIOs	9
se...	77.9	17.11 OIOs	939.91	633.83 OIOs	3
hd...	4.36	1.25 OIOs	890.18	460.62 OIOs	10
hd...	6.79	2.51 OIOs	879.43	319.89 OIOs	10
hd...	0.0067	0.0027 OIOs	868.89	8.22 OIOs	7

Let's take one of the VMs and drill down. This VM has regular spikes, with the last one exceeding 1000.



Their values should correlate with disk outstanding IO. However, the values are all low. That means the queue happens inside the Guest OS. The IO is not sent down to the VM.



Which in turn should have some correlation with IOPS, especially if the underlying storage in the Guest OS (not VM) is unable to cope. The queue is caused by high IOPS which cannot be processed.



Finally, it would manifest in latency. Can you explain why the latency is actually still good?



It's because that's from the IO that reaches the hypervisor. The IO that was stuck inside Windows is not included here.

The application feels latency is high, but the VM does not show it as the IO is stuck in between.

Can the disk queue be constantly above 100?

The following VM shows 2 counters. The 20-second Peak metric is showing ~200 – 250 queue, while the 5-minute average shows above 125 constantly. The first counter is much more volatile, indicating the queue did not sustain.

