# Private Cloud Management

## Transforming VMware SDDC into Your IaaS



**PART 1 OF 4**

**October 2026**

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

5 Edition

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



*The book is in your hands because of the couple above.*
*It is dedicated to the loving memory of Mama and Papa…*
*for your love and sacrifice in raising me in the old town of Suroboyo.*

# Foreword

Digital transformation is one of the most significant contributors to business transformation. In this agentic AI era, data center modernization, application modernization, and cloud repatriation are the norms. Broadcom VMware Cloud Foundation is at the core of these transformations for many companies globally.

Iwan has spent 3+ decades in the field working with companies of various sizes to make their "IT transformation" a success. He is the go-to person for our product managers, UX designers & engineering for mapping VMware Cloud Foundation metrics into day-to-day operations. I first met him back in 2015 VMworld and he has since become a trusted technical advisor to our product management team globally.

The book is deeply technical in content. Reading this book feels like having a conversation with Iwan. He has taken time to explain the concept, showing the value of each metric, and mapping them together to answer real-world questions. Many oddities make sense and complexities clear once you understand the underlying architecture.

I am always thankful to have him in my team and proud of his accomplishments

Kameswaran Subramanian

Product Management

VMware Cloud Foundation

Broadcom

# The Big Picture



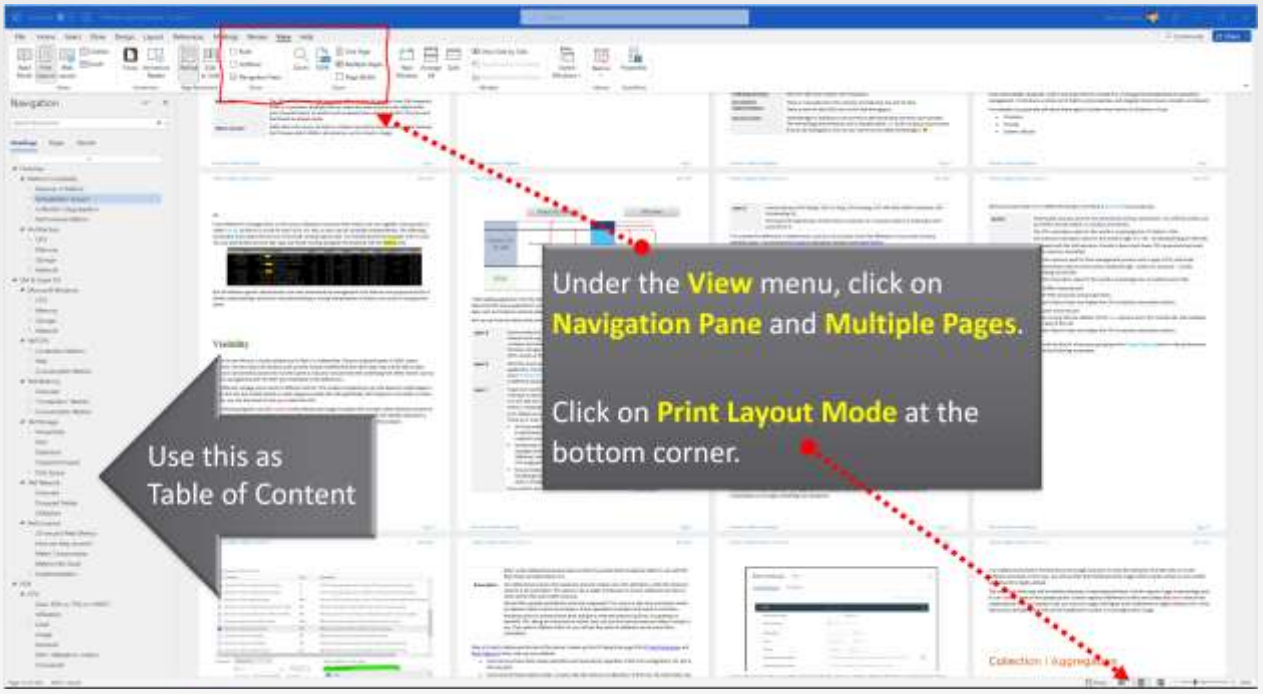

The program is delivered as a small and focused workshop among practitioners who are responsible for operating private cloud based on VMware Cloud Foundation.

It dates back to 2016, when Kenon Owens created Operationalize Your World and. We delivered dozens of sessions world wide, especially in Asia Pacific.

| | |
|---|---|
| **What this book is** | An informal book.<br>Think of it as I'm talking to you. We're just 2 IT Professionals having a discussion. |
| | A living book.<br>The main reason why I open source the book is it is a call for collaboration to the VCDX, VCAP and all VMware professionals |
| | An advanced book.<br>This is a level 300 book, written for VMware practitioners. |
| | An editable book.<br>Keep the good parts, throw away the bad ones, add your unique contents, and voila you have documented your operations! Post a screenshot of a cool customization you've done and you'll make my day! |
| **What this book is not** | It is a solution book, not a product book. It does not cover VMware Cloud Foundation feature by feature. It is not a place where you learn the product. There are many materials on the Internet on how to use the product. There is also the official manual. You do read them, don't you? 😉 |
| | It focuses on the management, not the architecture, aspect of private cloud. So, no coverage on VCF design, deployment and product troubleshooting |
| | It also does not cover all aspects of operations, such as process innovation, organizational structure, and financial governance. VCF Cloud Operating Model covers that. |

# How To Use This Book

The book is designed to be consumed as offline Microsoft Word document on Windows. It is not designed to be printed. Its table of content is the side menu of Microsoft Word. Follow the steps shown on following screenshot:




Use the navigation pane as a dynamic table of content, else it's easy to get lost even when using 43" monitor. If you simply read it top down, without having the navigation on the left, you will feel that the chapters end abruptly. The reason is each chapter does not end with a summary, which is required in printed books but redundant in online books.

# Table of Contents


**Part 1**
**Concepts**

Chapter 1: Your Private Cloud
Chapter 2: Performance Management
Chapter 3: Capacity Management
Chapter 4: Configuration Management
Chapter 5: Cost & Price Management
Future chapters:
Chapter 6: Compliance Management
Chapter 7: Availability Management

**Part 2**
**Consumption**

Chapter 1: Proactive Operations
Chapter 2: The Art of Dashboard
Chapter 3: Inventory
Chapter 4: Performance
Chapter 5: Capacity
Chapter 6: Configuration
Chapter 7: Availability
Chapter 8: Role-based Contents
Chapter 9: Logs Analysis
Chapter 10: Reports

**Part 3**
**Counters**

Chapter 1: The World of Metrics
Chapter 2: CPU
Chapter 3: Memory
Chapter 4: Storage
Chapter 5: Network
Chapter 6: Consumer Objects
Chapter 7: Provider Objects
Chapter 8: MS Windows


| | |
|---|---|
| **Concept** | It covers all the pillars of IT Operations, and the best practices to manage them.<br>It explains the root cause of why IT struggles with multi-cloud operations, and then shares a new paradigm that has resonated well with customers. It aims to correct decade-old misconceptions on "best practices". |
| **Consumption** | It covers dashboards, alerts, reports that implement the concept, so it can be consumed by the right persona. This part gets hands-on and assumes knowledge of the products. |
| **Counters** | The reference section. It goes deep into metrics used to implement the dashboards, reports and alerts in Part 2.<br>Reach out to collaborate on NSX and vSAN metrics.<br>Beyond metrics, we have events, logs, and properties. |

You do not have to read the chapters in sequence from part 1 to part 3, although it will be easier to understand if you read in order. Because each part stands alone, the chapter numbers within each part restart at 1. The other reason is I'm tired of manually renumbering the darn chapters!

## Assumptions

| | |
|---|---|
| **Version** | You have the latest release of VMware VCF Operations. |
| | Having said the above, some contents of the book can be implemented using earlier releases. To assure you that you can do that, I have kept screenshots from older versions whenever possible. It is also my excuse for not updating the old screenshots! |
| **Customization** | The ability to customize VCF Operations to your operations has been the hallmark of the product since it was released more than a decade ago. I was a pre-sales engineer when I first saw 1.0 a long time ago, in a galaxy far away. The ability to slice and dice an environment was a godsend, as I spent a lot of time working with customers troubleshooting and optimizing their environment.<br>This latest evolution continues the tradition of the book by showcasing the full potential of the product. |
| | IT Operations are like fingerprints. Two companies can have identical architecture, yet their operations will not be the same. Their policies and processes will differ, and so are the people and politics, hence this book supports the idea that you tailor the product to your unique operations.<br>A great tool to tailor is vCommunity adapter by Onur Yezseven and VCF Content Factory by Scott Bowe. |

## Part 4

I added Part 4 for miscellaneous stuff that I'm unsure where to put. Some chapters should eventually become its own book as they share a common theme.

Chapter 1: Super Metrics. Authored by Varghese Philipose.

Chapter 2: Automation. Authored by Thomas Kopton.

Chapter 3: SDDC vs IaaS

Chapter 4: Terminology

Since Part 4 is the last one, it's also where I put a personal note to wrap up the book.

## Removed Chapters

I've removed the following chapters. If you need them, just reach out to me on LinkedIn.

| Chapter | Reason |
|---|---|
| Availability Management | The content is not deep enough to exist as a separate chapter. |
| Compliance Management | As above |
| Kubernetes | As above. If you need the formula of the metrics, reach out to me via LinkedIn. |
| True Visibility Suite Dashboards | Authored by Brock Peterson |
| Green Dashboards | Authored by Varghese Philipose |
| Log Insight | Authored by Marine Harutyunyan and Samvel Israelian |
| VMware IT Story | Authored by George Stephen from VMware IT |

*This page is intentionally left blank.*

*Why? I don't know. Some people do it, so I just follow as IT behaves more like fashion nowadays…*

# PART 1

# Concept

# Your Private Cloud

## Part 1 Chapter 1

This first chapter provides a tour of IaaS operations management, starting with why reactive and hectic operations is common, and the paradigm shift required to proactive & predictive operations.

# Overview

What you architect is **SDDC**. But what you handover as a business result to your CIO is **IaaS**. What you bought from your vendor is SDDC, but what you sell to your customers is IaaS.

The transformation from SDDC to IaaS requires Operations Transformation. We transform from complaint-based to SLA-based, which requires fundamental process changes from alert-driven to insight-driven.

SDDC is a system, IaaS is a service. A system cares about its architecture, while a service cares about its service level.

Whether the Application Team or VM Owner pays for the service with a chargeback model or not, it is a service. VM Owners no longer own, hence care, about the underlying architecture.

They are 2 sides of the same coin. We can assess if the architecture is good or not, based on the actual result in production. Does it result in firefighting and blamestorming? Or do you have peaceful operations where alerts are meaningful and actionable?

Many operations rely on alerts as the starting point. Actions are taken based on alerts, resulting in reactive day-to-day operations.

IT Operations covers a wide area of systems. It's common to see more than 1K alert definitions across all systems under monitoring. As the team wants to be alerted early, a conservative threshold is set up. This results in alert storms.

Since automation is perceived as the holy grail of solutions, alerts are typically set to auto close if the symptom disappears. The creates a bigger problem, common in large enterprises with a large IT team. That problem is "lazy operations", where no alert is associated with no problem.

# Complaint-based Operations

How do you know that the Infrastructure as a Service (IaaS) Platform (be it on-prem private cloud or externally in the cloud) is serving its workload *well*? If you depend on complaints, then you run "complaint-based" operations.

Changing from reactive to proactive is unfortunately a complex undertaking, especially in large organizations where there are many roles and personas. It requires operations transformation and a paradigm shift. It is not easy to get customers to agree on a Service Level Agreement (SLA) when you've promised them "good" for years already. This book aims to provide practical guidance, something you can implement with the current version of Aria products.

# The Litmus Test

The following questions below help you assess the maturity of your IaaS business.

## Is your IaaS cheaper than public cloud?

The commoditization of infrastructure means your IaaS is being compared with similar platforms such as VMware Cloud on AWS and Amazon Web Services.

If not, your CIO may question your business value. The primary reason for having an in-house architect is so you can bring better price/performance, after taking into account your salary.

## Do your customers blame your IaaS?

If the answer is yes, take a moment to ponder why. There is a high chance you are relying on complaints in your operations, so you actually encourage them. No complaint, no problem. That's why it's aptly named Complaint-based Operations.

The reason why you rely on complaints is the operations team have no other means by which to measure success. You have not defined the performance of your IaaS. That's one of the goals of this book.

A sign of matured operations is that you have complete, correct, and accurate SLAs (Service-level agreements). Complete means you have Performance SLAs and Compliance SLAs, not just Availability SLAs. Correct means the SLA is measured on each *paying* VM, and not at the infrastructure level. It also means you use the right metrics. Accurate means the measurement has to be measured every 5 minutes, as any longer intervals than this can miss the problem.

## Does troubleshooting mean all hands-on deck?

Do you have a process that is followed by all teams (network, storage, server, OS, application)? Does that process end with Root Cause Analysis (RCA)?

As part of RCA, do you set up alerts so the same issue can be detected faster if it happens again? Without an alert configured, the RCA should not be closed. The alert is necessary as it will trigger the next RCA process.

## Does Help Desk provide a good first level defence?

If Help Desk simply passes issues through to the next level, you need to look at why.

Help Desk is your first line of defence. They are not as technical as you are. Equip them with Standard Operating Procedures and simple dashboards so that they can handle VM Owner complaints by discovering:

- Is the problem caused by IaaS not serving the VM well?
- If yes, which part of the infrastructure: CPU, RAM, Disk, Network?
- If not, how to prove it convincingly?

### Do you struggle with many over-provisioned VMs?

This is an indicator that you are operating as a System Builder as opposed to a Service Provider. As a System Builder, you are meddling with each System (read: Application). You size them and argue with the application teams, who are actually your customers. You are busy as there are many applications, and you are outnumbered.

If you are operating as an internal Cloud Service Provider, you should not be "in the way" of the business. You use an effective pricing model to drive the right behaviour. Does a public cloud provider block application teams when they buy 40 CPU AWS EC2 VMs when they only need 2 CPU? They don't, hence neither should you.

### Can you justify new infrastructure when utilization is not high?

This is not referring to additional money that comes with new projects. This is referring to existing workloads on existing clusters/storage.

Capacity is measured on utilization and performance. A cluster is at full capacity if it can't serve its VMs well. Since it takes time to buy hardware, you must have an *early warning* system to detect this performance degradation.

## Common Mistakes

"If you don't have a problem, I don't have a solution" summarizes how I engage customers. After >1.5 decades of engagements with hundreds of VMware customers and outsourced partners, here are typical mistakes I've observed:

- Using automation as the primary solution for transformation.
- Private Cloud is seen as *automation* project as opposed to operation. Private Cloud is not virtualization with automation and self-service. It is the required technical foundation to transform the business of enterprise IT from system builder to service provider. The automation, workflow and self-service portal are merely supporting features. The primary components of Private Cloud are SLAs and Class of Service, hence it's operations-centric, not automation-centric.
- VMware Cloud Foundation is architected with server-consolidation mindset. That means the system has no awareness of IaaS and SLAs. Different classes of service are mixed in the same cluster or datastore.
- There is class of service, but the system does not clearly state it. The naming standard does not include class of service.
- Performance is never defined properly. The infrastructure is designed for performance, but the benchmark does not align with what actually being sold. There is no Performance SLA, and often there are no Key Performance Indicators (KPI)[1].
- The infrastructure has no awareness of business units, applications, or application-tiers. The business is not reflected in the infrastructure.

[1] For explanation of the terminology, refer to the Terminology section at the end of this book.

# Maturity Model

It's a good practice to assess the level of operational maturity, as it allows you to summarize where you are. There are different variants of these models, so don't be hesitant to tailor to your goals. I've included a short assessment within Part 4 Chapter 1 to get you going.

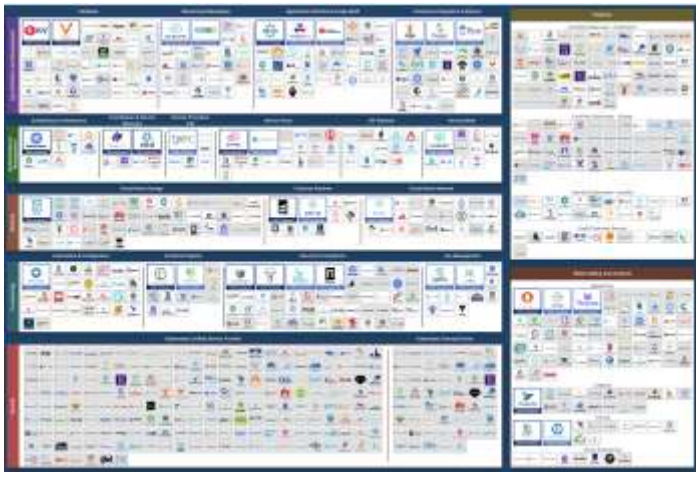
When scoring yourself, assign score on the following area:

- **Policy**: Is your policy outdated? Best Practice typically means proven or common practice.
- **People**: How skilful is the team vs the need? This includes the way the team is organised.
- **Process**: How effective are the key processes (e.g. planning process, troubleshooting process)?
- **Pillar**: How mature is each pillar of operations? For example, if your capacity management is mature, you are balancing cost and capacity very well. If your performance management is mature, you're not reactive to endless complaints because you have SLA formally agreed.
- **Platform**: This covers both the technology supporting the business workload, and the IT tools used by the operations team to support the former. For example, if you do not have clear visibility, you're flying blind.

## Multi-Cloud Management

A single private cloud - something you have complete control of - is hard enough to operate, let alone operating multiple incompatible infrastructures. Multi-cloud operations, where you are responsible for something that you do not have complete controls take the operations challenge to the next level. Don't be disheartened if your organisation is struggling with running multi-cloud operations.

The complexity is due to the immaturity of the architecture. There are simply too many components involved, as shown in the landscape diagram by Cloud Native Computing Foundation. The individual products that make up the architecture is not important, hence I intentionally make the diagram small.



Eventually though…, the architecture will slowly mature and turn into a commodity. CIOs will begin to focus on the operations, as the business will demand proper governance with SLAs.

Regardless of the underlying system architecture, CIOs are still required to manage cost, capacity, compliance, performance, and availability. The Pillars of Operations do not change just because you change the plumbing.

# The Business of IaaS

There are 3 variants of an IaaS business. They differ in terms of what you actually sell, how you do pricing, and what SLA you put on the table.

| Item | Pricing | Availability SLA | Performance SLA |
|---|---|---|---|
| **VM** | Price depends on VM size. A larger VM has a higher price.<br>Price depends on quality. A better tier has a higher price.<br>Example of an item purchased: 1 VM, 4 vCPU, 16 RAM, 200 GB disk, in Gold Tier. | Per VM.<br>Depends on the tier. | Per VM.<br>Depends on the tier. |
| **Resource Pool** | Sold per GHz, GB RAM, TB Disk.<br>Example of item purchased: 100 GHz CPU, 1 TB RAM, 80 TB Disk.<br>It can come with a 100% reservation; hence it's guaranteed. Alternatively, it may have partial reservation. It typically comes with best effort burst, in the form of expandable resource pool.<br>On the other hand, it may come with a limit but it's always higher than what you paid for. For example, you pay for 1 TB of RAM. You can have 0.5 TB guaranteed and a 2 TB limit. | Per VM.<br>Limited tiering capabilities. | N/A.<br>While Resource is reserved, customer is allowed to overcommit within their own limit. It's not something the IaaS service definition imposes. |
| **Hardware** | Price per ESXi host. Customers decide how many VMs they want to place. HA is provided by vendor.<br>Example of items purchased: 8 ESXi Hosts.<br>Example provider: Azure VMware Solution (AVS). | On the Host or Cluster, not VM. | N/A<br>Customer can squeeze as many VMs. |

Class of service is harder to implement in resource pools as there are more moving parts. You can have cascading resource pools.

## VM as a Service

The most popular variant of IaaS is VM as a Service. It is typical example of "buy wholesale sell retail" business. You buy in bulk (hardware, software) and commit DC space for years, then sell in small chunks (VM, K8 Pod). You make profit as your buy price is several magnitudes lower than your sell price, on a per unit basis. You probably pay 5x less per GHz than you sell.

| | |
|---|---|
| **Purpose** | Serve the workload. They take the shape of VMs. The VMs in turn can be K8 nodes or classic applications.<br>The workload must be grouped by tenant and business applications. |
| **KPI** | The key metrics used to measure the performance of the infrastructure.<br>Is it serving the VMs according to its SLA? |

| | |
|---|---|
| **Cost** | The total cost should be cheaper than public cloud.<br>Typically, customer aims at >2x cheaper, not just marginally cheaper. |
| **Pillars** | The key pillars that transform the SDDC into IaaS.<br>IaaS has multiple class of services. Each has their own Availability SLA, Performance SLA, Security SLA, and Service SLA. |
| **Proof** | The metrics demonstrating that the architecture works as intended.<br>Operations become proactive. It's based on insight, not alert. |

The business goal is to ensure the application and VMs are running well yet cost effective. In this way, you keep the customers happy.

The cost part is easy to quantify. You know what you actually spend on hardware, software, services and salary. The "well" in running well is the hard part as there is a big unknown. This is also the source of argument between application teams and infrastructure teams.

Say you are architecting for 10K VMs in 2 data centers. You envisage 2K VMs in the first month, 5K VMs in the first half year, and eventually to 10K within the first year. Do you know the basic info about each of these 10K VMs, so that you can architect an infrastructure to serve them well?

- How big are they? What are their vCPU, RAM, and Disk configuration?
- How intense are they? CPU utilization, RAM utilization, disk IOPS, network throughput?
- What are their workload patterns? Daily, weekly, monthly, no pattern, etc?

The answer is obviously no. Even application teams do not know as some of the applications may not be developed yet. Their vendors may not know either as the actual usage is not yet known.

Promising that the SDDC will serve all 10K VMs well is akin to promising the highway you architect will serve all the cars, buses and motorcycles well, when we can't predict how many they are and how often they will use it. We will cover this more in the Performance chapter.

So how can we promise that your IaaS will serve your customers well?

We can by using price/performance. The principle you share with your customers is the common sense principle used in all service industries:

- You want it cheap; it won't be fast.
- You want it fast; it won't be cheap.

This is where the Class of Service and the associated SLAs come in. The highest class of service provides the best uptime and performance but comes at a price. All these attributes are well defined in the SLA, leaving no room for ambiguity. The contract is not subject to interpretation. You define all the key metrics up front, assuring your customers that you are confident of delivering as promised.

You then architect your IaaS to deliver the above class of services. The class of service becomes your business offering. With that, you are ready to begin with the end in mind.

## Capabilities

The platform should provide a complete self-service portal for all types of users. The features should cover all stages in the life cycle, starting from provisioning. Provisioning should have an SLA and be supported with workflows and electronic approval.

### Key Metrics

You measure across the pillars of operations management. Each pillar is measured, hence managed, by a purpose-built metrics. This enables you to manage at scale.

| Pillars | Metrics | |
|---|---|---|
| Availability | Operational Availability (%) | Relative Availability, against your availability architecture and green zone. |
| | Actual Availability (%) | Absolute availability, reporting the fact as it is. This metric typically has lower value than Operational Availability (%). |
| Performance | KPI (%) | Absolute performance, reporting the fact as it is. |
| | SLI (%) and SLA (%) | Relative performance, against your promised SLA.<br>SLI = SLA Leading Indicator |
| Capacity | Capacity Remaining (%) | Relative to usable capacity, not total capacity. |
| | Time Remaining (days) | The number of days until Capacity Remaining (%) hits 0. |
| Compliance | Benchmark X (%) Compliant | Compliance against specific industry or internal benchmark, such as PCI-DSS.<br>1 metric per benchmark. |

## The Restaurant Analogy

Sunny Dua[2] and I use the restaurant analogy when explaining the need of SLA. The analogy has resonated well with many customers. Humans can always relate to food!

Essentially, a restaurant has 2 areas, often with a clear demarcation line:

- The Dining Area.
- The Kitchen.

Think of your IaaS business like a restaurant business. It has a dining area, where your customers live, and a kitchen, where you prepare the food. Guess which one is more important to the owner?

You are right. The dining area.

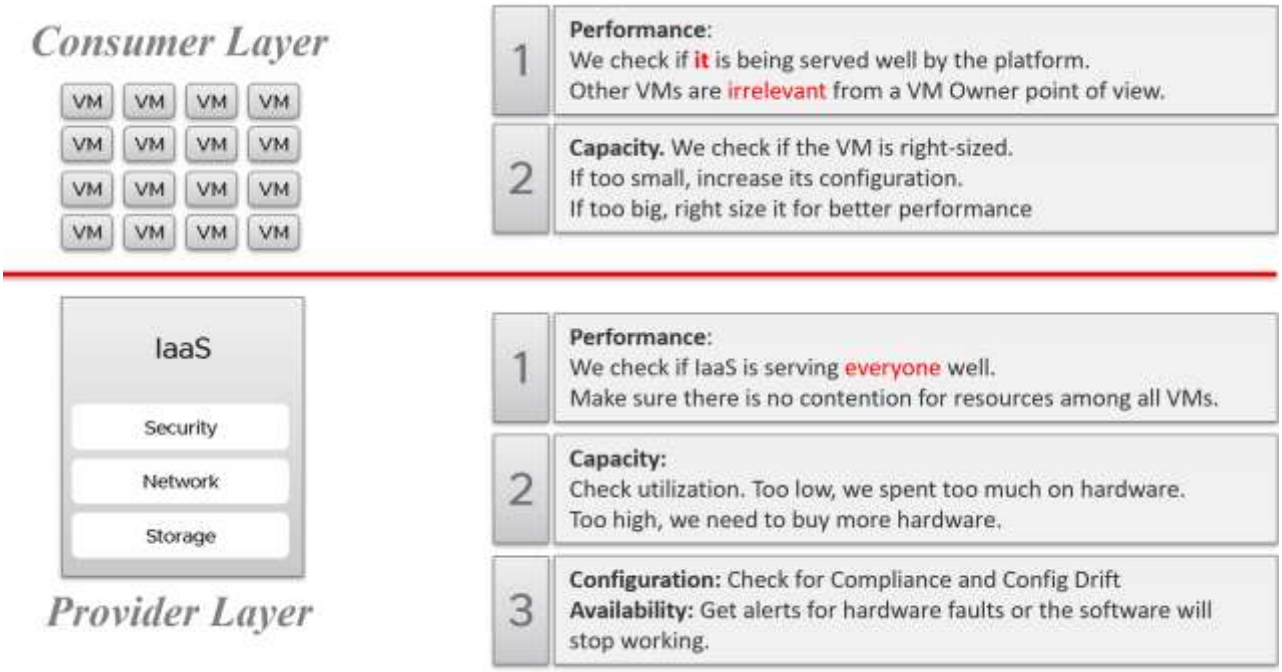
If everything runs smoothly in the dining area, customers are being served on time and on quality, and they are paying you well; it is a good day for the business. Whether you are running around in the hot kitchen is a separate, internal matter. The customers do not need to know about it.

We use the analogy to drive the message that you need to focus on the customers first, and your SDDC second. If you take care of your customers well, and they are happy with your service, the problem you have in your IaaS is a secondary and internal matter.

- The "dining area" is the Consumer layer. Look at the diagram below. It is where your customers' VMs live. In the public cloud such as AWS, that's all you can see.
- The "kitchen" is the Provider Layer. This is your infrastructure layer, where VMware and the hardware reside.

[2] Sunny and I went back a long way. We both came from the field before joining product team.

Public cloud is part of the kitchen. Just because you no longer own the infrastructure does not mean you don't take management responsibility. The structure of enterprise IT means the infrastructure team ends up being held accountable.



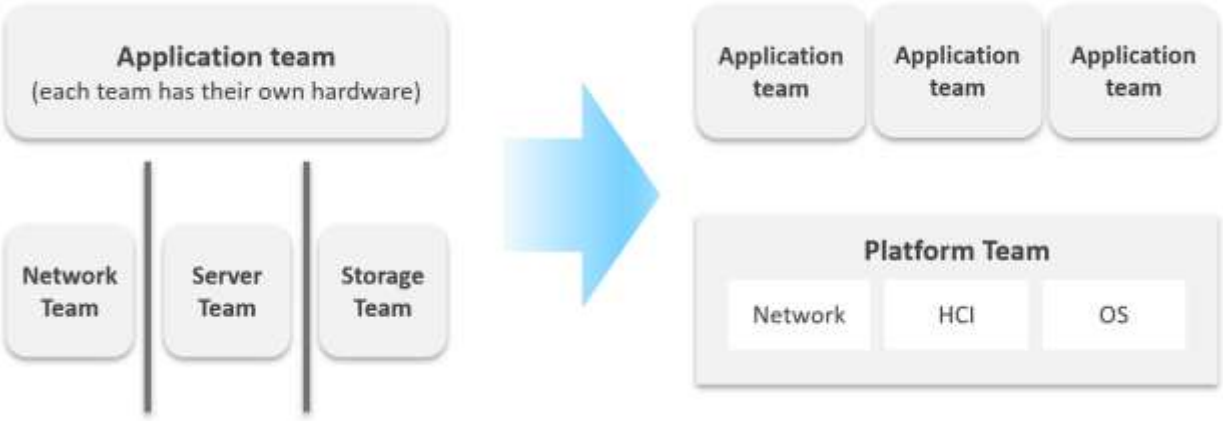

There is clearly a line of demarcation between the two layers. Your customers should not care about the details of your SDDC or EUC. The VM Owner does not care if you are firefighting in the data center. Because they do not care, whether you are using an older VMware Cloud Foundation or the latest, this is not something you want them to dictate to you. The same goes with your choice of hardware brand and specification.

Conduct regular sessions with the application teams on the following topics:

- How to run best on VMware, with optimal performance, highest availability, most secured while keeping cost minimal.
- How to monitor the performance, availability, and security when you're running on VMware. How to know you're being served well by the IaaS platform according to the promised.
- Windows and Linux performance best practices.
- Why rightsized is better than oversized for VM.

Understand their expectation of the infrastructure. In large environments, different VM Owners can have different expectations and levels of knowledge.

The application teams become consumers of a shared service—the cloud platform. Depending on the SLA, the application teams can be served as if they have dedicated access to the infrastructure, or they can take a performance hit in exchange for a lower price. For SLAs where performance is guaranteed, a VM running in the cluster should not be impacted by any other VMs. The performance must be as good as if it is the only VM running on the ESXi host.




Let's zoom into the kitchen area, as that's also undergoing a transformation. The Server team or Windows team or Linux team typically took the ownership of the shared platform and evolved to become the platform team. With the evolution of Hyper Converged Infrastructure, storage is being absorbed into the platform. The boundary with the Network team is also becoming blurry with network virtualization. Many network services such as Firewalls and Load Balancers are virtualized. Recently, with the arrival of the Kubernetes, the platform team began owning containers and K8s, plus there are new teams (DevOps and/or SRE) that sit in between Platform team and Application team.

# Purpose-Driven Architecture

When you architect IaaS or Desktop as a Service (DaaS), what goals do you have in mind? I don't mean the design considerations, such as availability and performance best practices. I mean the *business results* that your architecture has to deliver, viewed from the people who paid for the system, and by the people who will pay for the service. Set aside your opinion on the goal, as you neither pay for it nor use it.

Logically, the answer depends on what is being sold. You can either sell application or infrastructure, broadly speaking. Some popular examples are:

| Service | What you sell | Examples |
|---|---|---|
| **SaaS** | The software is provided as a service. Customers need not install it on-site. Common among ISVs who want to avoid on-prem installation, avoid outdated installations, or mine their customers' data. | Salesforce, VMware Skyline, Microsoft Office 365. |
| **DBaaS** | Database as a Service. There are 2 variants:<br>Instance: A customer shares the binary with others. Patching the software means all instances using the instance get patched. They all need to have common maintenance window.<br>Dedicated binary: Customers can have different versions, patch levels, and downtime schedules. | Examples such as Mongo DB as a Service or MS SQL as a Service are common among enterprise. The DBA provides this as service to the application team, who are not as deep on databases knowledge. |

| | | |
|---|---|---|
| **PaaS** | Platform as a Service.<br>A set of services used by business applications. AWS provides many such PaaS services and is the main reason why customers choose them. | Central IT provides a set of common services (e.g., login, payment) to all business units websites. |
| **DaaS** | Desktop as a Service. Typically, Windows 10 + End-User applications. Application teams must be involved, as a simple 10% CPU increase of your browser can impact performance SLAs as the ESXi host becomes heavily over committed.<br>The goal is to ensure End Users are getting a quality desktop experience while keeping the price per user low. | Many enterprises' IT provides this for better security and PC-manageability. They may deploy this with thin client.<br>VMware Horizon Cloud, Microsoft Windows Cloud are cloud examples. |
| **K8aaS** | Kubernetes as a Service. There are 2 variants:<br>Dedicated cluster.<br>Shared cluster. | Amazon EKS |
| **IaaS** | There are 3 variants here. As this is the topic of the book, let's explore in depth. | |

# Begin with The End in Mind

It's important to reflect the business in both the IaaS platform and your operations. It makes the infrastructure team aware of the context and impact to the business. In their day-to-day operations, they need to be Business Application centric. This calls for a paradigm shift.

Your CIO wants live information projected for his peers to see on how IT is serving the business. This requires you to have awareness of the business units and their critical applications.

In your service offering, you include the ability for customers to check their own VM health, and how their VMs are served by the underlying platform. This means your architecture needs to know how to associate tenants with their VMs. At the very least, create a structure so they can browse or find their applications and VMs.

## The Crown Jewel

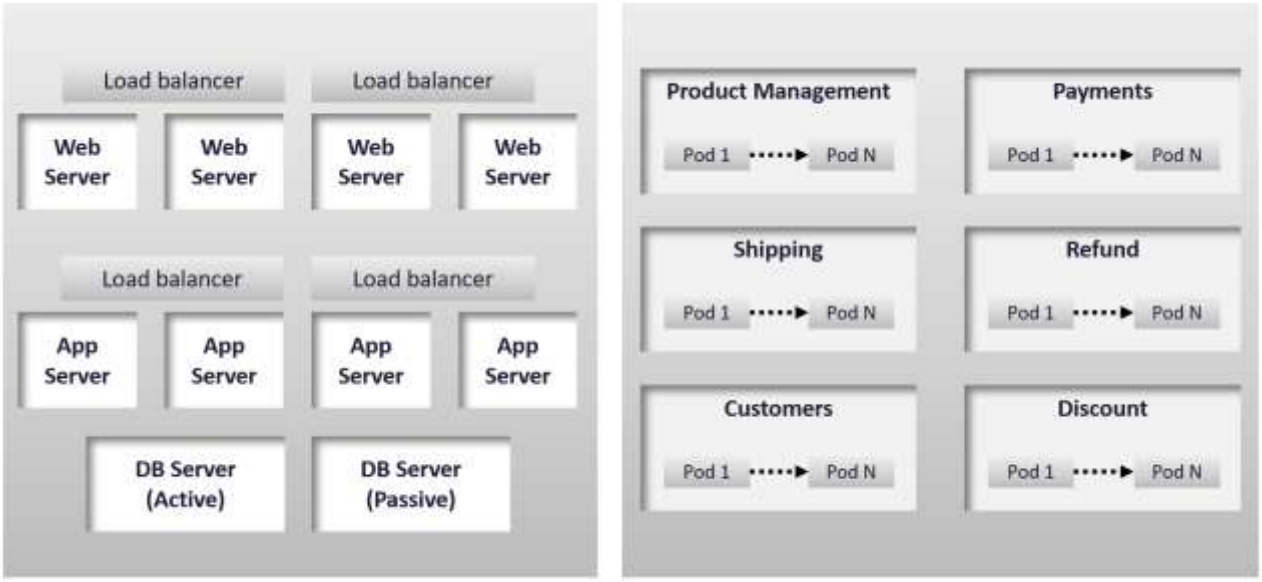
They are your mission critical business applications. They form a small percentage of your VM population. However, they are essential to the company revenue, profit or image. They are your crown jewel system.

Different industries have their own set of mission critical systems. If you are bank, it's your Internet Banking. If you're eCommerce, it's your online portal.

In order to be proactive, you need to monitor these applications closely. Likely a few times a day, not just one time in the morning. The application team is watching their own system throughout the day, so you can't afford to be blind to what's happening from infrastructure viewpoint.

Is the problem with your business application caused by your infrastructure? The problem is typically performance, although it could be availability or security.

You can create a universal model for all business applications since the infrastructure metrics are the same.




The health of a tier is the average health of its members. This is because a tier scales out across distributed instances. We are not taking the minimum value because processing within a tier does not happen in sequence. This is not a convoy. A good Load Balancer will balance both performance and availability.

"Hold on!", you might say. Since it is scale-out architecture, the Application Team has catered for this. If they only need 3 web servers, they will deploy 4 or even 5. So, both performance and availability are not affected if one web server goes down. The measurement of a tier has to take into account this extra capacity, and not simply do an average of all members.

This logic sounds reasonable. But is it correct?

It is correct for availability. It is not correct for performance. Performance needs to include all nodes as it reflects reality.

## Day 0 | Day 1 | Day 2

Planning is Day 0, Architecture is Day 1, and Operations is Day 2.

By Architecture, I mean the detailed technical work, including building and commissioning the system. While the business plan and high level marketecture[3] is defined during Day 0 (Planning), the real architecture work is done on Day 1.

It's a common misunderstanding to think the sequence is 0 → 1 → 2.

However, if we think deeper, Day 2 impacts Day 0, which is Planning. The reason is the End State drives your Plan. Your Plan drives your Architecture. So, it's 2 → 0 → 1, not 0 → 1 → 2.

[3] Marketing architecture. A jovial reference to beautiful PowerPoint based diagram that hides implementation complexity. Amazon jungle looks green and beautiful from 10 thousand feet, but on the ground it's a different matter.




Day 2 is not simply the first few days after you go live. It's the day you set sail[4].

Let's use an example to illustrate how Day 2 impacts Day 0, which in turn impacts Day 1.

Say you are an internal cloud provider, and you plan to charge per VM. You plan to have 2 classes of offerings:

- **Gold**: suitable for production workloads. Performance optimized.
- **Silver**: suitable for non-production workloads. Cost optimized.

For Gold, you plan to not overcommit CPU and RAM. If 1 CPU typically uses 4 GB RAM, then a 64-core ESXi host will only need 256 GB. If you buy a host with 1 TB RAM, then you may end up in a position where you are not able to sell the remaining 768 GB as you have no more vCPUs to sell. This means your hardware specification is impacted. That's an example of how Day 2 impacts Day 0.

For Silver, you plan to overcommit 4:1 for CPU and 2:1 for memory.

- You assume that 1 vCPU typically uses 4 GB RAM. Your customers are allowed to buy more or less memory, so this 4:1 ratio between CPU and RAM is just a guideline for overall planning.
- You plan to run vSAN with dedupe + NSX + vSphere Replication. You also expect heavy IO VMs, which requires kernel processing. For all these supporting, non-business workloads, you allocate 8 cores and 64 GB RAM.
- If you buy a 64-core ESXi, you have 56 cores left and you will be able to sell 224 vCPU.
- These 224 vCPU will need 896 GB RAM. Since you overcommit 2:1, you need 448 GB for VM. Total RAM you need is 448 + 64 = 512 GB.
- That means the hardware spec you need is 64 core and 512 GB RAM. If you buy more RAM than this, you may not be able to sell this extra RAM as you may not have vCPU to accompany them.

The above 2 examples show how your hardware spec can't be decided without considering the average VM profile and the overcommit ratio you plan. Yes, Day 2 does dictate requirements and constraints to Day 0.

You also promise the concept of Availability Zones for Gold class, as they host mission critical business services. Your company policy for Business Continuity dictates that in the event of an entire cluster failure, you plan to cap the number of VMs affected. If you limit to say 300 production VMs, then your cluster size should not be too big as you won't be able to fully utilize the resource. I've seen multiple customers having 32-node production clusters running 1K – 2K VMs.

[4] Day 2 will eventually hit technology refresh, which could be major overhaul in architecture. This will take you back to Day 0 as you need to plan for the new world.

## Promise vs Reality

In a large environment, you may have the luxury of designing different infrastructure for different workload types. Common examples are GPU Intensive workload, Disk Intensive, etc. If the infrastructure is superior to your standard offering, you need to be careful in setting the right expectation.

Let's take an example: you promise you can handle CPU Intensive workload as you've chosen the best CPU.

Notice the issues here?

There are at least 2 of them.

1. You probably heard of Winston Churchill quotation "Sometimes doing your best is not good enough." What you think is the best CPU may not be good enough for the workload, either in terms of GHz, number of threads, or power efficiency. For example, if your ESXi sports a 3.8 GHz speed but the application wants 5 GHz, giving it extra vCPU does not exactly meet the requirement.
2. Assuming you pass the first issue above, how do you prove that this so-called "best CPU" is actually able to handle the workload? What metrics do you use? Remember it's just a CPU. All you have as metrics are just GHz and vCPU. If you rely on the application team metric, you need to be prepared to spend time doing testing with them. You should also apply 100% reservation to eliminate infrastructure-level contention. The problem with reservations is that you cannot overcommit. It means you defeat the purpose of virtualization to begin with.

So, what can you do?

1. Set the right expectation. For example, you state that your infrastructure uses dynamic power management. In most cases, this is good for the application as they get Turbo Boost when they are running hard. In situation of light use, the application may run at lower speed.
2. Do not promise something you can't measure. In this case, the main metric you want to measure is "Is the CPU available when the VM asks for it". Metrics such as Ready, Co-stop, Overlap, and Other Wait track these moments of contention. You provide great observability by showing these metrics.
3. Measure what is relevant to your business. If what you offer (read: the SLA) does not guarantee that the *whole core* is available to the VM, then do not measure the time the VM vCPU runs on a shared core.

## VCDX | VCMX

Why do we draw a distinction between promise and reality?

My take is because IT Architects typically do not include Day 2 in the architecture. This is specific to VMware; hence I'm proposing VCDX should be accompanied by VCMX (Management). Designing the architecture and transforming the operations are 2 different skills[5].

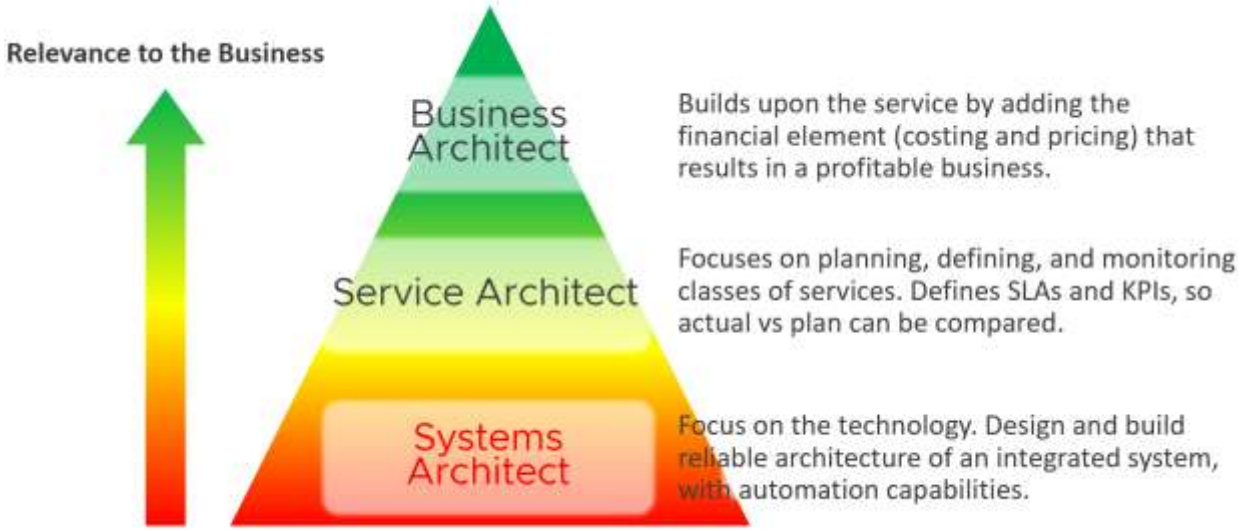
As a service provider, while your technical knowledge is important, your customer measures you on your service level. While they care about your systems architecture and its technological marvel, they measure you on service quality.

[5] For consistency with other VCDX certifications, we should call it VCDX Management. I use VCMX to drive a point. Certification wise, it can start at VCP level, such as VCP Management.




Architecture and Operations are two equally large realms. While we certainly consider Operations when designing a system, it is not a part of Architecture. This book is an example of Operations. Notice it goes deep into metrics as troubleshooting is at the heart of operations.

Architecture and Operations also differ in other industries. The person who designs the space shuttle is not the person operating it. You need to be an astronaut to be qualified to operate a space shuttle. The person who designs an F1 race car is not the person driving it. Different expertise is required. They complete each other and are inter-dependent, like Yin and Yang.




Since Infrastructure is becoming a service, you need to know how to architect a service (e.g., IaaS, Database as a Service, Desktop as a Service).

- What are the services the IaaS is providing? How do you define a service?
- What metrics do you use to quantify its quality?
- How many services? How do you distinguish between a higher-class service and normal one?

You also need to know what type of services are on demand. Service Architects go out, meet customers and understand their requirements. What classes of services are on demand now and in the future? From there, you can architect the corresponding services to anticipate the demand.

As a Business Architect, you not only know the cost of running the service, but you also know how & when to break even. You are not responsible for profit and loss, as you are not the CIO or Cloud Service Provider CEO, but you do play a strategic advisor role to them. You know what to price, how to price, and most importantly you know your price is competitive (or, at least you can provide a business justification for reasons other than price).

From my interactions with customers, I notice that Infrastructure Architects are not leading the Day 0 phase. They provide input to the Planning stage but are not the lead architects driving it. The Infrastructure Architect tends to focus on technical bits, something that CFOs and CIOs value less (hence they spend less time on it). They also do not architect the operations. I see many seasoned VMware Architects not extending their influence beyond architecture. I think that's a lost opportunity because Day 1 and Day 2 is actually part of the same side. Think of it as a Mobius strip.

Service Architect and Business Architect are the next steps for Infrastructure Architect. I shared story "The Chef and his cooking" back in 2014 during one of the VMUG session.

By the way, how do you know who is the real architect of a system? Let's say you have mega VDI system, with integrated components such as VMware, Omnissa, thin clients, office networks and many other things. Who is the true architect of this?

My answer:

```
The owner of the hands on the keyboard figuring out the root cause when there is a massive problem
that no one knows why. That's your real architect. The team who drew the architecture diagrams are
not.
```

# Automate | Operate

You need to account for situations where things go wrong, intentional or unintentional. Real problems happen in Day 2 as that's when you have business workloads doing revenue generating transactions. Do not architect something you are not willing to troubleshoot. Think of the roles and skills required to operate your architecture. Provide the necessary visibility into each component and define what constitutes health.

I hope the above examples show that Day 2 is where you want to start. As said in a famous quote: "Begin with the end in mind".

Did you notice something missing in the discussion above?

Yes, I did not cover Automation.

Why is that?

For me, that's part of architecture. You should not automate what you cannot even operate. So, automation is not part of operations. Automation is a feature of your Architecture, meaning you design the system with automation in mind. Using an analogy, it's like a plane with many automation features. Fly-by-wire. That's a *feature* of the plane. How you *use* the plane to ensure passengers arrive at the destination safely, comfortably, timely and fresh: that's operations.

In terms of transformation journey, automation should be placed last. Do not automate what you cannot operate. You're speeding up the problem if your operations is not well governed.




# Observability

Observability needs to be built-in, not bolted-on, in your system architecture. That means ensuring it can be monitored clearly, down to the smallest component. Do not deploy system into production that can't be properly monitored. Unfortunately, this is typically the last thing in our IT industry, especially with tight deadlines, limited skills and low budgets. Many applications, software, and hardware do not systematically leave trails for post-mortem analysis.

To "solve" the above problem, our IT industry came up with new buzzwords and make them larger than life. You hear jargons such as observability, unknown unknown, golden signal, SLO, SLI, and reliability. Observability is pitched as more than monitoring + troubleshooting, although it's just a nature of a system.

Observability is not a superset of monitoring. They are two different things.

| Observability | Monitoring |
|---|---|
| It is a property of the system to be managed | It is an action done by an actor |
| Observability and Debuggability are peers. Just because a system emits metrics and logs regularly, does not mean it has ways to be stepped through and debugged. | Monitoring & Troubleshooting are peers. Just because an admin has the skills to monitor a system, does not mean he can fix it. |

It is certainly important to elevate the proactive work to detect unknown problems that you are not even aware of. I love slicing and dicing millions of data points (metrics, logs, events, traces, NetFlow) and discover new insights. It also helps me understanding the behaviour of low-level metrics better. Many times, they are not what the manual says as documentation on metrics is often not deep. I've discovered many bugs in metrics from multiple vendors in the last 1+ decade.

CIOs should not only encourage, but also require the subject matter experts to allocate time to this proactive exploration. It should be part of a regular cadence to share work and findings. It's both a good exercise to keep the knowledge deep, plus you never know what you will discover!

Having said that, do we need a new term? I don't think so since it has caused confusion. Lots of software can do this, along with many other things. It's all part of monitoring and troubleshooting. There is no need to invent a new category of software. This is the classic "old wine in a new bottle" trick.

If we really want a new term, the word debuggability carries more value as just because a system is observable does not mean you can do something to fix it, let alone intervening to debug it. It is painful watching your system deteriorate with nothing you can do about it. In modern days, debugging is no longer limited to slow and manual process of stepping through code. It can involve in-line analytics as the codes processing. This certainly requires the system to be built with debuggability in mind.

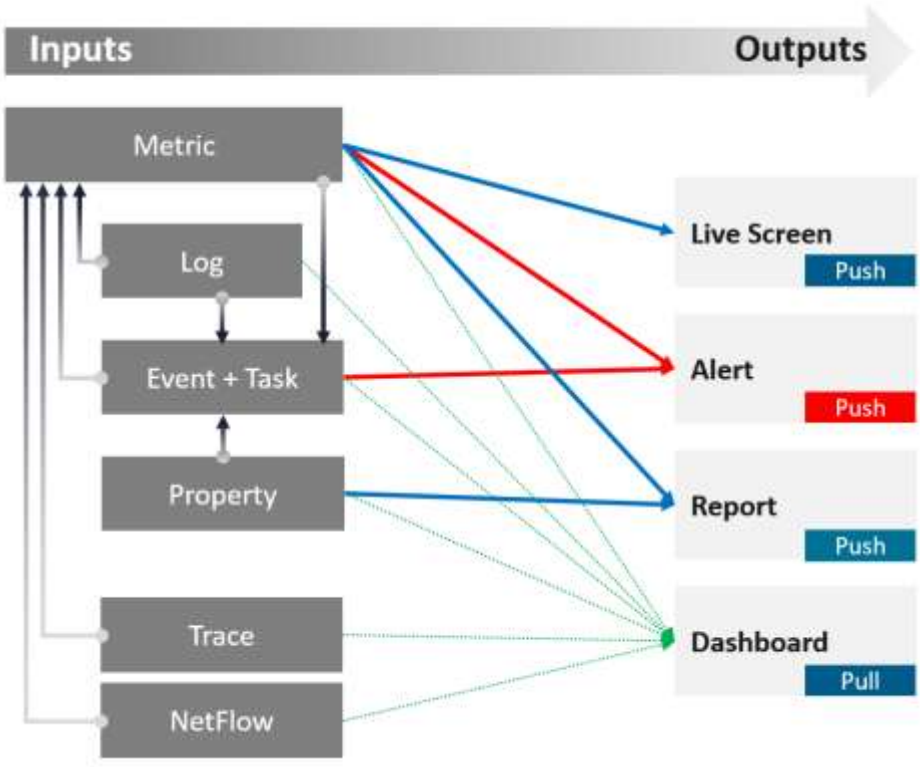
## Input | Output

There are 6[6] types of inputs which work together to give you your alerts, dashboards, and reports. Each of those input types has their own purpose and format. They also tend to overlap. So typically, different observability tools excel on each.

Output is documented in PART 2 of the book.

An alert may feel like an input, as you start from it. It is not an input as it's a trigger you create based on the values of the data types. For example, you create an alert when certain log events occur, certain property changes and certain metrics threshold being crossed.

Review this diagram. What do you notice?




| | |
|---|---|
| **Metric** | The definition is it is produced at regular intervals, even if the value is constant. That's why state is a metric, not a property.<br>Unlike property, a metric is not editable by user. Its valued is not entered by users. It's generated by system.<br>A metric always originates as a number. It may be translated into a string for ease of understanding. |

[6] I don't use MELT as Property is more fundamental than Trace. It also misses NetFlow.

| | |
|---|---|
| | The number could be a code, whose description is a string. For example, "-1" means no limit. By the way, I prefer infinity to indicate no limit as that seems more natural.<br>It can be a raw metric or computed metric. A computed metric is derived from existing metrics and properties. An example of a computed metric is a super metric in VCF Operations.<br>There are 3 types of metrics (contention, consumption and context). They are explained further in the vSphere Metric book.<br>If it crosses a certain threshold, it can trigger an event or generate a log message. From events, we can create a metric, such as a count of vMotion in the cluster.<br>If the number does not match expectation, we can trigger an alert. A daily proactive dashboard showing the trend across hundreds of clusters may give a clue if a problem will happen today. In this case, the proactive work avoids the alert to begin with. |
| **Property** | This is the opposite of metric, as it does not happen in predictable interval.<br>There are 3 types of properties:<br>**Label or Tag** — Label is always external; we attach is to an object. Examples: VM Owner Name, Cluster Class of Service, Cost Center.<br>**Setting** — Also known as configuration. Inherent or original part of the object. It also defines the object. Examples: VM Configured vCPU, ESXi Power Management, DRS Level.<br>**Non-setting** — It just provides context. Does not define or impact the object itself. Examples: date created, parent folder, parent ESXi.<br>Property change is a type of event, which can trigger an alert. Since not all properties are important, the significant of the event is also impacted.<br>For examples:<br>• Number of VMs in a cluster is not a property, even if it does not change for years.<br>• Maximum number of VM in a cluster is a property, as that's part of the cluster capability.<br>• Number of ESXi hosts in a cluster is a property as the cluster is configured with that. Number of running ESXi hosts is however a metric. |
| **Log** | A log is a raw message, typically produced by developers directly.<br><br>Numbers can be extracted to form metrics, while text can describe an event. Metrics and Events can then trigger alerts.<br>>99.9% of the logs are not useful. How do you minimize the cost while maximizing the benefit? |
| **Event** | An event is a record of something that happened. It could be bad, neutral, or good. It could be planned or unplanned. The bad ones may warrant an alert. Unlike a metric, it does not happen on regular interval. |

| | |
|---|---|
| | An event has a start time. It might also have an end time. For example, threshold bridged is an event. If the value drops below the threshold, the event ended. |
| | It can be a setting change, a state change, or a label change. A label is "external", meaning it is not an inherent property of the object. Events also trap the activities performed on those objects.<br>For example:<br>• "Marielle the vSphere Admin enabled DRS on Cluster 007 on 10 June at 10:06 am". Notice it has the actor, the target, the activity, and the date/time.<br>• "Cluster 007 DRS was enabled" is an example of the state change (from disabled to enabled)<br>• "Cluster 007 DRS automation level was changed from 2 to 5" is an example of setting change. The state remains fully automated.<br>• Cluster 007 will be decommissioned and has a tag assigned to mark it as such. This is an example of a label change. There is no setting or state change. |

In addition to the 4 basic inputs above, you need the following in certain situation:

| | |
|---|---|
| **Traces** | A trace is a record of code in-motion. Some can produce numbers, which are metrics.<br>This is needed is low level, function-level troubleshooting.<br>By the way, function tracing can be traced back (pun intended) to Solaris 10 Dtrace, back to 2005! |
| **NetFlow** | A NetFlow captures path taken, typically flows of network packets. It shows networked relationship and can produce metrics. |

## Symptom | Alert | Insight

The reason you have alert storm is you treat alert as To-Do List. You want to be reminded of everything so you do not miss anything.

Analogy: Think of Alerts as your Accident/Emergency Department. You can have dozens of departments dealing with all sorts of illness, but only 1 central location for urgent matters. So where are your "dozens of departments" in your private cloud? Some focus on security, some on storage, some on capacity, some of performance. If you monitor your blood pressure, weight, regularly, why not have daily health check for VCF?

Alert hopefully starts with symptom, a minor and non-urgent issue. This gives you a window to catch during your proactive daily health check.

Mild symptoms that do not go away over time becomes an alert, as it has become urgent.

While symptom and alerts are closely related, insight is something else altogether.

- The former is bottom up, the latter is top down. You typically gain an insight from a collection of alerts and symptoms, plus additional context. Insight uses many more metrics, especially the supporting metrics.

- Insight is much harder to realize as it requires both technology expertise and environment experience. It deals with "hint" instead of issue. You need to know the overall architecture and what's happening operationally, so you can derive an insight from the alerts and symptoms.
- Insights complement alerts, not replace them. Insights do not have the concept of "auto close" as they do not involve help desk tickets

Proactive Alert is an oxymoron.

Proactive means you're acting before something happened which forces you to react. The moment you react, you're reactive. Just because the business is not impacted does not make it proactive.

For example:

- vSAN shows high disk latency on Sunday midnight. You're called to investigate, before business become impacted on Monday morning. You fixed it on the weekend and save the business.
- Does that "weekend warrior" make it a proactive alert?
- What if the same alert happens during business hours and business impacted? Does that make it reactive?

Alerts rely on threshold, be it dynamic or static. A Threshold has an inherent limitation. It misses the big picture, as it can only see what has crossed the threshold.

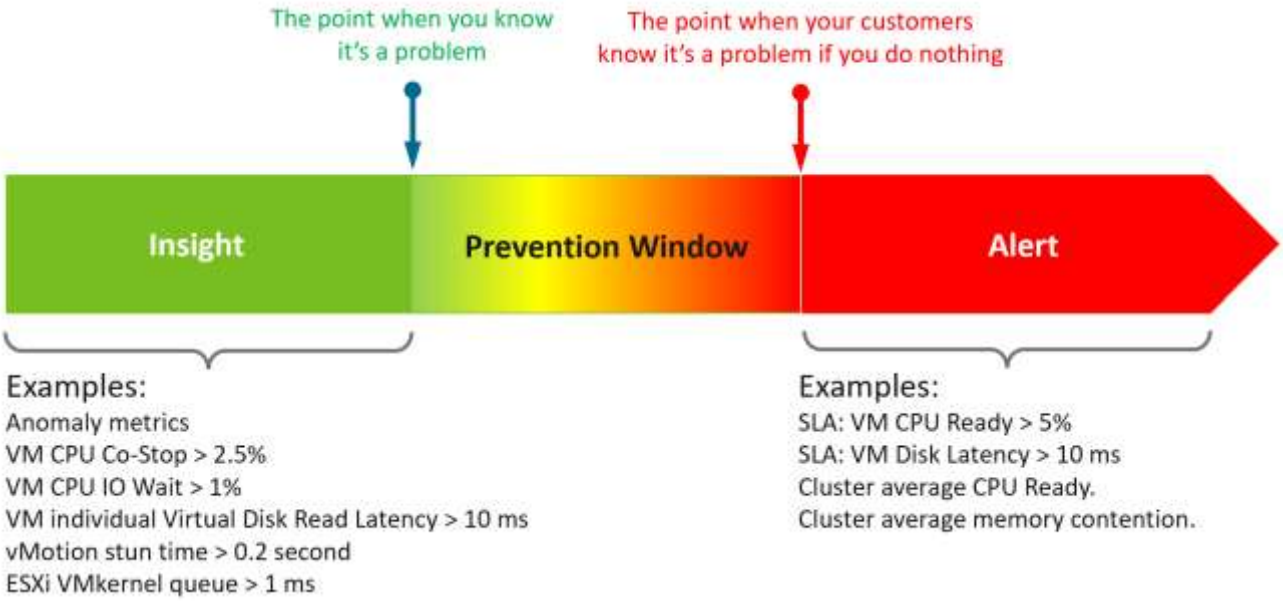
For one object that reached this threshold, there could be many just beneath the level. Think of an iceberg. The small portion above sea level, the tip of the iceberg, is an alert. It does not provide the total picture. In fact, the chunk beneath the surface is far larger.



Insights answer much harder questions, which are typically fuzzy hence they can't be defined as alerts. Examples of questions are:

- Are we being attacked? Are they events and activities that happens in parts of our environment where they are not supposed to happen?
- Is performance degrading? Is there any common pattern and cause?
- Is the environment behaving differently to what we expect?

Insights focus on the underlying problem. They also help buy you time so you can address the problem before the users complain. In the following example, the alerts use the SLA metrics and threshold. Insights require more granular metrics and supporting metrics




What do you think of the preceding example? Would it result in less alerts and less reactive troubleshooting?

The main limitation of proactivity is false positives. It also requires daily operational discipline.

An Insight is useless to those who are not familiar with the environment. The numbers shown by an Insight should match reality, and only a person intimately involved with the actual operations can sense if the numbers are not correct.

Let's take an example: the insight shows your total IOPS throughput is low. If you don't know what to expect on that day, that number is meaningless. But if on that day you expect high throughput because your company is in the retail industry and it's the day before Christmas, then you know the sales are not happening as per expectation. Proactively check before your CEO complains why business is not doing well.

| | Alert | Insight |
|---|---|---|
| **Goal** | To fix. You're ill. | To prevent. You're not sick. |
| **What it is** | A formal event with ticket recorded in the system. May have an incident associated. | Not a formal event. No incident. |
| **Situation** | Business or operations may be impacted. | No impact. |
| | Known problem. You may not know the root cause though. | No known problem. |
| | Urgent. You must look at it today. | Not urgent. Can do on the next business day, or even next week. |
| | Hopefully not important issue | Important issue |
| **Nature** | Reactive and unpredictable. | Proactive and regular. Daily, weekly, monthly. |
| | The system tells you. Response is mandatory. | Response is not applicable as you initiate. |
| **Person** | Low expertise. Follow steps or SOP. | Deep expertise. No steps to follow. |

| | | |
|---|---|---|
| | Does not need to know the overall environment and workload well. | Must know both the environment and recent operations. |
| **Metric** | Focus on primary metrics (the What). | Focus on both primary metrics and secondary metrics (which explains why primary metrics are bad) |
| **User Interface** | Start with Email or notification on your mobile phone. | Start with a big dashboard on desktop. |
| | A specific alert. You work bottom up. | The big picture of the overall environment. You work top down. |

## Lagging Indicator | Leading Indicator

| | Lagging Indicator | Leading Indicator |
|---|---|---|
| **Used in** | Alerts.<br>Reason is alert is your fallback, if you forget to proactively address. | Insights. |
| **Focus** | "Dining-area" metrics. Metrics that impact customers. | "Kitchen" metrics. Underlying metrics that impact the primary metrics. |
| **Technicality** | Simple to understand the meaning (not necessarily the underlying formula). | Tend to be low level metrics that require deep technical knowledge. |
| **Persona** | Level 1 and Help Desk. | Subject Matter Expert. Familiarity with the environment is required. |

## Alert

What, exactly, is alert?

The 3R of alert:

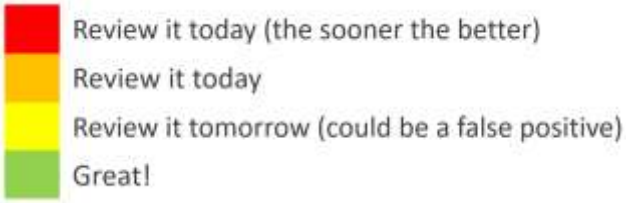
| | |
|---|---|
| **Rapid** | The issue is urgent; hence you want to know the problem as soon as possible.<br>If time is not an essence, then a daily SOP with dashboard is more effective as you can see the big picture. Avoid sending alerts to people that do not deal with day-to-day operations. Long term actions such as capacity management are best served with dashboard.<br>Do not treat alert as To Do List. It is also not a Reminder |
| **Real** | Don't confuse alert with symptom.<br>If there is nothing wrong, there is no need to trigger an alert. That's why in general you do not set up an alert on inventory changes, as inventory is merely an account of something. |
| **Rare** | If it happens too frequent, it will numb you.<br>Alert focuses on exception, not the big picture. As a result, you want this to be minimal. If the whole house is on fire, it's too late for an alert. |

What distinguishes red criticality versus orange criticality versus yellow criticality?

The following diagram provides the answer:




Let's summarize how the colors translate into day-to-day operations:




| | |
|---|---|
| **Yellow** | Symptom. Non urgent. Leading Indicators. Symptoms. Could be important, could also be false positive |
| **Orange** | Alert. Urgent, but importance is not high. Attend by end of day, do not delay until tomorrow. |
| **Red** | Alert. Urgent and important. However, it does not always mean you must drop everything you do and start firefighting in seconds. Attend within 1-2 hours. |
| | There is a small subset of Red where you need to drop everything. Think of it as emergency. It may not be important, but it's definitely urgent. If you get the alert at 0200 hours, you attend to it at 02:00 as each minute matters.<br>An analogy we can relate in life is you accidently cut yourself. Even though it's not life threatening, you want to stop the bleeding immediately, not 1 minute later. The good part is such emergency situation rarely happens. There is no alert defined for it in VCF yet. I'm keen to hear a real-world suggestion! |

# The 2 Sides of VCF




Think of it as "no workload" vs "with workload". Workload means the customers VM, not your infrastructure software.

Infrastructure software is virtual appliance or K8 pods that you need to have as part of your private cloud platform.

VCF belongs to a category of software called SDDC. You use the software on commodity hardware and create a private cloud on your premises. This is not so obvious as it's the only product in this category.

Because of its nature as a data center defined as a software, there are 2 sides of VCF:

- As an IaaS platform of customer.
  This is what the application team care.
- As a software product of Broadcom.
  This is what the infrastructure team care.
  The software takes the shape of one of these 2 forms:
  - Virtual appliance. Examples are vCenter Server and NSX Edge
  - OS kernel software. Examples are ESXi VMkernel, vSAN and NSX kernel modules.

The 2 sides can impact one another, requiring you to see them as one. You need to manage all aspects of operations such as availability, security, and inventory.

- There are 2 layers: consumer and provider.
- The consumer layer is where VM and containers run.
  This layer consists of 3 sublayers (application, Guest OS, and virtual machine).
  VCF scope ends at VM (BIOS & motherboard) and Tools, as the Guest OS is largely outside the influence of VCF the product.
- The provider layer is the virtualized platform (compute, storage, network).

# Service Level Agreement

The difference between an enterprise grade Cloud and non-enterprise grade Cloud is the SLA. A cloud provider can state that they have the best technology, the most experienced professionals, the most innovative process, industry certifications, blah blah blah to prove that they are the best. All that will not carry weight if they are afraid to back it up with the SLA in their contract. The SLA enables customers to hold the cloud provider accountable as it carries a financial penalty.

Once the SLA is defined, then customers want to know how it will be delivered. This is where the process, architecture, certification etc. come in. The **what** always comes before the **how**.

With that, let's define "SLA".

First, it is just a *component* of a business contract. The business contract is a legally binding document which has many other clauses outside the SLA section. The contract first needs to set the context and definition. After that, it has a set of agreements, with SLA being one of them. Examples of other agreements include confidentiality agreements, terms of payment, non-competitive agreements, and marketing agreements.

The SLA section has actual metrics that define the SLA. Google calls this SLO. It works for them as they do not have SLA (to you as their customers). As enterprise IT, you have SLA to your tenants. SLO creates confusion as it sounds like a peer to SLA, when it's just a goal. Private Cloud does not need SLO, as SLA and SLI are sufficient in practice. SLI is not a peer to SLA.




Guess how many SLAs do you need?

It depends on the type of services. Most service providers will only commit to the simplest and most obvious one, which is availability. It's the simplest as it's binary. The darn thing is either up or down. Google only covers availability in their SLA post here, which is based on Google Cloud's SLA. AWS only cover their infrastructure, and not your EC2 VM. I have read this and many other articles. While it makes sense for Google business, it's not suitable for IaaS. Happy to discuss my documented analysis .

Just because something is up, does not mean it is fast. In fact, a service that is slow to the point it's unusable is as good as down.

Just because something is fast, does not mean it's secured. This is why a Security SLA is necessary.

# The 4 SLAs of IaaS

The business of IaaS should provide four SLAs, as customers want complete coverage. These four are focused on Availability, Performance, Compliance, and Service.

The following diagram showing the first three: what they do, and what they measure.




In the Availability SLA, you measure downtime. In the Performance SLA, you measure "slow time". In the Compliance SLA, you measure unsecured time. Regardless, you measure something and express it in 0 – 100%, with 100% being perfect relative to the contract.

Wait, why am I not showing the 4th one?

Because it plays a secondary role. The first 3 covers the actual workload, while the 4th one covers the human (typically tenant or application team).

| | |
|---|---|
| **Availability** | This is the most basic SLA. It is the oldest and most well-known.<br>In reality, it is largely a given. It does not matter what the agreed number is. If the darn thing is down, you better hurry to bring it up before there is a complaint or things get worse! |
| **Performance** | The Performance SLA is far more valuable than Availability SLA. It is the solution to complaint-based operations by defining what exactly is "fast".<br>In IaaS, it covers CPU, Memory, Disk and Network, hence there are four metrics used. |
| **Compliance** | Also called Security SLA as the goal is secured environment.<br>This is hardly talked about, as customers and providers expect this to be 100%. This is why you need to provide an SLA, as promising 100% will lead to disappointment.<br>It measures compliance with industry security regulation or certification. |
| **Service** | Service provided by both human and system (typically in the form of self-service portal) |

## VM vs Guest OS

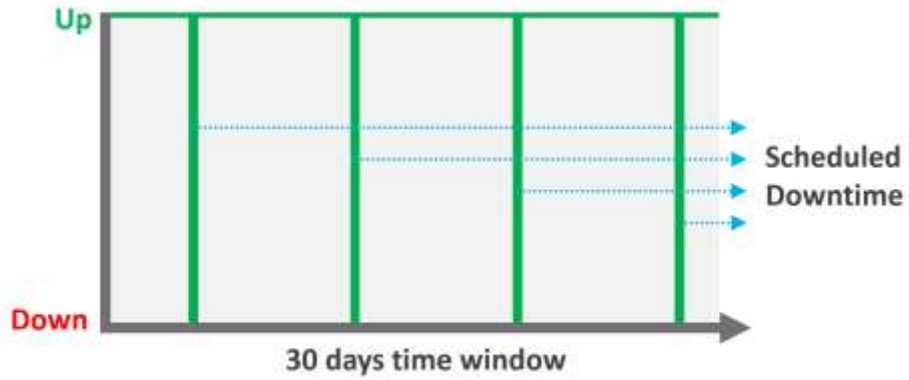
VM and Guest OS are 2 separate objects but they are 1 logical entity due to 1:1 permanent relationship. It is common for IaaS provider to cover both.

If your responsibility includes the Guest OS, then your SLA needs to include Windows or Linux.

| Type of SLA | Virtual Machine | Guest OS |
|---|---|---|
| Availability | VM is powered on. BIOS is up and running. A VM is basically a virtual motherboard.<br>Windows and Linux are not part of SLA. | Windows or Linux is up and running. This may include basic services such as security agents.<br>Application is not part of SLA |
| Performance | VM is getting the CPU, memory, disk, and network resources it demands | Windows or Linux performance counters are within expectation. |
| Security | VM is protected as per vSphere hardening guide or industry regulation. | Windows or Linux is protected as per respective vendors or industry regulation. |

In this book, I assume your IaaS offering includes Guest OS. However, the metrics for Windows and Linux are not yet complete due to the lack of maturity of their performance modelling.

## Availability SLA




Many vendors claim a very high availability SLA. This is misleading as they do not include scheduled downtime. This unique saving grace lets you state you're 99.999% available even though the actual reality, experienced by end customer, is lower. This is why you need 2 metrics:

- One for availability as defined in the SLA.
- One for actual availability. It reflects reality.

The 2 metrics names are:

- Actual Availability (%).
  This is much easier to measure as it does not consider context. Down is down, regardless of when, who and why.
- Operational Availability (%)
  This is harder to measure.

Example:

- All VMs has weekly scheduled downtime to apply urgent security patch. It's every Sunday 0000 – 0200 hours.

- Last month, the database server was brought down for patching 3 times. But all happened within the scheduled downtime. While Windows was only shutdown for 15 minutes, the large database took 45 minutes to fully restore.
- In this case, the Operational Availability (%) for the month of June is 100%. It meets SLA.
- The Actual Availability (%) accounts for the 3 hours of total downtime.
  It's 3x of (15 minutes + 45 minutes).
  In the month of June, there are 24 hours x 30 days = 720 hours.
  Actual Availability (%) = 717 / 720 = 99.58%

## Formula

| | |
|---|---|
| Definition | Defined as Guest OS is pingable, because running but isolated fails the availability test.<br>The Ping Source is predefined and set by the IaaS provider, not the customer. It pings the VM, not a specific process (e.g. web server. This is IaaS, not web server as a service.). |
| | The uptime only covers the Guest OS. If it takes the application 15 minutes to become fully operational as it has to load files and other services, that's not counted. |
| Inclusion | If the crash is caused by VMware Tools or IT Infrastructure owned drivers/agents, then it's counted. |
| Exclusion | Unscheduled downtime caused by customers. If the crash is caused by bad applications behaviors, the SLA is not affected as that's not within the control of IaaS provider.<br>As it takes time to figure out what caused the downtime, you need to be able to recalculate the metric. |
| | VM owner-initiated reboots as they might reboot their OS to solve problems or after installing software. How to track as developer may not inform the IaaS team, as Windows does not fully trap this event? |
| | Scheduled downtime. Guest OS upgrades, patches that requires reboot, Tools upgrade, VM Hardware version are not counted if you execute within the agreed scheduled downtime. |
| Complication | A challenge that impacts availability but not performance is recovery time.<br>Windows or Linux maybe up in 1 minute, but it needs to perform fsck (filesystem consistency check) before application can launch. This is considered as part of downtime. |

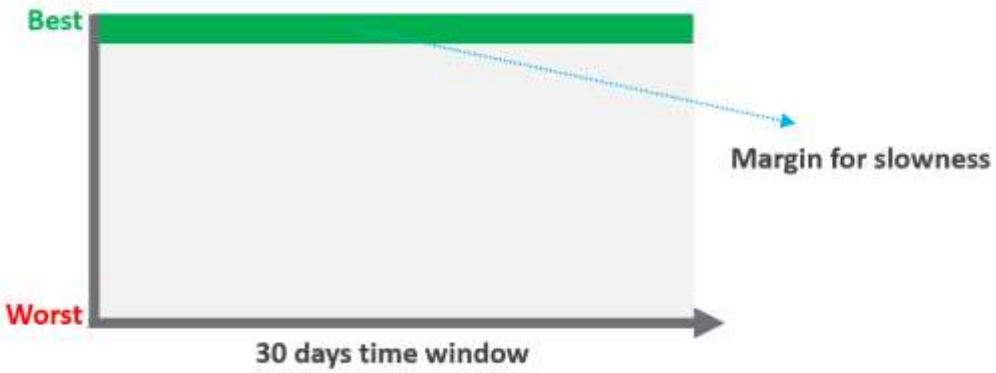
## Supporting Techniques

You need to back up your promises with solutions that are convincing for customers. Here are some solutions that you may offer to justify and support the higher availability SLA.

| | |
|---|---|
| Backup | Gold Tier provides application-level back up. It also provides more frequent full back up, and customers are provided with self-service individual file restore. |

| | |
|---|---|
| High Availability | Gold Tier provides application-level monitoring. Customers can also ask for specific boot-up sequence of their VMs, and ask for VM-Host affinity rules to minimize risk. |
| Disaster Recovery | Gold Tier provides lower RPO and RTO. Customers are also entitled to annual real-world tests, where the production workload is run from the DR site. |
| Snapshot | Gold Tier provides longer snapshots and larger snapshots. |
| OS Management | Gold Tier provides flexibility in patching. Customers can specify delay in patching and request custom patch packages, where not all patches from Microsoft or Red Hat is applied. |
| VM Management | Gold Tier provides flexibility in updating Tools and VM Hardware. Customers are allowed to defer the updates. |

## Performance SLA




Unlike availability, which has the luxury of scheduled downtime, there is no such thing as scheduled "slow time" with performance. You can't say that you're doing infrastructure upgrades and use that as excuse for why VM performance will be slow. As a result, you need to put your margin or buffer somewhere else.

Slow is not binary. It's a spectrum from 0 contention to absolute worst (as good as dead). Some metrics such as disk latency can never be 0. It will be a small number, but it's not 0. The same goes with the value of the CPU Ready metric. So, we need to define a threshold above 0.

| | |
|---|---|
| Definition | All are measured at VM level, not individual vCPU or vDisk. For disk, it's also the average of Read & Write.<br>All are averaged over 300 seconds, which is an appropriate balance for SLA monitoring. An SLA that is based on a 1-minute average will be too tight and result in either a cost increase or a reduction in threshold. |
| Inclusion | |

| IaaS | VM Counter | Threshold |
|---|---|---|
| CPU | Ready | 2.5% |
| RAM | Contention | 1% |
| Disk | Latency | 10 ms |
| Network | TX Dropped Packet | 0% |

| | |
|---|---|
| Exclusion | Received Packet Dropped. It's not reliable enough to be used in the SLA contract due to false positives. A packet could be dropped as it's not for the VM. More details in the Network metric chapter. |
| | Other forms of contention, such as CPU Overlap, CPU Other Wait, and vMotion. They are too granular for the purpose of a contract. You track them via KPIs instead. |

Why should you only use CPU Ready and exclude CPU Co-Stop and CPU Contention from the Performance SLA?

It took me years to vrealize the mistake.

You should exclude CPU Contention because its value can go as high as 37.5% without the application noticing any degradation. You can login to Windows or Linux and feel that it's responsive.

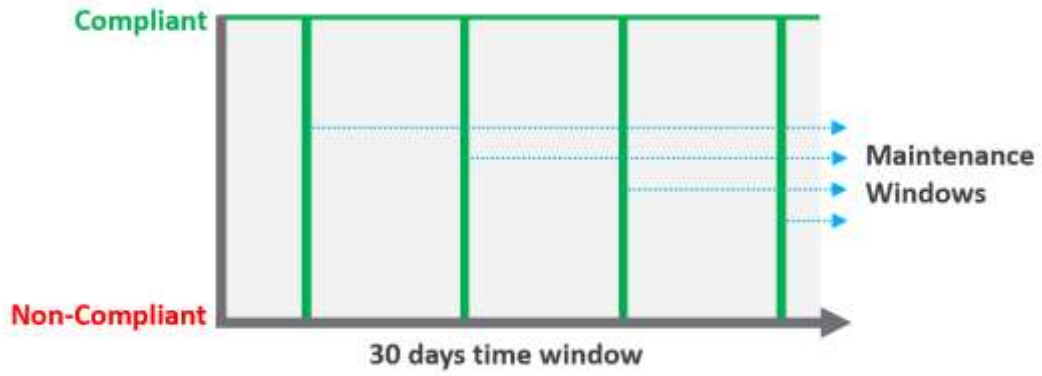
Use the above threshold as they are. There are two main reasons:

- Major changes in the value, such as changing CPU Ready from 2.5% to 5%, will require you to adjust your "nines" to a higher number. This requires you to profile your environment first.
- A common value in the industry will also enable you to compare with your peers and get an industry-acceptable numbers. You can then compare how well you serve your mission critical VMs, your Test/Development VMs, etc.

Just like in Availability, there are extra things you can do to give confidence to your customers. For example:

- Gold Tier provides priority on the network. Customers can opt for a periodic ping service to ensure network latency between their applications remain within the agreed threshold.

## Compliance SLA




| | |
|---|---|
| Definition | Percentage of compliance against an agreed security policy or benchmark. A compliance SLA differs from an Availability or Performance SLA in one key area: the compliance SLA should promise perfect compliance. Compliance is binary: you are either compliant or you aren't. You shouldn't be telling your customers that you will have less than perfect compliance with your own security requirements.<br>It has a window, typically ad-hoc, to enable investigation or maintenance which may result in temporarily becoming non-compliant against your compliance benchmark.<br>Compliance calculation is purely from Infrastructure Team point of view. |

| | |
|---|---|
| Inclusion | Internal security standards, typically an adaptation of VMware best practices or the VMware Security Hardening Guides.<br>Regulatory benchmarks, e.g. CIS, ISO, DISA, FISMA, PCI DSS, HIPAA.<br>For the vSphere infrastructure, compliance with these benchmarks (and custom ones) can be directly managed through the compliance features of VCF Operations. |
| | A VM compliance must consider its immediate surrounding. If the parent ESXi, the vSAN storage, and the distributed network and storage is not secured, the score need to reflect it. |
| Exclusion | This depends on the definition of your IaaS Service. You are only providing SLAs for what you control. |
| | If you provide the guest VM OS as part of your IaaS service, then you will need to maintain compliance by managing configuration (using something like Group Policy and/or a configuration management tool) and you will also most likely have some security tools and agents that run inside each VM. When the guest OS is part of the service, you should be measuring availability by whether the guest OS is running (e.g. via ICMP ping).<br>If you do not provide the guest OS as part of the IaaS service, then you do not need to worry about maintaining compliance in the guest OS, and you are also measuring availability by whether the VM is powered on. |

## Service SLA

IaaS is built on commodity hardware and provided as a utility. Having said that, there are many ways to differentiate your service vs your competitors. Use class of service to distinguish premium service classes. The following table lists some examples.

| | |
|---|---|
| **Provisioning Time** | In environment where the churn is high, the time taken to provision become important. You need to clearly define what "provisioned" means, as it can range from bare Windows or Linux to completely set up and configured with applications & database loaded. |
| **Provisioning Success** | Provisioning a complex set of multi-tier business applications with many VMs and many external integrations or endpoints may fail from time to time. If this is relevant to your environment, then add it as part of the SLA so you can focus on the higher class of service. |
| **Support** | Gold Tier customers will be alerted over email and messaging network within 10 minutes. |
| | The two popular examples are response time and path to escalation. Do not promise resolution time unless it's completely within your control.<br>Gold Tier provides faster response time and longer coverage hours (e.g. 24 x 7 x 365). Your ticket is also directly answered by Level 3, bypassing the front liners. |
| | Gold Tier comes with regular business reviews, attended by your management. |
| **TAM** | Gold Tier comes with a Technical Account Manager, acting as single point of contact for customers. The TAM is also the internal champion, representing customer interests within the vendor internal world. |
| **Monitoring** | Gold Tier VMs will be proactively monitored, not just relying on alerts. |

| | |
|---|---|
| | Gold Tier provides deeper visibility into the underlying physical infrastructure where customers VM are running. Customers are entitled to see lower-level internal metrics such as vMotion stun time and VMkernel latency. |
| | Gold tier provides self-service monitoring. Customers are given their own login to a portal where they can monitor their own VMs. They can initiate scheduled downtime |

There are other metrics you can add to differentiate one class from another. However, be careful of adding metrics that do not actually serve your business. For example, it can be tempting to put the accessible time of your self-service portal, to protect you from scheduled downtime. You need to work on the basis that your "office" is open 24 x 7 x 365 days.

# The Metrics of SLAs

Do you set it per week, per month or per year? Let's find out!

## The Time Window

SLA is a *monthly* metric, not daily or yearly. You use an entire month of data to calculate it, averaging 8640 datapoints of 5-minute averages.

In the following table, notice 99.999% in a year is more time than 99.95% in a week. Your customers would not accept a yearly metric as they can be exposed to a long downtime. You would not accept a daily metric as there is no room for error. The monthly metric provides a balance between service quality and cost to deliver the service. It also makes reporting easier as you simply follow the calendar month.

| SLA | Total Allowable Downtime | | | |
|---|---|---|---|---|
| | In 1 week | In 1 month | In 1 quarter | In 1 year |
| 99.999% | 0.1 minute | 0.4 minute | 1.3 minute | 5.3 minute |
| 99.99% | 1.0 minute | 4.3 minute | 13.1 minute | 52.6 minute |
| 99.95% | 5.0 minute | **21.6 minute** | 65.5 minute | 263.0 minute |
| 99.9% | 10.1 minute | 43.2 minute | 131.0 minute | 526.0 minute |

Each additional "9" shrinks your SLA window by 10x. That's why each decimal can cost a lot more money, as a different architecture may be required.

Even if you measure the SLA once a month, it can still be very difficult to meet. Take a look at the following table:

| SLA | Relative Window | Total Measurement Period in minutes | | | Allowable SLA Breached | | |
|---|---|---|---|---|---|---|---|
| | | 28-day month | 30-day month | 31-day month | 28-day month | 30-day month | 31-day month |
| 99.999% | 1 x | 40320 min | 43200 min | 44640 min | 0.4 min | 0.4 min | 0.4 min |
| 99.995% | 5 x | | | | 2.0 min | 2.2 min | 2.2 min |
| 99.990% | 10 x | | | | 4.0 min | 4.3 min | 4.5 min |
| 99.950% | 50 x | | | | 20.2 min | 21.6 min | 22.3 min |
| 99.900% | 100 x | | | | 40.3 min | 43.2 min | 44.6 min |

If you promise 99.99%, you only have 4 – 4.5 minutes of downtime per calendar month. That means your architecture must be able to detect the issue and then complete remediation in just a few minutes. That's a tight space to manoeuvre.

Let's analyze the size of the failure window we have per month. The table below gives you a better gauge into what SLA you want to set for each class of service.

| 30-day SLA | Failure Window | Failure Chance per SLA |
|---|---|---|
| 99.99% | 4.3 minutes | < 1 time |
| 99.98% | 8.6 minutes | < 2 times |
| 99.97% | 13.0 minutes | < 3 times |
| 99.96% | 17.3 minutes | < 4 times |
| 99.95% | 21.6 minutes | < 5 times |
| 99.90% | 43.2 minutes | < 9 times |
| 99.80% | 86.4 minutes | < 18 times |

## The 2 Sides of an SLA

This is one of those things in life where it's so obvious that we overlook it.

There are 2 sides of an SLA.

| | |
|---|---|
| **Promise** | What you promise to your customers.<br>Obviously, the higher the price, the better the service, the higher the SLA. So, there can be multiple numbers, matching the number of class of services. |
| **Reality** | What is delivered. Calculated at the end of the month.<br>There is only one number, regardless of the class of service. A Gold VM can fail its SLA even though it's getting a higher number than a Bronze VM. |

## SLA Calculation

Since an SLA is a monthly counter, it needs to be derived from thousands of 5-minute counters.




The four elements of IaaS (CPU, Memory, Disk, Network) are evaluated every 5 minutes. As this results in a metric, we need to give it a name. I call this SLA Leading Indicator, as it's telling you in advance if you're going to fail the SLA or not, which is at the end of the calendar month.

We need one metric for each service. In IaaS, the formula is:

```
If VM CPU Ready          >  2.5% then 100% else 0%
If VM Memory Contention  >  1.0% then 100% else 0%
If VM Disk Latency       > 10 ms then 100% else 0%
If VM Network TX Dropped >   0 % then 100% else 0%
```

VM SLA Leading Indicator (%) = Average of Above 4 metrics.

So, every 5 minutes, a VM gets a score of 100% or 75% or 50% or 25% or 0%.

The VM Performance SLA (%) value is simply the average of the 5-minute datapoints over the last calendar month.

Whether that's good or bad, it depends on what is being promised. The higher the class of service, the higher the price, and hence the higher the SLA.

## Performance Quantification

| | |
|---|---|
| **CPU** | CPU Ready time of 2.5% in a 5-minute collection period translates into 7.5 seconds of ready state. This 7.5 second freeze does not have to be a contiguous block. Likely it is sub-seconds, spread well over 300,000 milliseconds. |
| | The number is not measured against CPU Utilization. 2 VMs can have identical Ready time while having very different utilization.<br>VM 01: CPU Ready 10%. CPU Run 90%<br>VM 02: CPU Ready 10%. CPU Run 10%<br>To VM 1, the situation is not that bad as it still got to run most of the time.<br>To VM 2, the performance is bad as it cannot run half the time. |
| **Memory** | Memory Contention is relative to the amount of memory being used. Unlike CPU, it is not measured across time. Reason is memory does not "run". |
| **Disk Latency** | This is the average latency across 300 seconds. As disk IO is measured per second, a VM doing 1000 IOPS is doing a total of 300,000 IO commands over the entire 300 seconds. |
| | It's also an average of reads and writes. As each virtual disk can have its own latency, this number is normalized at VM level. |

# Class of Service

Now that you have the 4 SLAs, you compare them with the associated Classes of Service. The reason to offer multiple Classes of Service is that if you only have one Class of Service offering and you promise good service, everyone will expect the same first-class experience.

Kim Ramirez advises that from a pricing psychology standpoint it might make sense to offer Gold, with the expectation that nobody will buy it, and it only serves to make Silver look like a good deal. In life, one way we know something is good or bad is via comparison. Relative value can complement absolute value in educating customers.

Having a comparison also addresses potential confusion where customers wonder where Gold is, if they only see Silver and Bronze offers.

If you do not wish to make a certain class available, provide the reason to your customers and/or management.

## Price/Performance

The price-performance ratio is widely accepted as it is simple to understand and it's built on fairness. You've probably heard of this: "I offer 3 variants of service: Cheap, good, and fast. Pick any two. You want it cheap and good; it won't be fast. You want it cheap and fast; it won't be good. You want it good and fast; it won't be cheap!".

In IaaS, how do you apply the above principle?

- For Availability, this is measurable. If you reduce the downtime window by 2x, logically you should pay 2x.
- For Performance, how do you quantify this since it depends on utilization? Since utilization does not exist yet, you use overcommit ratios. If there are 2x vCPUs in the cluster, then each of them pays half price. This is fair as the cost must be distributed to all.
- For Compliance, this is a little different. Unlike with availability and performance, it is in your interest as a service provider to provide a perfect and consistent level of compliance across all classes of service.

The class of service impacts many parts of operations, so it needs to be central to your plan. The following diagram shows how the quality of the service and overcommit ratio serve as input to operations management.




The following table shows a basic and generic guideline to a class of service. The actual model that you will implement will differ, taking into account actual hardware model and business demand.

| Tier | Price | Overcommit | "Performance" | Positioning |
|---|---|---|---|---|
| **1 (Gold)** | 1.00 X | 1:1 | Perfect | Performance "Guarantee". Suitable for latency-sensitive mission critical applications. |
| **2 (Silver)** | 0.50 X | 1:2 | Great | 50% discount for a mere 5% penalty. Great value compared with Gold Tier.<br>Suitable for most Production workload. |
| **3 (Bronze)** | 0.25 X | 1:4 | Good | 75% discount for a mere 10% penalty.<br>Notice the price is half and the performance drop is doubled. This makes it fair and consistent.<br>Suitable for Test & Dev workload. |
| **4 (Free)** | 0.00 X | Max | Average | Suitable for temporary projects.<br>No Availability SLA, no Performance SLA |

I put "guarantee" in quote because for CPU this is not possible as the VM CPU Ready counter does not register 0.00% when there is no contention.

## Performance

The word is shown in quote as it's a broad definition. It includes all the types of SLA.

The performance column in the table above is backed by clearly defined SLA, because you need to quantify what the penalty exactly translates into.

- Let's take Bronze as example. I put 10% penalty to position the business value. In reality, the performance metric is not simply ≥ 90% for Bronze.
- Using CPU as example, it is not that the CPU Ready will be ≤ 10% at all times.
- 10% is too high a number. A VM experiencing 9.9% CPU Ready constantly for entire month will pass the SLA. This is obviously unacceptable. A fairer number is 2.5% since it is an average of 5-minute.

- "At all times" translates into 100% SLA. It's cost prohibitive. On the other hand, average is far too loose. SLA is expressed in "nines", such as 99.93 or 3 nines. It is never expressed in lower number such as 95% as that translates into 1 in 20 failure rates.
- 2.5% will serve all classes. You just need to adjust the "nines".

### Free Tier

The Free Tier is useful to convey the value of the SLA. Human nature tends to appreciate something when after it's taken away from us.

Business wise, the free tier must be funded by paid tiers. Since it is free, you are excused from providing SLA. It's acceptable for them to have unpredictable downtime and slow time. Commercial cloud providers provide free tier that are intentionally designed to be slower and less reliable, because they want you to upgrade and pay.

## The Benchmark to Rule Them All

Gold class has higher SLA than Silver class. For that to happen, that means they are measured against the same threshold or benchmark.

- For availability, you measure all classes against the ideal, which is no downtime.
- For performance, you measure them against the same threshold, which is the "slow time".
- For compliance, you measure them against the ideal, which is perfect compliance.
- For service, you measure them against the ideal, which is the best possible service.

A VM in silver environment will expect that it does not get what it demands as often as a VM in Gold. If the VM Owner wants to have more *consistent* service in performance, then simply pay more and upgrade to the gold cluster.

This approach is easier than setting up a different performance threshold for each tier. Say you set the following:

- **Gold**: VM Memory Contention: 0.5%
- **Silver**: VM Memory Contention: 1.5%

You notice the problem already?

It is hard to explain the delta or gaps between the class of services. Why is Silver 3x the value if it is only half the price? Shouldn't it be proportionate?

There is a 2nd problem. If you set *different* standards, it is possible that Silver will perform better than Gold, because it has lower standard. This can create confusion.

This means the performance in production is expected to have a higher score than the development environment. Development environments will obviously perform worse than Production environments.

How much worse, exactly?

Let's find out by applying all the above into actual numbers. You have 9 numbers, 3 SLA x 3 Class of Service.

## Recommended SLA

Let's put all the above SLA in an example. The following is what I'd recommend:

| Tier | Availability SLA | Performance SLA | Compliance SLA |
|---|---|---|---|
| **1 (Gold)** | 99.975% | 99.9% | 99.95% |
| **2 (Silver)** | 99.950% | 99.8% | |
| **3 (Bronze)** | 99.900% | 99.6% | |

Why they are all different numbers? Isn't easier to have 1 number for each class?

Well, they measure different things:

| | |
|---|---|
| **Availability** | It can afford to have the highest SLA because scheduled downtime and downtime caused by customer is not included.<br>I put 99.975% as Windows or Linux may need to run filesystem integrity check.<br>If you have many IT services or Security services that must be started before application services are started, adjust the SLA accordingly.<br>Notice the downtime windows is 1x, 2x, and 4x. It helps in justifying the price when the gaps are clear and consistent. |
| **Performance** | Lower SLA due to stringent standard. It's stringent so it can cover the mission critical environment. As a result, the Bronze environment will have a harder time meeting it. If you think that's too strict, reduce the SLA not the threshold. To see how bad the reality, use KPI.<br>Notice the number for performance is 1x, 2x and 4x. Silver failure window is 2x bigger because its price is 2x cheaper. Remember the Price/Performance principle? The basic concept is you pay double you get something 2x better.<br>Why not 100% for Gold since there is no overcommit?<br>Because that's just the compute portion. You do not control the network and storage portion. |
| **Compliance** | It has identical SLA for all classes as it's in your interest to secure everyone. You don't want to have a security loophole, which can be used as jump box to attack the rest. I did not put 99.99% as that's not even 1 chance of mistake. As the value for compliance is the last value of collection, that could be as short as 1 second.<br>Consider how fast it takes for you to effect a change. If you rely on manual change, then adjust the number accordingly. |

### Penalty Quantification

Let's drill down to see what the SLA numbers

In the following sample of 1 hour window, there are 12 measurements of the CPU Performance SLA for VM 007.

| SLA Window | CPU Ready | In 5 minute | Status |
|---|---|---|---|
| 00:00 - 00:05 | 2.40% | 7.20 sec | SLA pass |
| 00:05 - 00:10 | 0.10% | 0.30 sec | SLA pass |
| 00:10 - 00:15 | 48.25% | 144.75 sec | SLA fail |
| 00:15 - 00:20 | 1.30% | 3.90 sec | SLA pass |
| 00:20 - 00:25 | 2.49% | 7.47 sec | SLA pass |
| 00:25 - 00:30 | 2.51% | 7.53 sec | SLA fail |
| 00:30 - 00:35 | 0.10% | 0.30 sec | SLA pass |
| 00:35 - 00:40 | 0.90% | 2.70 sec | SLA pass |
| 00:40 - 00:45 | 1.24% | 3.72 sec | SLA pass |
| 00:45 - 00:50 | 1.92% | 5.76 sec | SLA pass |
| 00:50 - 00:55 | 2.19% | 6.57 sec | SLA pass |
| 00:55 - 01:00 | 0.06% | 0.18 sec | SLA pass |

The VM 007 suffers 2 SLA failures.

- The 1st failure is bad. The number is very high. Since SLA is binary, this is simply counted as failure. There is no severe failure in the book of SLA.
- The 2nd failure is barely a failure. Again, since SLA is binary, this is also counted as a complete failure. There is no partial failure.

So why did I recommend 99.9% and not 99.95% for Gold?

- The 0.05% matters operationally.
- 99.9% means the VM may not get the promised SLA for up to 0.1% of a given month. Using a 28-day month as it's the shortest, this translates into 40.3 minutes.
- Since SLA is computed every 5 minutes, the provider only has up to 8+ chances of SLA failure in the month of February.
- This means CPU Ready > 2.5% for 8 times in the February months.

## Sample Offering

Putting the above together, what does a sample class of service actually look like?

### Price & SLA

SLA is the hard product, the main thing that you get when you pay for the service. Using the airline business as an analogy, the hard product is the seat. Other things such as in-flight entertainment and meals are soft products.

In IaaS, since you do not typically care about the hardware specification (e.g. the type of SSD for storage, the memory brand and technology), you focus on the SLA.

| | Gold | Silver | Description |
|---|---|---|---|
| Price | 2x | 1x | Gold is priced 2x Silver. |
| Over-commit | CPU 2:1<br>RAM 1:1 | CPU 4:1<br>RAM 2:1 | The reason why Silver is half price is because there are twice as many consumers. |
| Availability SLA | 99.975% | 99.950% | Gold has 2x less downtime |

| | | | |
|---|---|---|---|
| Performance SLA | 99.9% | 99.80% | Gold has 2x less "slow time" |
| Compliance SLA | 100% | 100% | You should maintain perfect compliance with your compliance requirements for all classes of service. |
| Service SLA | Gold | Silver | See the Service section for details. |

If you mix environments, it becomes operationally challenging. Your only tool is to apply reservations and shares. For example, give Gold 2x the value over Silver to justify the 2x price.

The preceding table is a generic guideline. As part of your planning with IT Management, you help them define and decide on each Class of Service. This planning session requires vendors input as you want to optimize cost. Use vendors discounting and licensing model to complement the plan, not dictate the plan.

Yes, it's a balancing between ideal solution and what actually gives the best bang for the buck.

At the end of the planning session, you may end up with something like this.

| Tier | Price | Compute | Storage |
|---|---|---|---|
| Gold | Highest | CPU 1:1<br>RAM 1:1 | All Flash |
| Silver | Good<br>(67% of Gold) | CPU 1:2<br>RAM 1:1.5 | All Flash |
| Bronze | Low<br>(33% of Gold) | CPU 1:6<br>RAM 1:2 | All Flash |
| Free | Free | N/A | Magnetic |

For Silver and Bronze, to maximize consumption of hardware, you base on average ratio of 1 vCPU to 4 GB RAM. This is a common ratio. If you want to tailor this to your own environment, use VCF Operations to give you the actual ratio.

For Silver, you decide on 3:1 CPU overcommit and no RAM overcommit. Based on 1 vCPU 4 GB RAM, you need to buy 12 GB RAM for each physical core. Buying 64 cores maximizes your vSphere license. That means you need 64 x 12 = 768 GB of RAM. If you run vSAN and NSX, you need to account for the overhead accordingly.

The 3:1 overcommit enables you to lower the price by 3x for CPU portion. If you do 50/50 split between CPU and RAM, the overall price is 67% of Gold, yielding a 33% discount.

For 33% discount, what's a reasonable performance penalty? What's acceptable to both your CIO and application team?

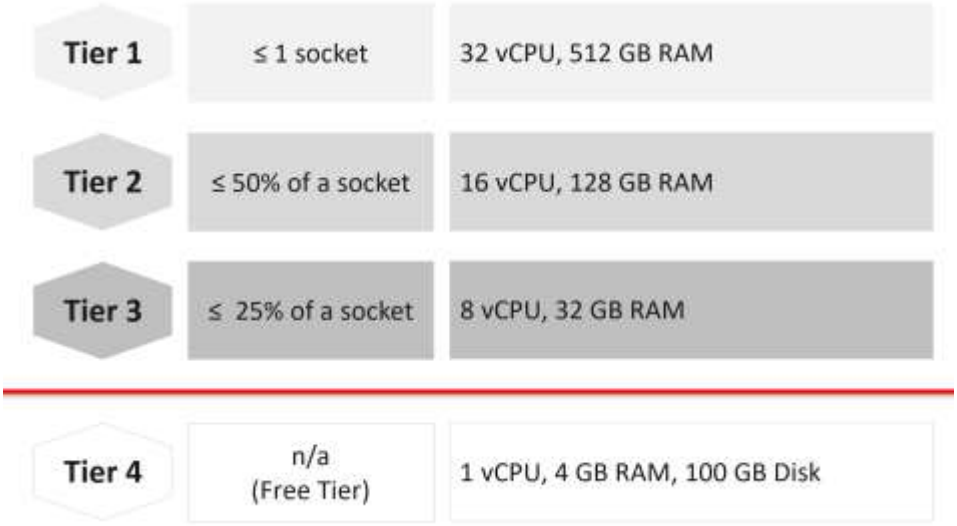
CIO also decided that free tier is not offered as that's not the business you want to be in.

As you can see, the hardware spec is driven by the business model. Decide on the overcommit & price first.

If you want your customers to right size in advance, then a 64 vCPU VM needs to be *more* than 64x the price of 1 vCPU VM. If the pricing model is a simple straight line, there is no incentive to go small and no penalty to over provisioning. You will end up forcing rightsizing in production, which is a costly and time-consuming process.

## Size Limit

Because you overcommit, you run the risk of contention. One way to minimise this risk is to control the maximum size of VMs. You want to avoid monster VMs dominating your overcommitted ESXi hosts. The following table provides an example of the size limit you associate with each class of service.




Capping the size at each tier is a good way to prevent monster VMs from causing performance problems in environment with higher overcommit ratio.

If you allow a single VM to be the same size as the ESXi host, you practically can't do overcommit if you want good performance.

The Free Tier may also be limited further by capping the number of free VMs per customer, else you'd go bankrupt.

For comparison, AWS's free tier for EC2 is only 1-2 vCPU & 1 GB RAM as it's based on t2.micro and t3.micro.

## Service

Beside the hard product, soft products play an important role in further differentiating Gold from Silver. Support, both reactive and proactive, is one major area where different classes of service can show the value of higher tiers.

| | Gold | Silver | Description |
|---|---|---|---|
| Response Time | 2 hours | 4 hours | Gold has 2x faster response time |
| Support Hours | 24 x 7 | 12 x 7 | Gold has 2x support hours |
| Support Process | Level 2 | Level 1 | Gold bypasses the first level help desk. |
| Root Cause Analysis | Provided | Additional Charge | The RCA includes setting alerts to ensure the same incident is identified immediately. |
| Proactive Support | Daily | None | Gold VMs gets daily health check (KPI and alerts). |
| NOC Screen | Yes | No | Silver VMs are not displayed in a live screen of Network Operations Center room. Reason is the business need to focus on the Gold VMs. |

| | | | |
|---|---|---|---|
| Report | Weekly | Monthly | Gold has more frequent report. |
| Business Review | Monthly | None | Face to Face discussion where we review the SLA and support issues. |
| Critical Patch | Higher priority | Lower priority | We patch all Gold VMs first before protecting the lower tier VM. |
| Optimization | Yes | No | Performance and capacity optimization (via proactive right-sizing, for example). |

## Availability

This is another soft product in IaaS. Ensure that Gold has much better offering here so the 2x price is clearly justified.

| | Gold | Silver | Description |
|---|---|---|---|
| Snapshot | 1 week | 1 day | For Gold, we will also remind you just in case you forget. For Silver, it's auto-delete. |
| Size of snapshot | No limit | 100 GB | |
| No of snapshots | 2 | 1 | Gold can have 2x as many snapshots at any given time. |
| Back up | Higher priority | Lower priority | We back up all Gold VMs before backing up Silver VMs. |
| Full back up | Weekly | Fortnightly | Gold has 2x full back up frequency. |
| Back up level | Application | OS | We will install & manage back up agent. |
| Back up report | Yes | No | We provide report for back up status. |
| Disaster Recovery | Separate add-on service | | Both services do not have DR as default. |

# Compute & Storage

A VM runs in a cluster but is stored in a datastore. How do you architect both the compute and storage subsystems to support the class of service?




From a performance management point of view, vSphere clusters with vSAN form the smallest logical building block of the resources. While resource pools and VM-Host affinity rules can provide smaller subdivisions, they are operationally complex, and they cannot deliver the promised quality of IaaS service. Resource pools cannot provide a differentiated class of service. For example: your SLA states that Gold is two times faster than silver because it is charged at 200% more. A resource pool can give Gold two times more shares, but whether those extra shares translate into half the CPU readiness cannot be determined upfront.

Same with storage. A VMFS datastore is not able to provide differentiated service levels. Even if they can, it's not based on latency, which is what your customers care about. The same limitation applies in vSAN. Let's take an example.

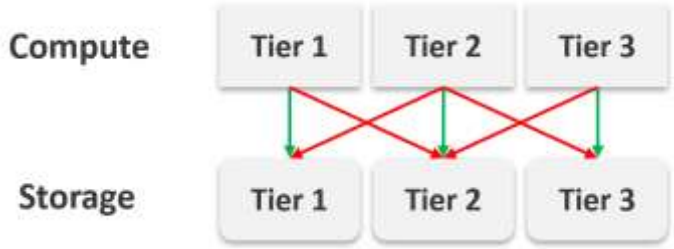
- You promise Gold VMs will get 10 ms latency at worst. For Silver VMs, you promise up to 20 ms latency. The justification for 2x the latency is they pay half as much. So far so good.
- You set your storage policy and IO Control, favouring Gold while trying to ensure Silver gets something.
- Since storage IO control configuration is not based on latency, you may end up with 9 ms for Gold and 21 ms for Silver, where you could have gotten 10 ms and 20 ms.

## Mix and Match

Should we provide flexibility, where customers can choose Gold Compute but Bronze Storage?

In the following example IaaS, mixing is partially allowed. Tier 1 cannot mix with Tier 3, as seen by the missing lines.

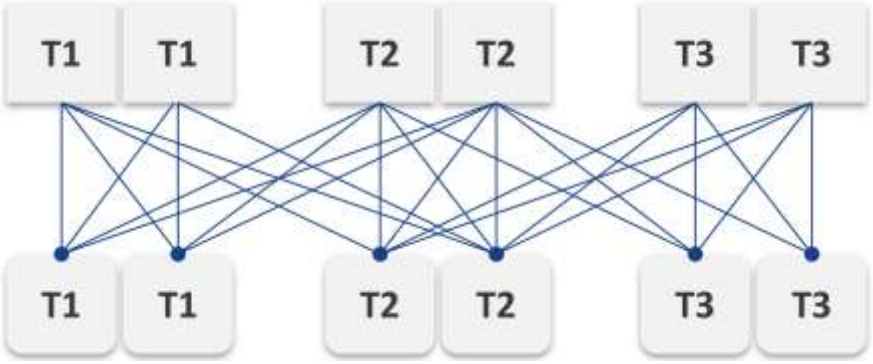
The result is 7 combinations instead of 9, as shown by the 3 green lines and 4 red lines.




It looks operationally visible.

Now let's see if the above idea scales….

We will just add one extra cluster for each tier. Only 6 clusters in total.




Simply adding 1 cluster for each tier dramatically increases the permutation. What if you have 10 clusters each?

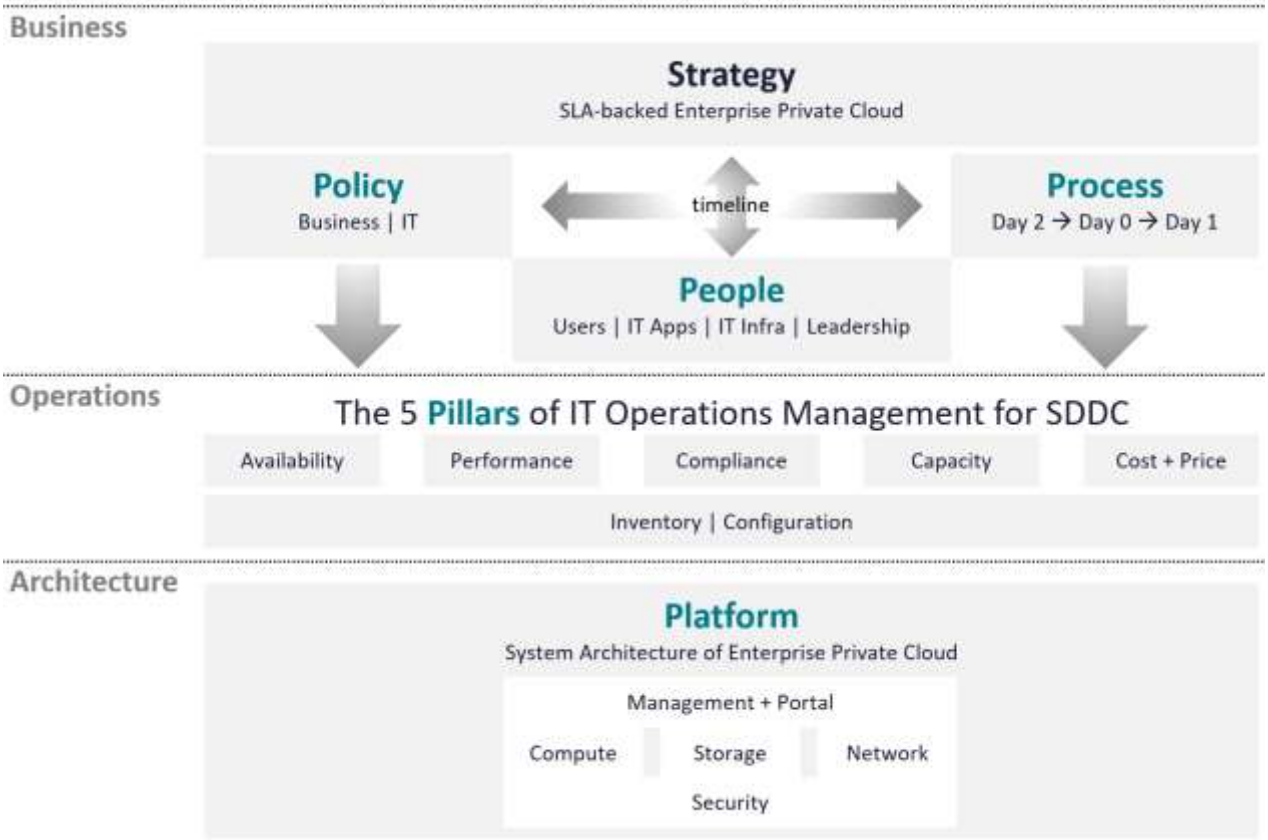
This is a classic case of flexibility becoming complexity. Flexibility has its price, so you need to make the business call. In smaller environment, you can afford to be more flexible.

There are further complications. As examples:

- You need to have two tags: one for compute and one for storage. What do you call a VM with Gold compute and Silver storage?
- Separating storage from compute also means 2x the tagging. You now need to tag both the clusters and datastores.
- We can create a logic where the virtual disks inherit the class of service from the datastore. But what if we have RDMs? Yes, that's still a rather popular choice.
- If you further allow different storage classes for different VMDKs, then you need to tag each virtual disk. This makes reporting difficult as now you need to list each disk attached to each VM.
- Performance troubleshooting also becomes difficult. Say you allow a VM to buy Bronze Compute but Gold Storage, because the load is storage sensitive, but the application team does not care about CPU. What if the VM disk latency is caused by CPU contention?

# SP5 Framework

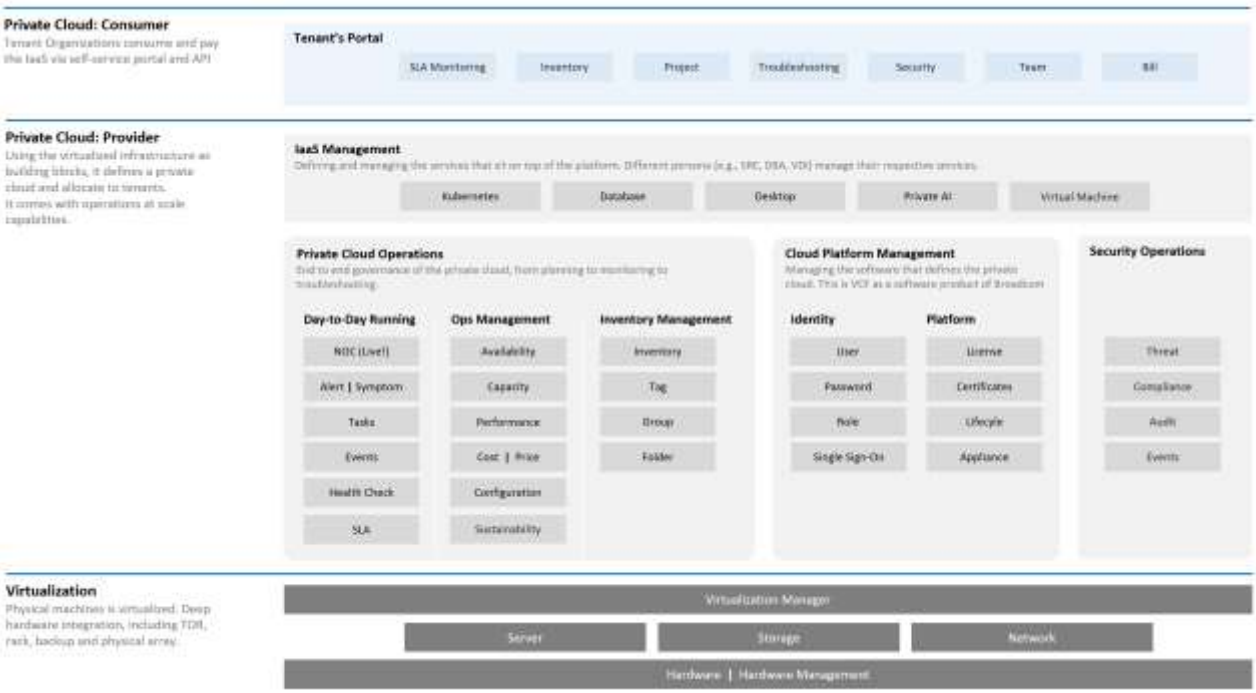
VCF as the core of your private cloud require a new IT operations management framework. I call it The SP5 Framework. It's the IT Business Strategy supported by its 5 components (Policy, Pillars, Process, People, Platform).




There are 5 pillars of operations management. They are the management disciplines a CIO directly cares about, which is why you need to ensure they are "good". To achieve that, the people (IT professionals) is organised accordingly. Processes and policies are then put in place to balance between governance and agility. A complete set of policies and SOPs answers "who does what when where and how". Finally, you implement all these using technology product, such as VMware Cloud Foundation.

What are the capabilities of a private-cloud-ware? How would you explain to your CIO and CTO?

Using a vendor-neutral diagram, we can say that virtualization is the foundation technology that enables the creation of flexible pool of resources.




What do you manage, actually? What do your customers want you to manage? How do you manage what you need to manage?

The most basic thing is you need to have visibility into the environment. There are thousands of objects (e.g. VMs, applications, firewalls) with complex relationships and interdependence. Inventory gives you this. That's why it's the first box in the diagram below.




Once you know what you manage, you can then move on towards making sure those things are healthy. If there is no real problem, then you move to address potential problems. If there is no risk, then you look for optimization.

How do you know you've mastered operations management? What's the acid test?

When you can go on long vacation in peace, without the worry of urgent escalation calls asking for information or decisions that only you can make or provide.

# Timeline

The clock is emphasized as human and agents operate on a timeline. All the processes are implemented on a rolling timeline.

The roles conducting the tasks are:

| | |
|---|---|
| **Level 1 Team** | Watch the Live Screens on the NOC wall |
| | Manage alerts and perform the SOP associated with the alerts |
| **Level 2 Team** | Conduct the daily health check with the goal of preventing alerts of the day |
| | Analyse the logs to catch issue early |
| **App Team** | Watch their own critical applications. Frequency depends on the business cycles of the application. |

Human and AI agents are 3D being. We operate on a timeline. Human operators also comply to the nature of days and nights.

Different roles have different time pace. The following diagram shows only some of the roles.




The NOC room team runs at the most pressing speed as they deal with real time situations. They are the only role watching the operations live as it unfolds. As a result, they are typically equipped with multiple large screens on the wall.

Their focus is to keep the environment healthy. Typically, they do not have the skills nor time to improve the environment. This is where the Level 2 team comes in. Unlike the generalists in Level 1, the various groups that make up Level 2 team are specialist on their own area.

These Level 2 team perform proactive check, at the start of each day. The goal is to prevent alerts firing during the day, as that would make their life easier that day.

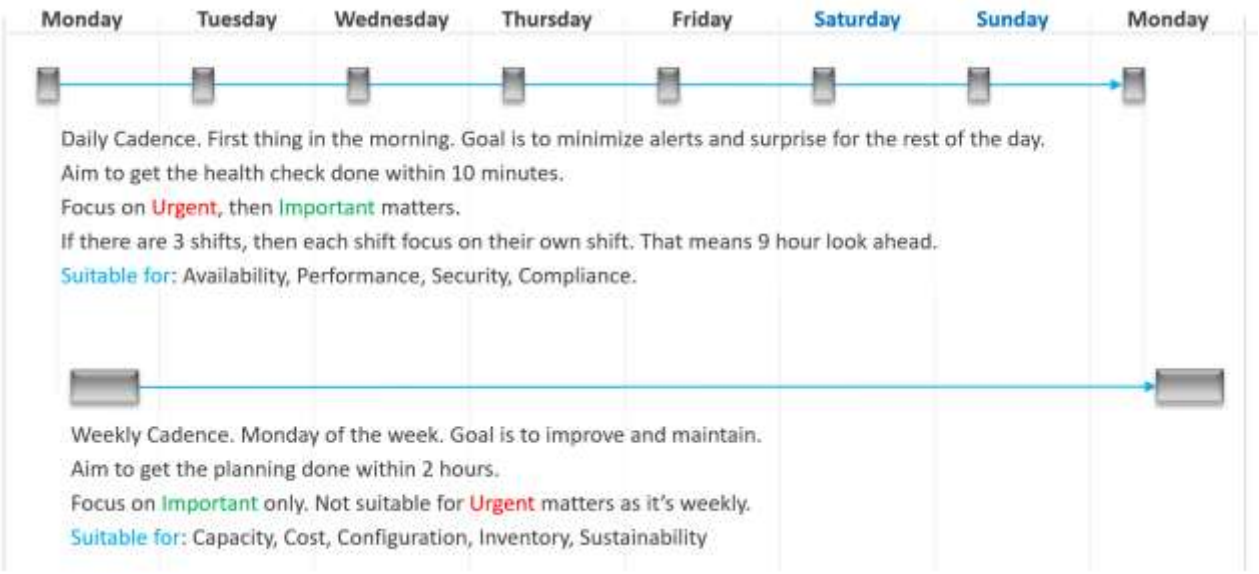
I added Capacity team and Audit team to show examples of team that do not get involved in day-to-day operations.

## Cadence

Let's now see the diagram differently. This time, we focus on the timeline as human operates along a timeline. We adhere to the natural cycles of nights and days.

As part of proactive operations, there are at least 2 cadences:

- Daily
- Weekly




Let's highlight the differences:

| | Daily | Weekly |
|---|---|---|
| **When** | First thing in the morning. Do it before you jump into the first problem.<br>You are planning your day here. | On Mondays.<br>After you settle with your daily cadence. Does not have to be first thing. However, you need to block longer time |
| **Duration** | Minutes.<br>Aim for 5 minutes. This will allow you to do it a few times a day. | ~ 1 hour.<br>1 hour is probably what most administrators can practically afford. |
| **Other Tools** | Use the dashboard together with the Alerts home page and NOC screens. So yes, ideally you have multiple large monitors. | |

### Weekly SOP

Weekly or longer is suitable for different types of tasks. As you have more time and are working on longer time horizon, you should look at both the big picture and look ahead.

The end of the week is a good time to document the changes and lesson learned from the week, and plan ahead for the next week.

| | |
|---|---|
| **Capacity** | Capacity monitoring and planning. Check actual growth vs projection (plan). |

| | |
|---|---|
| | Reclamation Process. VM rightsizing, Idle VM, etc. This can be done weekly as you need to deal with the VM owners. |
| **Compliance** | Root cause for non-compliance is documented, and preventive measurement is put in place so it does not happen again. |
| **Configuration** | Minor updates. For example, from vSphere 7.0 U1 to U2. This is typically part of standard IT tech stack hygiene, where you keep up with the update from all your vendors while making sure they are compatible. This protects you from security non-compliant and emergency patching during business hours.<br>Review does not mean immediate implementation. For example, there is a newer version of VMware Tools. You may decide to start implementation in 2 months, as you have 15K VM and you need to prioritize and batch them. |
| | Major upgrade. For example, from vSphere 6.5 to 8.0. This is typically a one-off project, as opposed to regular maintenance. The implementation is typically executed within a green zone, so other regular maintenance may be deferred to make space. |
| **Overall** | Weekly Management Report. Focus on reviewing the operations of the week, and plan for next week. |
| | Review of ad-hoc events. What are the lessons learned, and can they be turned into an SOP and alerts set up? |

## Monthly SOP

The month serve as logical time period as human and business relate well with calendar months. There are different activities at the end of the month or the start of the month.

| | |
|---|---|
| **Availability** | Restore test of backup. Make sure it can be restored and the data is readable. |
| **Configuration** | Less urgent update. Review new versions and ensure you do not fall too far behind. |

## Longer Cadence

You complement the above frequent SOPs with a regular cadence with a longer time horizon. Naturally, the focus is on the big picture, major projects, and strategy.

| | | |
|---|---|---|
| **Quarterly** | Overall | Quarterly Management Report. Focus on longer term items such budgeting. |
| | Cost | Budgeting. Review actual versus plan. |
| | Availability | DR Test (Production is still running). Isolate the network. To ensure users are comfortable with the procedure when actual DR strikes. |
| **Half-yearly** | Availability | HA Test. Actual test that your HA works as intended. Covers vSphere, physical switches, storage array, etc. |
| **Yearly** | Availability | Actual DR Failover (Production not running) and Failback to primary DC |
| | Capacity | Inventory Stock Take. To discover unused VMs and physical items in data center |

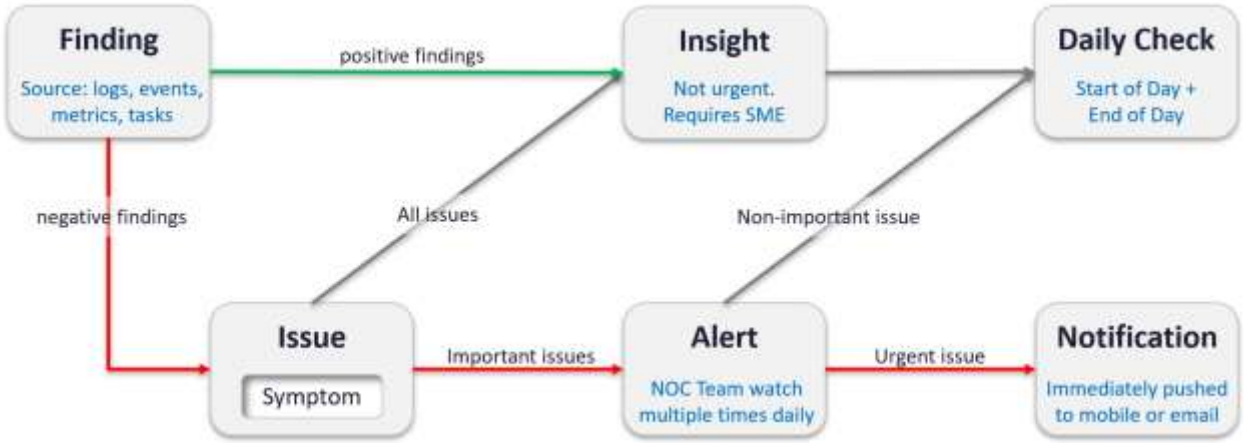
| | | |
|---|---|---|
| **Ad-hoc** | Capacity | Unexpected demand. This is why it's important for capacity teams to stay close with the business, especially the ones working on major initiatives. |

# Basic Terminology

It is as if observability and monitoring are not confusing, there are other English words used to monitor a system:

- Finding
- Insight
- Issue
- Alert
- Symptom
- Notification

The following diagram shows how they are related.




| | |
|---|---|
| **Finding** | Finding is what is found. It is the result of diagnostic or analysis.<br>It can be bad things or good things. Bad = Issue.<br>Since the sources of finding (logs, events, metrics and tasks) focus on negative, a no entry typically means it's a good day. For example, you find no login to a confidential cluster.<br>Findings are input to Insight. Generally speaking, it takes someone who is familiar with the environment to derive an insight from findings. |
| **Insight** | A useful observation with real business benefit. This typically requires someone with both technology expertise and familiarity of the environment.<br>Findings are input to Insight. An expert can derive an insight from findings. |
| **Issue** | An issue is identified by its 1 or more Symptom. No symptom simply means no issue, you are healthy. |
| | Type of Issues:<br>• Availability: Hard failure, soft failure<br>• Configuration: Drift, unsecure |

| | |
|---|---|
| | • Performance<br>• Capacity<br>• Security |
| **Alert** | It's an urgent Issue with Symptom crossing threshold.<br>An issue that is urgent (may not be important) needs to reach Admin fast. Admin wants to be notified.<br>An issue that is not urgent (but important) can be analysed after Admin are done with time-sensitive issue |
| **Notification** | A mechanism for VCF Operations to inform users of insights or alerts that need their attention.<br>This takes different format:<br>• Email<br>• Messages<br>• On-screen when user login |

## Monitor | Troubleshoot

Monitoring is **What**, while Troubleshooting is **Why**.

Monitor is where you compare Plan vs Actual. That's why the goal must be clearly defined. Does the reality match what your architecture was supposed to deliver? If not, then you need to adjust your plan. That's why Plan and Monitor form a circle.

For example, you plan to deploy VDI for 10K users. At 1K, you find out that the users are consuming more resources than plan. You either need to scale down your deployment or add more resources.

You do this when reality is worse than planned, or something is amiss, not when there is a complaint. You want to take time in troubleshooting, so it's best done proactively. And quietly with no one rushing you for results.

Troubleshooting is often wrongly elevated in importance or status, as if it's a singular or major area of operations. It is not. What you can troubleshoot, and how you troubleshoot it, should be planned during system architecture phase. Another word, think of what can go wrong, how they can be detected, and the remediation action. If you do not plan the troubleshooting element of your architecture, each actual troubleshooting event will be more painful that it needs be.

| | **Monitoring** | **Troubleshooting** |
|---|---|---|
| Question answered | What is the problem? | Why does it happen? What is the actual cause of the problem? |
| Nature | Proactive. | Reactive. |
| Expertise | Low. A junior IT person is better suited, as it's repetitive tasks with the aid of predefined dashboards and alerts. | High. Needs experienced IT Pro as there are wide variances in the steps taken. Also needs someone who understands the environment. |
| Metric | Generally, 1 metric. And this metric is also the SLA. This is the 1st metric you or your customer check.<br>Primary metric. You check it proactively as part of your SOP. | Always many metrics. There are layers of metrics, one impacting another.<br>Secondary metric. You only check if the primary is reaching threshold. |

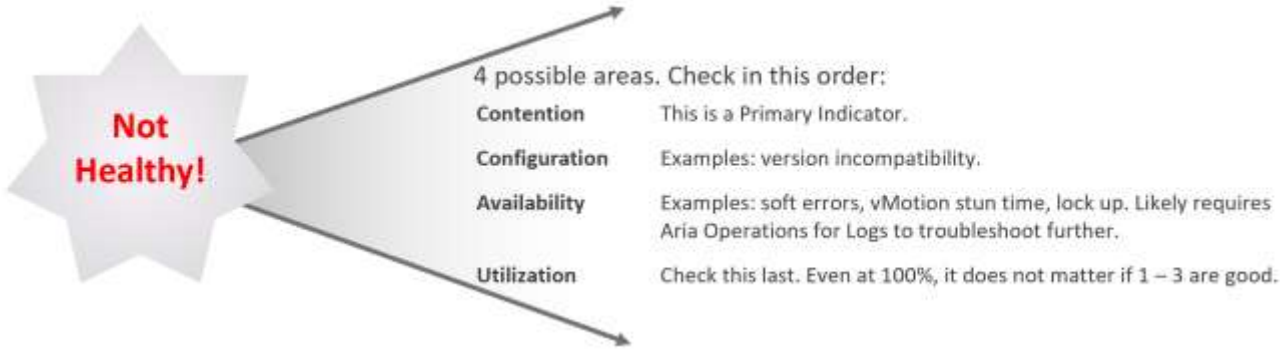
| | | |
|---|---|---|
| Duration | Should take just 5 minutes. | Can take days, with back-and-forth discussion amongst various teams. |
| Frequency | Performed daily.<br>Gold Class will have higher frequency of regular monitoring than Bronze, as part of the SLA. | On demand. |
| Timeline | Now and Future. You consider future load and anticipate. | Now. Future is irrelevant. Your focus is to put out the fire or potential fire. |
| Logs | Not required. Metrics, properties, and Event suffice. | Almost always required. For network troubleshooting, you also need NetFlow data. |
| SLA | SLA is applicable in monitoring. | Yes. It becomes urgent if SLA is breached. |
| KPI | Use KPIs in monitoring instead of individual metrics. | Yes, but as a starting point. You then drill down into supporting metrics, which are often raw metrics. |

In most cases, monitoring is best done using a 5-minute interval, as 1 minute of bad metrics may not have business impact. Troubleshooting on the other hand may require per-second granularity. However, that does not always mean you need to see each and every counter if your remediation action is the same.

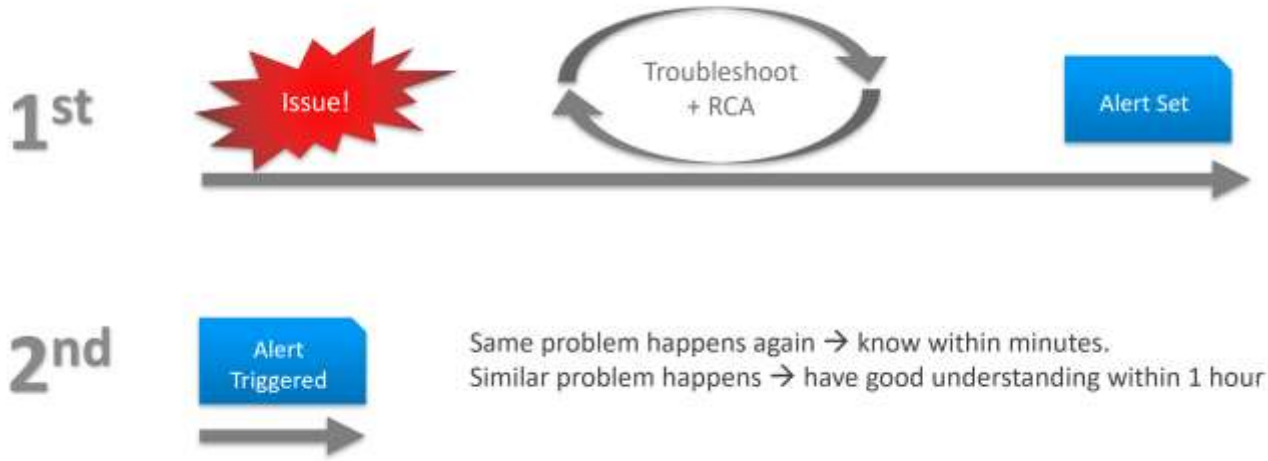
## Troubleshoot

There are many things that can go wrong, especially in production and on the eve before you take a vacation.

Troubleshooting requires expert team. The expert team is also the team setting up the thresholds used by the Level 1 team. Troubleshooting involves logs analysis, as many systems do not generate complete metrics, and there can be many different causes behind a common problem. At the end, the actual root cause may not even be closely related to the problem. Troubleshooting is much more than simply "finding out" and goes beyond just gathering facts. It focuses on why, and then formulates a solution to prevent future incidents. Incidents mean something is dead, slow, or breached. You troubleshoot availability, performance, and security. Capacity and cost are not something you troubleshoot.




Due to its complex nature, the first time an incident happens is forgivable.

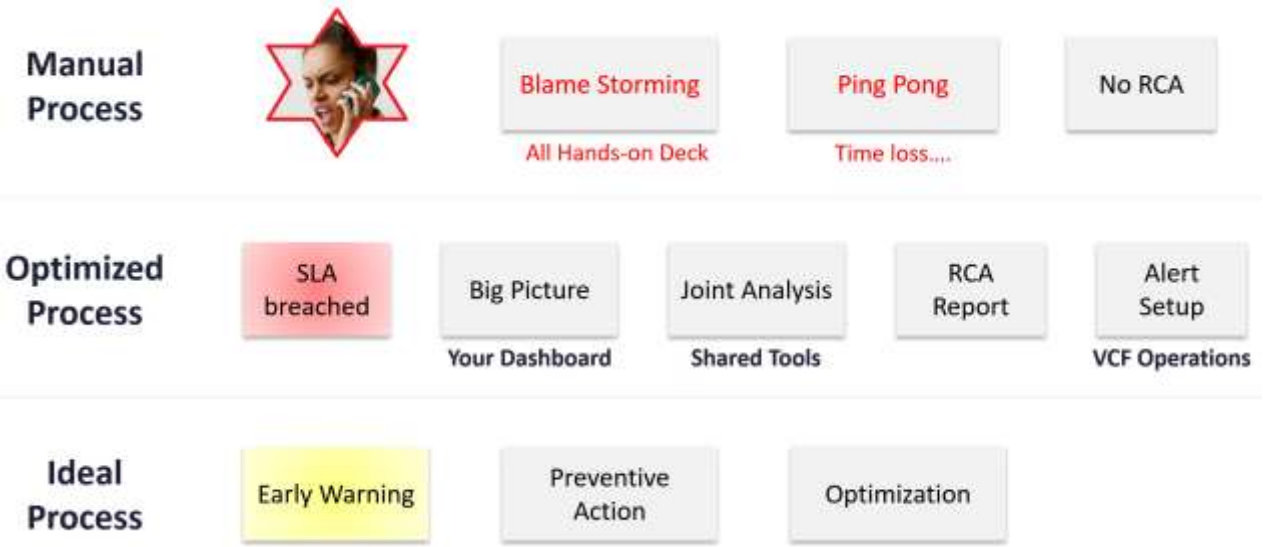



To codify troubleshooting, consider a layered approach. This makes it easier for the less technical teams. Classify your troubleshooting metrics, events and properties into two categories:

| | |
|---|---|
| **Primary Metric** | This defines performance. It is the What. This is always expressible as 0 – 100%, so it's easier for the Level 1 team. It's almost always a hybrid metric.<br>Example: Kubernetes Cluster Performance (%) |
| **Secondary Metrics** | This provides some explanation to the primary metric. It is the Why. Aim for this to be expressible as 0 – 100%, so it's easier for Level 1 team.<br>Example: Highest ESXi Memory Consumed (%) in a Cluster.<br>Metrics that can't be color coded are harder for Level 1 teams, as the meaning depends on the context.<br>Example: VM Disk IOPS |

## Root Cause Analysis

Root Cause Analysis (RCA) is an important component of an optimized process for operations management.




The structure of an RCA report varies among customers, even if the issue they are troubleshooting is essentially the same. Regardless of structure, what do you think is the most important content in the report from an operations management perspective?

The most important content in the report should be the corrective actions to prevent recurrence, such as alerts that are configured to proactively highlight indicators of recurrence. Without these alerts, you cannot reliably detect the issue before it reoccurs and take corrective actions if required.

There is a good chance that the root cause is different than the symptoms. It may happen on a different object altogether and the error message could be seemingly unrelated. A root cause typically starts as a log message, meaning it has not bubbled up into the screen (UI) as formal alarm. When the vendor support team recommends you a specific log message to trap, how do you validate it is correct?

You need to ensure that the alert is valid. That means it should not result in false positive.

Let's take an example. Take a Virtual Desktop Infrastructure (VDI) system mass disconnect issue, where >100 users had their sessions disconnected at the same time. The analysis concludes that the problem started with a log message ("resuming traffic on DV port"), so we need to trap this message when it appears again.

**vmware.log**

- 2016-02-01T23:42:15.266Z| vmx| I120: GuestRpcSendTimedOut: message to vdiagent timed out.
- 2016-02-01T23:42:15.266Z| vmx| I120: GuestRpc: app vdiagent's second ping timeout; assuming app is down

**vmkernel log**

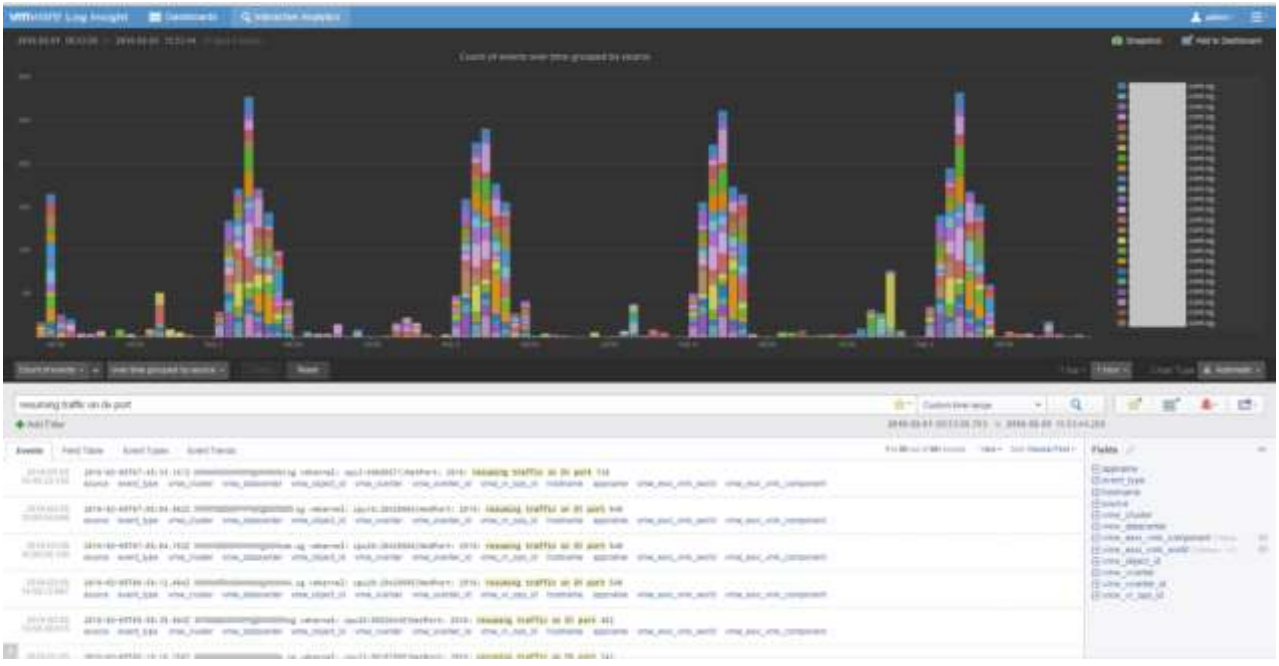
- 2016-02-01T23:42:14.842Z cpu3:49141417)NetPort: 2916: resuming traffic on DV port 588

**Analysis:**

- The message "resuming traffic" means VM was unavailable for a brief duration and resumed traffic transmission.
- This implied that the issue occurred on the server side and not on the client side. Coinciding with the GuestRPC timed out messages, we conclude that the Guest was unresponsive during this time and resumed subsequently.

The first thing you need to do is validate the above alert. Using tools like VCF Operations for Logs, you cross check the message against your entire environment, especially the healthy (in this case, unaffected users). Ideally, you cross check for entire week, not just during the time the incident happened.

The following was the result when I cross checked against all the users in the last 5 working days. The log message has happened more than 1000 times, meaning that "resuming traffic on DV port" is not the message that I should base my alert on. There are too many of them and there is a clear pattern following office hours.



# Health

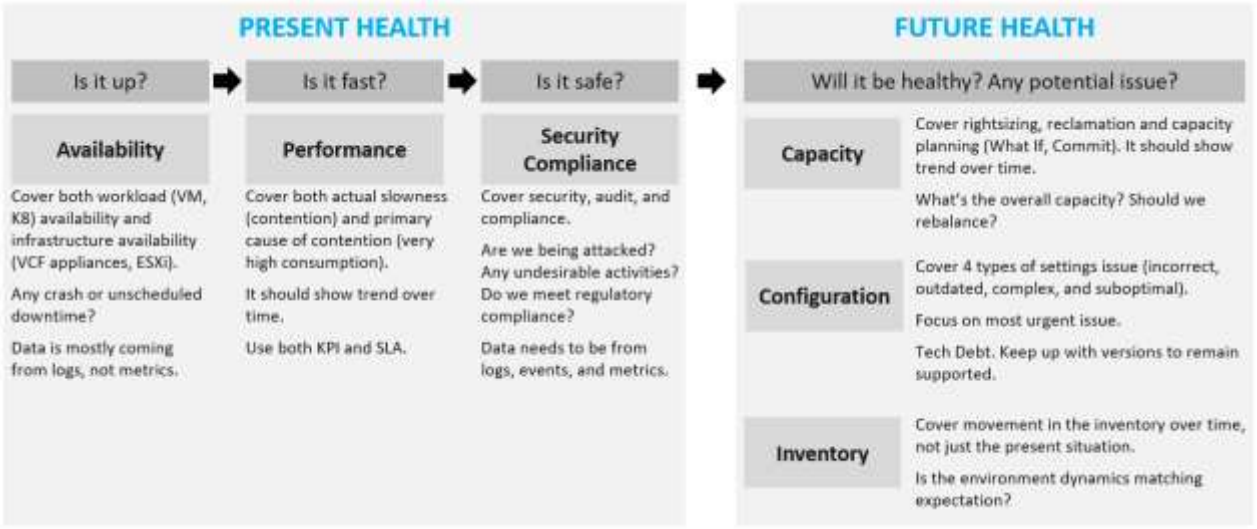
The English word Health is used extensively in IT industry for decades, but nobody takes the time to define it subjectively.

You certainly want your environment to be healthy. The desire to achieve that nirvana state results in a definition that is too broad as you try to cover everything. When the health metric covers too many things, you can end up with low score and yet everything is running well!

Health is hard to define, as it depends on the context and object. The English word health itself is subject to interpretation. How healthy are you? For example, I exercise regularly and can perform many rounds of pull ups, push up, deadlift and squats. I'm physically healthy. Biologically though, I've been suffering from irritable bowel syndrome and sleep disorder. As for mental health, my wife thinks I have a big problem 😊

Let's try another real-life context. How healthy is your country? Let's take the world superpowers (USA and China). Both are well accepted as superpowers, in both economy and military. But how healthy are they?

The answer depends on which aspect and which provinces you're talking about. It needs to have more context. That's why you do not have a single score for health.




If you insist on defining "health", then my recommendation is map it to the pillars of operations. When you do that, you vrealize that there are 3 sides of health, not 1. Since there are 3, you need to have 3 different metrics.

| | |
|---|---|
| **Present Health** | Your health in the present time, especially right now. It covers real problem that has happened and/or is still ongoing.<br>The present health only includes reality. It does not include possibility. That's covered under Risk. Just because you have security risks does not mean you're being hacked. |
| | There are 3 problems that impact the present health:<br>• Availability<br>• Performance<br>• Security (present)<br>Availability and Performance are related, which means they are not the same thing. Your environment can be 100% up but slow. |

| | |
|---|---|
| | Security has 2 parts: present and future. The present health only covers actual security issues. For example, your environment is running fine, but you notice unauthorized access & suspicious commands being issued in your ESXi consoles. |
| **Future Health** | It covers potential problems. There is no problem at this moment, but if you do not act on it, you increase the risk of it becoming a problem.<br>Using a day-to-day life analogy, you are healthy now but have a risk of heart problems if you do not stop smoking, are lacking sleep, consuming unhealthy diet, and are overweight. |
| | There are 4 problems that create risk in operations:<br>• Security (future)<br>• Compliance<br>• Configuration<br>• Capacity |
| | In all the above problems, the present health is not impacted as there is neither slowness, downtime, nor security breach. What you have is a risk, as your applications and operations continue as if nothing happens. Your data has not been stolen. Your customers do not notice, and your business is not affected.<br>Let's take an example. You do not configure HA in a vSphere cluster. If all ESXi hosts are running, your availability is 100%. Your performance is also not impacted. However, you have an availability risk. |
| | If you improve, your future health will improve. This is about effectiveness and efficiency.<br>Effectiveness is about doing the right things. Efficiency is about doing things right. You can operate the wrong architecture correctly.<br>**Efficiency** is about optimization. There is no health problem at present, nor is there a health risk for future problems.<br>You want to increase efficiency as it lowers cost, reduces complexity, reduce capacity footprint and improve application performance. |
| | There are multiple ways to increase efficiency, hence the definition varies among objects.<br>• Hardware refresh is one common way to increase efficiency as you often get faster performance and more capacity at the same price point.<br>• Wastage (oversized, unused EC2, idle, unmapped disk space, orphaned file, etc.)<br>• Cost. Compare your cost with other cloud providers as IaaS is essentially a commodity.<br>Green Operations fits efficiency as sustainable operations call for lean operations. |

# Why a New Framework?

Before I came up with the pillars of operations, I studied other IT frameworks as it's easier to use existing as adoption will be faster. There are many frameworks in running the business of IT. ITSM, IT4IT, ITIL, COBIT, Microsoft Operations Framework, and many others where you can learn about overall IT Management[7]. In this book, I'm focusing on areas that are related to VMware Cloud Foundation, and only on Day 2.

The philosophy of ticket and incident-based operations are reflected in these frameworks. I'm not against the principle of using tickets as a unit of work, but most implementations have been disappointing to users. That's odd as the very purpose of the help desk is to help the users. When you call your telco, bank, airline, etc. for support and you get a ticket, how has the experience been to you?

It's terrible.

So why impose that bad experience to your customers?

The SP5 Framework differs fundamentally as it aims to do away, not just minimize tickets and incidents. The proactive, insight-based operations require a small number of seasoned IT professionals with deep technical expertise and knowledge of the business. It does not work on a very large number of teams spanning many departments, each following rigid and siloed policies.

## Complexity

I found many frameworks to be heavy. This means they are complex and costly to operationalize. ITIL 5 has 34 management practices for the entire IT Business. I find 34 are simply too many to manage. You end up managing the management. This defeats the purpose to begin with.

| Service Management (17) | Technical Management (3) |
|---|---|
| Business Analysis | Deployment Management |
| Service Catalogue Management | Infrastructure & Platform Management |
| Service Design | Software Development & Management |
| Service Level Management | |
| Availability Management | |
| Capacity & Performance Management | |
| Service Continuity Management | |
| Information Security Management | |
| Service Request Management | |
| Service Desk | |
| Incident Management | |
| Problem Management | |
| Change Enablement | |
| Service Validation & Testing | |
| Service Configuration Management | |
| Release Management | |
| IT Asset Management | |

[7] There are other frameworks, such as eTOM and BIAN. Since they are domain specific, they are not suitable for VCF.

In addition to managing 20 types of services and technologies, you also need to manage 14 general management:

- Strategy Management
- Portfolio Management
- Architecture Management
- Service Financial Management
- Workforce & Talent Management
- Continual Improvement
- Measurement & Reporting
- Risk Management
- Information Management
- Knowledge Management
- Organizational Change Management
- Project Management
- Relationship Management
- Supplier Management

Even with 34 management practices, ITIL 5 merges Capacity and Performance as 1. In the context Broadcom VCF private cloud, they are distinct disciplines, done by a different persona and are having different processes. This book covers both disciplines in-depth, while ITIL 5 does not.

## Duplication

Frameworks have lots of big words that are subject to interpretation in the real world as you try to implement them. Some of them also overlap, or even mix up concepts altogether. For examples:

| | |
|---|---|
| **Monitoring and Alerting** | They are not peers.<br>When you monitor a system, you will arrive at one of these 3 conclusions:<br>1. You need to troubleshoot (because it is broken or wrong)<br>2. You can optimize. It is not broken or wrong, but you can save money or improve performance.<br>3. There is nothing you can do.<br>Alerting is one of the 3 ways for your monitoring system to communicate to you. You interface to the tools via alerts, dashboards, and reports. |
| **Changing vs Optimizing** | They are not peers.<br>Optimizing is part of changing. You can't optimize without making any change at all. You change something because it is broken or it needs improvement. |
| **Operating vs Supporting** | Support is part of day-to-day operations. It mostly revolves around inquiries and incidents. If you're doing operations right, it also involves insights and proactive changes. |
| **Automation and Orchestration** | Automation is the "big item". Orchestration is just one of the techniques for automation. It typically strings together a number of different scripts, which could be in different programing languages. |

| | |
|---|---|
| **Measurement and Reporting** | I'd rather call this Monitoring, as Measurement is a noun.<br>Reporting, as covered in Part 2 Chapter 1, is just one way for computer systems to talk to humans. The other two are alerts and dashboards. |
| **Monitoring and Event Management** | Monitoring is a process, an activity. It is not something you manage. The diagram in the next section shows how Process and Pillar are related.<br>Unlike alerts, events could be neutral or even positive. An event is an accounting of something that happened, and related to tasks or activities. Just like a metric is an accounting of utilization or contention. A log is a record of something that happened. You don't manage metrics, event, logs as you don't have full control over them.<br>So events are just inputs to your operations. It's one of the 5 data types to monitoring. The other 4 are metrics, properties, logs, and netflows. These 5 measure the observability of an architecture, and they feed into your monitoring tool. Their values or meaning are defined by the context of the pillar. |
| **Operations and Governance** | They are the same. You manage operations by having governance. Operations management covers risk (availability risk, capacity risk, security risk) and applies control (compliance to approved configuration, clear accountability for each role, manage cost so they don't overrun) . |

## Clarity

Being precise is essential, hence the need of jargon. But using jargon when it's not required is adding complexity. Even worse, using common English word that means something else in real life is confusing. PCMCIA stands for People Can't Manage Computer Industry Acronyms. It is a lighthearted reference to the confusing terminology in IT industry. To be fair, other industries also have their fair share of issues.

There are many IT management areas that are not pillars of operations. They are either unnecessary or just providing support to Operations Management. For examples:

| | |
|---|---|
| **Service Level Management** | An SLA is not something you manage; it is a by-product. You manage availability, performance, compliance, and if you do that well you will pass your promised SLA. |
| **Service Continuity Management** | This is part of your architecture, not something you manage. A good architecture is a living system, as it's designed to handle updates and upgrades without compromising the SLA. |
| **Demand Management** | This is part of Capacity, as the art of capacity management is about making sure there is just enough supply to meet demand. Both insufficient and too much capacity are problems. |
| **Continuity Management** | This is part of Availability, which cover snapshots, HA, DR, replication, active/active, back up, etc. |
| **Incident management** | To me, incidents (e.g. your website goes down) andproblemsm (e.g. your core routers hit a rare bug) are not something you manage. You manage the impact, such as availability and performance. As part of availability, performance and compliance management, you are bound to have incidents, which were caused by problems (e.g. incompatible configuration, bug). |
| **Service request management** | |
| **Problem management** | |

| | |
|---|---|
| | You should focus on why you have so many availability, performance, compliance, configuration, or capacity problems. By reducing them, you will reduce your incidents. Aim to minimize incidents and problems, instead of accepting their prevalent existence as acceptable and investing on their management. |
| **Service Desk** | This is a technique, a solution you design as part of your monitoring strategy. While the team needs to be managed, it can span thousands of people; it is not a pillar of operations. |
| **Continual Improvement** | To me, this is more culture, realized as a set of processes, policies and structures. See Kaizen as an example. While it's important, it's not a pillar of operations |

# SP5 Framework: Pillars

What do your CIO actually care? Because that is what you will manage.

Well, the first thing your big boss care is availability. You have neither performance nor security issue if the system is down. Hacker cannot even login to an vCenter that is not even available.

Next, just because something is up, does not mean it's fast. Your boss can argue that your system is so slow it is basically not available to users. So you need to manage performance**.**

Next, just because the system is up does not mean it's secured. Security and compliance matter.

There are 5 pillars of operations. Each pillar is an individual unit of management. They represent individual disciplines and are compatible with one another.

- Availability management
- Performance management
- Security and Compliance management
- Capacity management
- Value management

How do manage them? What are the key processes required to run multi-cloud operations?

The following diagram maps the process and the pillars.

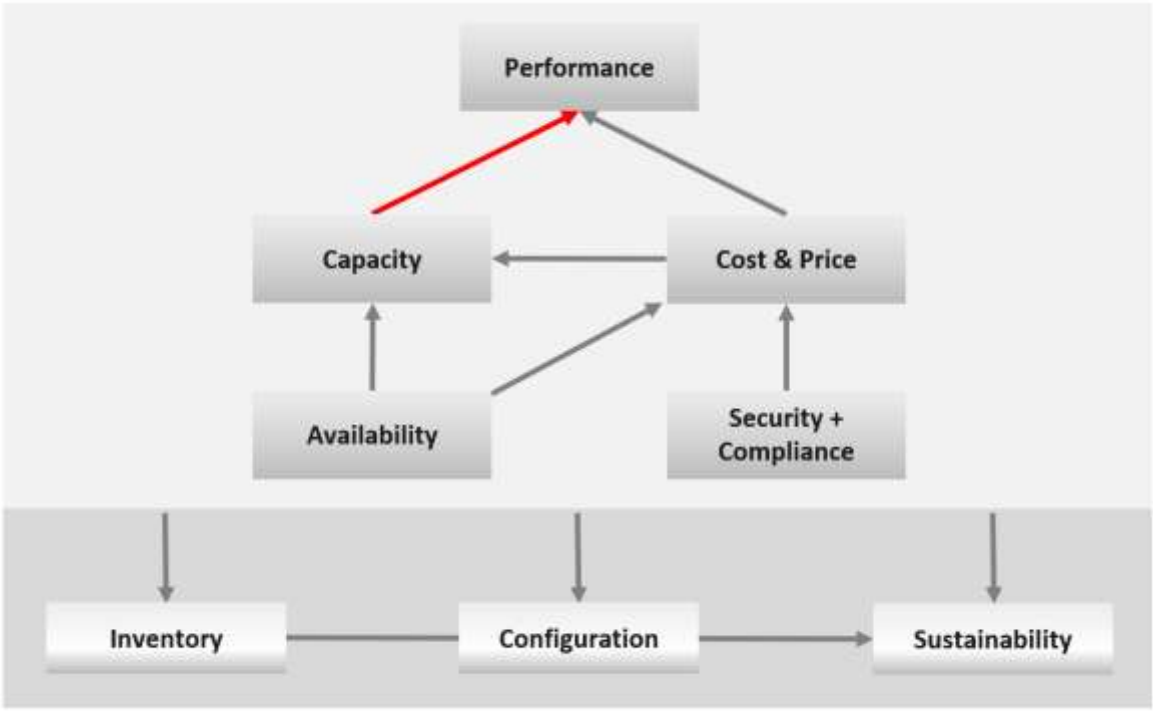
The 5 Pillars of Operations Management

| | Availability | Performance | Compliance | Capacity | Cost |
|---|---|---|---|---|---|
| Day 0: Planning | Yes | Yes | Yes | Yes | Yes |
| Day 1: Design & Build | Yes | Yes | Yes | Yes | Partial |
| Day 2: Monitoring | Yes | Yes | Yes | Yes | Yes |
| Day 2: Troubleshooting | Yes | Yes | N/A | N/A | N/A |
| Day 2: Optimizing | Partial | Yes | N/A | Yes | Yes |

Major technology upgrades require planning and redesign, hence it repeats the cycle above

The complexity of each pillar depends on the technology: for example, vSAN capacity is more dynamic than a traditional SAN. In vSAN, changing the storage policy could create a sudden spike in consumption.

The 5 pillars of Operations Management are interdependent. Knowing the relationship is as important as knowing the individual component. Relationship matters as the symptom and the root cause are often two different things. A performance problem could be caused by a configuration problem, such as an outdated configuration or incompatible versions.




I chose a red line as it's a complex relationship. As Capacity goes down, Performance remains steady. As Capacity drops below 0%, performance suddenly takes a hit.

I put Security and Compliance together outside as it's typically managed by a different department. Their scope extends beyond the Data Center to areas like physical building security and employee work-from-home solutions.

Lines with arrowheads means there is impact:

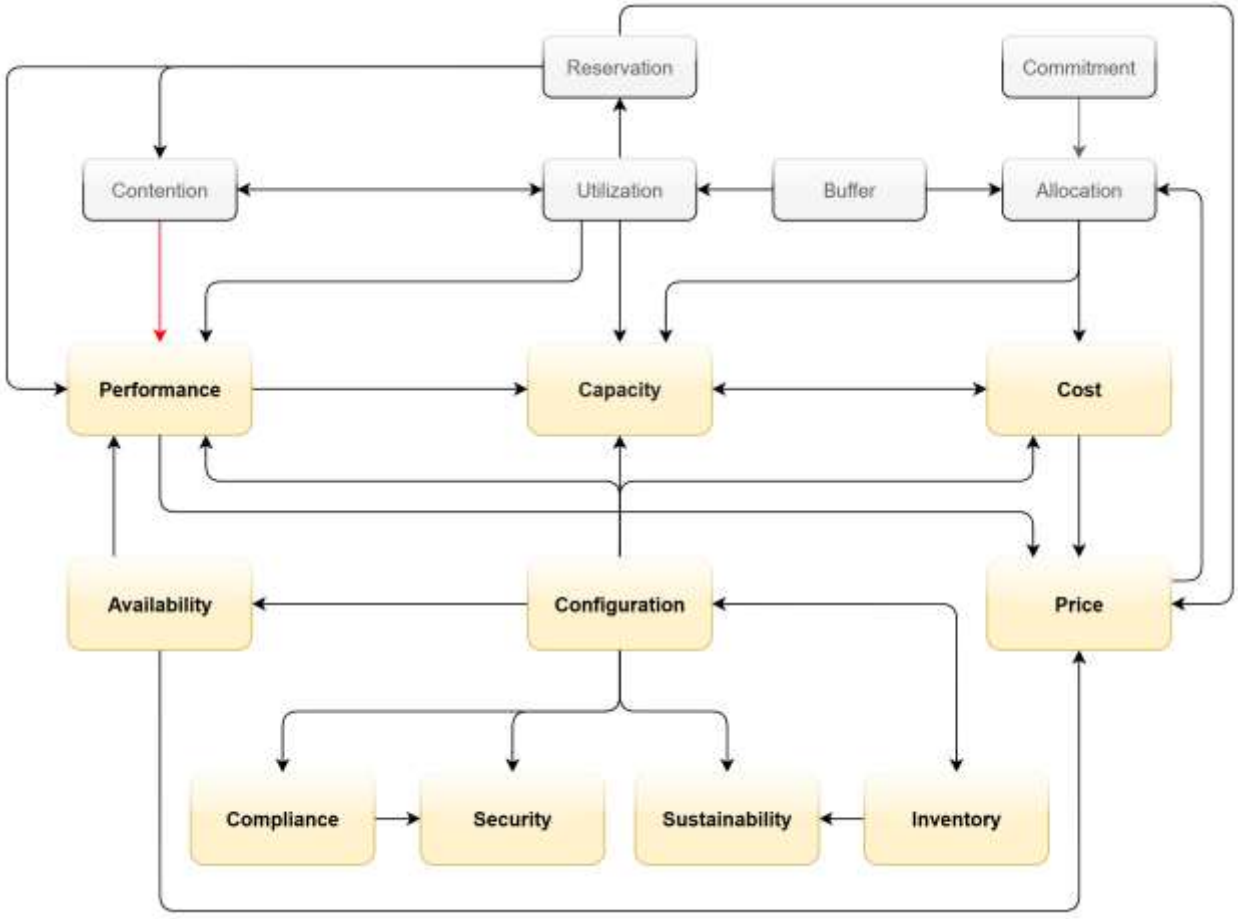
- Availability increases your cost, as you're adding extra resources for no capacity benefit.
- Compliance typically increases your cost. The more compliance requirements you must maintain, the larger the operational overhead involved in managing and measuring it.

Lines without arrowheads means there is relationship but not impact:

- Availability and Capacity should not overlap. You calculate capacity after taking out availability.

- Inventory and Configuration do not impact each other.

The following diagram provides more details.




Let's elaborate on each pillar.

# Availability

There is a spectrum of availability solutions, including snapshots, back up, redundancy, fault tolerant, clustering, and SRM. Each can have impact on cost. Most of them require additional resources so they must be part of capacity planning, so they do not impact performance. For example, load balancers and replication needs to be accounted for.

You also need to include the potential workload caused by DR events in your capacity planning.

There are two metrics for availability:

- Actual Availability (%)
- Operational Availability (%)

The first one simply measures the fact as it is. It does not consider the HA (high availability) configuration and scheduled down time.

Operational Availability complements it by considering the above. So its value will be higher, as it reflects the operational impact. This is also the number used in the SLA.

The higher the Availability SLA, the higher the cost of the service. There is a big increase for each additional 9 of availability. Five 9s of availability costs 10x more than four 9s as the margin of error is reduced by 10x.

Why isn't there a line to Performance?

Availability events like a host outage, when accounted for in the design, should not impact performance, as it does not lead to a drop of usable capacity.

# Performance

Ever heard of "the system is so slow it is as good as down"? Just because something is up, does not mean it's fast. On the other hand, if you have catered for HA, you can have part of the system down with no impact on performance and capacity.

From day-to-day operations, performance is the most important pillar of operations. This is why the next chapter is Performance Management.

Performance is often confused with capacity as "more work done means good performance". This "more work" requires higher utilization. This simple thinking has drawback as it associates idle with low performance.

The primary metric for performance is contention. You don't have performance issues if there is no contention. One common cause for contention is utilization over capacity or limit.

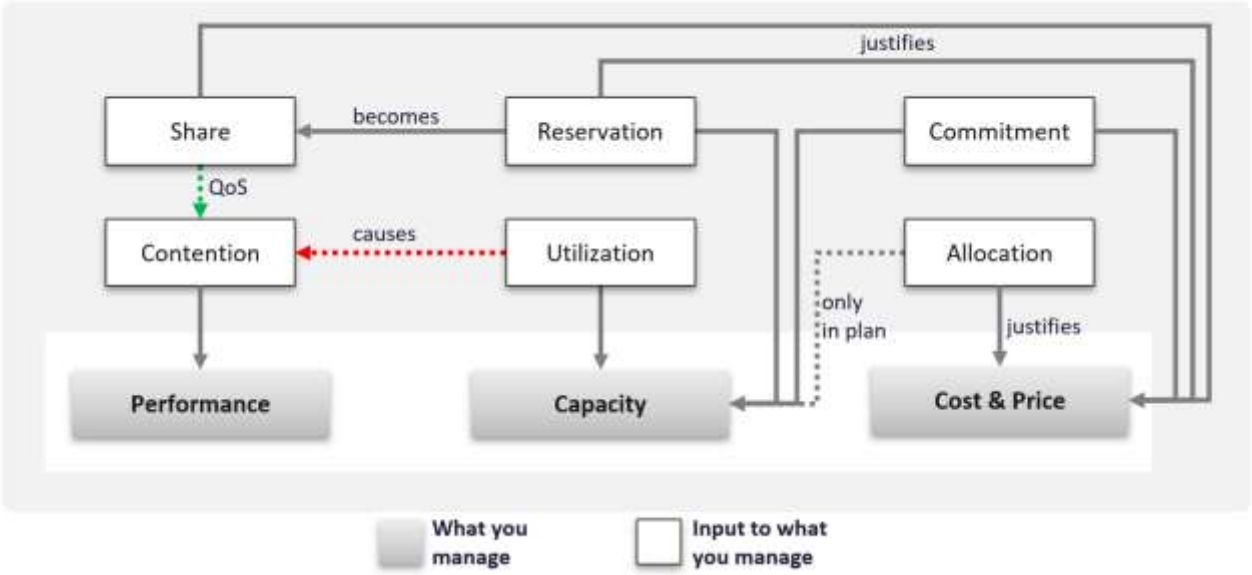
# Capacity

There are 2 types of capacity:

| | **Short Term** | **Long Term** |
|---|---|---|
| Performed by | Day to day Operations | Capacity Planner |
| Decision Point | Stop provisioning new load | Buy new hardware |
| Availability Management | Highly relevant | Not so much |
| Performance Management | Highly relevant | Not so much |
| Value Management | Not so much | Highly relevant |
| Timeline | Present.<br>This means now or today. It cares less about tomorrow. | Future.<br>Depending on the lead time to add capacity. If procurement is required, this can be months. |
| | Projection is not applicable. | Projection is mandatory |
| Scope | Resources that can be provisioned within an hour or so | Resources that are not yet purchased |

| | | |
|---|---|---|
| Focus | Hardware | Software (as they are more expensive) |

Short term capacity is related to performance. They are interdependent; hence one is often mistaken for the other.




Capacity is affected by Performance as it needs to consider contention metrics not just utilization. If you can't satisfy existing demands, then you won't provision new workload, hence capacity is practically full. The utilization metrics may not be high yet, but that's a secondary consideration as you stop adding new workloads until you figure out why.

Capacity also considers demands that do not manifest yet. This is why you need to consider reservations and allocation. They also minimize the consumption to go too high and cause contention.

Price can help address capacity. The best way to address oversized VMs is having a pricing plan that encourages smaller sized VMs. If the price is free then use a business justification to right-size instead.

Using highway transportation as example: You do not add lanes because of congestion that lasts a short time or rarely happens.

But what if the workload is revenue generating? For example, during annual sales you expect a much higher demand that happens only 1 day in 365 days. Do you build permanent capacity for the rest of the 364 days? Are you able to provide a burstable capacity, either in the cloud or on premises?

Applying the above into vSphere on-premises, perhaps on that special day you need to limit other workloads, power them off, take hardware from other clusters, or move workloads to other clusters.

It focuses on the consumer, while the long-term capacity view focuses on the provider. The short-term view answers questions that care about today. Examples:

- "Can you deploy additional workloads?"
  The now part of the question is important, as the deployment is typically done by automation. You want to prevent the deployment if it will cause problem. This means you need to check both the present situation and the future situation.
- "Can we take out resource providers temporarily?"
  Examples are putting ESXi hosts into maintenance mode, or doing a cluster upgrade.

## Reclamation

Infrastructure team should not force their internal customers to reduce their own consumption. This is a hard political battle, as there are more developers than administrators in a typical large enterprise. A better way is to show the CFO the total cost of wastage, money that can otherwise be saved. CFO will simply hold the budget for new hardware, and force the respective line of business to optimize their current consumption[8].

# Compliance and Security

Just because a system is up and fast, does not mean it's secure.

Security is related to but not the same as Compliance. Security covers issues such as attacks (be it by internal employee or by external threat). Compliance deals with configuration settings or values that may expose security loopholes or are required to conform to specific sets of standards. Compliance is closely related to configuration as you comply by controlling the values of specified configuration items.

Compliance is measured against both internal and/or industry standards. It's also measured continuously.

In Security you worry about being attacked. This means you track malicious activities, such as inappropriate usage of administrator accounts, or denial of service attacks.

Compliance is binary: you are either compliant with your defined compliance requirements, baselines, or benchmarks, or you aren't. Compliance is also unaffected by complex and qualitative affairs like resource contention and performance. For this reason, as a service provider, you should guarantee to your customers that you will be perfectly (100%) compliant with your own compliance benchmarks for all classes of service.

However, you only provide a compliance guarantee on what you manage. If you don't provide the VM's guest OS as part of the service, then you don't manage compliance for it, and you don't provide an SLA against it.  If you do provide the guest OS, then you must also manage its configuration via something like Microsoft AD Group Policy.

# Cost and Price

Last but not least: money.

Cost and Price are necessary  in the transformation towards becoming a real service provider.

| | |
|---|---|
| **Price** | With hardware becoming commodity and infrastructure becoming invisible, price has naturally become a common denominator among all IaaS providers. The general expectation is the price per VM is similar across cloud providers. One way to provide differentiated pricing is SLAs.<br>While Price should be higher than Cost, it can be set independently of cost. Use discounts and progressive pricing to set the correct price for the right terms and conditions. Progressive pricing will also discourage oversized VMs from being |

[8] Thanks Tyson Then for this idea.

| | |
|---|---|
| | provisioned in the first place. It's easier to handle then, rather than once VMs are already in production.<br>At an individual VM level, the better the Performance SLA, the higher the price customer is willing to pay, hence the term Price/Performance. |
| **Cost** | At the platform level, cost also goes hand in hand with capacity. The higher the utilization of the IaaS, the lower the cost per VM. Cost is separate from capacity as it can be optimized without reducing capacity.<br>Cost and capacity can also go independently of each other. You can increase capacity without increasing cost via technology refresh. You can reduce cost without reducing capacity by lowering non-capacity costs such as the rate you pay for services. |

# Non-Pillars

What's missing in the 5 pillars?

- **Inventory**.
  Inventory is simply what you have. Your responsibility is a lot more than accounting of what you have where and how they change over time.
  Inventory is just a by-product. You plan for capacity, with certain configurations. As you manage your cost & capacity, the inventory will adjust accordingly.
- **Configuration**.
  Configuration is just a means to an end. You care about configuration only because it impacts security, capacity, performance, and availability. So it is often both the problem and solution to all the above pillars.
- **Sustainability**.
  It's covered by both capacity and cost. The only thing unique is the source of power. It's simply looking at the same things, albeit from a different angle. It does not replace any of the existing pillars of operations management.
- **Reliability**.
  This is just a characteristic of the system. Obviously, you want performance & available to be consistent, hence reliable and as you gain confidence, it becomes predictable. For example, if your website performance is not reliable or its uptime is not stable, you will troubleshoot from a performance or availability angle. There is no need to elevate the nature of something into an entity by itself. It complicates operations as you end up with overlap.
- **Recoverability**.
  This is just a property of availability. Just like debuggability is a function of performance & availability.
- **Manageability.**
  This is just a property of configuration.

Inventory is related, but not identical to configuration. They tend to be confused easily. For example, "configuration maximum" actually means inventory maximum.

| Inventory | Configuration |
|---|---|
| An account of what you have where. So, the location, the movement, and the count matter. | Properties of your inventory. The location is just another property of the object |
| Inventory uses a small subset of configuration as the focus is on counting the number of objects. The majority of properties managed by configuration are not relevant to inventory. | The full set of settings, that you either intentionally set or have to accept. |
| Configuration drift is not relevant. | Configuration drift is important. |
| Inventory movement is important. | Inventory movement is not relevant. |
| Inventory has stock stake concept. This can involve physical or virtual items. | Configuration does not. |
| Deals with distribution. You want to keep them "balanced".<br>e.g. number of VMs across all your vSphere clusters. | Deals with variance. You want to keep them minimal.<br>e.g. version of VM hardware versions. |

Examples:

- Number of VMs in a cluster is a part of inventory, not configuration.
- Number of ESXi hosts in a cluster is a part of inventory. But it's also part of configuration as that's the design of that cluster. The cluster is configured with 8 ESXi hosts for a reason, and deviation may need to be explained in design documentation.

Inventory also deals with the *change* of inventory. A high number of churns is an inventory problem, not configuration problem. For example, as part of inventory management, you may want to check how many VMs were added, were deleted, were changed in your environment in the last 1 month.

There is a subtle difference between volume of change and the rate of change. Volume typically covers a longer period (e.g. 5 minutes, 1 hour, 1 day), while rate covers shorter period (e.g. per second, per minute).

To make inventory management easier, you typically group items by function. For example, in IaaS, you may have:

| | |
|---|---|
| **Consumer Layers** | Business Application (typically it spans multiple pods or VMs).<br>Software (e.g. Microsoft SQL Server).<br>Container or pod.<br>VM (which could be in cloud). |
| **Provider Layers** | Cloud services (AWS something).<br>Kubernetes.<br>vSphere + vSAN + NSX.<br>Physical (network, arrays, servers, UPS, racks). |

# SP5 Framework: People

There are many personas required to keep operations running well. Some are directly involved in the day-to-day operations, while others focus on the big picture, hence requiring a longer time frame. In small branches of a large organisation, the roles are played by the same few people, backing each other up. You can have 3 people doing everything with no structure, or 300 people with clear demarcation and formal hierarchy. Regardless, the jobs still need to be done, so document all the roles and responsibilities. Now you know why I make this document editable.

There is a many-to-many relationship between "jobs to be done" and persona. It reflects the dynamic nature of live operations. You have team members taking leave or away, hence you need someone else to step in on an ad-hoc basis.




## Day to Day Roles

| | |
|---|---|
| **Level 1 Operators** | Deal with the production environment. Perform a regular check on the overall environment. Use both insight and alerts.<br>Responsible for closing alerts. Alerts should be closed only when root cause is known, not when symptoms disappear. Closing alerts without knowing why they happened prevents lesson learned and can potentially backfire.<br>Perform simple troubleshooting, following SOP. Typically, these SOPs do not require reading logs.<br>SOP is ideally automated, taking input parameters, so the chance of human error is minimized if the number of manual steps or frequency is high.<br>Focus on Availability, Performance and Security.<br>Typically stationed at Network Operations Center room and called the Help Desk. |
| **Platform** | Activated when Level 1 is unable to solve the problem. For each problem solved, this role should update the troubleshooting guide so Level 1 can be empowered. |

| | |
|---|---|
| | Focus on insight, not alerts. Look at the big picture and try to prevent alerts from happening. They focus on risk (e.g. configuration risk, compliance risk).<br>More senior than Level 1. May specialize is some areas (e.g. vSAN, networking). They document the knowledge base and develop SOP. They automate the SOP as much as possible.<br>Perform advanced troubleshooting, which often requires logs analysis . Work with the Architecture Team.<br>Lead or involved in the evaluation of operations management tools. Design and maintain VCF Operations dashboards and alerts.<br>In larger organization, there can be more levels. |
| **Operations Manager** | Manage the level 1, level 2, level 3 operators.<br>Deal with tenants on SLA. |

# Other Roles

There are other roles that Ops Teams need to deal with. Here are some of them. In larger organisation, each could be a team on their own.

| | |
|---|---|
| **Security** | In large organizations, this could be a separate department by itself. It could also report outside IT if the scope covers beyond computer system.<br>They work closely with the Compliance persona. |
| **Compliance** | Set the compliance settings to agreed internal and industry standard.<br>Verify that non-compliance alert was addressed timely and correctly by the operations team. Report & discuss the compliance status with upper management.<br>Focus on Risk (Configuration, Compliance). |
| **Capacity** | Plan the supply side of capacity, working with the architect role.<br>Plan the demand side of capacity, working with line of business or sales team.<br>Does not get involved in the day-to-day capacity. ESXi hosts going into maintenance mode is an operational problem, not capacity management matter. |
| **IT Finance** | Typically work with Capacity Team on purchasing. |
| **IT Management** | There can be multiple levels here, all the way to the CIO and CTO.<br>Look at the big picture, trends over time, and future. Not so much what's going on hour-by-hour.<br>Weekly report, focusing on the overall health and not individual users or pools.<br>Monthly presentation and review, supported for live dashboard for an interactive discussion.<br>Generally, does not get involved in troubleshooting and architecture.<br>Primary focus is Compliance and Cost. Performance is not the focus, as that was likely promised to be "good" by the Architect as part of the design. |
| **Architecture** | Design the system architecture. Needs to get input from Day 2 team, and design with the end in mind. |

| | |
|---|---|
| | Look into the future. Evaluate new technology and assess if technology refresh or migration to a new architecture makes business sense.<br>May lend support on complex troubleshooting.<br>In larger organizations:<br>• The storage team is typically separate, unless it's virtual such as vSAN.<br>• OS-specific teams. They are responsible for setting the build standard for each OS. |
| **Network** | The network team is typically separate. This is due to the nature of networks. |

Guess what role is missing?

Yes, Site Reliability Engineer (SRE).

The role focuses on automation. I see them as a variant of platform team, because in operations you should only automate what you can operate. Doing automation without mastery of operations have resulted in many DevOps turned into Dev Oops!

# Exit Criteria | Entry Criteria

Among the roles and departments, there are handovers and boundaries. This is especially important for escalation. What's the Exit Criteria and Entry Criteria?




Take for example the escalation process from Level 1 to Level 2 support.

L1 Exit Criteria defines the set of analysis they have performed on the issue. If it's not in their capability, they have the right to escalate to L2.

| | |
|---|---|
| **L1 Staff** | Perform check following their SOP to ensure the Exit Criteria is satisfied.<br>The guide to analyze is provided by the respective L2 as part of the alert definition. |
| | Add additional information that L2 needs, giving all the context an L2 personnel needs to perform their job. The context is part of the handoff process, meaning Level 2 will not accept the escalation if the necessary details are not provided<br>The list of information can be basic information, such as VM owner. It can also be a set of additional low-level counters or test results that the L1 Team ran but does not have the skills to analyze. It's analogous to seeing a specialist doctor who asks you to bring your test results. |
| | Since L1 is generalist, while L2 is specialist, the L1 staff must escalate to the right Level 2 group. |
| **L2 Staff** | The 2 pieces of information above becomes the Entry Criteria for L2. |

| | |
|---|---|
| | L2 Team automatically accept the issue and now the ball is in |

## "It's on my court"

Each team need provide the standard list of metrics, events, error that shows that the problem is on their court. These items are monitored by their respective team, and alerts are in-place.

The following list examples metrics and events for different team.

| | |
|---|---|
| **Application Team** | Naturally, the actual metric and event are application-specific, and even version-specific. |
| | There is general abnormal behaviour regardless of application. Examples:<br>• Memory that grows over time and does not go down despite loads going down. This indicates memory leak, or application does not release the memory back to Linux or Windows.<br>• CPU on a thread that runs 100% constantly. While high utilization is good, a perfectly flat line over minutes could indicate process run away. |
| **Database Team** | The counter depends on the specific database software |
| **Windows Team** | CPU: Context Switch, Run Queue |
| **Linux Team** | CPU: Context Switch, Run Queue |
| **K8 Team** | K8 Pod: CPU Throttle, Out of Memory event. |
| **Virtualisation Team** | For VMware vSphere VM:<br>• VM CPU: Ready, Co-Stop, Overlap, Other Wati<br>• VM Memory: Latency, Zipped, Swapped, Ballooned<br>• VM Storage<br>• VM Network:<br>• VM System: vMotion down time |
| | For Infrastructure :<br>• ESXi Performance (%)<br>• ESXi Performance Risk (%)<br>• Cluster Performance (%)<br>• Cluster Performance Risk (%) |

# SP5 Framework: Process

Day 0 & Day 1 are typically done by the Architecture team, and Day 2 is typically done by the day-to-day operations team. In large organisation, Day 1 could be different as there is enough workload for them to continuously upgrade. For example, if you have 6000 ESXi hosts, and you have a 5-year depreciation policy, you're basically replacing 100 boxes per month. That means 25 hosts per week on average.

## 3 Categories

### Day 0

| | |
|---|---|
| Plan | This is where you set the goals. The goal should follow the SMART criteria. Make sure they are aligned with the business deliverable.<br>For example, if you plan to build a private cloud, how do you measure success? What are the metrics, so you know how successful you are? |
| Size | A big chunk of planning is sizing the environment.<br>Some companies perform stress tests and load tests, so they know what to expect when the real load occurs. Without planning and testing, you don't know if the reality should be, as you do not have a measured goal.<br>When you architect vSAN, how many milliseconds of disk latency did you have in mind? For example, you set the goal of 10 ms measured at VM level (not at the vSAN level and not at the individual virtual disk level) based on 5-minute average? |

### Day 1

| | |
|---|---|
| Design | This is where you design, build the system, and launch the service. This includes configuring the various operations inputs such as cost drivers (e.g. application license costs, electricity rates). This is not part of the book as there are plenty of materials on it.<br>Day 1 is generally closer to Day 0, as it's the realization of your plan. |
| Build | Ideally this is automated to ensure what is designed is 100% identical to what is built.<br>This includes deployment of the infrastructure, physical and virtual. |
| Test | Ensure what is built works as intended. For example, if you build a vSphere cluster, test that each ESXi host can recover from an HA event. |
| Upgrade | I do not consider them as part of Day 2. |
| Migrate | Upgrade varies from a simple patch to complete rearchitecting, where you are running the old and new system side by side. This type of upgrade is basically a migration, as you have the luxury of a new build. It also involves substantial planning, so you might have to circle back to Day 0. |

## Day 2

Day 2 is when the real challenge happens, as this is the stage you need to ensure critical workload is served well.

As a result, it includes proactive tests of the system resiliency and security. Monitoring is more than just checking. It's about ensuring that the system is able to handle emergencies and remain secured.

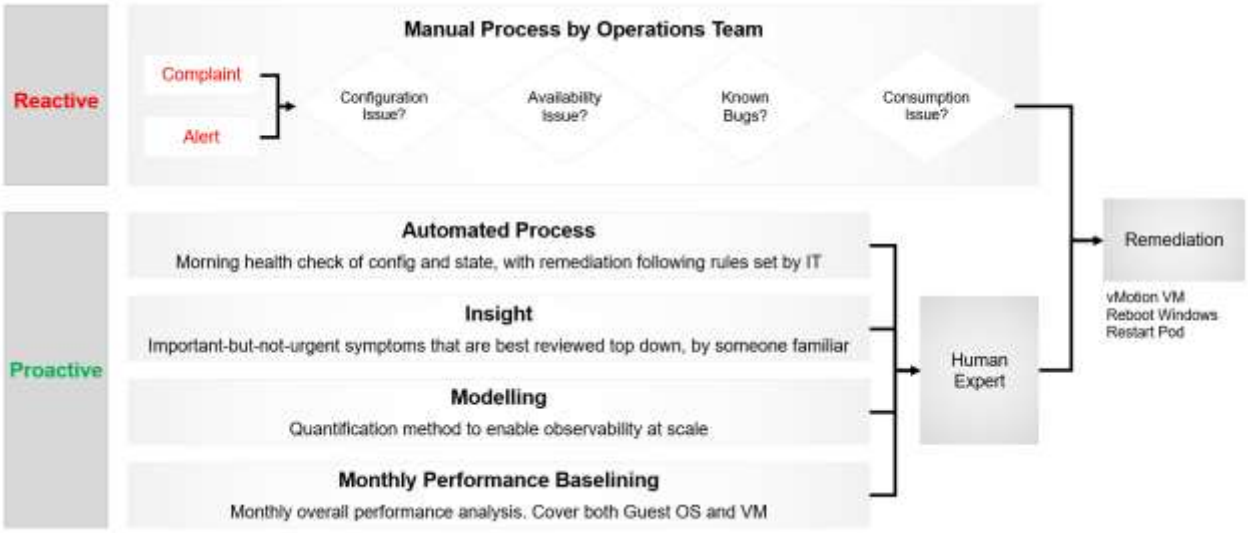
Define your Standard Operating Procedures (SOP) to ensure knowledge is captured and lessons are learned. It's also important to know "who does what when" to ensure gaps are covered. If your operations are partly or fully outsourced, ensure the vendor provides the documentation.

Having SOPs does not mean that you do not have ad-hoc tasks at all. You do, however, want to keep them minimum and so they are manageable.

| | |
|---|---|
| Deploy | This refers to the deployment of workloads, not infrastructure. Workload lifecyle management happens throughout the years, hence it's considered Day 2. |
| Prove | You have redundancy (HA) and DR, right? How do you know they work if you never test them? A pair of core switches that have been left untouched for years may give you a surprise when there is an unplanned HA event. Once a year, test your HA to make sure they work.<br>Annual reboots could be a good idea to clear cache and logs. As part of firmware, driver, OS, and VM Tools updates you may need to reboot anyway. Take advantage of this downtime by testing the resiliency of your infrastructure.<br>Another area you need to test is security. Conduct independent penetration tests. |
| Monitor | These 2 will be elaborated as they form the bulk of Day 2 activities |
| Troubleshoot | |
| Optimize | As part of your monitoring, you may not discover a problem, but you spot an opportunity to make performance even better, reduce costs even further, and free up wastage in capacity. It's common for new versions to deliver performance improvement. Again, you do this proactively, not waiting for complaints to happen. |

# Proactive Operations

While we cannot eliminate troubleshooting, we can perform regular health check + auto remediation to minimize the reactive action.




The preceding diagram shows the dual tracks.

It uses proactive as the primary, while still having reactive as the fall back.

There are 3 techniques we can apply. They must cover all the things you check manually today. Automate your entire check list.

- Daily Optimization.
- Insight review.
- Performance Quantification
- Monthly Baselining

The techniques are based on the fact that there is only a finite set of rules. The logic that human experts use to troubleshoot performance is known. This means it can be automated. The final solution does not even need AI for almost all cases, as the final remediation action is basically moving some VMs out of the cluster. This too can be automated.

This daily check + remediation, coupled with human verification, should free up IT Team to deal with corner cases.

The check must be quick, ~1 minute, so you can do 2x a day (Start of Your Day and End of Your Day).

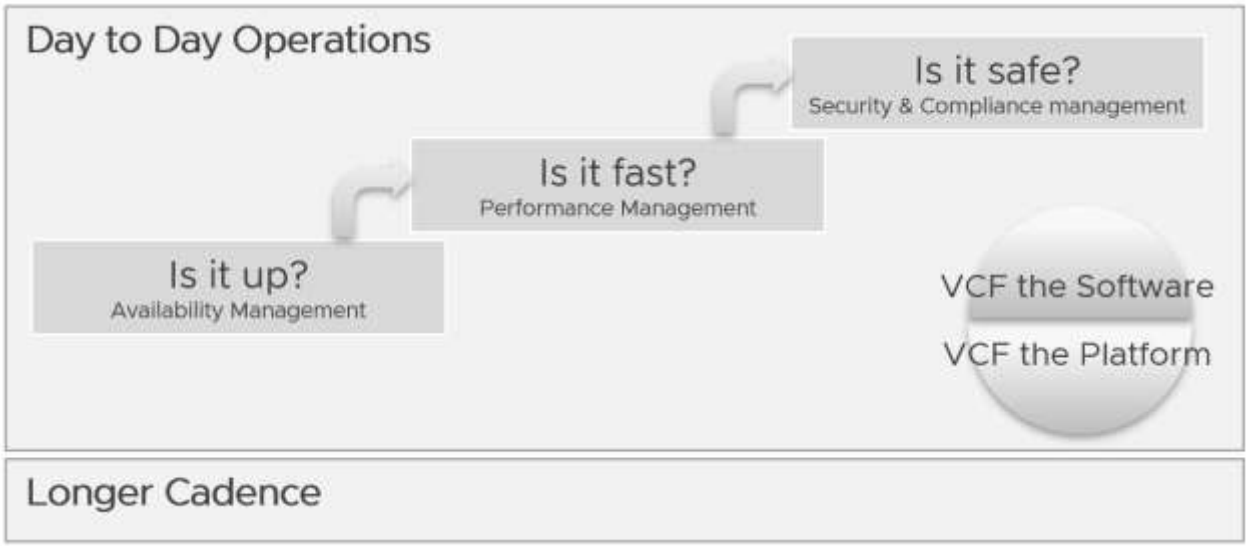
The following diagram shows how Level 1 Team (the frontliner) and supporting team (Level 2, Application Teams, etc.) work together to avoid alert in the first place.




## Day-to-day Operations

The daily discipline of who does what everyday matters.

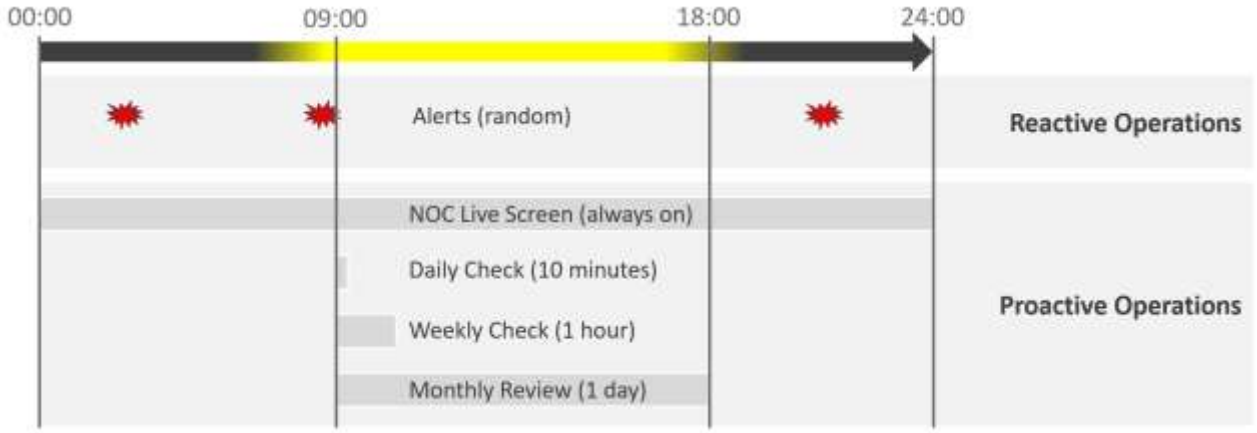
1. The most important and the most basic is availability. If your system is down, you cannot have performance problem.
2. The next thing you ensure is performance. A system that is so slow is as good as down, perhaps to the point the hackers get tired of waiting 😊
3. Yes, security is number 3, not number 1 in the grand scheme of operations.




You assess the above on both "sides" of VCF, as they are intertwined. We covered this in earlier part of this book.

Day to day operations become more systematic when you distinguish between monitoring and troubleshooting.

The following shows the timeline of actions taken during a 24-hour period.




| | |
|---|---|
| 08:00 – 09:00 | Automatic Health Optimization script runs. It flags VCF components needing attention that it cannot remediate. |
| 09:00 – 09:30 | Human Expert reviews as part of Daily SOP.<br>• The script outcome.<br>• The daily dashboard, which also has the alert trend. |
| 09:30 – 10:00 | Having reviewed in totality, humans proactively prep the environment for the expected load of the day. For example, approves inter-cluster rebalancing. |

Live NOC screen and dashboard work together to realize proactive operations. You need both.

| | |
|---|---|
| **Live Screen** | They provide live, real-time, information. They are conveniently projected on the big screens for NOC Team to glance quicky. It is always available, and auto-refresh. |
| | The NOC Room screens enable the L1 Team understands the situation better. Coupled with clear SOPs, it should result in less quantity and higher quality escalation |

| | |
|---|---|
| | They complement daily check well. Since you only check the dashboard in the morning, what happens to the rest of the day? |
| **Daily Check** | They focus on insight, not alert. We covered both in-depth in Part 1 of the book. |
| | Insight and alerts are complimentary. Open both pages side by side. |

They complement alert well. The main limitation of alert is bottom-up. It can't show the complete picture. Alert is the tip of the iceberg.

Why do we add critical business applications in the methodology?

As covered in chapter 1, we need to begin with the end in mind. Since the purpose of IaaS is to run the applications, we need both the application team and the infrastructure team to track the critical applications. I put 2x daily as the minimum.

Proactive operations should result in less alerts, both in frequency and intensity. Volume wise, the bulk of alerts are the yellow alerts, due to their lower threshold. They can be replaced with insight, which is discovered as part of a daily proactive health check dashboard.

| | |
|---|---|
| **Alerts** | A better name for alert is guard rail. This way, the purpose is clear. It is to catch when human forgets, or system fails without warning.<br>It is not the starting point of operations, where operators passively waiting for alerts to happen then do something. |
| | A well-run operations have a low number of alerts. |
| | The nature of alert means its use case is narrow. You do not want to run your operations based on alerts. Too many and you're overwhelmed. Too few and you lack the early warning.<br>This is why insight is the main driver, with alert playing secondary role. |

What's the limitation of the above?

They focus on day-to-day operations. Both have short timeline. Maximum 1 day. If you have quarterly workload, you will not see them.

You also need a deeper and longer analysis, to implement major improvement. You do this via a set of interactive and powerful dashboards, such as the performance baseline profiling dashboard. Because you do it monthly or longer, you can afford a longer time to execute it. Use it to solve larger, more complex, and political problems.

Is there another benefit of this "out of band" operations management?

Yes. Use it to improve your daily operations itself. Both the Daily Check and the NOC Screens should help in preventing the alerts in the first place. However, you may have alerts that fall through the crack. On a weekly basis, review these alerts and improve the dashboards accordingly. Once this is repeated over a long period of time, you're bound to catch many of the alerts and get better in preventing them.

## Real Time

This is assisted with a set of live dashboards on the big screen in the NOC, showing real-time metrics and events. As the time duration is basically minutes, capacity is typically not relevant. You focus on availability, performance, and security.

The focus is to gain early warning, hence buying time if operations turn into an emergency. That means there is no alert triggered yet.

You want to put out fire fast.

| | |
|---|---|
| **Availability** | Focus on undesirable events. Make sure they match your expectations of what is actually happening in your environment at that time. For example, if you're not deleting VMs and the live screen shows many VMs being deleted, something's amiss. |
| **Performance** | Focus on overall performance and make sure the fluctuation is within your expectation. |
| **Security** | Focus on suspicious activity such as loosening of firewall rules or usage of administrator accounts. |

## Daily

The daily SOP should look ahead, using the historical data and business context as a guide.

| | |
|---|---|
| **Availability** | Check for soft errors. This likely requires you to look at the logs from each hardware and software vendor. For example, vSphere has proactive HA. |
| | Ensure backups are performed. For example, if you take a snapshot of 1549 VMs and only removed 1518, the backup likely failed to consolidate the snapshot. |
| **Performance** | Insight analysis . Look at the overall environment.<br>Mission Critical VM Performance. Focus on looking forward to the next 1 day, by considering the pattern over the last 1 week. Know the expected workload, by working closely with the business owner. |
| **Compliance** | Ensure non-compliance is remediated. |
| **Security** | Check for potential security attacks, such as loosening of firewall rules and usage of administrator accounts. |
| **Configuration** | Check for misconfiguration that can cause issues. |
| | Implement urgent patches, after appropriate validation |
| **Capacity** | VM provisioning. Ideally this is automated, so it happens throughout the day. In this way, your developers need not wait for the end of the day. What you need to ensure is the aggregate load does not overwhelm the shared environment. |
| | Snapshot deletion. Set up a policy to delete snapshot older than say 3 days. You do not want to wait until 1 week.<br>Ensure the exclusion is properly implemented, so you do not delete from VM whose snapshot you've agreed to keep. |
| | Powered-off VM deletion. Once a VM passes the powered-off definition, there is no need to wait for another week before you delete them. You do have backups, don't you? |
| | Powering off Idle VMs. Once a VM passes the idle definition, there is no need to wait for another week before you gracefully shut the OS down. |

## Optimize

The outcome of monitoring is not always troubleshooting. You may discover nothing to fix. However, that does not mean you discover nothing as you may discover the opportunity to improve.

Optimization delivers many practical benefits and real business results. Here are some of them:

| | |
|---|---|
| Lower Cost | Reclamation: Orphaned VMs, powered off VMs, Idle VMs, Oversized VMs, snapshots.<br>Reduce DC footprint: Saves Software (MS, RedHat, VMW, etc.) and Hardware (server, storage, network) + Data Center (rack, space, cooling, UPS)<br>Move burst capacity from Own to On-Demand. |
| Better Performance | Performance Profiling. Enable proactive monitoring via actual baselines.<br>Establish Performance SLAs, complementing Availability SLAs.<br>NOC Dashboards. Insight, then Alerts.<br>Faster business service via self-service + approval workflows. |
| Lower Complexity | Standardize architecture.<br>Simplify business policy or security policy.<br>Standard Operating Procedures (SOP).<br>Reduce human error via automation.<br>Replace ageing hardware and upgrade outdated software. |
| Higher Customer Satisfaction | Internal IT department. Reputation among Applications Team.<br>External SP. Repeat business.<br>Ability to justify or defend pricing. Price/Performance. |
| Higher Compliance | Internal compliance with evolving, mandatory and/or optional security benchmarks, such as VMware security hardening guides, CIS, ISO, or FISMA baselines, or industry regulation such as DISA, PCI DSS, or HIPAA. |

I'm sure there are more of them. Drop me a note with your real-world stories!

## VM Life Cycle

This should be supported with an approval system, so all Change Requests and associated actions are properly recorded. This eliminates finger-pointing in the future. It will also support audit: who is keen on "who did what change to which object on when"?

| Stage | Notes |
|---|---|
| Request | If a VM is free (price is basically $0), then rely on business justification and IT policy. Policy states the criteria for the different class of services.<br>VM size is requested by the application team, approved by their management. For size or quantity above certain thresholds, IT should review. |
| Creation | The actual deployment of the VM in vCenter. This is ideally automated.<br>This stage generates the actual VM name, create folders if necessary, and places the VM into the correct folder.<br>Once the VM object is registered in VCF Operations, create the custom group is necessary, and set the custom property. |

| | |
|---|---|
| Changes | Changes in VM size need to be approved as it impacts capacity and performance. If you have to use shares and reservation, ensure they are updated accordingly. |
| Retire | Delete the VM and remove it from inventory. |

# Performance | Capacity

Think of it as Quantity and Quality. Or Space and Speed.

They are heavily intertwined.

- Rightsizing belongs under capacity, but it uses performance as the primary consideration. For mission critical, the metric is highly granular (1 – 20 second), not 5 minutes average.
- Infrastructure capacity is about maximizing utilization, but it gets overridden by performance.

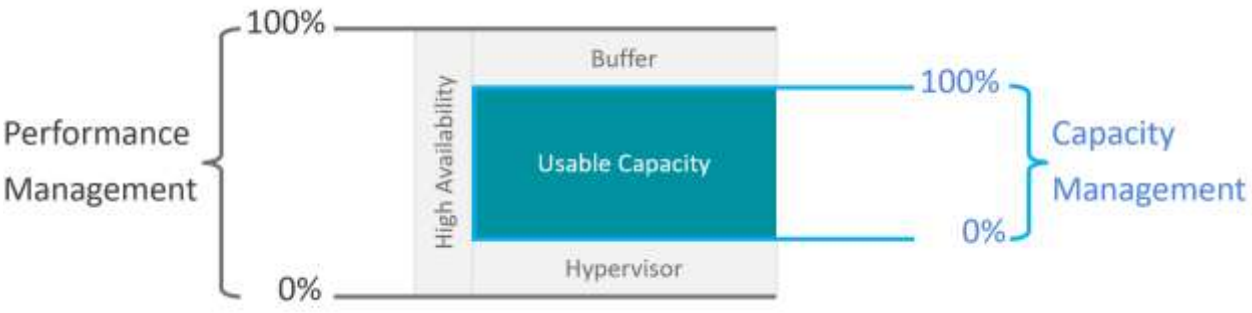
Performance is more time sensitive and important than capacity. Manage performance first, capacity second. Using the restaurant analogy, you focus on the dining area first, then the kitchen.

In larger organizations, they are typically managed by two different teams. The capacity team does not get involved in the day-to-day operations as they focus on longer-term resource availability. They also consider latent workload and future demand, which performance does not consider.

The capacity team may not have the technical skills to troubleshoot performance. On the other hand, the day-to-day operations deals with "what's on the floor" of the data center. Their primary focus is meeting the demand from applications at that hour.

Capacity involves factors like HA, Buffer, Overhead and Reservation. None of these are relevant to performance monitoring. In Performance, you don't care about them as performance is about reality (what actually happens). Those factors may cause performance problems, but they are not considered in the performance metric.

Capacity uses a smaller subset of the resource than performance. One main concept is Usable Capacity, which is unique to capacity. There are no such things as usable performance, usable compliance and usable availability.




The relationship between capacity and performance varies depending on the object.

- Consumer objects (e.g. VM, K8S Pod) have different natures than provider objects (e.g. vSphere Cluster, vSAN Cluster).

- For provider objects, performance is always bottom up. You start with the VM running inside in the provider object, and then aggregate the metrics. Capacity is always top down. You look at the big picture first, then drill down. For example, you start with the vSphere cluster, then drill down to ESXi.

For an IaaS provider, the following table explains how performance and capacity differ.

| | Cluster Performance | Cluster Capacity |
|---|---|---|
| Focus | VM<br>Are the VMs being served well? | ESXi<br>Are the ESXi running hot? |
| Time Duration | 1 second – 1 hour | 1 day – 1 year |
| Counter | Contention<br>Utilization is for troubleshooting, not monitoring. | Consumption<br>Drive it high, so long there is no contention. |
| Availability SLA | Does not take into account.<br>Not relevant | Takes into account.<br>Gold Class is in fact Availability-driven. |
| Utilization | Measured against total capacity | Measured against usable capacity |

# Utilization vs Demand

Utilization is not something you manage. It is just an input to what you actually care, which is capacity and performance. The nuance is both use utilization differently. In addition, capacity uses demand metrics, which takes the highest of utilization & reservation.

Performance will be absolute (real value), Capacity will be relative (it depends on settings). Unlike performance, Capacity is measured against usable capacity, not absolute capacity. There is no such thing as usable performance.

| Performance | Capacity |
|---|---|
| 100% utilization is not only good, it's perfect.<br>So long there is no queue, drive this high | 100% demand is bad, especially it breaches Usable Capacity |
| Low utilization may mean bad performance. Check the queue | Low demand always mean good capacity, but cost is likely bad |
| Look at actual utilization. Ignore Buffer and HA. | Look at relative demand. Include Buffer and HA. |
| Use 20-second peak interval, together it 5-minute average. | Use hourly or daily average.<br>Consider performance separately. |
| Look at 1 day time period | Look at 3 – 6 months |

Now that we've looked at purpose, now let's look at object.

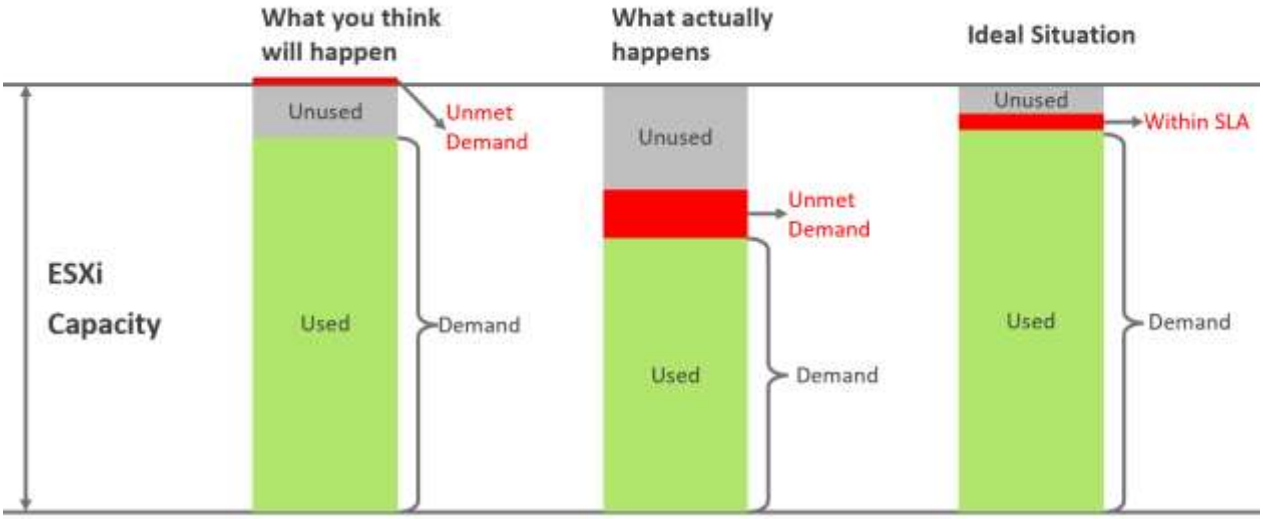
Take a 16-node vSphere cluster, for example:

- For performance, taking the average utilization of 16 hosts is too late. It's also not practical, as you don't typically wait until all 16 have a problem. In this case, you want to take the highest among the host as your primary counter for cluster utilization. If the counters show no issue, then there is no need to look at the remaining hosts.
- From capacity, taking the average makes sense, as you do capacity at cluster level. You will continue adding until either you run out of capacity or you hit performance problems.

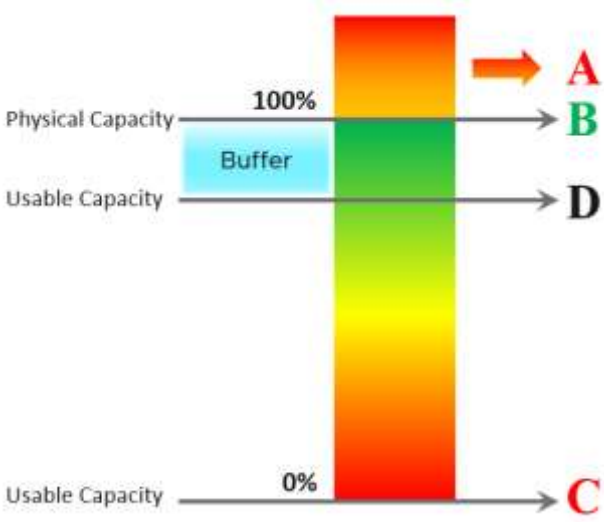
# Contention vs Consumption

The following diagram shows 3 different scenarios on how contention and consumption can play out:

- What you think will happen: you theorize that contention will only happen when utilization is high, and the unused capacity acts as cushion to prevent unmet demand from happening. This is unlikely as there could be imbalance.
- What actually happens in most environments: demand is unmet even though utilization is not high, due to suboptimal configuration or constraints. Imbalances and incorrect cluster configurations are two typical causes of contention at low utilization.
- What would happen if your environment is optimized: you have very high utilization yet you keep unmet demand within the promised SLA.




Don't confuse "ultra-high" utilization indicators as a performance problem. High utilization does not compromise performance, so long as there is no queue or contention. Just because an ESXi Host is experiencing ballooning, compression, and swapping does not mean your VM has memory performance problems.

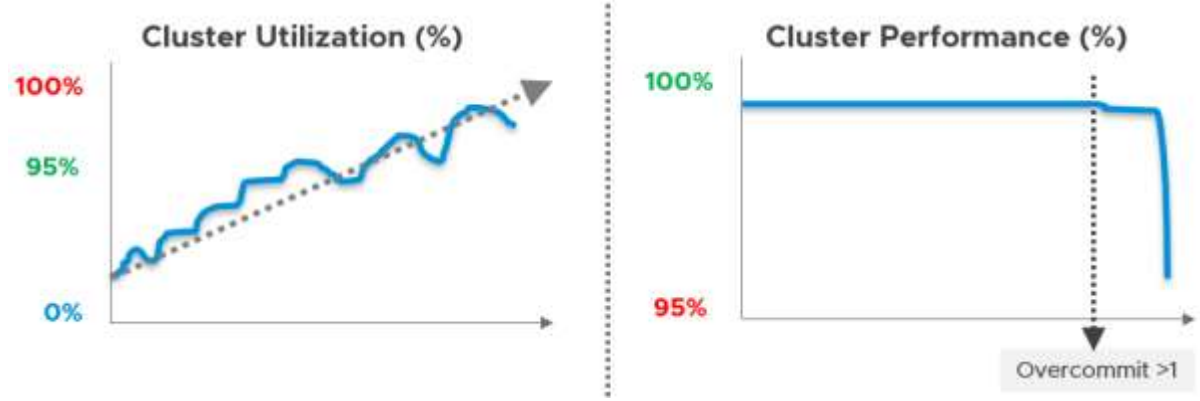



| | |
|---|---|
| **A** | It's a common misperception that performance problems happen mostly here. It actually rarely happens here as utilization rarely exceeds 100% in real world due to buffer created by high availability. |
| **B** | Maximum utilization is achieved at 100% utilization. Consequently, the overall Performance is best here as the system completes the most amount of work. |
| **C** | Worst performance actually happens when utilization = 0% as nothing gets done. The demand is not being met at all (for whatever reason). |
| **D** | This is the threshold of Usable Capacity. It has nothing to do with Performance. Performance is in fact better above this threshold. |

# Pattern Difference

You can't forecast performance using capacity. Their metrics have different patterns.

Let's take an example to see how contention and utilization differ. The following is using a cluster object as the example. There are two metrics, each expressed in percentage.




You want your utilization to be as high as possible, as you've paid for the hardware already. So, you start from 0% but want to move up as far as possible.

Performance is different as it depends on the class of service. Your Gold Class should deliver higher performance than your free tier else you may breach your SLA. Metrics wise, there are 2 metrics: SLA and KPI.

# Performance Management

## Part 1 Chapter 2

The goal of performance management is to address problem before customers complain. If you can't detect the problem before customers do, then you don't have performance management in place. The word manage implies proactive. Passively waiting for an alert or complaint to trigger the troubleshooting process is not performance management. Proactivity requires regular cadence to prevent the problem. This cadence requires an internal threshold that is more stringent than the external, formally agreed SLA.

# A Day in the Life of a Cloud Admin

Here is a common story often told in the virtualization community, which will resonate with you as an IaaS provider.

A VM Owner complains to you that her VM is slow. It was *not* slow yesterday. Her application architect and lead developer have verified that:

- The VM CPU and RAM utilization did not increase and are within a healthy range.
- The application team has verified that CPU Run Queue is also in the healthy range.
- The disk latency is good. It is below 5 milliseconds.
- There are no network packets loss.
- No change in the application settings. In fact, the application has not had any changes in the past month.
- No recent patches were installed into Windows.
- There was no reboot. It has been running fine for weeks prior to this issue.

She said your VMware environment is a shared environment, and perhaps an increase in the number of VMs and an increase in the workload of other VMs are straining your IaaS.

She also said that her other VM, which was P2V recently, was performing much faster in physical.

If you think she is saying it's your fault, you are right!

What do you do?

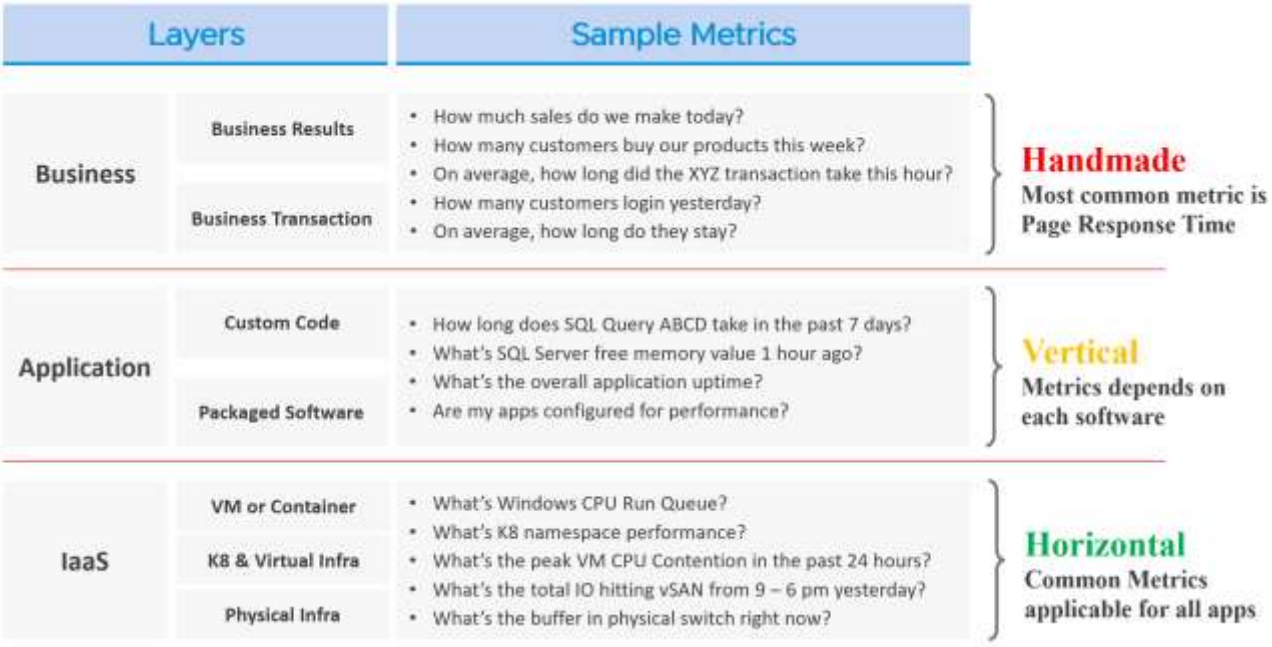
It is certainly a difficult situation to be in. You oversee more than 10,000 VMs. You have successfully consolidated them into 500 ESXi Hosts, saving the company 9500 servers, not to mention a lot of money. You built your reputation during the process, so this is not just a matter of her VM not performing. Your reputation is at stake here.

You also recall that your team has been adding new VMs regularly over the past several months so she could be right about the increasing number of VMs straining the IaaS platform. In addition, there have been several soft errors in your network and storage, and your team has been investigating for weeks. But why did she say it only happened today, but not yesterday or a few days ago?

It's a hard question. To answer it, we need to take a step back and elaborate. Let's dive in!

# The 3 Realms

In the big picture, there are 3 realms of IT in business. Each realm has its own set of teams. Each team has a set of unique responsibility and hence skills required. The following diagram outlines the 3 realms, alongside with typical layers within each realm and questions being asked.




The fight typically happens between application team and infrastructure team, as both are IT. The business folks are real consumers or end users and not technical enough to provide solutions.

By the way, for those who use enterprise architecture framework, the 3 realms above actually map to TOGAF[9] by The Open Group.

- Business maps to Business Architecture
- Application maps to Information Systems Architecture
- IaaS maps to Technology Architecture

Each team need to have their own KPIs, and there must be SLAs between them.

| Layer | Owner | Observability |
|---|---|---|
| Business | Non-IT<br>(specific departments owning the business functions, such as Internet Banking) | Service Response Time. This is business-function specific. For example, if it's an eCommerce website, a good example of function is the payment for a shopping cart. |
| Application | IT (Application Team) | It depends on the application software and codes. Typically requires agents with specific knowledge of that application, as it has unique metrics. As a result, it is not possible to develop a generic KPI. |

[9] I'm TOGAF Certified, but it's a long time ago in a galaxy far away. I believe in the principle of enterprise architecture, but not its heavy implementation.

| | | |
|---|---|---|
| Infrastructure | IT (Infrastructure Team) | It's possible to develop a universal KPI that works on all applications as it's basically about CPU, memory, disk, and network.<br>This is the focus of our work. |

While businesses run on IT, not all business KPIs depend on IT. Some depends on marketing, pricing strategy, and what your competitors are doing. Take an ecommerce portal: its top business KPIs probably involve revenue and gross profit. As you can imagine, if your competitors are doing a massive discount, you may not be able to achieve your sales and profit target.

That means there are 2 types of business KPIs:

- Metrics based on IT software.
- Metrics not based on IT software.

Examples for the ones relying on IT are business transactions (e.g. buy a product, update personal information, transfer fund) that are provided by the IT system. They can be online (real time, a user is waiting) or batch (runs in the background) in nature. They can be user to system, or system to system. Online transactions are often measured in time taken to process a single transaction (typically less than a few seconds) and is called Online Transaction Processing (OLTP). Batch transactions are often measured in time taken to process many transactions (which can be hours).

As you can imagine, different transactions require different amount of time. Even the same kind of transaction (e.g. generating a report), can vary as it depends on the amount of data or records. As an application architect, it's important to define the expected time taken to complete commonly used business transactions. Obscured business transactions that are rarely used can be given low priority. However, this effort is not scalable if you have hundreds of business applications.

If you want universal metrics for all business applications, then page response time and error returns are the only metrics. I like the principle behind apdex, but not its implementation. Happy to share an improved formula if you are interested.

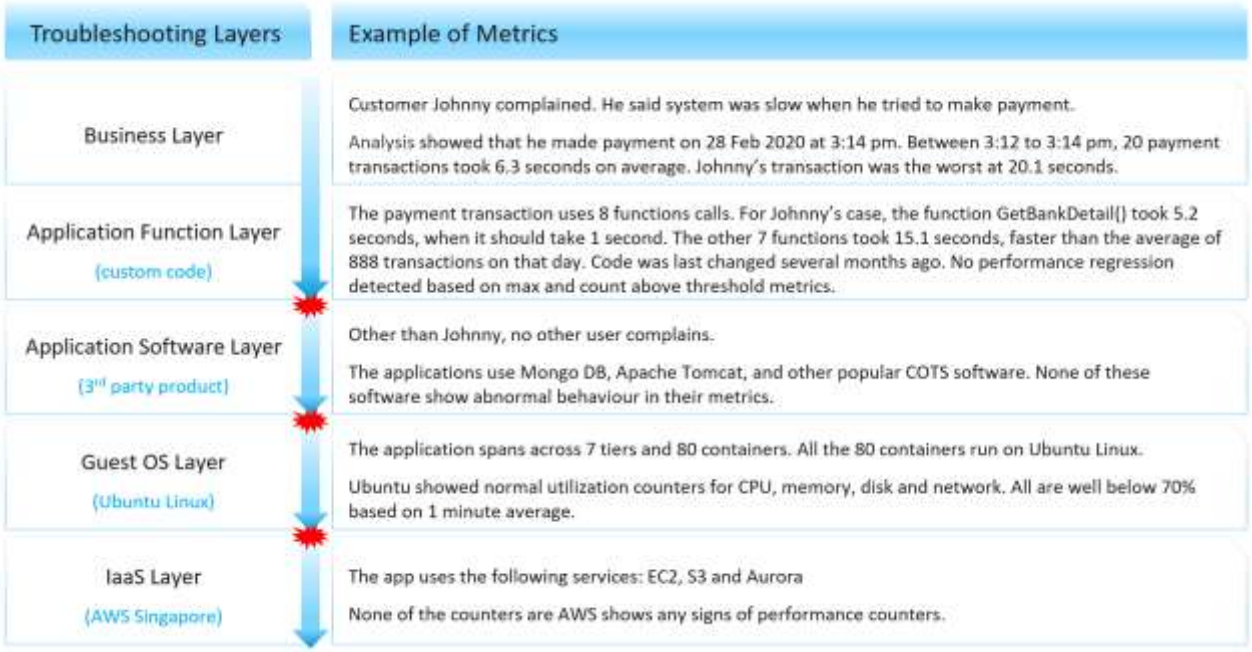
Performance troubleshooting is largely an exercise in elimination. The methodology slices each layer and determines if that layer is causing the performance problem. Hence it is imperative to have a single metric to indicate if a particular layer is performing or not. This primary metric is aptly named Key Performance Indicator.

Higher layers depend on the layers below, and hence the infrastructure layer is typically the source of contention. As a result, focus on the bottom layer first, as it serves as the foundation for the layer above. The good part is this layer is typically a horizontal layer, providing a set of generic infrastructure services, regardless of what business applications are running on it.

Now, we don't know the impact to the application when there is latency in the infrastructure. That depends on the application. Even on the same identical software, e.g. SQL Server 2019, the impact may differ as it depends on how you use that software. Different natures of business workload (e.g. batch vs OLTP) get impacted differently even on the identical version of the software.

# The Layers in IT Systems

To continue the discussion of layers in IT systems, the following example of an IT system shows 5 layers. The challenge in performance troubleshooting is the layers may not share context. Depending on the application and infrastructure architecture, there could be more layers.




Using the example above, we can demonstrate how the lack of visibility is making troubleshooting virtually impossible. Let's run through the hypothetical example depicted above. The story starts with a complaint as that resonates better.

At the Business Layer, you can see the performance of each business transaction. You not only know which user was affected, you also know what transaction was affected as the metric has transaction ID. You can trace it in the code as you know how long each function calls take place, assuming you log for every single transaction.

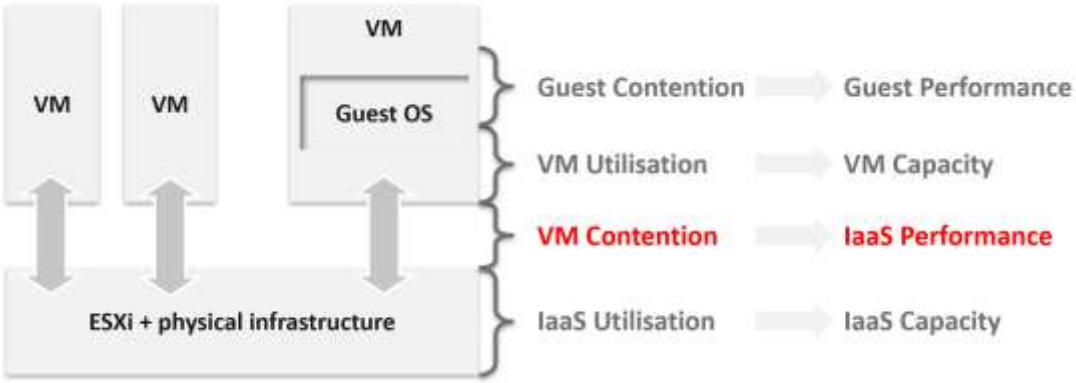
The problem starts when you move beyond your code and into Commercial of the Shelf (COTS) software. The software may show that its queue is 10000, which is 5000 more than what the manual say it can handle. But you have no idea if the user's transaction was in that queue or not. The COTS software metrics do not relate to users anymore, let alone individual transactions. The red explosive icon marks where context is lost.

Moving from application to infrastructure resulted in another loss of context. Windows or Linux has no idea what applications you're running. As far as the OS is concerned, every application is just a process. It will report basic CPU, Memory, Disk and Network utilization per process. More advanced metrics are reported at OS-level, system wide. For example, you do not know if your process was the one experiencing network packet loss. The packet loss metric is a system-wide metric.

Moving from individual EC2 or VM to the shared infrastructure results in another loss of context. In the case of public cloud, you may not get visibility into the physical layers at all. Your troubleshooting ends at the consumer layer.

# The Layers in IaaS

Let's apply the layers to IaaS.




The 1st level is Guest OS contention. This is beyond the control of the IaaS platform. It's completely up to each Guest OS to manage their given resources.

The 2nd level is useful for tracking VM capacity. Say a VM is given 8 vCPU. If you want to know if it's using it, this is the area where you measure it. Whether you measure it In-guest or at the VM level, it depends on the specific use case because each layer lacks visibility into the other.

The 3rd level is where the VM meets the provider. This is where you track whether the IaaS platform is serving the VM well.

The 4th level is the underlying IaaS platform. This level is irrelevant to the VM Owner. This is normally the domain of the capacity planners and the troubleshooting specialists.

Let's drill down further, breaking the layers more.

## IaaS Metric Types

Broadly speaking, there are two categories of metrics we are interested in: consumer metrics, and provider metrics.

Consumer metrics consist of:

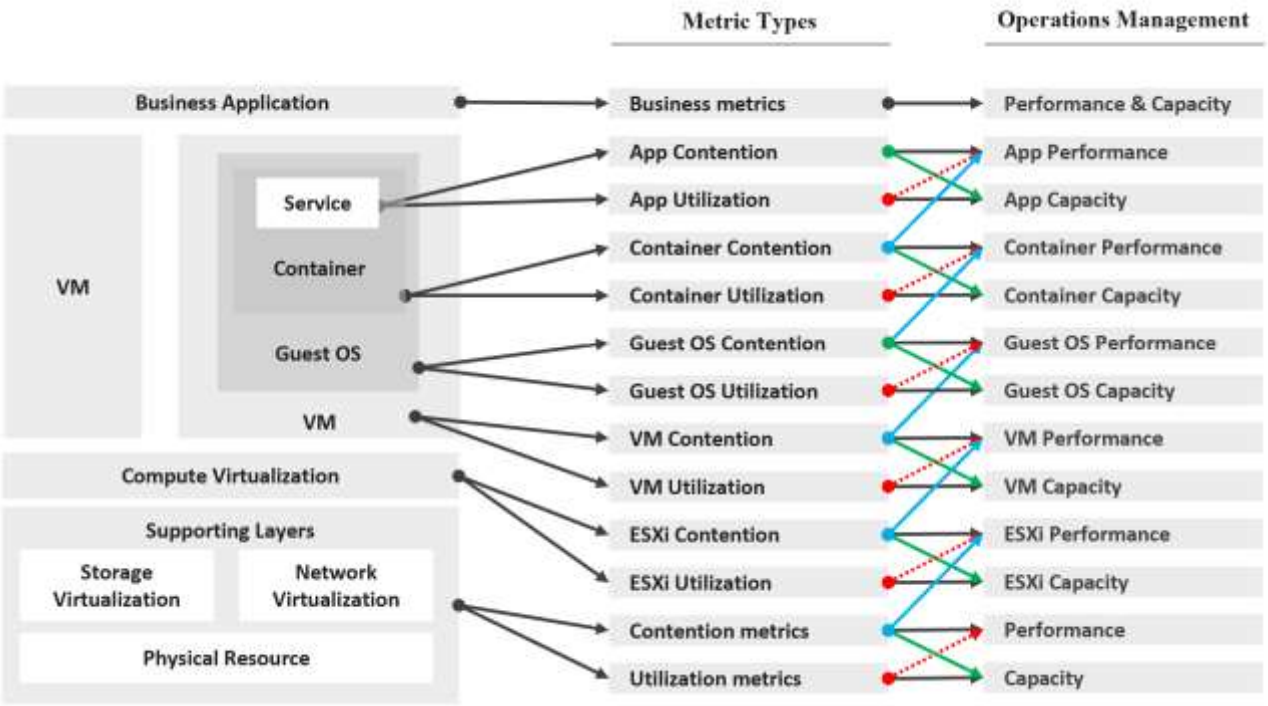
- Business Applications: this could span multiple VMs, containers, and serverless functions. The metrics here are business metrics, not IT metrics. They are the best at measuring the application performance, but they can't explain why it's not performing,
- Service: this is a process running inside Windows or Linux. We normally call this the application. An example is a database server.
- Guest OS: there are two kinds of performance metrics; One at the Windows/Linux layer, and one at the driver layer (e.g. PVSCSI, vmxnet).
- Container: this typically runs inside a VM. If there is more than 1 container in a VM, it makes operations management harder.
- The Virtual Machines themselves.

Provider metrics consist of:

- Compute virtualization, also known as the hypervisor. This is where the VM or container is running.

- Storage and Network virtualization: they act as subsystem, supporting layer to the hypervisor. Ideally, they should have VM-level metrics, so you can trace how a single VM is performing deep into the storage and networking stack.
- Physical resources: with hyperconverged, the only physical resources which are not already included in the categories above are basically network equipment due to their function as interconnects.

The following diagram captures the interrelationships between these metrics.




Notice how consumer and provider metrics are intertwined. A consumer contention metric becomes the performance metric for the provider.

Contention metrics are placed above utilization metrics as that's what you should drive your operations. As Mark Achtemichuk said in this article, "drive by contention". For each layer, you have a set of metrics. The black line indicates that contention is the primary counter for performance, and utilization is the primary metric for capacity.

| | |
|---|---|
| **Green lines** | show that contention metrics give valuable input to capacity metrics by showing how much additional capacity is required. For example, the number of queues in the CPU should be used to determine the amount of CPU to add. |
| **Blue lines** | show that contention in underlying layers directly impact performance in the layer above. For example, if a Guest OS experiences disk latency, the application will feel the impact. That can result in a ripple effect to the top layer. |
| **Red lines** | are not solid, as they're highlighting a misconception. If contention = 0, then utilization at 100% is in fact maximum performance. If contention is not measurable, then add buffer to utilization as queue tends to develop at high utilization. |

On the other hand, you can have poor performance at low utilization. Many things can cause this as there are many possible configuration errors.

# Optimized Performance

Optimized performance is difficult because the best performance is achieved when utilization/throughput is at 100%. This is when the most work is being done by the system overall. Running at that level requires a perfect level of mastery due to many dimensions of inter-dependencies. In addition, majority of loads have peaks so on average you could be well below 100% during idle periods.

| Type of Dependency | Description |
|---|---|
| **Vertical** | There are layers in the stack, and a problem in a lower layer can impact an upper layer. |
| **Horizontal** | The four elements of IaaS are not standalone. When CPU is paused, RAM & Disk will experience latency as time shifts as far as the Guest OS is concerned. |
| **Flow** | A problem in your NSX Edge VM on the NSX Edge Cluster can impact a business VM sitting on another cluster, because of the traffic flow. If you don't understand the flow, you can waste time troubleshooting at the wrong place. |
| **Version** | There are valid reasons behind "What Works With What". It's a known problem that not all versions of all components work well together. Drivers, Firmware, etc. can cause interoperability problem, which can manifest itself as performance. |

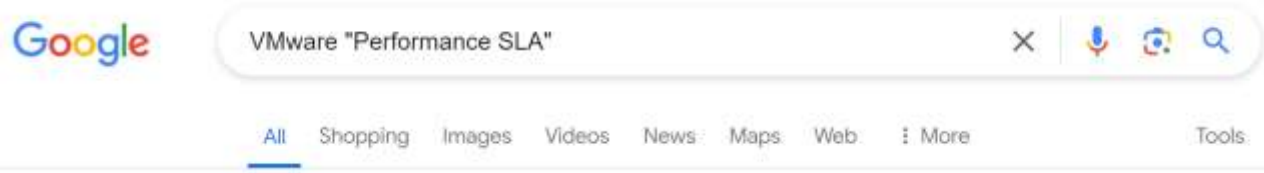
# Key Performance Indicator

In this book, my definition for KPI is strictly on *performance*, because the word *performance* has a specific meaning in enterprise IT. To me, KPI as a term does not apply to availability management and compliance management. We should call the key indicators that determine availability as **KAI**, and the key indicators that determine compliance as **KCI**[10]. This prevents confusion as implementation-level solution requires us to be non-ambiguous with terminology. For a general overview of KPIs, Norman Dee has written a series of blog post starting here.

We've covered SLAs in-depth. They are complex to operationalize, especially the performance SLA.

Google VMware "performance SLA", and you will find only a few relevant articles. The string performance SLA must be within a quote, as it is not "performance" and "SLA", but "Performance SLA". Yes, I'm after web pages with the words Performance SLA together. You will get many irrelevant results if you simply google VMware Performance SLA without the quotes.

[10] You see, I'm not allergic to jargons so long they are actually needed in live operations. If it's just an objective, then no need to label it with jargon, else we have to create jargon for every single type of goals.




I checked the first few dozen results. Other than my own articles, Google returned only a handful of relevant articles. The rest were not in fact relevant once you read them carefully. The relevant articles did mention Performance SLA, but did not define and quantify what a Performance SLA is. If something is not quantified, it is subjective. It's hard to reach formal agreement with customers quickly and consistently when the line is not clearly drawn. If you have a disagreement with your customers, especially paying customers, guess who wins 😊

# KPIs as a Stepping Stone

As an SLA is hard to vrealize, what can you do today?

You walk from where you stand. Adopt KPIs first as you don't have to worry about classes of service. If you have different expectations set for Production and Non-Production VMs, then create a group for each. Create a super metric that averages the performance of each group. You should expect production VMs to have a better KPIs overall.

KPIs complement SLAs by providing the stepping-stone in your operations transformation. It is a necessary step towards operations with formal business SLAs.




| | |
|---|---|
| **Complaint** | Metrics based on user complaints.<br>Blamestorming among teams. Politics come into play.<br>Alerts-driven, not insight driven. Reactive and firefighting. |
| **KPI** | Using both 20-second average and 5-minute average.<br>Performance for both Consumer and Provider objects are quantified & measured.<br>KPI includes Guest OS, if IT owns VM sizing. |

| | |
|---|---|
| | 1 common policy for all. No formal class of service defined. Performance is based on production environment.<br>Insight based, with SOP.<br>RCA is closed with alerts set up.<br>Customers can track the SLA via self-service portal. |
| **SLA** | SLA is measured on 5-minute average.<br>SLA = 1 month average of SLI of that specific month.<br>SLA is part of business contract.<br>1 policy for each Class of Service.<br>Pricing & QoS are used to differentiate each class. |

# KPI | SLA

KPIs and SLAs work hand in hand.

<table>
<tr><th>Layer</th><th>Owner</th><th>SLA (%)</th><th colspan="2">KPI (%)</th></tr>
<tr><td rowspan="2">Business</td><td rowspan="2">LOB<br>(non-IT)</td><td rowspan="2"></td><td colspan="2">Business metrics, such as transaction volume and concurrent customers.</td></tr>
<tr><td colspan="2">IT metrics, such as page response time & error rate</td></tr>
<tr><td>Application</td><td>IT Developers</td><td rowspan="3">App SLA:<br>Availability<br>Performance<br>Compliance<br><br>IaaS SLA:<br>Availability<br>Performance<br>Compliance</td><td colspan="2">2 types: custom codes and packaged ISV products.<br>Examples: Web Server KPI, App Server KPI<br>Database KPI.</td></tr>
<tr><td rowspan="2">Infrastructure</td><td rowspan="2">IT<br>Administrators</td><td>Consumer Layer</td><td>VM KPI (%), K8 Pod KPI (%)</td></tr>
<tr><td>Provider Layer</td><td>vSphere Cluster KPI (%)<br>vSAN KPI, NSX Edge KPI</td></tr>
</table>

KPIs are also better suited for when the infrastructure team takes ownership (and responsibility) for sizing the VM. Since sizing the VM basically means sizing the Guest OS resource, you're now responsible for metrics at Windows and Linux level.

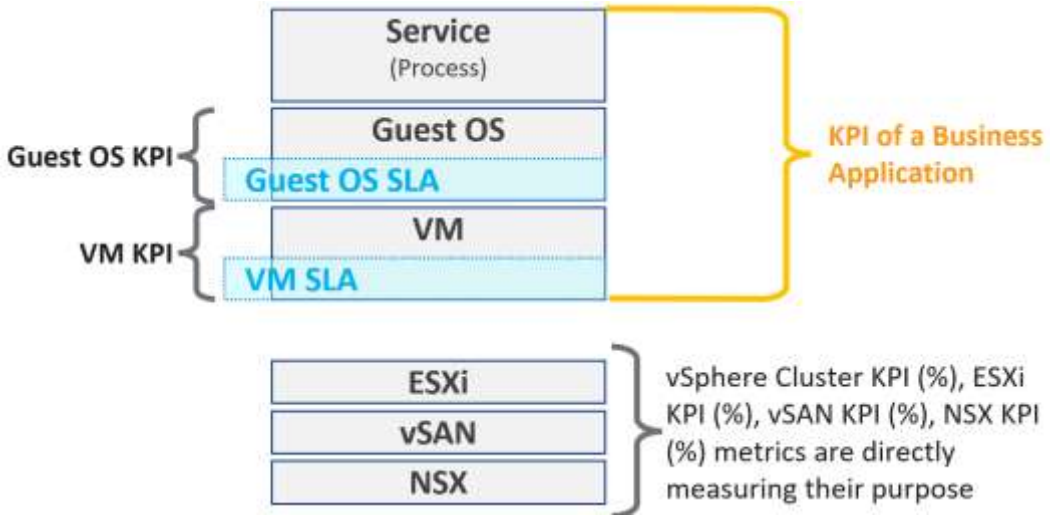
Regardless of SLAs, the main reasons for creating a KPI metric is ease of monitoring. A KPI is color coded. Metrics can be color coded like traffic lights (e.g. green, yellow, orange, red), so you can understand their indications and react faster. The goal is in fact to enable proactive remediation, before the situation degrades too far.

## Differences

| KPI | SLA |
|---|---|
| A single metric that quantifies the true performance of a logical entity. | A set of metrics written in the business contract between the service provider and service consumer. Typically, this is between the IaaS provider (the infrastructure team) and the IaaS consumer (the application team or business unit). |
| VM and Guest OS are 2 separate objects, but they are 1 logical entity due to 1:1 permanent relationship, hence they are combined in 1 KPI number. | What happens inside the Guest OS is not even relevant to the VM SLA. |
| Comprehensive coverage.<br>Covers the SLA metrics, plus relevant additional metrics that provide early warning before the SLA metrics are breached.<br>There are many KPI for a given SLA because not all metrics should be in the contract, while almost all performance metrics need to be monitored. | Limited.<br>Only important metrics are included. Having too many metrics in SLA makes it harder to comply.<br>Guest OS metrics should not be included as that's part of "application KPI" or VM KPI, not IaaS SLA. They impact the VM performance, but nothing the IaaS can do, meaning the remediation is at the Guest OS layer. |
| Tend to be lower in value.<br>A value of 75% is acceptable. | Tend to be near 100%. A typical SLA is > 99% |
| Simple to operationalize. | Complex to operationalize.<br>It needs **Operations Transformation**, much more than technical changes. You need to look at contract, price (not just cost), process, people, class of service, etc. |
| It tends to be absolute, as it's reporting raw metrics | It is always relative, compared to an agreed threshold. |
| Leading Indicator.<br>5-minute window. They are used as the starting point to troubleshoot, before user complaint | Lagging Indicator.<br>30 days backward looking, to be precise. You complement it with SLI, a 5-minute tracking indicator.<br>If you breach, you're talking penalty already. Typically it's a credit for the next billing cycle as opposed to actual refund. |
| 20-second accuracy.<br>It tracks at higher intensity, and it covers more metrics and events. | 5-minute accuracy. |
| Based on day (e.g. last 1 hour), which is moving as time passes. | Based on date (e.g. June 1972), which is a fixed period. |

# KPIs in IaaS

Within the context of VM as a Service, here are the KPIs and SLA.




Why are there more KPIs than SLAs?

There is only 1 SLA per service.

| | |
|---|---|
| **VM KPI** | It excludes the application or service running inside Guest OS as infrastructure team stops at Windows or Linux. |
| **Business Application KPI** | There are actually 3 levels of KPI. The highest and most important tracks the business transaction (e.g. user login). If the number is below expectation, you check for application software KPI (e.g. Apache web server) and infrastructure KPI (which is basically the VM KPI) |
| **vSphere Cluster KPI** | It includes only the contention metrics of the running VMs. The utilization metrics are excluded as that has no direct correlation to the cluster performance.<br>If the cluster is running vSAN, then the KPI should include metrics measuring vSAN performance. |
| **vSAN KPI** | I do not add in the diagram to keep the diagram simple. vSAN KPI includes VM disk latency of all VM accessing the vSAN datastore (from the same vSphere cluster or from other cluster). |
| **NSX KPI** | I do not add in the diagram as it has a many to many relationship to vSphere cluster. |

# Baseline Profiling

How do you determine what's an acceptable performance by your customers? Is their expectation unrealistic (read: not impossible but costly to implement)? What's the value of profiling your environment if the application team is unhappy with your service level?

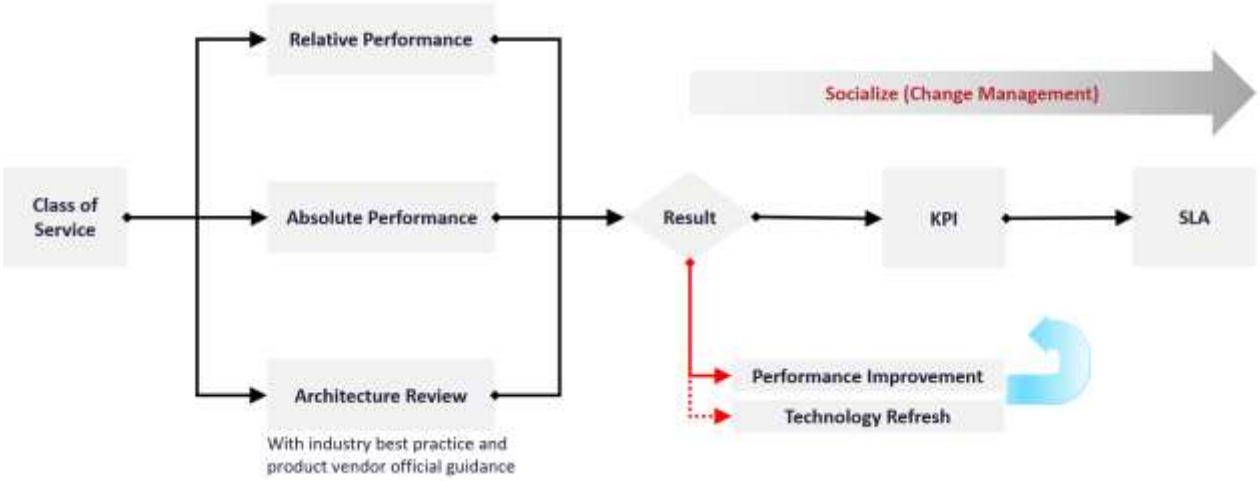
Let's look at 3 scenarios. The first 2 are corner cases, and the 3rd one is the most common, especially in large environment with many application teams.

| Happy | Unhappy | Mixed |
|---|---|---|
| If your application team is happy, then the profiling is a proactive exercise to ensure performance is maintained as your environment grow or change. | If your application team is not happy, then the profiling quantifies the complaints so they can be measured. It reveals the full picture as it analyzes the entire VMware environment, not just the complaints. | Likely, some are happy and while others are not. |
| This prevents infrastructure from being blamed for application-level changes. For example, if your database size grow, it can slow down the application. | It might prove the blame on infrastructure is not right, if the metrics are good. | In a large environment with many VM Owners, it's common to see different owners have different threshold. A 10% CPU Ready may be fine for Owner A yet a 2.5% CPU Ready causes Owner B to be upset, even though both VMs are in the same class of service |
| It helps you establish the SLA while everyone is still happy. | It shows how bad the situation is, and how much improvement is needed. This can help in justifying additional capacity | Profiling brings up these facts to facilitate discussion to get everyone agree on the same SLA, because it's a shared platform |

This is why education is important. As you have promised to serve everyone well, you need to have a threshold that works with 99% of your customers, and not 98%. While 99.9% is a better target, it's also far more expensive.

# Methodology

Now that we know the value and purpose, let's discuss how to implement it. The following flowchart shows the overall process




Do you have a performance SLA implemented?

If yes, that's great!

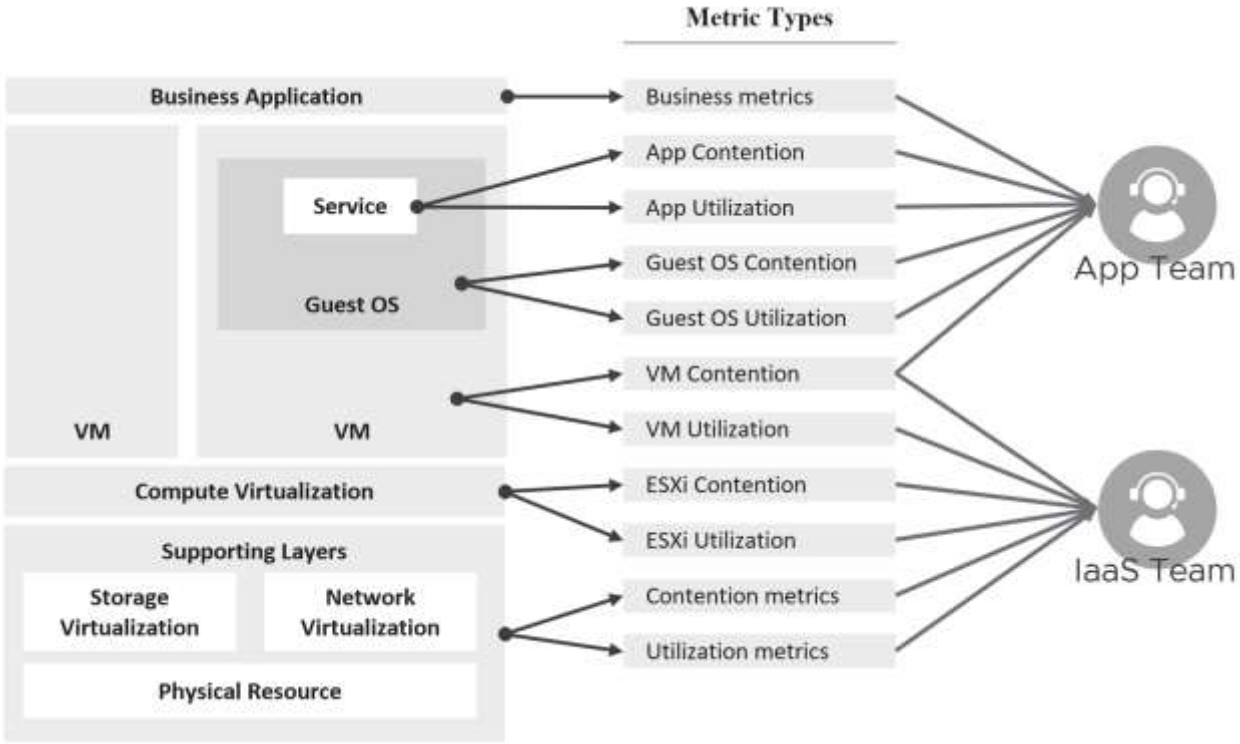
- Do you get complain from your customers? If yes, go back and follow the no performance SLA flow.
- If there is no complaint, how is your actual SLA compared with your contractual SLA? The first one is what was actually delivered in your DC, while the contract is just a promise written on a piece of paper.
- How is your promised SLA compared with industry best practices? If there is a room for improvement that both your customers want and your management supports, then it's worth looking at the system and process to improve it. This may result in a technology refresh, reducing your cost while increasing both capability (availability, performance, capacity, and compliance).

If no, then how do you set? There are 3 factors to consider:

| | |
|---|---|
| **Complaint** | This gives the relative number, as complaint is subject to the person giving the complaint. Different people have different tolerance levels when it comes to how slow a system is.<br>This is the most time-consuming, as most of the time there is neither data nor analysis . You might have to interview the actual users one by one. In some political cases, you will need to get the business users, the application developers and the infrastructure team in a room or zoom.<br>Compile the list of actual complaints. Analyze each complaint to ensure the problem is indeed with the infrastructure (and not application-level issue). If it is due to infrastructure, write down the actual metric or event that prove the problem. |
| **Actual** | This gives the absolute number, as it's based on actual number.<br>This number should be based on the 20-second peak, as the 5-minute average is not sufficient. |
| **Best Practice** | This comes from the makers of the product. For example, Microsoft has a guideline for disk latency for its SQL Server database |

## 2-level Profiling

Profile both "inside the VM" and "outside the VM". The following diagram show the metrics of interest for each level.




## Time Factor

You definitely need enough data points per VM. I recommend 1 – 3 months so you get at least the month end business cycle.

| Period | Data Points | 99th percentile |
|---|---|---|
| 1 hour | 12 | These are too short a timeline as there isn't enough data points, or it will not capture monthly peak. |
| 1 day | 288 | |
| 1 week | 2016 | |
| 1 month | 8765 | What I recommend so habit it formed. |
| 3 months | 26298 | Too infrequent. You may forget. |

Taking the worst value of 8765 data points can result in outlier.

I recommend taking the 99th percentile. This eliminates the worst 87.65 data points per VM if you take 1 month' worth of data. That means ignoring the worst 7.3 hours of the month. As this can be significant, that means the data cannot be ignored.

How about maintenance, upgrade, and other exceptional activities?

They should not take up 7+ hours per month.

## Outcome

What are the possible outcomes of the analysis ? There are 1 – 4 action items you can do. The action "align with best practice" include configuration changes, design changes and version upgrade.

| Relative Data | Absolute Data | Recommendation |
|---|---|---|
| Valid Complaint | Bad Performance | Adopt industry best practice. Review gaps between actual performance and **industry best practice**. Align design & configuration with performance best practice.<br>If hardware is old, tech refresh to increase capacity and performance, while reducing cost. |
| Valid Complaint | Good Performance | If the performance issue is short lived, analyse further with **20-sec peak**. If this is the cause, improve performance further.<br>Discuss and quantify the required performance with application team. Use tech refresh but focus on performance, not capacity. |
| No Complaint | Bad Performance | Assess if there is a risk for future complaint. If yes, this is your window to improve. Ideally, align design & configuration with **industry best practice**.<br>If hardware is old, tech refresh to increase capacity and performance, while reducing cost. |




What if you have no complaint and the profiling proves the performance is good?

- That's certainly a good news. It is the ideal situation and you may not have to do anything. However, make sure you set expectation at the right level (balancing price/performance).
- Proactively, check if present both performance & capacity are enough for future demand.

## Complaint Analysis

Why do we start with complaint?

Because this is what the business experience.

End users normally complaint on 2 types of problems:

- The system is down. This is easier to confirm as the proof is clear. You typically have log files too.
- The system is slow. This is much harder as it's subjective.

They don't normally complain about the system is not secured.

The problems above can be major or minor. Major problems are easier to analyze, and they tend to be reproducible. You have many users, or many applications, or many VMs affected over long period of time. In this case, the problem is likely something common, and you can see the metrics.

Minor problem is hard to analyze. You likely need to interview the actual user. When you do, gather the following

| | |
|---|---|
| **Time** | This is not the time the complaint was lodged, but the time the problem actually happened. For example, the developer may complain at 12:34 pm that her application was slowed at 09:51 am, and it lasted 14 minutes. |

| | |
|---|---|
| **VM** | The VM affected. Ideally, this was analyzed by developer as the VM causing the problem, not the VM receiving the problem. |
| **Analysis** | What is the root cause? If the actual cause is not directly related to the symptom, it needs to be explained as it becomes less convincing. |
| **Proof** | What's the metric or event that prove that the root cause was indeed correct? The acid test is alert. The expectation from senior management is if the same problem happens again, you know right away. How can you know it right away? The alert. You set up an alert to trap the root cause.<br>If your root-cause alert does not detect the problem, you have not found the root cause. |

# Consideration

What things must you consider when doing the profiling?

| | |
|---|---|
| **Resolution** | Profile based on 5-minute average, not the 20-second metrics. This matches your SLA. |
| **Scope** | Profile Compute and Storage.<br>Why is network not included?<br>Network needs to be analyzed separately due to its nature as interconnect and not nodes. |
| **Focus** | Profile all the VMs, not just the VMs you care. This gives you the complete picture. The only time you exclude VMs is when you're 100% sure they cannot impact the performance of the VMs that matter to you. Typically, this means the environment is physically isolated, including separate physical network and storage. For example, if there is a physical data center where you don't care at all, you can exclude them.<br>In large environment, split by Class of Service. The higher one should be performing better than the cheaper tier.<br>You profile the VM, not the cluster or datastore, as that's where the SLA is. |

# Noisy Neighbour

I use the word Noisy Neighbour instead of "resource management" as it's clearer in meaning. When you have enough overall capacity, but some consumers (VM, K8 Pod) aren't getting what they are asking, you have imbalance in the shared environment. One major root cause of imbalance is noisy neighbour.

In a shared environment where there is practically no limit or control placed on consumer (e.g. VM, Container), you can get into Victim Villain problem. This is where a small percentage of consumer cause performance problem to others. It could be as simple as 1 constant consumer causing the problem to the same few victims, so you know who the villain and who the victims. It could be as complex as transient problems, where you can not even see a pattern. Both the villains and victims are random, and they may even trade position over time. Sometimes the victim is the villain, causing self-harm.

The real villain may not be the VM, meaning the real request of the load is coming from outside the VM. Windows or Linux was simply executing what a central command center asked.

The cause of the problem may not even be related to the symptom.

A story I remember well is the Head of IT Operations asked me to "find the b@$t@rd that did this to me". He believed there is a villain VM in this environment that randomly attacked other VMs. The symptom is the victims became unresponsive, sometimes to a simple ping, but the problem always disappeared within 15 minutes. The victims were spread across multiple clusters, multiple datastores and multiple data centers. There is no pattern on the victim. They varied in size, and both Windows and Linux were affected. There is no pattern on the time, so he did not know when the next attack would be.

We did a lot of analysis , working with both storage team (they were on EMC high end array) and network team (they have synchronous replication for mission critical datastores over Cisco switches and routers).

At the end, we found out it was the anti virus signature update. It was not randomized over long period. When hundreds of VMs were updated, the combined IOPS saturated the WAN link, reducing the synchronous replication. That basically paused IO commands, so the VMs were frozen.

What are the 3 knobs you have to control noisy neighbour?

- Share
- Reservation
- Limit

We cover them in Configuration Management chapter. Go to the Resource Management section.

What knob is missing from the above?

Do you think Priority is missing? I think priority is too strong a knob. It gives 0% to the lower priority ask, which is not what you want. You still want each consumer to get something, which is basically what share does.

Let's now apply the above knobs into the four main resource types. In future I hope to address GPUs as they have different functionalities.

Just like there are many nuances on metrics, there are also nuances in addressing noisy neighbours.

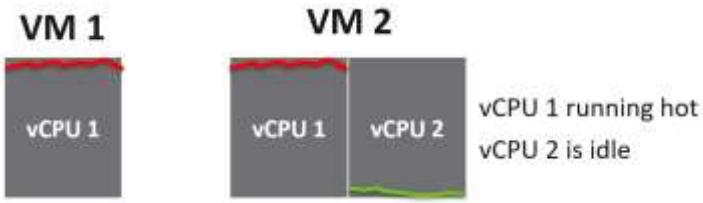
You will see that the settings are complex. Adding Resource Pools complicates further as you have another management layer.

# CPU

Let's begin with CPU as this is typically where the problem is.

| Name | Reservation (MHz) ↓ | Limit (MHz) | Type | Shares | Shares Value | % Shares |
|---|---|---|---|---|---|---|
| Window... | 4388 | Unlimited | Fixed | Normal | 2000 | 13.33 |
| vropsCP... | 0 | Unlimited | Fixed | Normal | 2000 | 13.33 |
| LinuxThi... | 0 | Unlimited | Fixed | Normal | 1000 | 6.67 |
| linux2n... | 0 | Unlimited | Fixed | Normal | 1000 | 6.67 |

| Control | Analysis |
|---|---|
| Limit | There is a default limit, which is the configured CPU size of the VM or container. The problem with this limit is most of the time it's too high, driven by cheap pricing. A VM that only needs 8 vCPU has a 32 vCPU configuration. Why? Because the buyer can afford it.<br><br>Let's take an example:<br>• Cluster size: 10 ESXi hosts<br>• ESXi specification: 40 cores, HT enabled, 2.5 GHz each. Total 100 GHz, not counting HT.<br>• Total cluster capacity: 1000 GHz (not counting HT benefit)<br>• Overcommit ratio is 4:1. So 1 core is assigned to 4 vCPU. The core can run 2 vCPU at the same time, but at 37.5% performance penalty.<br>• No of running VMs: 400<br>• Average size: 4 vCPU. Total is 1600 vCPU on 400 physical cores.<br><br>In the above example, all you need is a few 32 vCPU VM to create the problem. If you have 10 of them, there is a chance each fills up 1 ESXi host. This is why both overcommit ratio and maximum VM size are both in your pricing policy.<br><br>While you can set a limit on the VM, it can result in unpredictable performance as Windows or Linux does not know that it can only really use a portion of the resources allocated. The limit is also applied in GHz, not vCPU. So the impact can vary each second as the frequency fluctuates. |
| Reservation | Can you solve the above problem by reserving CPU to all the VMs?<br><br>The answer is no. You can only reserve 25% of the configured capacity of each VM, because the overcommitment ratio is 4:1. That means a 2 vCPU VM is only given 1.25 GHz. This loosely translates into 0.5 vCPU only, not enough to run anything useful.<br><br>One adjustment you might think of is to guarantee the first vCPU. So each VM gets 2.5 GHz, regardless of size. Since there are 400 VM, you reserve 1000 GHz.<br><br>Your total capacity is only 1000 GHz, so the cluster is filled up. This actually exceeds your usable capacity, as you need to account for HA.<br><br>On the other hand, you have 25% boost due to HT.<br><br>Reservations become less effective as you increase the overcommit ratio. At 8:1 overcommit, you cannot guarantee more than 0.5 CPU worth of resource across the environment, on average. |

| | |
|---|---|
| Share | Share is per vCPU, not per VM.<br><br>While a larger VM has more overall CPU shares, the share is not shareable among its vCPU. In the following example, the vCPU 1 of VM 2 does not get 2x the share as it does not benefit from idleness of vCPU 2 of VM 2.<br><br>IO cost the associated VM. A VM doing excessive network packet and disk commands may experience CPU ready if it runs out of share. The reason is the share is measured against CPU Used, and not CPU Run.<br>CPU Used includes CPU System, which account for the work performed by hypervisor on the vCPU behalf.<br>`%USED = (%RUN + %SYS - %OVRLP) * Frequency Factor * HT Scaling Factor.`<br>Hence a high CPU System will eat the share of the vCPU. |

# Memory

Memory is basically an extremely fast disk, such that you care about space more than speed. As a result, the memory utilization metric tends to be flat over time, both at the VM level and ESXi level. The exception here is transient VMs or containers. ESXi clears its consumed metrics when the associated VM is powered off. So in environments running Horizon Instant Clones, you get higher fluctuation.

Because of this flat utilization, the overcommit ratio is lower. This results in relatively less contention, and therefore problems. The problem with this solution is cost. You are basically paying for a lot of cache.

| Name | Reservation (MB) ↓ | Limit (MB) | Type | Shares | Shares Value | % Shares |
|---|---|---|---|---|---|---|
| WindowsV... | 4096 | Unlimited | Fixed | Normal | 40960 | 12.9 |
| windowsT... | 4096 | Unlimited | Fixed | Normal | 40960 | 12.9 |
| vropsCPPM | 0 | Unlimited | Fixed | Normal | 81920 | 25.81 |
| LinuxThick... | 0 | Unlimited | Fixed | Normal | 10240 | 3.23 |

| Control | Analysis |
|---|---|
| Limit | The memory utilization at both Guest OS level and VM level consist of mostly cache, pages that are not used. Since there is no limit within Guest OS, should you apply at VM level then?<br>This carries the risk of memory contention, as the 2 layers are independent of each other. You need to track the memory contention closely if you are doing this. Considering the lack of real-time remediation solution, I'd recommend you stay away from this, and focus on right-sizing at the Guest OS level instead. |
| Reservation | Unlike CPU, memory reservation is "sticky". Once the VM touches that page, it's permanently reserved regardless of subsequent usage. The problem is the |

| | |
|---|---|
| | VMkernel does not know if that page is useful or not. As a result, I'm not recommending you use this.<br>Unlike CPU, where you frequently go beyond 2 : 1 overcommit, you typically do not overcommit beyond 2 : 1. If you overcommit, tracks the VM memory ballooning counter for an early warning of memory contention.<br>For your highest class of service, where you guarantee memory performance, there is no overcommit. Per-VM reservation becomes irrelevant when the entire cluster does not overcommit. |
| Share | See CPU, as the behaviour is the same (although the default share values themselves are slightly different). |

# Disk

There are actually two types of metrics, as disk has both space (GB) and speed (IO and throughput). The speed dimension is more problematic as the limit is much higher and the spike is unpredictable. You can have either IOPS or throughput giving you problems.

Unlike compute, there is no defined upper limit by default. A VM can generate unlimited IOPS or throughput. This creates risk, as it can take just a single developer running IO Meter to saturate your shared storage.

Compared with compute, storage typically has more layers, as the actual storage provider is often outside the hypervisor. Even with hyperconverged storage like vSAN, the vSAN kernel module is a separate stack and you need to deal with the vSAN network. In the case of central storage array, you likely need to deal with a storage fabric.

Because of the separate layers, there are multiple points of control:

- Non vSAN: VM, datastore, storage array. If the fabric has control, you can set it there also to throttle IO going into the central array.
- vSAN: VM and vSAN.

The control for VMs can be seen below, which shows that I've set a 4000 IOPS limit and give Hard disk 1 a higher share value.

| Name | Disk | Datastore | Limit - IOPs | Shares | Shares Value | Datastore % Shares |
|---|---|---|---|---|---|---|
| WindowsVMvs... | Hard disk 1 | vsanDatastore | 4000 | High | 2000 | 20 |
| vropsCPPM | Hard disk 1 | vsanDatastore | Unlimited | Normal | 1000 | 10 |
| vropsCPPM | Hard disk 2 | vsanDatastore | Unlimited | Normal | 1000 | 10 |
| vropsCPPM | Hard disk 3 | vsanDatastore | Unlimited | Normal | 1000 | 10 |
| LinuxThickVM | Hard disk 1 | vsanDatastore | Unlimited | Normal | 1000 | 10 |

The problem is: should you set it? If yes, what numbers?

| Control | Analysis |
|---|---|
| Limit | There is a limit for IOPS, which is per virtual disk. Can you see the challenge in setting this number?<br>There are different variations (block size, read/write ratio, random or sequential). That means the resulting performance of 1000 IOPS can vary. The problem is your application team typically do not know this level of details. Guess who loses when you argue with your customer? |

| | |
|---|---|
| | If you set too low, the VM latency may go up.<br>If you set too high, it defeats the purpose. Your central physical array may be saturated. For example, if your storage is supporting 2000 VM, if the limit of VM 1% of what your array can deliver, then it only takes 5% of the VM population (40 VM in this case) to saturate your storage. And this is assuming their IOPS are perfectly distributed across your disks. In reality, you have hot spots.<br>For VMs, you can consider limit for the OS disk, if applications do not use it. This at least prevents rogue agents or system-level service for running amok. |
| | What about the control at datastore level (non vSAN)? SIOC provides 2 choices for congestion threshold. As you expect, both have their own pros and cons, else you just need the best and no need a choice.<br>What about the control at a vSAN level? We certainly need to dive a lot deeper in this area, before applying limits to each virtual disk.<br>What about disk space? How do you prevent your datastore running out of space when you overcommit? For thin provisioned, you can't control when the disk space will be used. The only thing you can monitor is snapshot age. Remove them within a few days to minimize the risk of huge snapshot. |
| Reservation | There is no reservation for VM Disk. See earlier screenshot listing 5 VMs in a table. None of the column show reservation. |
| Share | Shares come into play only when latency hits some threshold.<br>Unlike CPU, the share does not care about the disk space size. Ideally, the control you want is IOPS per GB. So when you buy more capacity, get faster disk. The reason is when you sell disk space, it should come with a certain amount of IOPS and throughput at certain latency threshold. This is how you justify gold storage higher price per GB. |

## Storage I/O Control

The control at the datastore level is for a different purpose. Storage I/O Control (SIOC) is a datastore-wide throttling. There is share and limit, but not reservation.




You can enable SIOC on each datastore, including local datastore.

Take note because vSAN has its own mechanism, SIOC does not apply to it.

# Network

Network is the hardest to solve, due to its unique nature as interconnect. I cover it in-depth in vSphere Metric book. Read at least the section "Why Network Monitoring is Unique"."

Done?

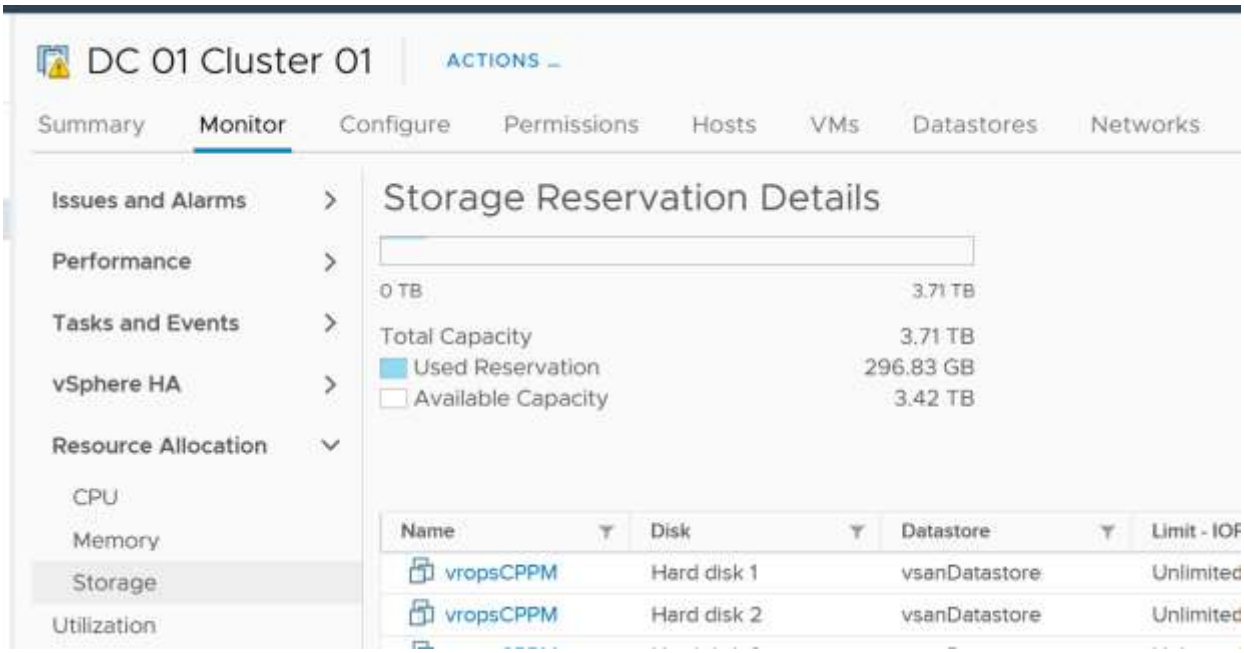
Great, now let's discuss how the noisy neighbour problem is harder in the network.

We start with a single ESXi, not distributed switch, as that's where the villain VM is running. Let's say the ESXi has 2x 25 Gb/s physical NIC. Total is 50 Gb although you typically count this as 40 Gb, giving you 20% headroom. In these 2 cards, you run both the VMK traffic (vSAN, vMotion, vSphere Replication, etc.) and VM. Let's say you have 25 running VM on average. Assuming you allocate around 20 Gb for VMK, that leaves 20 Gb for 25 VM.

Now, a single villain VM can hit 10 Gb. For example, Hadoop worker node receives large amount of data over the network, sustaining 5 Gb over 300 seconds period. That leaves 10 Gb to be shared among the remaining 24 VM. Each only gets 400 Mb/s on average. If they are not network intensive, they may not feel the impact.

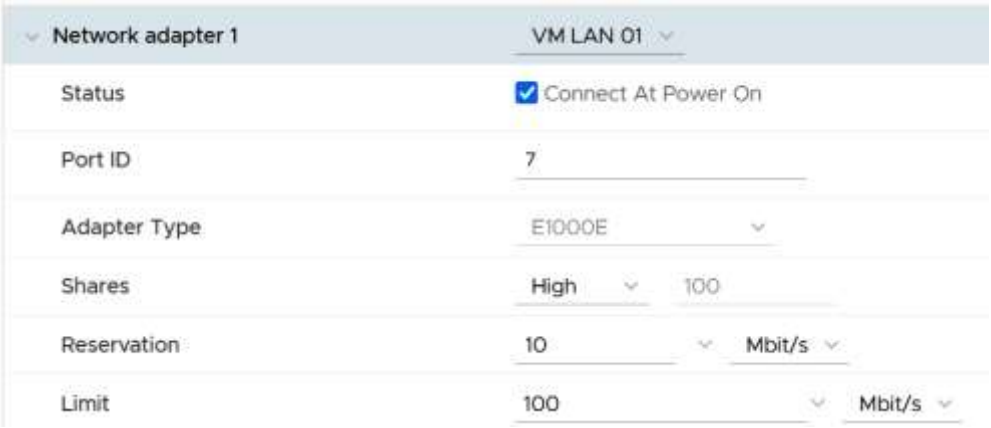
But the above calculation is done on 2 x 25 Gb network. If you only have 2 x 10 Gb, you increase your risk significantly. In a sense, it goes back to fundamental of capacity management, which is your overcommit ratio.

Review the following diagram, which shows a vSphere Cluster. Why is network missing on Resource Allocation?




The reason is you configure it on the distributed switch.

Just like compute, you can set on each VM network adapter the shares, reservation and limit.




| Control | Analysis |
|---|---|
| Limit | Just like compute and storage, if you set limit you need to ensure it's high enough for the VM, yet low enough for the ESXi physical NIC card. So what number do you set?<br>I think it should be 8 Gb/s. That's good for the VM, while giving some buffer at ESXi assuming it sports 2 x 25 Gb/s physical NICs. There is also management challenge, as the settings are buried deep and set in multiple places. So at the end of the day, it might not be practical. |
| Reservation | If you have plenty of bandwidth, consider setting 100 Mb/s per VM. Use the metric profiling technique to arrive at number specific to your environment. |
| Share | The main thing you need to check is actually between VM and non VM. The share should be larger for production VM network and development VM network.<br>Regarding VMkernel, it's tricky. Is vMotion more important than vSphere Replication? What if you're replicating a mission critical VM but migrating a development VM? |

# Solution

So what can you do to avoid it altogether, instead of just minimizing it?

Well, if you overcommit, the answer is it is not possible. It's like the highway. You overcommit the lanes, so during peak hour there will be some congestion. You can minimize the impact by having the right capacity planning.

Start with the right pricing policy.

- It should be a function of your overcommit ratio. In a nutshell, your message to your customer is "if you all want to pay half price, there will be 2x as many of you as I need to break even".
- Have progressive pricing tier, so a 64 vCPU VM is not simply 64x the price of 1 vCPU VM.

If you do not charge, then use Class of Service as the policy. The size of the VM or Container is less in the lower class of service, as the class has higher overcommit ratios.

Educate your customers, and communicate clearly the different quality of service. For example, the following shows the SLA for memory.

| Class of Service | Overcommit | Performance Threshold |
|---|---|---|
| Gold | 1 : 1 | VM contention is 0% for 100% of the time in the entire month |
| Silver | 1.5 : 1 | VM contention is <1% for 99.99% of the time in the entire month |
| Bronze | 2 : 1 | |

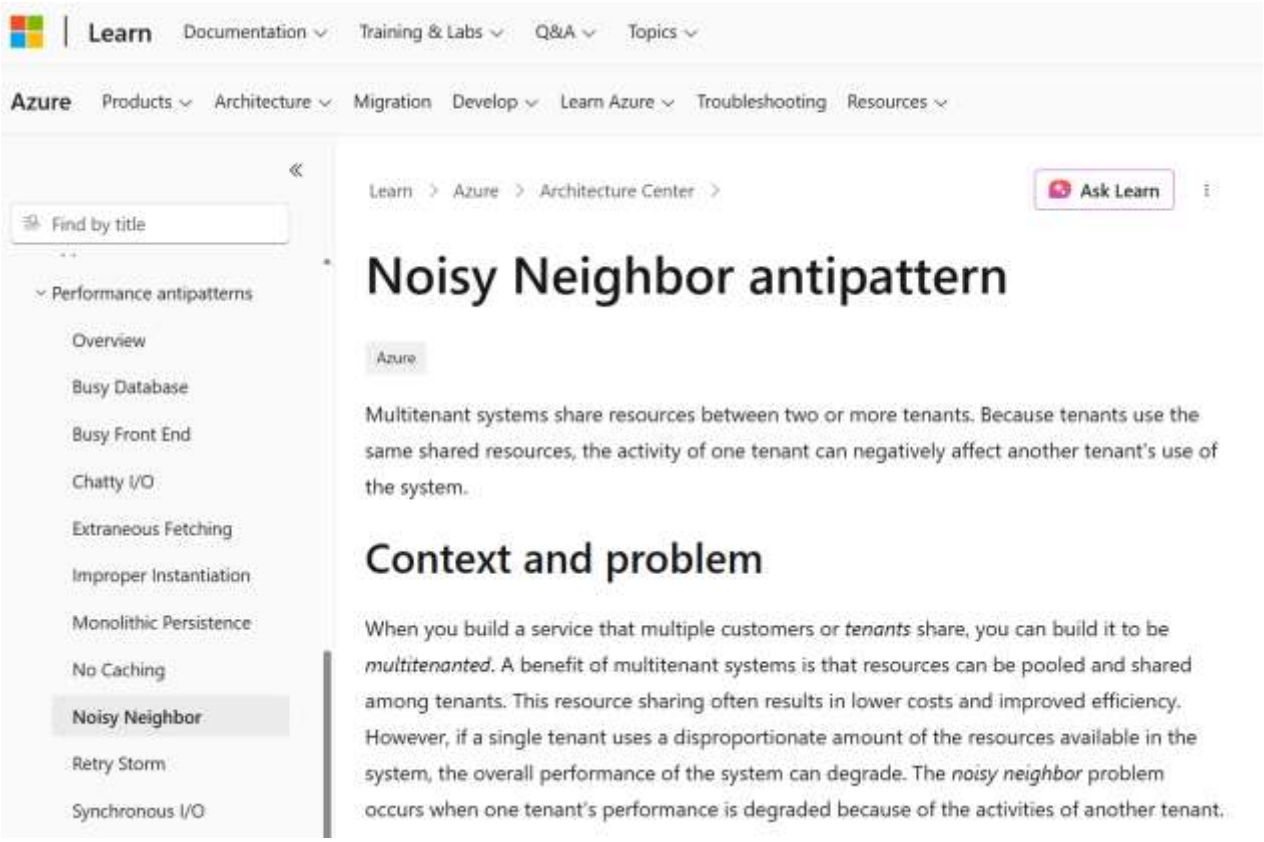
## Maximum VM Size

The following table provide guidance on the maximum VM size.

| Overcommit | Maximum VM Size |
|---|---|
| 1 : 1 | Maximum size = ESXi logical CPU.<br>Logical means HT enabled. An ESXi with 48 cores 96 threads means 96 logical CPUs. However, take note of CPU throughput degradation and CPU NUMA effect. |
| 2 : 1 | Maximum 0.5 of ESXi logical CPU<br>As there can be 2 large VMs running, they will feel the 37.5% penalty as each of them actually want the entire physical core. To avoid that, reduce from 0.5x to say 0.4x, giving 20% headroom. |
| 4 : 1 | 0.25x<br>Same as above. You can have 4 medium size VM, each of them taking up 1 entire socket in a 2 sockets ESXi host. So a pair of VMs compete for 1 physical socket. |
| 8 : 1 | 0.125x.<br>This level of overcommit is generally only suitable for lab, dev or simple VDI (where the desktop is 2 vCPU each). It's not suitable where a lot of VMs are 25% of entire cores. For example, if your ESXi has 48 cores total, then avoid doing 12 vCPU VM. Keep them at 4 – 6 vCPU instead, so they can be slotted more easily in the cluster |

## Other Reference

By the way, the noisy neighbour problem happens in non-VMware environment too, as it's a function of overcommitment on shared resources. Here is an article about noisy neighbours in Microsoft Azure.



# Capacity Management

## Part 1 Chapter 3

At the heart of capacity management is balancing demand and supply. It is about meeting demand with the lowest possible cost while remaining compliant to security and availability best practice.

# Overview

## "Good" Advice

Let's begin with this as I keep seeing it in VMware-based environment. The scope of the advice is about a VMware vSphere Cluster, but the principle applies to others such as Kubernetes, VDI or AI.

Can you figure out why the following statements are wrong? They are all well-meaning advice on the topic of Capacity Management. We're sure you have heard them, or even given them.

Regarding vSphere Cluster RAM:

- We recommend 1:2 overcommit ratio between physical RAM and virtual RAM. Going above this is risky.
- Memory Usage on most of your clusters is high, around 90%. You should aim for 80% as you need to consider HA.
- Memory Active should not exceed 50-60%. You need a buffer between Active Memory and Consumed Memory.
- Memory should be running at high state on each host.

Regarding vSphere Cluster CPU:

- CPU Ratio on cluster "XYZ" is high at 1:5, because it is an important cluster.
- The rest of all your clusters' overcommit ratio looks good as they are around 1:3. This gives you some buffer for spikes and HA.
- Keep the overcommit ratio to 1:4 for Tier 3 workload as they are not mission critical.
- CPU usage is around 70% on cluster "ABC". Since they are UAT servers, don't worry. You should get worried only when they reach 85%.
- The rest of your cluster's CPU utilization is around 25%. This is good! You have plenty of capacity left.

Can you figure out where the mistakes are?

The mistake is they are simplified. Capacity may appear simple:

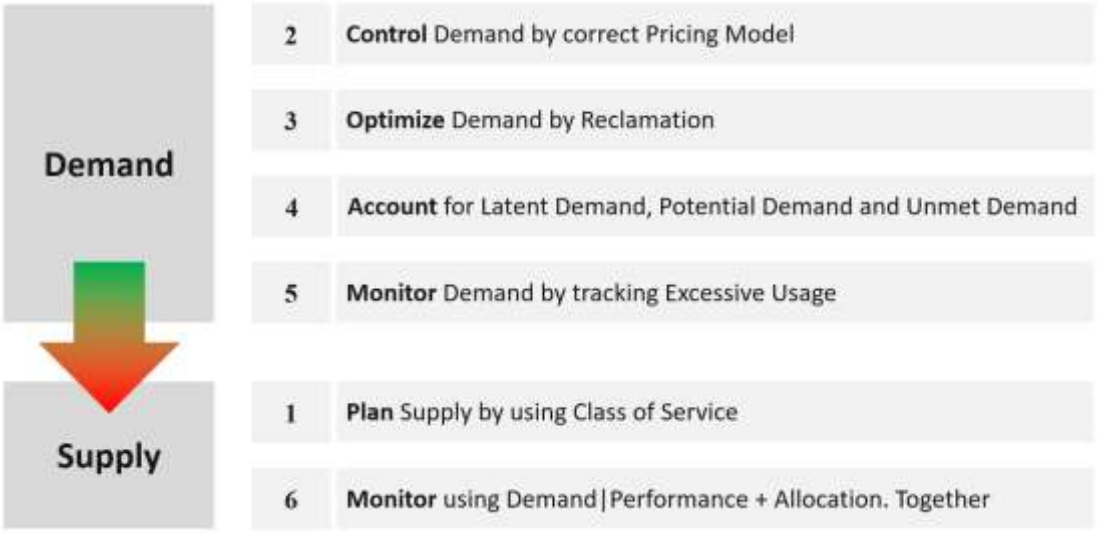
- Can you architect a cluster where the performance matches physical?
  Easy, just don't overcommit, or put 100% reservation for that VM.
- Can you architect a cluster that can handle monster VMs?
  Easy, just get lots of cores per socket.
  Easy, just get lots of core in the box.
- Can you architect with very high availability?
  Easy, just have more HA hosts, more vSAN FTT with failure domains spread across different racks, more NSX Edges.
- Can you architect a cluster that can run lots of VMs?
  Easy, just get lots of big hosts.
- Can you optimize the performance?
  Sure, follow performance best practices and configure for performance. Just be prepared to pay.

- Can you squeeze the cost?
  Sure, minimize the hardware and software cost, and choose the best bang for the buck. You know all the vendors and their technology. You know the pro and cons of each.

But how to put all the above together that optimize cost, performance, security and availabiity?

# Concept

Balancing demand and supply require you to look at these 6 components below. Steps 1 and 2 are done together, and the remaining 4 steps can be done in parallel.




The above is less harder to do if you do it right from the start, which is why we need to begin at the planning phase.

For example, imbalance cluster with low ESXi utilization can be caused by monster VM. Can the applications scale horizontally instead?

If you start from Step 6 and ignore Step 1 and 2, you will play the lead role in a Mission Impossible movie, because you can end up with many over-provisioned VM issue. These VMs are typically the larger ones, and more important to the business. It is hard to solve this in production environment as it will involve downtime and the burden is on you to prove it will not have performance impact. Politically, it may make the team who sized the VM and justified the cost look bad.

Your best bet is to prevent the problem from happening in the first place.

VM impacts capacity in 2 ways:

- Rightsizing
- Reclamation.

## Private Cloud Capacity

Your IaaS consists of 3 large components

- Compute

- Storage
- Network

| | |
|---|---|
| **Compute** | It covers ESXi, cluster, resource pool, and physical server.<br>It gets the most attention as the consumer (VM or Pod) runs in a cluster of ESXi host. It's a good starting point, especially if you have 1:1 relationship between compute and storage. |
| **Storage** | It covers datastore, datastore cluster, vSAN, RDM, physical array, back up infrastructure, etc.<br>It needs to be managed equally well, if you have many to many relationships between cluster and datastore. If you use vSAN HCI Mesh, then you also need to manage the storage portion carefully. |
| | Storage Capacity differs to Compute Capacity and presents a challenge on its own. Unlike compute capacity, which is basically vSphere cluster, storage varies in shape. The two major ones are datastore and vSAN, as local datastore and RDM are rarely used. In addition, storage has thin provisioning at both virtualization layer and physical layer. We will discuss vSAN capacity separately as it has its own unique factors such as FTT, plus it needs to consider compute too. |
| **Network** | It covers virtual and physical. It also includes switch, router, firewall, load balancer, etc.<br>It is typically less of an issue for VMware Architect, as it's typically done by the Network Architect. In addition, it's common for ESXi to sport 50 Gb of bandwidth. So unless you run high bandwidth applications, such as networking VMs and web servers, running on the same ESXi hosts, you may not hit the limit. |

# Stages

Capacity Management requires an end-to-end plan, typically spanning multiple years, not months. Why?

There are 2 reasons:

- At the provider, the physical layers form a constraint. A rack, top of rack switch, cabling, cooling all have limits.
- At the consumer level, production workload tends to live for several years, and they change along the way.

Such a long plan requires adjustment along the way, because at the end of the day it is about comparing the reality you face with the plan you set. Good or bad is relative to your plan. If you plan for no overcommit because performance is absolute and budget is not an issue, then you'll never run out of capacity. In other cases, that could be considered bad as you could end up with a lot of wastage.

There are 4 phases of capacity management:

| | |
|---|---|
| **Plan** | Within this phase, you perform sizing and estimate how long the infrastructure will last. Depending on the project, you may even size a multi-year capacity up front. The longer the plan, the bigger the margin you need to allocate. Higher margin or buffer certainly increases the risk of excess capacity.<br>As not all plans are confirmed, you may run multiple What If scenarios. |

| | |
|---|---|
| | This is the phase where you buy hardware. |
| **Monitor** | This phase starts after deployment. You begin tracking and compare the reality against plan.<br>For example, you expect the capacity for your new DaaS project to last for 1 year. 3 months into deployment, you have 25% of your users consuming the DaaS. However, your overall utilization is already at 70%. This is a red warning, indicating your plan is off by significant margin. |
| **Optimize** | You optimize both the supply and the demand.<br>To optimize supply, you typically perform a tech refresh. Newer hardware can bring 2x the capacity at the same cost.<br>To optimize demand, you perform reclamation and rightsizing. |
| **Upgrade** | As the hardware reaches end of life, you either upgrade or migrate. The workload likely need vMotion somewhere else.<br>This typically involves technology version refresh and design changes. |

Capacity Management becomes easier if you begin at the planning stage. This is where you define your offering, setting the price and performance expectation. Without expectation being quantified as metrics, your customers will demand high performance as you've promised them "good" performance.

e first place, by using progressive pricing. This is covered in the Cost and Price Management chapter.




Discipline in capacity optimization is necessary due to excess wastage. Establish a weekly cadence that is executed regularly (depending on the size of the environment).

In large environment, set up upgrade cadence. Let's take an example:

- You have 1200 ESXi hosts. Hardware depreciation and warranty is 5 years, so you replace after 5 years. That means you replace 240 hosts per year. If you do a monthly cadence, you replace ~20 servers per month.
- To balance between keeping variations low and harnessing new technology, you set the standard per year. Technology Refresh is an integral part, as new technology delivers both lower cost and higher capacity, not to mention faster performance and tighter security.

## The Planning Stage

Capacity management begins long before hardware is deployed. It begins with a business plan, which decides on what class of service will be provided to serve which locations. Class of Service was covered earlier in SLA. You should also read the Performance SLA portion here, as it's required when you overcommit the capacity.

| Input | Consideration |
|---|---|
| Location | The physical location, which could be data sovereignty or network latency requirements. |
| Cost | It impacts the architecture and location. |
| Class of Service | It's related to the type of service. In addition to IaaS, you may have VDI, Database as a Service, K8 as a service, etc. |
| Security | It might require traditional physical or air gap separation. |
| Availability | This includes HA and DR. |
| | Depending on the business policy, you may have to comply with Business Continuity Policy or security policy. Examples:<br>• Define the blast radius to contain major outage. This may translate that you do not design a large cluster containing large amount of VMs, as a cluster may unintentionally go down.<br>• Define the concentration risk. The number of mission critical VMs in the same datastore is capped below 250 VMs.<br>Contain the security risk. Internet-facing VM and internal-facing VM do not share the same network. |
| Environment | Other than production, you may need to provide test, development, staging environments. |
| | What if an application needs to do a scalability test? A revenue-generating application going live may need to simulate their full load to ensure it can meet the sales demand. This scalability test can't be performed in production environment. One solution is to triple purpose the DR cluster. It's DR + Dev + Test, where test includes scalability test. This means the cluster size needs to be larger. If this is not possible, then burst to cloud, as it's temporary workload. |

Group your operations by clusters or group of clusters. Take note that the vCenter Data Center object for example. It can contain clusters of different purposes. It will not make sense to combine the metrics into a single data center capacity remaining (%) metric, if the member clusters are not interchangeable.

| | Cluster A | Cluster B | Cluster C | Average of All Clusters |
|---|---|---|---|---|
| CPU Remaining | Very High (wastage) | Low | Low | ? |
| Memory Remaining | Low | Very High (wastage) | Low | ? |
| Disk Space Remaining | Low | Low | Very High (wastage) | ? |
| Capacity Remaining | Low | Low | Low | ? |
| Time Remaining | Low | Low | Low | ? |

## Architecture Plan

As you can see, it's complex to balance all the above. To make it worse, what worked last year for you may not work next year as many factors change. Regardless, make an overall plan that lists all the input you need. In a large environment, list all the input considerations for vSphere Clusters.

From a capacity monitoring point of view, vSphere cluster is the *smallest logical building block*, due to HA + DRS + DPM. So it is correct to assume that we do capacity planning at Cluster level, and not at Host level or Data Center level.

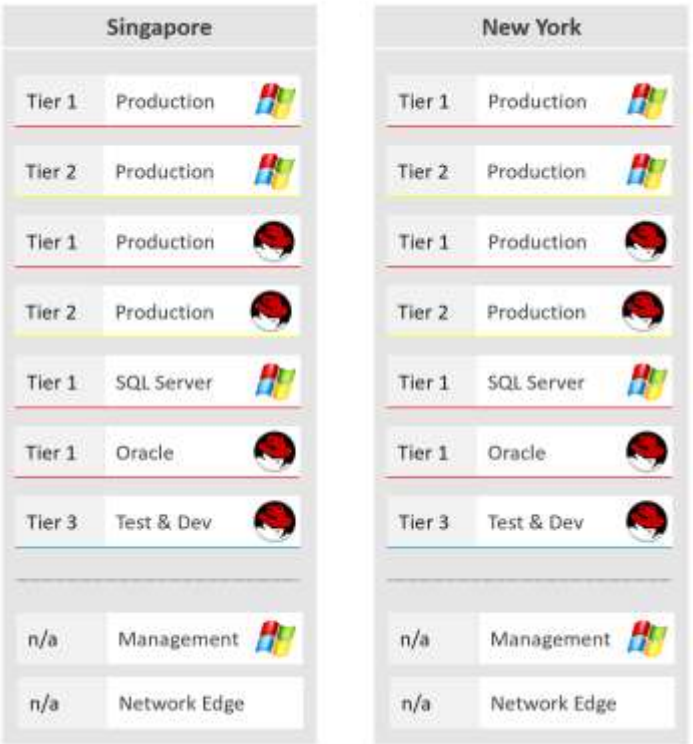
| | |
|---|---|
| **OS License** | Windows, RHEL |
| **Environment** | Production, Test/Dev |
| **Network** | NSX Edge, DMZ, Internal |
| **Location** | Singapore, New York, etc. |
| **Software ($)** | Oracle, SQL Server, etc. |
| **Service Tier** | Tier 1, Tier 2, Tier 3 |
| **Function** | Management, NSX Edge |

DR solution such as Site Recovery Manager impacts capacity as you need to consider both the DR test and actual DR.

## Example

The following shows a VCF environment with just 2 physical locations but 18 unique clusters.

It has 7 clusters for business workloads and 2 clusters for non-business loads (overhead).



# Optimized Capacity

Optimized Capacity means you fully consume what you bought, without wastage or compromising performance.

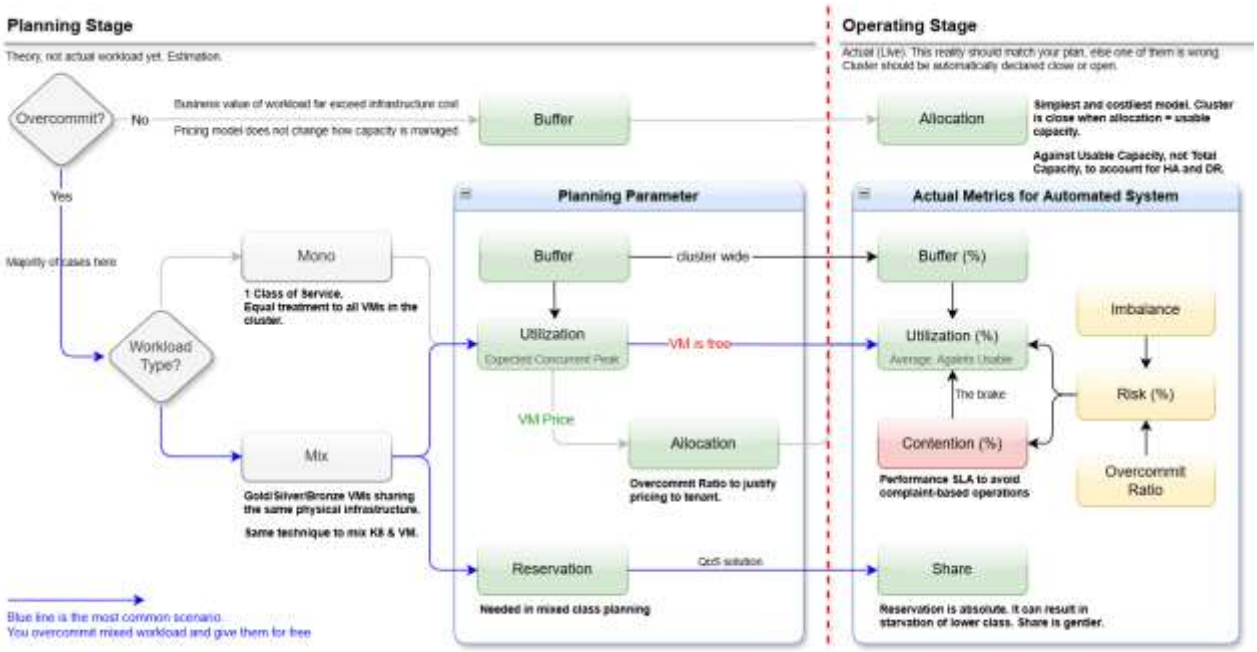
In the provider layer (ESXi, cluster, datastore & datastore cluster, distributed switch and port group, hardware), you can optimize the following:

| | |
|---|---|
| **Compute** | Reduce pockets of resources by using larger cluster, removing Host/VM Affinity and setting DRS to be fully automated.<br>Increase supply while keeping cost the same by doing a technology refresh.<br>Avoid usage of CPU pinning. |
| **Storage** | Use larger datastore to minimize island of buffers.<br>Use local datastores for agent VM or applications that do not need HA.<br>Reduce islands of datastores by consolidating datastores that are lowly utilized. |
| **Network** | Use larger physical pipe. For example 2x 100 Gb instead of 4x 10 Gb.<br>Use load-based teaming. |
| | Remove unused network. While network is good for segregation, VXLAN or VLAN sprawl make both management and security harder. |

# Capacity Model

Proactive capacity management is only possible if we can model it. That means defining the different types of capacity dimension. For each dimension, we need to define the formula for total capacity, usable capacity and consumption metrics.

The model depends on what your goal. The following flow charts show that you apply different models based on the need.




Mixed class is unfortunately common due to budget & physical environmental constraints.

If you operate a multi-cluster load balancing, ensure all member clusters have consistent settings.

## Workload Level

The first consideration is the level of load. There are 2 types:

| | |
|---|---|
| **Not Overcommit** | The simplest and most expensive model. This happens in mission critical where the business value of the workload far exceeds the cost of infrastructure. |
| | The ratio between consumer and producer is 1:1.<br>Utilization, reservation, and contention become irrelevant as each VM can get what it wants when it wants it. |
| | Allocation is the only useful input. Measure this against usable capacity, not total capacity. |
| **Overcommit** | Once you overcommit, you run the risk of contention. The model becomes more complex. See below. |

## Workload Type

Let's begin with what you run on the cluster.

There are two main variants:

| | |
|---|---|
| **Mono** | This means the cluster only runs a single class of service. It does not differentiate among VMs. They are all gold, or all silver, or all bronze. |
| | This makes operations easier. You don't have to set reservations or shares as they are all equal. |
| | How high can you push utilization? |

| | | |
|---|---|---|
| | That depends on your expectation on when the combined workload hit the highest. You plan for concurrent peak instead of individual VM peak. | |
| **Mix** | You're mixing workload with different service level. The gold VMs have to be given higher share when there is contention. You want to avoid absolute reservation.<br>Mixed workload is far more complex. You need to apply different techniques during planning phase and in actual live running phase. | |
| | **Plan** | During planning, since there is no actual workload, you need a calculator to simulate the different combinations.<br>Obviously, you need to put less VM if the cluster will be running mostly gold VMs. |
| | | Since your goal is to maximize consumption, you put as many VMs until you hit 100%. You're gated by 3 dimensions:<br>• Utilization<br>• Allocation<br>• Reservation.<br>Contention cannot be used during planning stage. |
| | **Actual** | In actual run, DRS and ESXi need settings that differentiate Gold VM versus Silver VM versus Bronze VM. |
| | | Use Share.<br>Reservation is absolute and has side effects.<br>Allocation cannot be used as it's not in-line.<br>Utilization cannot be used as it's not a setting. |

## VM Price

Even for internal customers, where no actual money leaves the company, setting a price makes inter-departmental collaboration much easier. Money is the best solution to value something, including your private cloud service.

If VM is free, then you simply put as many VM as possible until you have a complaint. Since you don't want to be reactive and deal with escalation, implement the Performance SLA.

If you're charging $ for VM, then allocation model is the solution to justify why Gold vCPU costs more than Bronze vCPU since they run on the same models of hardware. You simply explain to your customers that you charge 4x less for Bronze because you overcommit 4x. So at the end your cost is covered. It's fair.

# 4 Dimensions of Capacity

There are 4 sides that determines capacity.

| | |
|---|---|
| **Utilization** | This is the primary metric. |

| | |
|---|---|
| | Against usable, which includes both availability and buffer. |
| **Contention** | This is the override. Only used to stop provisioning. |
| | The primary input to “Cluster is closed” decision. |
| **Allocation** | This is the leading indicator. |
| **Reservation** | This is a guarantee. Used in planning. |
| | In actual environment, use it sparingly. Use shares instead. |

They have their own consideration, so let’s dive in.

## Utilization

It reflects the actual, live usage of the resources. If utilization is high, it does not matter if the overcommit ratio or reservation are both below your target, the cluster is full.

The ideal scenario is the cluster is running at

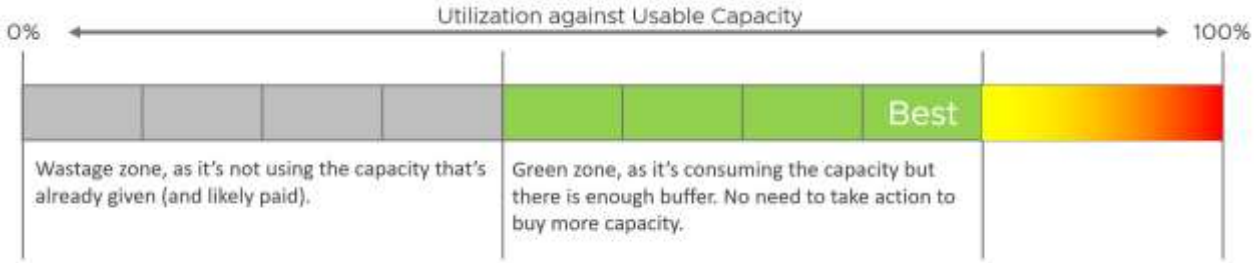
- 100% utilization
- 0% contention
- 0% risk.

This means it’s working as productively as possible and you’re safe. You get your investment well used.

If utilization is low and won’t go high for foreseeable future, that is not a good thing. Unless it’s a newly provisioned object that is yet to grow to its full usage, or disaster recovery protection, that indicates wastage resources.

How should we fit wastage in the utilization model?

The following is my recommendation. I divide the capacity into 10 equal chunks. That means the usable capacity is green when it’s around 40 – 80% used, with ideal target of 80%




Accounting complication:

- It is relative utilization, measured againsts usable capacity. You need to exclude the HA host.
- The hypervisor overhead on the other hosts should not be excluded as they are part of the load.

## Performance

If the cluster is unable to serve existing workload, the remaining capacity becomes practically 0 because provisioning has to stop.

This means you need 2 metrics:

- Performance (%).
  It quantifies the actual performance.
- Stop Provisioning (Yes/No).
  It decides if the cluster is closed for new VM/Pod based on performance (%), capacity remaining (%) and risk (%).

Ideally, you have a predictive metric, a leading indicator showing that the risk is getting higher. This enables you to still provision those VMs in the queue, or better still give 1 week worth of heads up. The problem is the nature of performance and risk do not lend themselves well for projection. You can get wrong result.

## Reservation | Share

They need to be reviewed as complimentary.

| | Reservation | Share |
|---|---|---|
| Nature | Absolute | Relative. |
| | No overcommit as it's a guarantee. | Overcommit is not even applicable. |
| Focus | Throughput.<br>What is reserved is the CPU cycles. That's why the unit is Hertz, milliseconds. Due to Turbo Boost, there is spare capacity at ESXi level. | Runtime.<br>The share is relative to time or unit. It's time sliced for CPU run time, and apportioned for memory consumed. |
| Drawback | If set directly at VM level, it impacts HA and DRS. Lower provider utilization and higher consumer contention due to imbalance. | No guarantee |
| Use it… | …conservatively.<br>Avoid applying the full reservation value. | …aggressively.<br>Apply a higher ratio between gold and silver to give higher priority to gold. |
| Used in the: | Planning stage | Running stage |
| Apply… | … indirectly. | directly. |
| | Set it at Resource Pool. Take note that a gold VM may not get what it wants 100% timely. DRS scheduler lags ESXi scheduler. | Set it at VM. |

Both introduce operational complexity. Over time, you can lose tracks what VM or resource pool has what reservation or share and when.

There is no deterministic correlation between VM reservation or share, and its performance. A VM CPU Ready does not improve 2x because you increase its CPU reservation or CPU share by 2x.

### Reservation

Reservation complicates operations because of these reasons:

- It has 2 parts: allocated and used.

- Resource Pool and its children VMs can have independent settings.

Where to set reservation?

Frank Denneman recommends in this blog article that you "create a resource pool and set a reservation at the RP level. If a reservation is set at the VM object-level it has an impact on admission control and HA restart operations (Are there enough unreserved host resources left after one or multiple host failures in the cluster?"

### Demand

Just because you set a reservation does not mean it's actually consumed.

Reservation that is not yet consumed impacts capacity but not performance.

For example, a VM that uses 4 GHz but reserve 5 GHz should be considered as having 5 GHz, as that extra 1 GHz is guaranteed for that VM.

```
Demand = Max (Utilization, Reservation)
```

For VM, this needs to be calculated at individual VM, before being summed up at the ESXi level. Only powered on VMs are included. While the VM that is already provisioned can be turned on anytime, including them will result in overly conservative capacity.

A restaurant analogy will make the above clearer.

- Your restaurant has 2 floors. Equal capacity.
- First floor is full of diners eating.
- Second floor is empty, but it has been reserved for wedding. It's paid for.
- Capacity: Is your restaurant full?
  - Well, it depends on who is entering. If it's a guest wedding, the answer is no. If not, the answer is yes.
  - That means for each guest, you need to ask if he has reservation or not.
- Performance: can you serve the diners well?
  - Yes. The second floor is empty so your staff can serve the first floor easily.

Applying the above to vSphere, how do you know who are the reservation holders that are yet to consume what they are entitled to?

- For compute, those are VMs already powered on but have not consumed CPU and memory at their reserved threshold.
- For storage, there is actually reservation. It is just not called as such. The thin provisioned VMDK files that are yet to grow to their full size implies that it has reservation.

## Allocation

Allocation is the metric you use when selling the capacity to your consumers. If you charge half price relative to full price, then your overcommit should not exceed 2:1.

The total demand could be more than the visible demand, which is the active load that is consuming your capacity. There is demand that is not yet visible, because it has no utilization at present.

Use Allocation when you want to protect the shared infrastructure from sudden spike. It acts as a leading indicator.

Allocation is not in-line. ESXi and vCenter do not use allocation for actual resource scheduling. As a result, you can overcommit in allocation.

Allocation model also has usable capacity concept. Deduct the hypervisor overhead. This means VMkernel, vSAN, NSX, and vSphere Replication must be deducted from total capacity.

Allocation is only counted when the VM is powered on.

The CPU is based on thread, not core. So take note the performance degradation. Avoid promising the specific speed in GHz.

Take note the number for memory includes memory tiering, a new feature in vSphere 8. The overcommit is based on the physical RAM, meaning it does not count the NVMe device.

## Invisible Demand

Allocation is useful due to situations like the following:

| | |
|---|---|
| **Rare Demand** | This can wreak havoc in a shared environment. A group of highly demanding VMs can collectively impact overall performance of the cluster or datastore. An example of this is annual sales. In this case, the capacity team should set an appropriate overcommit ratio and drive by allocation as the demand is low most of the time.<br>A rare part of sudden demand is disaster, like stock market crash. It can't be predicted. Whether your CIO wants to pay in advance for such rare thing is a business decision, not within the call of Capacity Planner. |
| **Unexpected Demand** | Many critical VMs are protected with Disaster Recovery. During a DR drill or actual disaster, this load will 'wake up' and consume. You should consider the Site Recovery Manager Recovery Plans into your capacity.<br>Be careful with the complexity as 1 Recovery Plan can have many Protection Groups, yet 1 Protection Group can be included in many Recovery Plans. |
| **Potential Demand** | Many newly provisioned VMs take time to reach their full expected demand. It takes time for the database to reach the full size, the user base to reach the target, and the functionalities to be complete.<br>Newly provisioned VM tends to be idle (which can be months) and may suddenly grow. If you have many of them, plan for their eventual size. |
| **Unmet Demand** | There are 2 parts to it:<br>• inside the VM<br>• outside the VM. |
| | If the VM is undersized, the unmet demand will *not* be visible to the underlying infrastructure. Unless that is intentional, it is wise to include undersized VM in the cluster capacity monitoring.<br>The visible part of unmet demand becomes part of IaaS KPI and SLA, covered in Performance Management chapter. |

## Limitation

The allocation model has the following limitations:

| | |
|---|---|
| **VM Size** | VM size is not considered in the overcommit ratio. It assumes that scheduling two monster VMs is as easy as scheduling many small VMs. The ESXi scheduler |

| | |
|---|---|
| | can juggle higher number of small VMs than a few large ones, especially if they peak at different times.<br>Utilization is completely ignored. The consumer part is simply based on the configured amount. In cases where there is additional workload, the allocation model can report lower capacity consumed than actual. |
| **Overhead** | Any form of utilization is not considered, including consumption because of virtualization. For example, software-defined storage such as vSAN actually puts the availability protection data in the same datastore with the actual data. So you end up with double the consumption inside the datastore. |
| **IaaS Workload** | Agent VM is included as part of demand as it takes the shape of a VM, although it tends to use local datastore. |

### Overcommit

Overcommit is the main technique to reduce cost of shared infrastructure or shared service. So long the contention (real and risk) is acceptable, it reduces the cost to everyone. In daily life, queueing or waiting for service is common.

As a cloud provider, if you do not overcommit, you may not be able to compete on cost. Public cloud players (e.g. AWS) uses 1:1 overcommit for CPU but they count the thread (not just the core). They do not overcommit memory.

## Planning Phase

How do we put together the utilization, reservation, allocation and contention into a real-world example? Can this example include 3 class of services (gold, silver, bronze) for a more realistic implementation?

To start, we need to quantify the relative value of gold vs silver vs bronze. Keep it simple, so it's easy for tenants to understand the business values of each.

I recommend a 2x gap. It's easier to explain to senior leadership and application team. Operationally, it's easier to manage at scale than 1.5x gap or 3x gap.

2x gap means gold class is 2x better than silver class, and silver is 2x better than bronze. To achieve this promise, you put in place techniques such as reservation and allocation. You also guard the performance and keep the headroom accordingly.

| | Gold | Silver | Bronze |
|---|---|---|---|
| Relative Contention (slowness) | 1x | 2x | 4x |
| VM Size (configured, not utilized) | 4x | 2x | 1x |
| VM Reservation Rights | 4x | 2x | 1x |
| VM:ESXi Overcommit Ratio | 1x | 2x | 4x |
| Likely Cluster "headroom" | 4x | 2x | 1x |

You get what you pay for, which is fair and simple.
Gold Price ($) = **2x** Silver Price and **4x** Bronze Price

Separate what you promise (or sell) and how you deliver that promise.

- To the tenant, the gold class is 2x better than silver class. The chance of encountering contention is 0.5x silver. This is what you sell.
- You prove to the tenant that you assign 2x reservation, have half the overcommit ratio, and has 2x the headroom. These 3 are the techniques as you can't directly guarantee contention is half.

How do you implement the above technique?

The following table shows an example of. Notice the consistent 2x gap.

| | Gold | Silver | Bronze |
|---|---|---|---|
| **Reservation**<br>Relative to VM configured size.<br>Set at Resource Pool level, not VM. | 50% CPU<br>100% RAM | 25% CPU<br>50% RAM | 12.5% CPU<br>25% RAM |
| **Allocation**<br>Thread, not Core. Usable, not Total Capacity.<br>Memory Tiering is used beyond 1:1 overcommit | CPU 1 : 1<br>RAM 1 : 1 | CPU 2 : 1<br>RAM 1.5 : 1 | CPU 4 : 1<br>RAM 2 : 1 |
| **Contention**<br>Minimum Performance (%) threshold.<br>KPI (%), not SLA (%). | ≥ 99.5% | ≥ 99% | ≥ 98% |
| **Utilization**<br>Expected, not ceiling. It is what will happen due to above.<br>Against ESXi physical thread, not core.<br>Against usable capacity, not total (physical). | CPU ≈ 50%<br>RAM ≈ 80% | CPU < 70%<br>RAM < 90% | CPU < 90%<br>RAM < 95% |

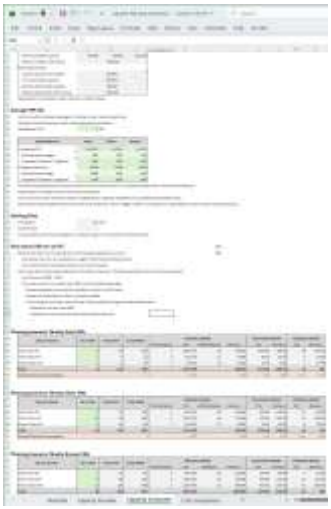
If the VMs are mostly Gold, there is a high chance that the utilization is well below 60%.

For Bronze, since you overcommit, the utilization becomes the upper limit. Do not go beyond this threshold as you run a risk of contention.

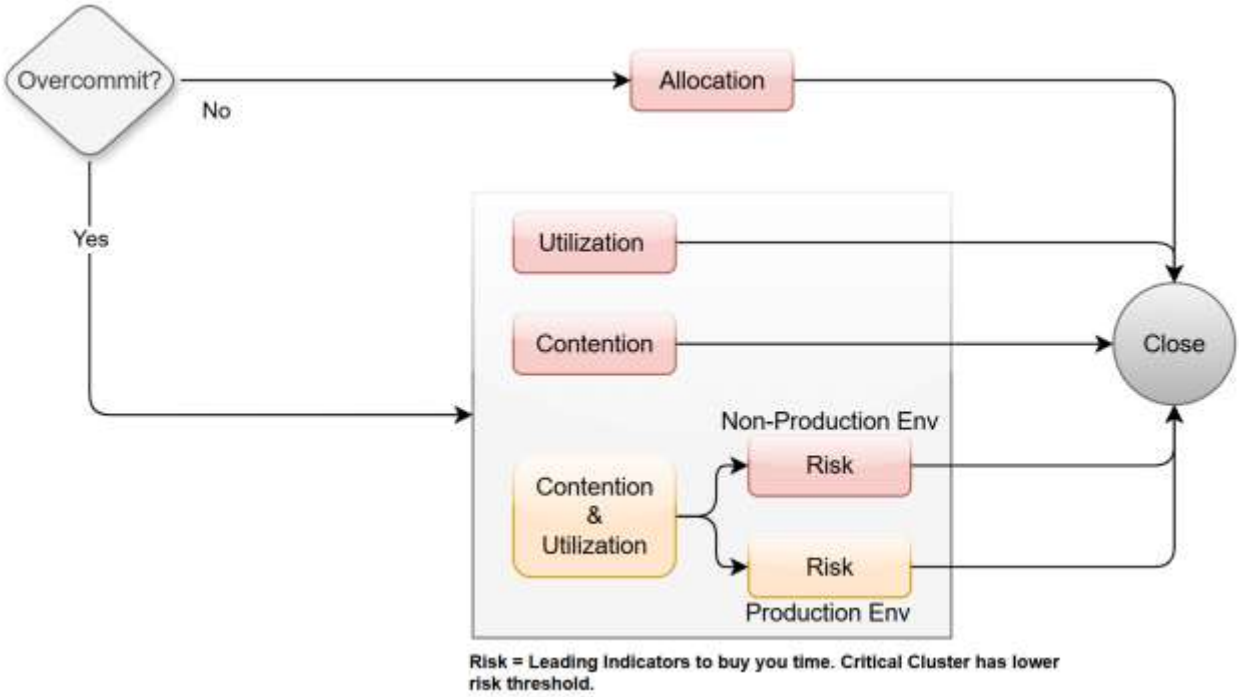
### Planning Template

I've created a Microsoft Excel spreadsheet to help you plan capacity based on the above model.

Download from isvara.io.

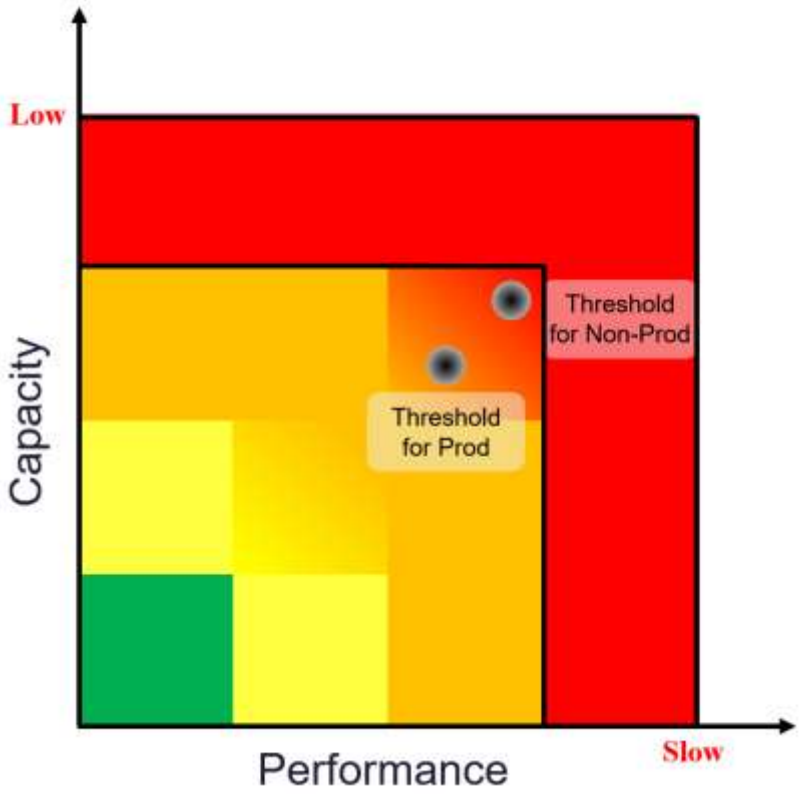


## Cluster Closure




Risk = Leading Indicators to buy you time. Critical Cluster has lower risk threshold.

Once you have the cluster capacity metric, performance metric, and risk metric, you can model a logic that automatically decide if the cluster is close for business. It won't accept new VM or allow any VM to be upsized. This paves the way for self driving capacity management.




## Logic

If either capacity or performance is red, then cluster is closed.

If both capacity and performance are orange, then check risk.

- For production environment: If risk is orange or red then cluster is closed.
- For non-production environment: If risk is red then cluster is closed.

All other permutations mean the cluster is open to accept new VM or VM upsize.

Implementing the above with actual threshold:

| | Prod Cluster | Non-Prod Cluster |
|---|---|---|
| **Threshold for Capacity Remaining**<br>Projected value, 3 days into the future | **Red: < 10%**<br>**Orange: < 20%** | **Red: < 15%**<br>**Orange: < 30%** |
| **Threshold for Performance (%)**<br>95th percentile for the last 24 hours | **Red: < 40%**<br>**Orange: < 60%** | **Red: < 20%**<br>**Orange: < 40%** |
| **Threshold for Risk (%)**<br>95th percentile for the last 12 hours | **> 50%** | **> 75%** |

Once you open the spreadsheet, the first thing you need to confirm is the size of your ESXi host.

The spreadsheet comes with default values that I think provides a good balance between cost and size.

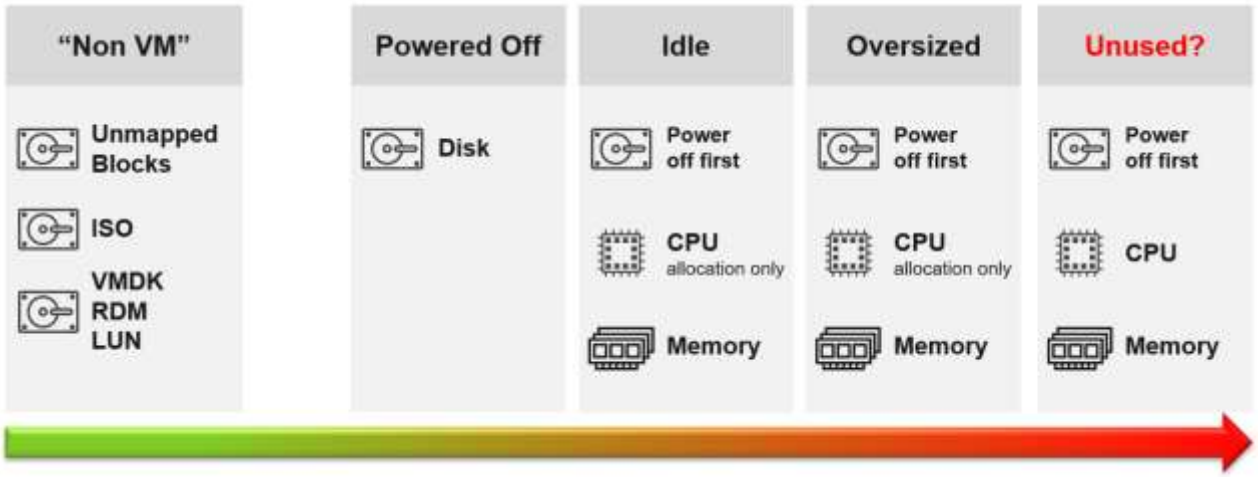
# Reclamation

It is only applicable when VMs are free. When VM is the main source of budget for IT department, the responsibility to reduce VM cost will naturally shift to the application teams as infrastructure team will not want their budget reduced.

Reclamation delivers many benefits, and some of them are listed below

| | Area | Benefit |
|---|---|---|
| **Unused VM** | CPU<br>Memory<br>Storage | This delivers the highest benefit, but it's also hardest to find as the VMs may not be idle nor undersized. They appear like an active VM.<br>Improve on the underlying IaaS capacity and performance.<br>Savings on storage only happen when you delete the VM. |
| **Oversized VM** | Memory<br>Storage | VM Performance. Especially if Guest OS does a lot of CPU context switch and VM size exceeds whole box total core counts.<br>The rest of benefit is the same with Idle VM as the portion you're reducing is idle. |
| | CPU | The oversized part is idle. So it only benefits allocation model. |
| **Idle VM** | Memory<br>Storage | Cluster Capacity, especially if you use allocation model.<br>Cluster RAM as idle RAM pages tend to occupy ESXi memory. One quick way to free up is to reboot the VM.<br>Negligible savings on CPU demand as idle loop hardly occupies real CPU cycles.<br>Savings on storage only happen when you delete the VM. |
| | CPU | Idle VM only saves you from allocation model. |
| **Powered off VM** | Storage | Datastore Capacity |
| **Orphaned VMDK** | Storage | Datastore Capacity. Does not impact cluster capacity |
| **Snapshot** | Storage | Datastore Capacity. |
| **Unmapped** | Storage | Datastore Capacity |

There are 5 areas of reclamation, from the easiest to the hardest. Naturally, the logic differs for each.




Non VM files are the easiest, because they are not owned by someone else. They are yours! Non VM objects, such as templates and ISOs should be kept in 1 Datastore per physical location. Naturally, you can only reclaim Disk, and not CPU & RAM.

An orphaned file is a file in the datastore that is no longer associated with any VM. Orphaned VMs and orphaned VMDK's are not even registered in vCenter. If they are, they may appear italicized, indicating something wrong. They may not have owners too.

For orphaned RDM, look from the storage array if there is any ESXi mounting it. You need an adapter for the specific storage you want to monitor.

Snapshots are not backups, and they do cause performance problems to the VM if kept for extended periods of time. Keep them only for the purpose of protection during change. Once the change is validated as successful, keeping the snapshot does a disservice to the VM. A Snapshot is easier to reclaim, hence VCF Operations lists them separately.

# Reclamation Approach

Active VM is politically the hardest, as they serve business workload. Focus on large VMs first. Take on CPU and RAM separately as they are easier to tackle when you split them. Divide and conquer. If you reduce both, and application team claim performance impact, you need to restore both. Claiming CPU and RAM from small VMs can be futile, regardless of idleness. An idle VM with one vCPU cannot be further reduced. Focus on the large VMs, for the reason covered here.

## Focus on Monster VMs

When reducing oversized VM or powering off idle VMs, focus on large VMs. Let's take an example for comparison:

- Reduce 20 large VM. Average reduction is 10 vCPU.
- Reduce 100 small VM. Average reduction is 2 vCPU.

In both scenarios, you reclaim 200 vCPU. But the large VM option delivers more benefits and is easier to realize. Here is why:

- Every downsize is a battle because you are changing paradigm with "Less is More". Plus, it requires downtime, which requires approval and change request process.
- Downsizing from 4 vCPU to 2 does not buy much nowadays with >20 core Xeon.
- No one likes to give up what they are given, especially if they are given little. By focusing on the large ones, you spend 20% effort to get 80% result.
- Large VMs are also bad for other VMs, not just for themselves. They can impact other VMs, large or small. ESXi VMkernel scheduler has to find available cores for all the vCPUs, even though they are idle. Other VMs may be migrated from core to core, or socket to socket, as a result. There is a counter in esxtop that tracks this migration.
- Large VMs tend to have slower performance. ESXi may not have all the available vCPU for them. Large VMs are slower as all their vCPU have to be scheduled. The counter CPU Co-stop tracks this.
- Large VMs reduce consolidation ratio. You can pack more vCPU with smaller VMs than with big VMs.

# Powered Off VM

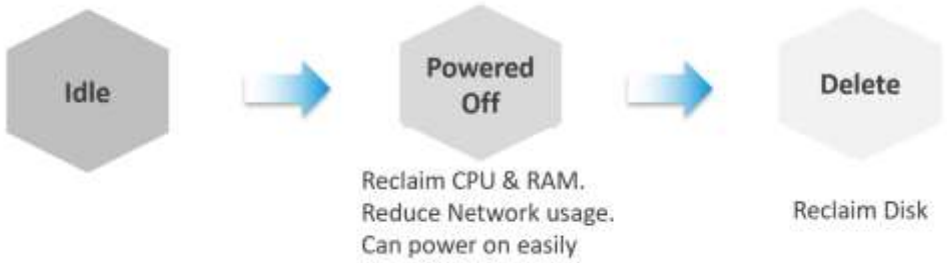
Compared with orphaned VMDK, Powered Off VMs are harder to remove, as there is now an owner of the VM. You need to deal with the VM Owner before you delete them. This is where tagging them with the owner email or Business unit would have been useful. We discussed proposed tagging in Chapter 1, specifically here.

There are different techniques to define power off:

- Non-Stop. In this technique, you want the VM to be continuously powered off. A quick power on to check something in the VM will remove the VM from powered off list.
- Percentile. In this technique, you can turn on the VM for a short period of time.

Each technique has their own pro and cons. VCF Operations use the non-stop as it is safer.

## Powered Off as Brake




Why do cars have brakes?

So they can go faster!

Take advantage of Powered Off as the brakes for your Idle VMs. If you treat Idle and Powered off as 1 continuum, you can power off the Idle VMs earlier. You get the benefit of CPU and RAM reclamation. It's a safer procedure too, as you can simply power it back on if you find that the VM is actually being used.

One major caveat if you do this, is the average utilization of the remaining VMs in the cluster becomes higher. As a result, you may not be able to achieve the overcommit ratio needed to break even.

# 2 Sides of a Running VM

There are two reclamation formula for running VM (idle or not). The formula is complex as it has 2 different stages:

| | |
|---|---|
| **Before** | Determine *if* the VM falls under the category. For example, does the VM qualify as an Idle VM? This should look inside the VM, as that's where the workload runs. Measuring at the ESXi level could yield incorrect results because:<br>• Some metrics include loads not generated by the VM that are charged to the VM, such as vSphere replication and vMotion.<br>• Some CPU metrics affected by power management. |
| **After** | Determine *what* can be reclaimed. Since what is being reclaimed is ESXi resources, the usage inside the Guest OS is irrelevant. The queue inside the Guest does not impact the hypervisor, so there is nothing to reclaim at the ESXi layer. All metrics are from ESXi. Guest OS metrics are not applicable as we're not reclaiming from inside the Guest. |

So you need to apply 2 different types of logic.

# Idle VM

By definition, idle means it's not doing useful business workload. A VM that is doing only non-business workload (e.g. AV scan, Windows regular update) should be considered as idle. This non-business workload is hard to detect via unless you have process-specific whitelisting. This is also not fool proof as some non-business software uses Windows or Linux system services as proxy.

Idle VM is a great target, as you can now claim CPU and RAM when you power them off. You cannot claim disk yet as you are not deleting them yet. Take note that you are not reclaiming real CPU cycle as it's idle to begin with. Idle VM does not actually consume any ESXi CPU cycles. So reclaiming a 10 vCPU VM running only 1 vCPU does not give you 9 vCPU. You are reclaiming blank air. For memory, you will reclaim real ESXi memory as idle VMs tend to have its consumed memory remained on ESXi.

Idle VM has a default threshold of 100 Mhz. This means 5% utilization in a single vCPU VM running on a 2 GHz ESXi. This also means 0.25% on a 20 vCPU on the same ESXi. The reason for static is idle by definition is absolute, not relative to the VM size. Oversized VM is relative.

While a VM uses CPU, RAM, Disk and Network, we only use CPU as a definition for Idle. There is no need to consider all 4, and require all 4 to be idle, because they are inter-related. It takes CPU cycles to process network packets and perform disk activity. Data from the network card and disk must be copied to RAM before processing, and the copying effort requires CPU cycles.

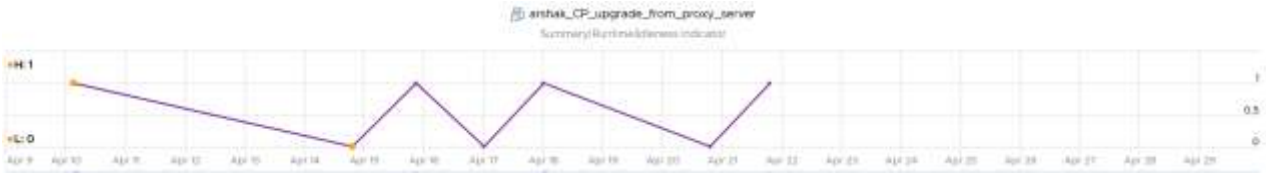
Take note of a corner case limitation of VM with runaway CPU, where CPU is high but no meaningful memory access, network transmission (TX) and disk processing. Idle VM will fail to detect it. It's a corner case, hence I think it's not worth the complexity. Also, the CPU runaway typically happens on a process, which likely a single threaded. Use the CPU Usage Disparity (%) metrics to detect that.

Idle has to be defined so it's measurable and not subjective. Declare it as a formal policy so you don't end up arguing with your customers.

VM that is rarely used can appear idle, if you measure idleness over a long period of time. For example, if a VM is only productive (from business viewpoint) for 2 hours a week, that means the remaining 166 hours should be classified as idle. That's 98.8% idle.

To counter the above, you want to evaluate idleness on a daily basis. A VM has to be idle every single day for 'N' number of days. This daily calculation is stored in the counter **Idleness Indicator**. It is a rolling counter. That means it is calculated every 5 minutes, but each value takes the last 24 hours of data. This is better than calculating only once a day so that VM does not have to wait before it gets declared as idle or not.

As you can see in the following example, its value is only stored if there is a change. This makes it easier to see when and if it changes.



Now that you have it daily, it's a matter of rolling up to the whole period (default value is 7 days). We set the value to just 7 days so you can see the calculation result within a week. Note that we set to 100% and we ignore newly provisioned VM.




Think of the various situation before extending from 1 week. For example, a VM has been in production for a few years. A new version of the application has been developed, and this VM is being decommissioned. The VM goes idle. As the application team does not inform infrastructure team, the VM will take at least 7 days before it's marked as idle. If you change that to 1 month, it will take longer.

On the other hand, a month-end VM that processes payroll can be idle for 29 days.

The counter that covers the whole period is called **Reclaimable Idle**.

It is a daily counter, that's set to 1 (true) if the VM meet the idle criteria.



To list the idle VM, you need both metrics to be true.

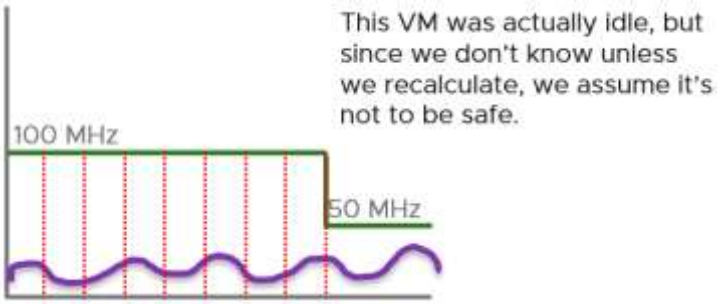
Why can't you just use Reclaimable Idle?

Because it's a daily counter. You can mistakenly assume a VM is idle even though it's recent activity shows it's not idle anymore.

In some environment, it can take time before a newly provisioned VM is used. Check the creation date of the VM before powering it off.

Have we got all cases covered?

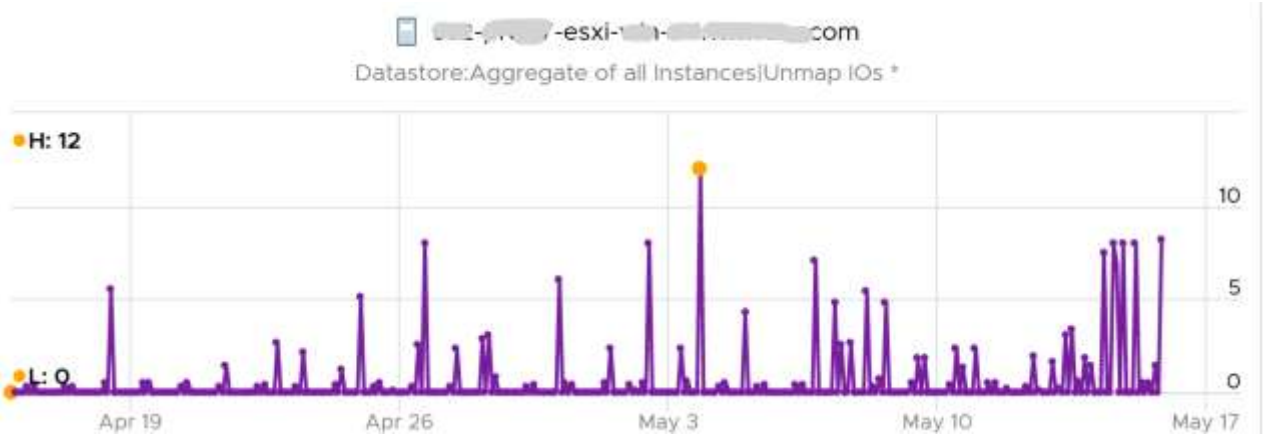
Nope. There is a corner case, where you tighten the definition (say from 100 MHz to 50 MHz). What was Idle may no longer qualified for idle. We can recalculate the daily metric, but this consumes performance. So to be safe, we will restart again from Day 1. So if the Idle Window is 2 weeks, customers have to wait 2 weeks.



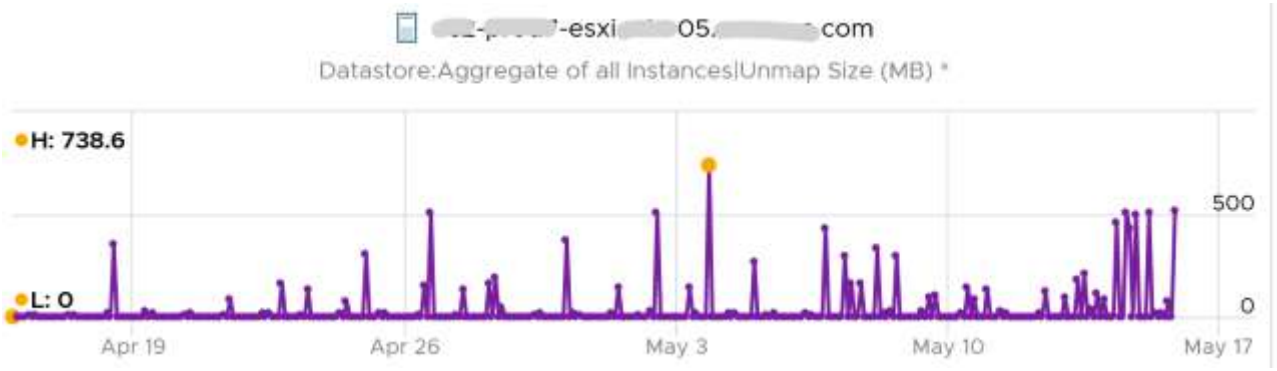

# TRIM and Unmap

When Guest OS delete files or parts of it, it does not replace the value with 0 and just leave the block. This is more efficient and also enable recovery. But this cause the underlying VMDK to grow. The same thing happens at the array level. This is where Trim and Unmap come in.

VCF Operations tracks the unmap operations via 2 metrics at ESXi Host. The first one is Unmap IO, which tracks the number of unmap SCSI instructions. For example, if the value is 100, that means ESXi has sent 100 requests of unmap to its datastore. So think of it like IOPS, except the IO is not writing/reading actual block, but more of a request to delete (unmap) the block in the back end array. The value is the sum of 20 seconds since vSphere reports per 20 seconds, then averaged over 5 minutes. In the example below, you can see the host sends unmap commands frequently in the last 30 days.




The second metric is Unmap Size, which tracks the total unmapped space from the operations above. The value is shown in MB.




You can track both operations on each datastore, but you can't aggregate them per datastore.

For further reading on TRIM and Unmap in vSAN, read this detail article by Patrick Kremer.

The problem only happens on thin provisioned disk. So if you want to check how much space you can reclaim, create a view that compare the value inside the Guest vs the value shown at VMDK level.

# Unused VM

Unused VM is not idle, but they do not provide *business value* anymore. The application team may have stopped using it, but left the application running just in case they need in the future. The VM is not idle as it still generates CPU activity. The activity can be business workload, IT workload, or both.

This makes unused VM much harder to find, as what works for VM 00001 may not apply to VM 10000.

The IT workloads take many forms. Guest OS upgrade, Guest OS patches, and application patches can be 3 different workloads with different patterns. VMware Tools patches, anti-virus scan, intrusion detection scan and agent based back up are other common examples. In an environment with high security, there can be many security related agents running. Or worse, they can be agentless, executed via network.

Business workloads can be batch jobs, reports or monitoring. No one is using the application anymore, but the application continues running. It could be generating report and send email to someone who just ignore that emails. This is harder to identify than the one running pure IT workload as it's more unique.

Unused VM is hard to detect as the infrastructure team lack the business context, and the patterns vary widely. The owner verification is required before you power off the VM. This is why it's important to have ability to relate a VM to a department or owner. We discussed the necessity of business-centric infrastructure in Part 1 Chapter 1.

There are some checks you can do to find the unused VMs when their CPU usage is not low. Find VMs that exhibit multiple of these behaviours. Take note that each of these can be false positive, so you need multiple of them for more accurate conclusion.

## Configuration

Configuration is easier to interpret than utilization, as they tend to have clear cut rules. The following list some configuration items you can consider.

| | |
|---|---|
| Isolated | It's no longer connected to a virtual NIC, so it's not communicating with other machines. |
| Guest OS | It's running older version, especially those nearing End of Life. It indicates the application team may have written a replacement applications running somewhere else.<br>Running old version of Windows, Linux and/or Kubernetes.<br>Expired license.<br>Temporary license. |
| Application | It's running older version, especially those nearing End of Life.<br>It's running application that you no longer license. In this case, there is urgency to power it off.<br>It's running without license, or with evaluation license, or expired license.<br>It has no business application installed. Just base OS + IT security applications. |
| Owner | It belongs to a folder in vCenter that is no longer owned.<br>Unable to figure out the tagging for the VM.<br>Request to contact the owner has gone unanswered.<br>Owner has changed organization. |
| Report | The application is no longer listed in any of the reports to the department owning it. This could be performance report, capacity report, compliance report, chargeback report, etc. |
| Location | It runs on a cluster that is due for decommission. It is stored in a datastore, that sits on an array that is due for decommission.<br>Its folder has names like "old", "decommissioned", and "archived". |
| Relationship | The VM is talking to other VMs. You can check which services is talking at which port. Check what services it's using over the network. This is useful for VMs in the cloud, which uses cloud services from AWS, Azure, etc.<br>As these neighbors could also be unused, using this alone is not reliable. |
| Alert | Alerts associated with VM have disabled. |

## Utilization

| | |
|---|---|
| CPU | While usage is not idle, or even high, it's the same CPU. There is very little context switch, indicating the same processes are running. If the CPU Usage Disparity (%) metric is stable, that indicates constant run. |
| Memory | While the In Use metric is high, it's passive. There is lack of paging, both in and out. |
| Disk | While IOPS and throughput are not low, its filesystem is stable, rarely changing in both size and activities.<br>The IOPS and throughput also eventually form a pattern over the long run. |
| Network | The network it belongs to is no longer reachable, or has been isolated. That means access to it is non-existent or highly restricted.<br>The VM sends very little packets out, indicating it is not talking much on the network. At the same time, it's only talking to a fixed and small group of other |

| | |
|---|---|
| | servers. Take note that secure applications that store data typically are restricted. |
| | It belongs to a VLAN that is due for decommissioning. |
| Log | The amount of logs or Windows Event is much lower relative to its peers. The pattern is also predictable over time. |

## Other Signs

| | |
|---|---|
| No login | No one has ever logged into the Guest OS, be it from UI or the console (e.g. SSH into Linux) for a long time.<br>If user log in, it's for a very brief period of time. |
| Process | It's the same set of processes that are running. The number of processes also remains steady and predictable.<br>The process that takes up the most CPU is not business software. It's system process or IT application. |
| Availability | It never gets rebooted, or it gets rebooted often. Essentially, it seems like no one cares about its state.<br>Reboot happened during business hours and no one complained. |
| Performance | Severe performance issue and you don't get a complaint. If someone owns it, you will get a formal ticket if the performance is terrible for a long time during business hours. At night, a VM can experience slowness for 1 hour and no one may notice. |

Bottom line, what can you do if you are *unsure*?

You've announced to everyone that the VM would be deleted if no one claims it repeatedly, yet no one replied. Anything safer you can do than powering off the VM, as powering off disrupts the running process and close opened files? Powering off does not guarantee that it can be brought online successfully.

You have 2 choices:

- Disconnect it from the network. This makes the VM isolated, without shutting down the application. If it's used, the VM Owner will know it.
- Apply CPU limit to it. This slows down the VM. The VM Owner will feel the impact and complain it's slow.

When you do that, ensure you tag the VM or move them into a "Unused VM" folder. If these unused VMs spans multiple vCenter servers, use VCF Operations custom property. If there are many of them, create a dedicated dashboard so you can see them at a glance.

## Annual Stocktake

Unused VM is hard to detect. If you don't have the VM owner, perform a stocktake on those unidentified VMs.

Stocktake is applicable if your IT business is not profit oriented. This means it applies to internal IT department even though you have chargeback as you aim to be a good corporate citizen.

The stocktake actually starts from Day 0, where VM is being requested. Make expiry date mandatory, to catch those temporary VMs. For permanent VM, set it to 1 year. If you set beyond 1 year, you increase the risk of change of owners and you lose the contacts. Reorganisation can result in the department or team owning the VM no longer exist. What was meant to be permanent VM suddenly becomes unused VM.

You need to have a process to keep unused VM in check. Have a simple process so that you can get an agreement from all your customers. As business owners may not know the VM name, include the following information

- Hostname. Take the one from inside the Guest OS, not from vCenter.
- Application name
- IP Address. They may login with IP address and it rings a bell to them.
- Guest OS name and version.
- vCenter Folder name. This should be the business unit it belongs to. See Part 1 Chapter 1.
- 95$^{th}$ percentile utilization in the last 3 months.
- Any other information and context you think will help them remember it's their VM.

# Rightsizing

What do you rightsize? Not all objects are relevant for rightsizing. Take for example, an ESXi host. Once you buy it, you rarely change the size over the lifetime of the server. Same with datastore and vSAN.

Typically, what you rightsize is VM and Kubernetes.

Let's dive into VM. Why so many oversized VMs?

Over Provisioning is a common malpractice in real life SDDC for these reasons:

| | |
|---|---|
| **Legacy** | Physical machine was P2V, bringing its configuration as it is. |
| **Cost** | The price is low for the business paying for the VM. The private cloud is either free or much cheaper than public cloud. |
| | No progressive pricing. An 10 vCPU VM costs exactly 10x of 1 vCPU VM |
| **Education** | The mindset that bigger capacity means better performance is hard to change. |
| **Vendor** | Some sizing is dictated by the vendor owning the commercial software. They will not support if you deviate from it. |
| **Micro Burst** | This is the hard part. Some applications have sharp but short CPU bursts. They only last a few seconds, so a 20 second averaging fails to show them. |
| | Highly volatile burst typically does not apply to memory. |
| | **Solution:**<br>Collaborate with VM owners since they have business transaction level monitoring.<br>Use the 2 second metric in VCF 9.1. |

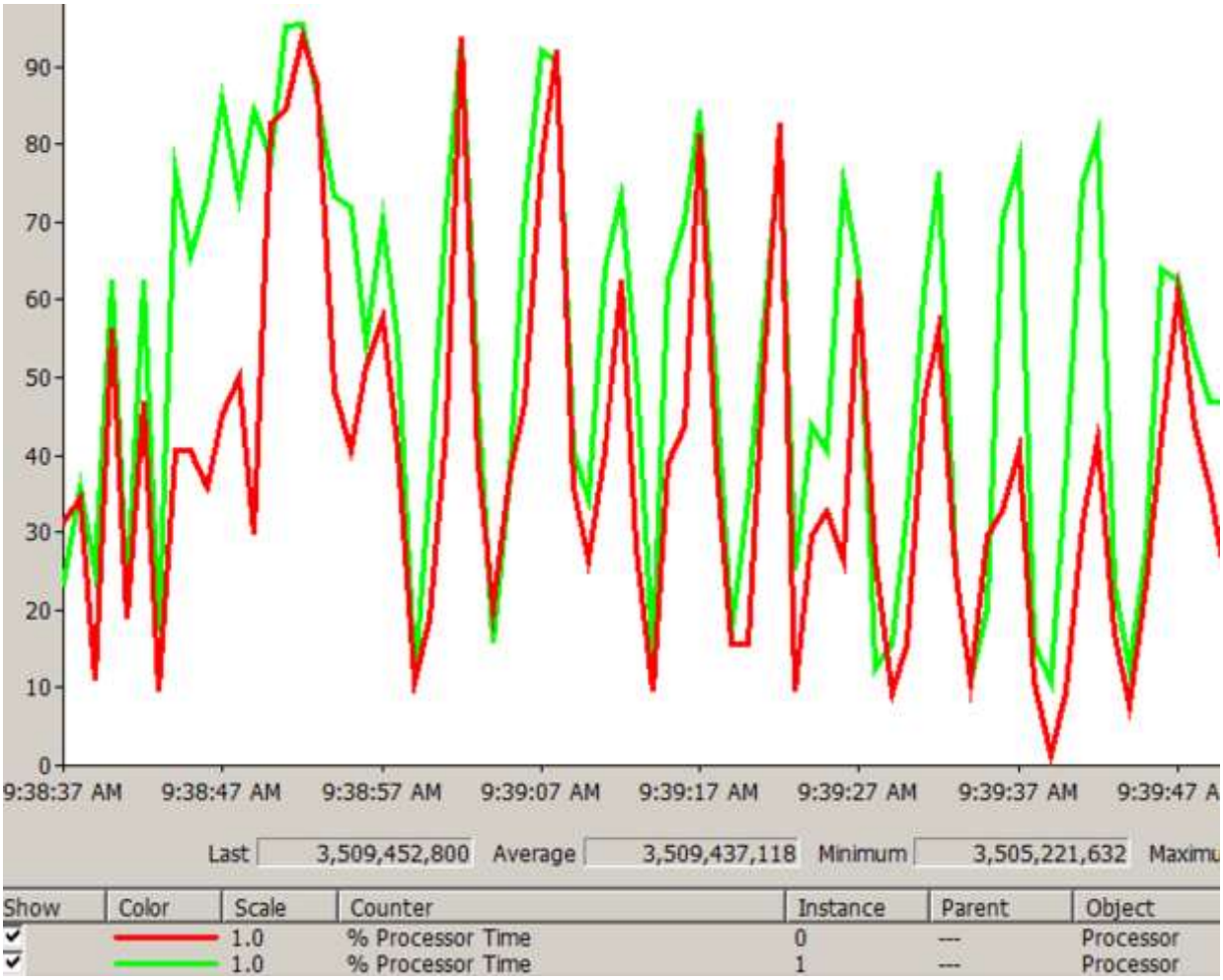
# Challenges

Taking away resources from VM owner is notoriously difficult. The political science part is harder than the rocket science part.

| | |
|---|---|
| **Fear** | Will it be slow after the VM downsized? |
| | You need to prove, using metrics, that there is no performance degradation.<br>What if the slowness is caused by other factors? It is possible that other factors other than CPU or memory were the actual culprit. How do you prove it? |
| | **Solution:**<br>Establish a formal and transparent process where VM owners can see their VM performance and usage pattern before and after. The comparison should span 1 month just in case there is month end peak. |
| **Paid For** | If the VM was already paid for, how do you position it, so the original size did not look like a mistake by other people in their planning? You don't want to come across correcting other departments. |
| | **Solution:**<br>Project them as the real hero that saves the company, and the infrastructure team as just the facilitator. |

## Micro Burst

In the following screenshot of Windows Performance Manager, the 2 CPU shot up to >80% for just 1 – 3 seconds.




The data point above is per second. If you average the number over 20 seconds, it will show 50%. If you downsize to 75%, you will likely have higher CPU run queue.

There are 2 main approaches to identify the applications:

- By default "No".
  The onus is on the application team to inform the infrastructure team that their application has short and sharp CPU burst.
- By default "Yes".
  The infrastructure team conducts a company wide scan. Since agent cannot be used, your choice is esxtop, VCF Operations 9.1, or build your own adapter. Since 2 seconds results in high amount of data generated, limit the number to 1000 data points to avoid impacting the system being monitored.

# VM Footprint | Guest OS Sizing

VM right-sizing is a commonly misunderstood term because there are actually 2 distinct formulas: 1 internal and 1 external.

- Internal means sizing the Guest OS, which lives inside a VM. Let's call this Guest OS Sizing.
- External means sizing the footprint of the VM to the underlying ESXi and infrastructure. Let's call this VM Footprint.

To see the difference between internal and external sizing, let's examine a few popular use cases:

- Your application team asks for extra vCPU. In this case, the hypervisor overhead is irrelevant. When you size NSX edge vCPU, you do not need to add extra vCPU for the overhead to do the packet processing. This means you're sizing Guest OS.
- Infrastructure team is migrating a VM to a new ESXi with CPUs that are 2x the speed. For example, from a 2 GHz ESXi to 4 GHz. All else being equal, you can cut down the VM size by 2. A 16 vCPU becomes 8. In this case, the hypervisor counter is more accurate.
  However, you are worried about causing queue inside the Guest OS as the application may expect 16 slow threads vs 8 fast ones. In this case, you need to look inside Guest OS but do that after rightsizing.
- You're migrating a VM from classic VMFS to vSAN. In this case, use the hypervisor metric as it will be adjusted according to vSAN FTT policy.
- You're converting VM virtual disks from thin to thick. In this case, the consumption at Guest OS level is irrelevant as you're inflating the virtual disk into its configured size.

From the above use cases, we need two different formulas:

| Purpose | Method |
|---|---|
| Guest OS Sizing | Using Windows/Linux counters. Excludes VM overhead, includes Guest OS Queue.<br>Used to size the "VM", meaning the CPU and RAM requirements of Windows or Linux. For disk, that means the size of the Guest partitions, but expressed in terms of virtual disks. |
| VM Footprint | Using vSphere counters. Includes VM overhead, excludes Guest OS Queue.<br>Used to size the "infrastructure footprint" of the VM |

Once we know what the VM needs, we need to project based on past data and recommend the new size. The new size is then adjusted to comply with NUMA.

You'll see below that CPU, RAM and storage all require different approach.

# Oversized Guest OS

Oversized VM depends on the size of the VM. A 64 vCPU VM running 7 vCPU is oversized, while an 8 vCPU running 7 vCPU is not.

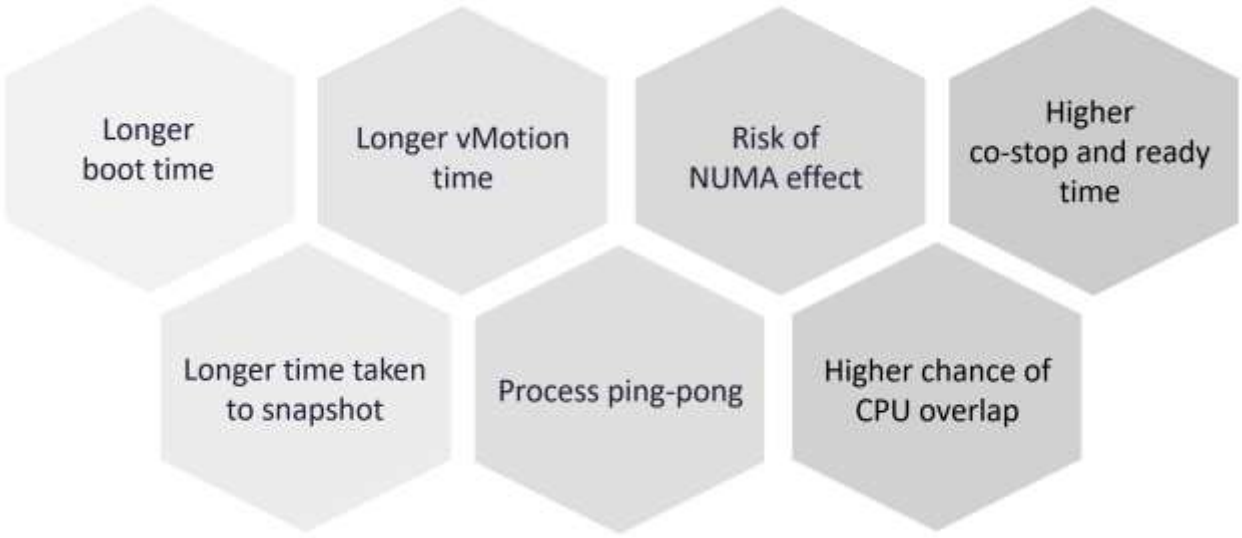
| | |
|---|---|
| **VM Is undersized** | Calculated based on CPU & RAM total capacity and recommended size values. |

| | |
|---|---|
| | If for at least one of the containers (CPU or RAM) the recommended size > total<br>The lowest value for increasing the CPU is 1 vCPU and for memory is 1 GB |
| **VM is oversized** | The VM is oversized if it is possible to reclaim a CPU or Memory.<br>Calculated based on CPU & RAM total capacity and recommended size values. |

Note: the implementation in VCF Operations is based on projection, not a mere 5-minute or 1 day data. So if you power off an oversized VM, it remains considered as oversized until it passed the definition.

## Cost of Oversized

More CPU, memory and disk do not translate into faster performance. In fact, it carries additional overhead.




## Optimization

In the consumer layer (process, guest OS, container, VM), optimize the following:

| | |
|---|---|
| **CPU** | Use CPU Run Queue as primary counter, as utilization should be 100% to minimize ping pong and NUMA.<br>Make sure all the CPU is used well by checking the CPU Usage Disparity metric, as some applications tend to gravitate towards the first 8 vCPU. |
| **Memory** | Utilization should be near 100% as it contains cache (majority of pages not active). Check page fault to see if there are excessive page fault. |
| **Disk** | Rightsize the filesystem. Note this requires Windows or Linux partition modification.<br>Reduce the usage of RDM. If you do, and you use thin provisioning at array level, check for wastage using unmap. |

# Best Practices

I've seen large enterprise customers try to do a mass adjustment, downsizing many VMs, only to have the effort backfire when a tiny percentage of VM performance suffers. Take time and effort to educate the VM Owner that right sizing actually improves performance, despite the seemingly odd logic. A carrot is a lot more effective than a stick, especially for those with money. Saving money is a weak argument in most cases, as VM Owners have paid for the VMs.

| | |
|---|---|
| **Hardware** | Map to the underlying hardware (CPU, RAM) architecture.<br>AMD EPYC and Intel Xeon use an 8-core block.<br>A VM with > 8 vCPU risks hyperthreading effect.<br>A VM with > 16 vCPU risks NUMA effect. |
| | NUMA aware. Configure the vSocket x vCore following the hardware. |
| **Performance** | Rightsizing is not about capacity. The capacity is there to ensure performance, especially during peak times. |
| | **Solution:**<br>Include contention metrics in the formula. For time-sensitive business transactions, measure at this level and Guest OS level. It needs to be sampled at 20 second interval, not 5 minutes. |
| **Collaborative** | Agree upfront on the metrics and methods to quantify performance. Ideally do this before relationship turns defensive. |
| | Agree on the date of the change. Make it a joint execution. |
| | **Solution:**<br>Make a dashboard where everyone can see Before vs After for each VM that was rightsized. The comparison is the proof that performance is not affected and capacity remains sufficient.<br>Agree on the process to roll back. Ensure during the process, no other changes, especially data-level and application-level changes. |
| **Show big picture** | While a few VMs may not attract the attention of the C-level leaders, the total may be financially significant. |
| | **Solution:**<br>Show a company wide number showing the excess. Report regularly and send it to all stakeholders. |
| **Encourage small** | Small VMs are immediately provisioned, available via self service.<br>Larger VMs require more justification and management approval. They are also subjected to regular review of their consumption. |
| | **Solution:**<br>Progressive pricing. Discount for small VMs is subsidized by premium pricing of large VMs. |
| **Application-aware** | Certain applications such as Java VM and databases manage their own memory. Exclude them as Windows or Linux is unaware of their resource management. |
| | Kubernetes Node does not run applications directly. They run containers, which in turn run the processes and threads. |

| | |
|---|---|
| | **Solution:**<br>Exclude them from the standard formula. Work with the DBA or K8 SRE. |
| **Open** | Transparency build trust as it's fairness to all applications. Have a public dashboard that any application owner can use. |

## The Big Picture

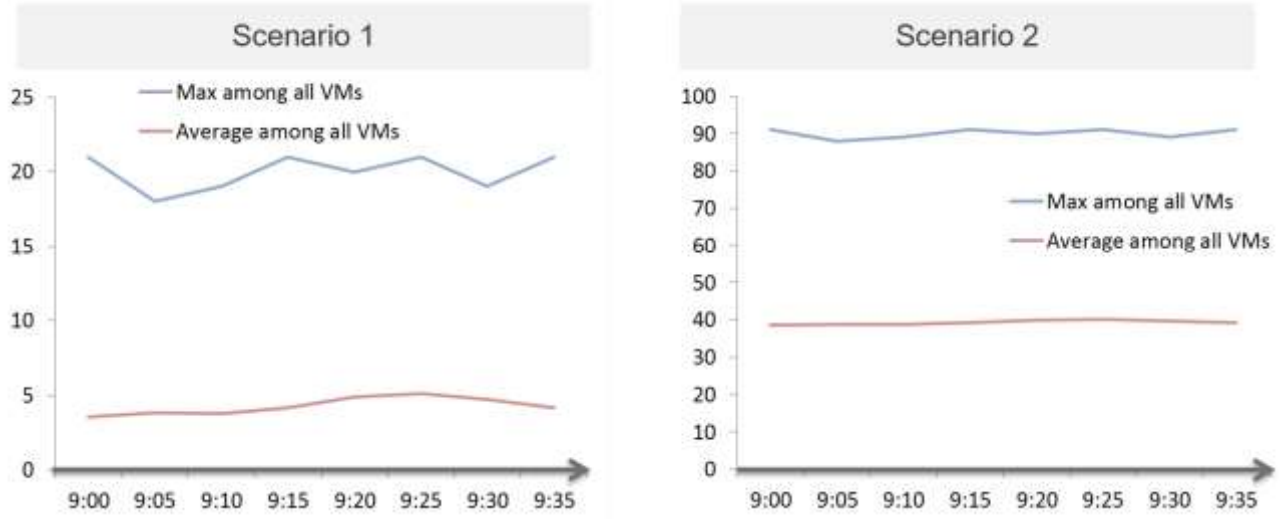
If you have thousands of large VMs, how do you communicate easily to your senior management that many of the large VMs do not use the CPU given to them in the last few months?

You need to present a convincing chart, that shows the utilization of hundreds of large VMs (which you defined as having > 16 vCPU) every 5 minutes, so a short peak is not excluded in your presentation.

The first thing you need is to create a dynamic group that captures all the large VMs. Create 1 group for CPU, and one for RAM. You then plot their utilization, every 5 minutes, in the last 3 months.

In a perfect world, if all the large VMs are right sized, which scenario will you see: scenario 1 or 2?

Both scenarios show the average CPU utilization of the large VMs.




That's right. Scenario 2.

Because the group has hundreds of members, there is a good chance that one of the large VMs is using the CPU given to it. On average, they should be hovering around 40 – 50%, as at any given 5-minute interval, some may be idle while others may be busy.

The technique we use for both CPU and RAM are the same. I'd use CPU as an example.

Once you create a group, the next step is to create two supermetrics:

| | |
|---|---|
| **Maximum()** | Maximum CPU Workload among these large VMs.<br>You expect this number to be hovering around 80%, as it only takes 1 VM among all the large VMs for the line chart to spike.<br>If you have many large VMs, one of them tends to have high utilization at any given time. |

| | |
|---|---|
| | If your Maximum line is constantly ~100% flat, you may have a runaway process. To find out which VM, list the VMs and set the 95th percentile of the time period you're interested. The runaway VM will be at the top showing 100%. |
| | If this number is low, that means a severe wastage. |
| **Average()** | Average CPU Workload among these large VMs.<br>You expect this number to hover around 40%, indicating sizing was done correctly. |
| | If this chart is below <20% all the time for the entire month, then all the large VMs are oversized. |

Why is it not needed to create the Minimum?

There is bound to be a VM who is idle at any given time.

The 2 line charts show us the degree of over provisioning. Can you tell a limitation?

It lies in the counter itself.

We cannot distinguish if the CPU usage is due to real demand or not. Real demand comes from the application. Non-real demands come from the infrastructure, such as:

- Guest OS reboot.
- AV full scan.
- Process runaway. This can potentially result in 100% CPU Demand if the application is multi-threaded. How to distinguish a runaway process from legitimate high workload is the challenge.

## Progressive Pricing

How do you prevent oversized VM to begin with?

Hint: why doesn't cloud providers like AWS have oversized VM issue?

They have no issue as it's good for their business. In fact, their profit margin is higher on oversized VM.

One effective solution is progressive pricing. We cover this in Part 1 Chapter 5 Cost & Price Management.

If you do not charge for VM, then you're left with official approval and corporate policy. For example, the bigger the VM, the higher the approval chain. You can also make the form more complex, needing more justification for monster VM.

Regardless of the pricing, make sure each VM has life span. While they can live forever, they are subjected to annual confirmation that they are still required by the business.

## Start Small?

Considering the above problem, how do you prevent the problem to begin with?

One idea is to give every VM a minimal size regardless of their requirements. As this standard is small, majority of VMs will end up needing an upsize over time. So you need to be prepared for CPU Hot Add and memory Hot Add.

There are a few things to consider before taking this approach:

- It goes against the service provider business model. This is classic System Builder, where IT acts as the infrastructure architect, getting involved on VM sizing discussion. Ideally, you use price as your primary lever for sizing, as you may not be familiar with the load of their applications.
- Upsizing logic is more complex than downsizing. You need to consider NUMA impact on performance. The maximum size also depends on the ESXi hosting the VM.
- It can be abused. A synthethic load can be added in the code. Counter this by having a continuous and long-term monitoring.
- Upsizing needs to be more responsive. While application team can tolerate weeks before you downsize their VMs, they probably want their VM to be upsized within the same day. And if performance is affected, they may even ask for it to be done within an hour or so.

## Before vs After

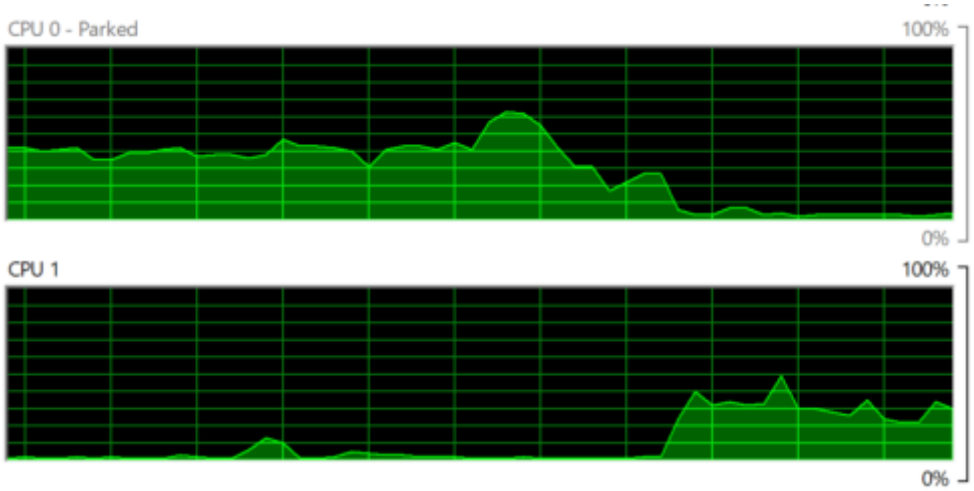
Since you're reducing CPU and/or memory, it's essential to show the key statistics before and after the changes.

### CPU

The overall utilization should remain the same. If the CPU cycles drop (in GHz), increase the share instead of adding back the vCPU.

If the usage was spread across all CPU, the remaining CPU will likely show higher utilization.

If the usage was uneven, the remaining CPU will show similar utilization. In the following Windows machine, the CPU basically took turn to run.




The CPU run queue should not go up. It should remain the same. If it does, check the thread states metric.

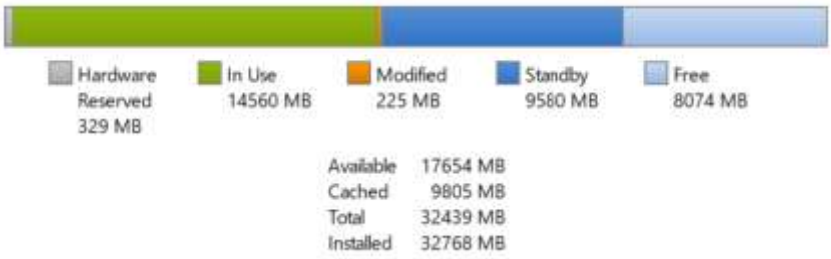
The CPU context switch should go up, if the application runs many threads. Ensure the increase is negligible.

### Memory

When memory is reduced, you will likely see less free memory, and more active swapping.

If the VM is large and suffers from NUMA, the Local NUMA metric should go up. This should improve performance, especially on memory intensive applications.

Using Microsoft Windows as an example, you will likely the Available (MB) metric drops. This is fine as the memory is not used. It is not deleted as deleting it serves no purpose as there is no demand.




### Other Changes

Changes in CPU and memory utilization can be caused by disk and network demand. Ensure you're comparing apple to apple by plotting disk IOPS, disk throughput, network throughput and network packets/second.

# Sizing Logic

| Rule | Description |
|---|---|
| **It's not just utilization** | It needs to consider unmet demand. CPU wants to run, but it cannot. Memory has lots of page faults in Guest OS memory. |
| **It's not just demand** | Size base on what the Guest OS needs to perform well, not just base on what it demands at present. Applicable for RAM, where Guest OS can't operate optimally without buffer.<br><br>In capacity, we size not just for demand, but also for performance. While we can satisfy the demand for memory with just the In Use, it might come at the expense of performance. The only thing faster than memory is CPU. So make sure CPU is not waiting for data. This is done by caching as much as possible, as it's hard to predict what pieces of data is required by the program. |
| **Includes peak** | Consider the busy or peak period, because that's when the VM needs to work the most. |
| **Consider big picture** | A single 5-minute burst is too short a timeframe to determine the entire next 3 months. Consider long term pattern. This alone makes sizing an art, as you need to know the nature of the workload. |
| **Excludes IT load** | Exclude the time when the Guest OS is not doing business workload. There are a few IT workloads that cause high utilization. Common ones are Guest OS reboot, Guest OS updates, anti-virus full scanning, agent-based full back up. So long as these tasks don't prevent the Guest OS from doing useful work, you can exclude them. The exception is when your VM needs to run during these non-business hours too. So it depends on the VM. |

| | |
|---|---|
| | This is the hard part, as it requires awareness of the footprint (read: process name) |

Sizing upwards and downwards should have identical consideration.

- The only difference is they have different boundaries. The lower boundary applies to downsizing, and the upper boundary applies to upsizing.
- For downsizing, Guest OS needs a minimum amount of RAM to operate.
- For upsizing, consider the NUMA boundary. Also, a VM should not be larger than the total number of logical processors on the ESXi Host, else it won't even boot. In fact, it should be smaller as you want to account for the VMkernel overhead.

As you can see from above, sizing is complicated. And the above is just Guest OS. We have not considered other things that need sizing such as Containers and Business Applications.

The art of sizing has 2 parts: time and metric.

- First, we calculate the value for a given point in time. The correctness of the input value matters, else you have GIGO effect.
- Second, we plot thousands of these values over time, and project it over time. The projection has to consider the peak cycle, meaning it has to be geared towards conservative sizing. It also has to consider the business cycle. If you have annual sales, then consider annual data.

# Migration

The arrival of the cloud makes migration more common than before. While the destination differs, the sizing methodology does not. There are many examples of migration. Popular ones are:

- From old DC to new DC.
- From on-premises to cloud. This is typically VMware-based cloud as you can simply move without changing VM. Examples are Amazon VMC and Microsoft AVS
- From Cloud to on-premises. This is typically due to high cost. It's hard to beat owning with renting if you apply a 5-year TCO. Cloud used to give newer hardware, which is no longer the case.

In the above, you typically change all infrastructure. New server, new network, new storage, new SDDC. You may virtualize network & security by adding NSX. You may also virtualize storage by going vSAN.

Migration ranges from a simple 1:1 to complex M:N migration. It also ranges from a single cut over done over the weekend to multiple migrations lasting years. Destination can be on-prem (e.g. cluster upgrade) or cloud. I've seen both directions.

# Challenges

Regardless of the migration type and scope, there are some common changes and basic requirements. These could create challenge in the migration project.

## Faster Destination

You are using faster & bigger hardware. You have higher CPU speed, more CPU cores, faster RAM, faster storage, bigger network, less network hops, etc. It's faster by at least 4x compared to the ageing environment it replaces.

And that's exactly where the problem might *start*.

A VM that takes 8 hours to complete its batch job may now take 2 hours, all else being equal. So it completes the same amount of work, doing as many disk, network, CPU, memory operations in 4x shorter duration.

So what happens to the VM IOPS? Yes, it went up by 400%, all else being equal.

What happens to VM CPU Usage? It also went up by 400%, as it has to complete the same amount of logic. Suddenly, a VM that runs relatively idle at 20% becomes highly utilized 80%. What was an oversized VM has become an undersized VM.

I call the above as Performance Multiplier. Unfortunately, it's hard to guess the impact. This is why you are better off testing with a few well known VM first, such as infrastructure VMs or applications that are owned by IT. Examples are email servers, file servers and your Active Directory services.

Because of the above, my recommendation is to keep the VM size. Do not rightsize and migrate at the same time. If you change the size, you will be in the defensive position if there is performance issue.

## Different Architecture

The destination could be using software-defined storage and network. Both vSAN & NSX consume ESXi CPU and memory, not to mention storage and network. You must also be aware that certain vSphere disk space counters got affected by vSAN FTT policy.

If you're migrating to the cloud, take note that the management load and SDDC load, such as vCenter and NSX Edge appliances are also residing on the same cluster. The unique nature of VMware-based cloud migration creates situations that you need to address. For example, you need to watch Elastic DRS if you do not want it to get triggered.

## Cost Pressure

How do you typically justify the budget for the new infrastructure, since it's both faster and bigger?

Yes, you promise higher consolidation. You have more CPU cores, more RAM, so logically you use higher over-commit ratio. As Mark Achtemichuk said in this article, use it carefully.

Since you have to increase overcommit ratio, how do you then prove that performance will not be affected as you drive utilization higher? That calls for a Before vs After performance comparison.

Enterprise IT (read: Infrastructure Team) is also using the opportunity to right-size VM, but VM Owners are against downsizing. How to down-size VM without impacting performance?

## Long Migration Project

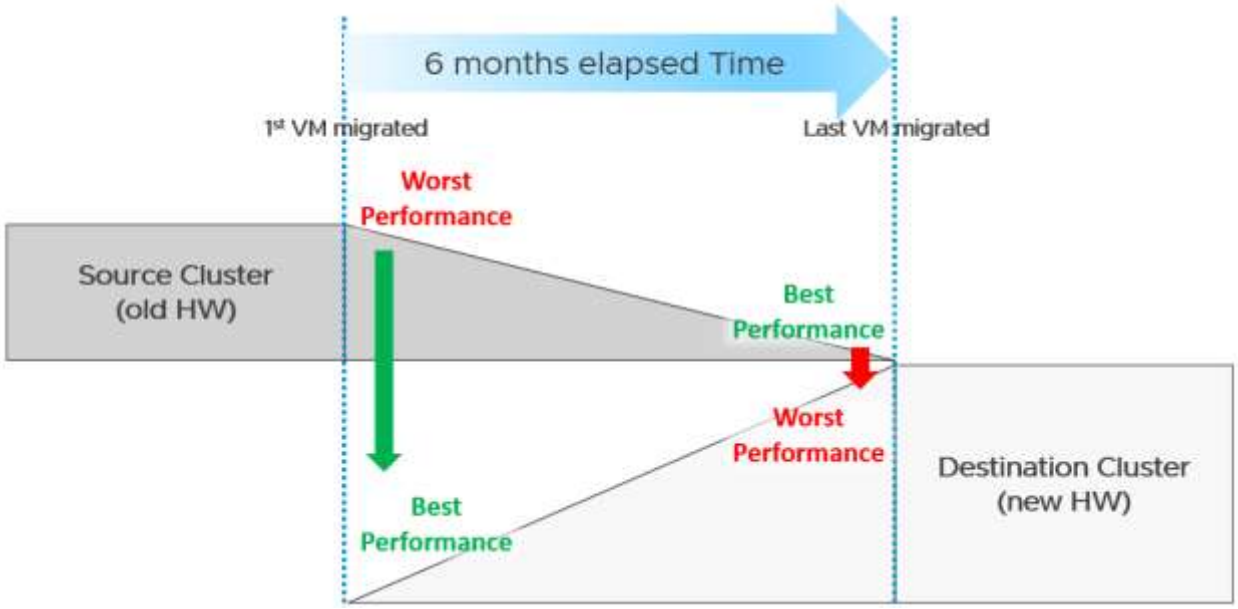
The problem with project that lasts months is things change. The application may change due to business or technology requirements. The people (application team, infrastructure team, management team) may change and politically this can complicate matter. In large scale project, the inter-DC pipe can be a choked point if large number of VMs on both sides are communicating at the same time. For example, if you only have 10 Gb/s bandwidth for inter-DC, it may not be enough when you have 500 VM on Site A + 500 VM on Site B using the pipe. You essentially only have 10 Megabit/sec per VM. You can try to group the applications, but you can't control if they change.

This is why I'd rather choose intensity over time. Migration is not one of those projects that are best done slowly.

The following shows an example where the migration takes 6 months.

I drew 1 source cluster and 1 destination cluster. In reality, there can be many to many relationships.

The source cluster has both less capacity (hence smaller area) and slower performance (hence darker color). This is a typically scenario as the new hardware typically delivers faster speed and more space for the same cost.




What potential problem did you spot?

This long migration period created an undesirable situation where the first few VMs enjoyed the whole cluster. They could run with 0 contention as there was enough resource for everyone.

Application team felt the responsiveness. Everyone was happy. They might start new features or load the systems even more. Overall utilization went up. So far so good as there was enough capacity.

As more and more VMs get added, the new infrastructure hit the point of *overcommit*. At this point, the VMs would begin experience contention.

On the other hand, the remaining VMs in the old clusters began to experience less contention. So their performance actually improved. The application team felt good, and started taking advantage of the newly found performance. They begin changing the application or increasing the data size. The users' expectation also went up as they can do more work. Everyone is feeling more productive.

The last VM to be migrated might get a shock. The performance might actually drop from the end user's viewpoint.

The above could result in mismatch expectation. This is why SLA matters. You also need to have the SLA agreed *prior* to the migration. Do not rely on users' complaint or user-level metrics as that's beyond your control.

## Best Practices




Migration is best done as soon as possible, ideally in one migration window. This minimizes inter-DC traffic. For example, if VM 1 talks to VM 2 and VM 2 talks to VM 3, if you somehow forgot to migrate VM 2, you have a ping pong traffic. The latency and bandwidth could cause application performance. So pick the longest window you can get, such as major public holiday or company shutdown.

Migrate 1:1. This means 1 source cluster and 1 destination cluster. Obviously, you exclude the powered off VMs 😊 This makes migration management, and VM troubleshooting easier.

Within a week, get a sign off. Before sign-off, do not allow changes as that can make comparison invalid. Changes include application, business functions and infrastructure.

Do not right-size using the utilization data of the old Data Center. Wait until the new pattern establish itself. I recommend resetting the capacity engine starting date.

# Infrastructure Sizing

You are planning a tech refresh for Cluster X. It has 24 ESXi and 1000 VM. You are hoping to reduce infrastructure to 12 ESXi, hence you buy newer CPU, increase the clock speed and add cores per socket. With such major changes, do you consider individual VM one by one, or you do see how they behave as a group?

The answer is the latter, as 1000 VM will not peak at the same time.

Do you consider what happens inside Windows or Linux, or do you see their footprint on your ESXi? The correct answer is the latter, as what happens inside is irrelevant.

## Simple Migration

Aim to do a 1:1 migration. 1 cluster to 1 cluster. After you migrate successful and got the sign off, you then move the VMs into their final destination cluster.

If you do this 1:1, your sizing becomes much simpler. You simply add headroom for the next 3 years or so at the cluster level. If your new cluster sports vSAN and NSX, you need to consider their overhead. Speaking of overhead, you also need the VMkernel overhead, which varies.

The issue of you have with the single VM gets amplified here. Instead of dealing with just 1 faster VM, you now deal with many. Even if the workload is identical, since they complete the job in much less time, your workload pattern turns to spiky. What took 1 minute now take 20 seconds. If your monitoring is 5 minutes, your observability is 300 second average. You can have microbursts that you cannot see. This is why using the 20-second peak metrics is necessary.

Be aware of heavy hitters VM. Consider 2 VMs. Both have 16 vCPU. Both are running hot, but one of them is heavy on IO. It sends a lot of network packets and doing lots of disk IOPS. This 2nd VM has a different footprint on the ESXi. It's much more demanding. All those IO processing need to be processed by other physical cores. That's why your sizing should include disk IOPS, disk throughput and network throughput.

### Generic Formula

New Sizing = (Sum of VMs Usage x Future Growth x Performance Multiplier) + Overhead

Where:

| | |
|---|---|
| **Sum of VM Usage** | It is the chosen number to represent all the data in the assessment period. Ideally, you capture 1+ year so you do not miss annual data. Now that's ~105K datapoints as there are that many 5-minutes in a year.<br>So what numbers do you pick?<br>If the chart is trending, especially upwards, you need to account that the workload is increasing. So I'd use a projection, instead of simply take say the 95th percentile. The percentile also treats recent data as more relevant.<br>For such large data sets, there can be outlier. Both projection and percentile handle this. |
| **Future Growth** | The headroom you need before you top up hardware. So if you buy every year, then you need at least 1 year of buffer |
| **Performance Multiplier** | An additional buffer you put to account that faster performance. |
| **Overhead** | Virtualization layer, which is vSAN + NSX + VMkernel load.<br>This is the reason why you can't take your existing cluster load. Your new cluster overhead is different. It's likely higher as your hardware is bigger, and you may have vSAN and NSX. |

Notice something missing?

Yes, it's reservation. I exclude it as you need to avoid using it. If you want to use it, take the maximum of Usage or Reservation.

## Specific Formula

The generic formula we have needs to be applied to each of the 4 elements of infrastructure. CPU, memory, disk and network need to be treated differently.

We also need to apply to specific object. Let's start with vSphere Cluster as that's the most common.

| | |
|---|---|
| **CPU** | Your main number is in GHz.<br>You support that with another number, in vCPU. For vCPU, the performance multiplier does not apply as you maintain constant. Usage is based on allocation. Overhead is expressed in physical core, so you apply it on the Usable Capacity instead.<br>Allocation Sizing = (Sum of VMs vCPU configured x Future Growth)<br>Make sure this number is within your comfort level. |
| **Memory** | You only have 1 number. It's in GB.<br>Performance Multiplier does not apply to memory as it's just a storage space.<br>New Sizing = (Sum of VMs Usage x Future Growth) + Overhead |
| **Disk** | Your main number is in GB. It only covers disk space, hence Performance Multiplier does not apply here.<br>You typically do not migrate snapshot. You will also exclude non VM such as orphaned vmdk and template.<br>New Sizing = (Sum of VMs Usage x Future Growth) + Overhead<br>You support that with 2 numbers, one for IOPS and one for Throughput. Performance Multiplier applies here.<br>vSAN migration needs to consider vSAN overhead. You're moving the redundancy from hardware level to VMDK level. |
| **Network** | Your main number is in Gigabit/second.<br>You support that with number of packets/second.<br>Make sure both numbers are within what the new hardware can deliver. |

## Sample Output

You do the above exercise per cluster. You get something like this:

| | CPU | Memory | Disk | Network |
|---|---|---|---|---|
| Old Cluster 01 | 145 GHz<br>3504 vCPU | 3.9 TB | 8.2 TB space<br>8374 IOPS<br>49 GB/s throughput | 38 Gbps<br>83K packets/s |

The above is only valid for a 1:1 migration. Anything more complex takes us to the realm of complex migration.

# Complex Migration

From that simple migration above, you can see that a simple sizing exercise becomes complex when the destination is much faster. This turbocharges the VMs, changing the workload pattern.

This becomes a real issue in complex migration, defined as migration that you cannot complete in a single shot or migration where you mix the VMs across clusters.

## Long Migration Project

If you need to migrate over multiple green zones, stretching the period into months, you need to treat sizing as an iterative process.




In complex migration, it does not begin with what you to migrate. It begins with the destination. You plan your end state first, add what you want to migrate, and then work back and forth between the 2 sides.

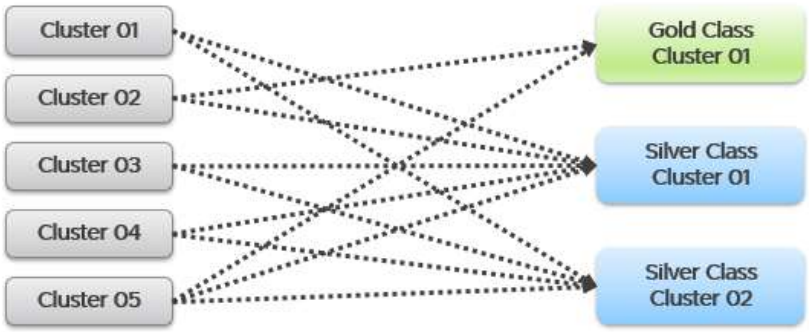
The reasons why you go back and forth are:

| | |
|---|---|
| **Speed** | See Performance Multiplier |
| **Time** | In a large-scale migration, it can take months. During that time, the workload may change. The new features added by the application team may alter your sizing |
| **New workload** | Because the migration period stretches over time, you get new VMs that do not exist in the old clusters. |
| **Selection** | You typically do not migrate everything and you may use the migration as opportunity to regroup. For example, you may move from per department to per class of service, as your IT business changes from System Builder to Service Provider. |

## Migration Group

What happens when your consolidation is not One to One, but Many to Many?

For example, you are consolidating from 47 clusters to 19, but the VMs in a single source cluster will end up in multiple destination clusters. The following examples shows 5 old clusters being consolidated into 3 new clusters. However, the VMs are being reclassified.




This is more challenging as each VM can potentially have its own usage pattern.

Take for example, you have 500 VM in a cluster, but you only need to migrate 100 of them.

Do you take the individual utilization one by one, estimate the size for that VM, repeat for each VM, and finally simply sum all the VMs up?

You can't do that as they may peak at different time.

This makes the sizing process cumbersome as you must create group. 1 group for each destination cluster. If you have 30 destination clusters, you need 30 groups. This migration groups become a central piece of your migration monitoring and sizing. They form a pair of "Before" and "After". The group should consist of just VMs because that's what you're migrating. As a bonus, these groups help you in tracking and adjusting later on in the project. As you migrate a VM, remove it from this group.

The main limitation of the migration group is it has no historical data. The data starts from the time the group is created. To see the past, use super metric preview as a workaround.

## Rolling Up

As part your overall planning, you need a total number. How do you prevent this number from being inaccurate?

The answer is you cannot.

Take for example, 2 clusters. They have highly cyclical workload, which goes up to 90% and down to 10%. Their workload pattern happens to be complimentary. When Cluster 1 is highly utilized, Cluster 2 is lowly utilization. If you combine them, you get a flat line utilization.

Each cluster has 10 hosts. But because their workload do not overlap, when you combine, the total utilization is only 11 hosts, not 20 hosts.

So what is your sizing requirement at vCenter level?

11 hosts or 20 hosts?

The answer depends on whether you plan to combine them or not in the destination cluster. That's why you need to begin with the end in mind. If you plan to combine, the answer is 11. If not, the answer is 20. Big difference.

What if you do not know yet?

That means you have 2 numbers, serving as rough estimate of the minimum and maximum.

Use these numbers as the guide. As you migrate, you adjust your sizing.

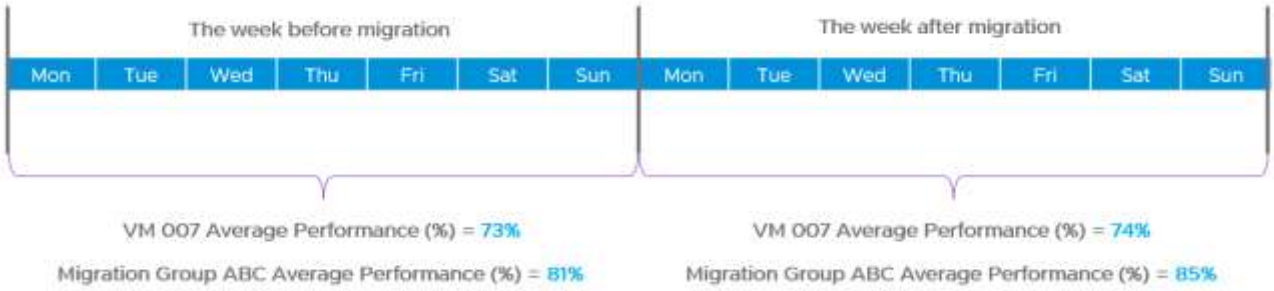
# Reporting

When you are migrating your customers workload to another infrastructure, the onus is on you to prove that you are not causing problems to the VMs or Applications. This is especially true if it's your idea to migrate, and you are not giving them a choice.

To you as infrastructure team, the migration is a capacity exercise. To your customer the application team, they care more about their VM availability, performance and security. This means you need to quantify the "Before" and "After" to avoid misunderstanding at the VM level. If you have 1000 VM, you need to do 1000 comparisons.

This comparison cannot be done immediately after a VM is migrated. It needs to consider a longer timeline, such as 1 week, for a more complete comparison.

On the other hand, you do not and cannot make the post-migration comparison too long. Compare 3 months of before against 1 week of after.




How do you report that the migration does not result in availability, performance and security degradation? If you promise improvement, how do you quantify that?

| | |
|---|---|
| **Consumption** | First, you prove that the VM is not doing less work. That means CPU utilization, Disk IOPS and Network Throughput.<br>Expect the numbers not to drop.<br>We cannot combine these counters as they can't be standardized into 0 – 100%.<br>Why is memory not included? Memory is basically disk space and it's a form of cache. The memory counter at VM level is irrelevant, and the memory counter at Guest OS level is not within your control. |
| **Contention** | Second, you prove that your IaaS is serving the VM well. That means CPU Ready, Memory Contention, Disk Latency and Network Dropped Transmitted Packet.<br>Plot the contention number using the 20-second peak counter.<br>Expect the numbers to drop. If it increases, it remains below the promised SLA.<br>If you are responsible for the Guest OS, then you need to show that queues are not higher post migration. This is why I recommend to keep the VM size. Let the oversized VM be migrated as it is, and right-size it after migration is signed off. |

Once you have the above numbers, it's a matter of plotting them over time.

The above solves it for 1 VM. How to measure for many VMs?

As usual, you create a group to help manage at scale. For each migration batch, you create 1 group. So if you plan to migrate over 35 windows, you have to create 35 groups.

1 migration window should be kept within the day. This makes your Before vs After comparison easier. If you're migrating 3x on Friday night, Saturday night and Sunday night, then it's 3 migration windows.

# Configuration Management

## Part 1 Chapter 4

Configuration Management is about preventing issues caused by incorrect settings. ensuring the actual configuration settings matches the intended or desired value. The configuration of hardware and software products must be identical with the documented systems architecture.

We should be able to quantify configuration manage. This makes it easier to track over time. We can assign 100% where there is no configuration issue, and 0% where none of your desired configuration rules are being followed.

# Approach

Configuration management starts with a plan, where you decide what settings are suitable for what objects. So have your plan documented as corporate standards.

The actual configurations you have in production should reflect your current architecture standard and security policy. Your architecture or standard may change over the years, but it should be documented. You then use the configuration dashboards to compare the reality versus intended standard. If they differ, one of them is wrong and needs to be addressed.

Standards make operations simpler and are often required for compliance. For example, you have a standard for VMware Tools versions, and you choose one version as your standard, but allow 2 other versions across your environment as it takes time to upgrade. You can create a pie chart showing the distribution of VMware Tools version. Each slice in the pie chart counts the occurrence of a particular value. You should expect to see only three slices. If you are seeing more than three, then the reality differs to your standard.

The plan should cover beyond VCF due to dependency. For example, ethernet jumbo frame needs to be configured end to end, including all the physical switches in between. Settings must also be checked across the entire stack, especially the lower layer as problem in a layer will impact the layer above it.

The challenge is balancing conflicting requirements, while meeting constraints such as cost and capacity. For example, you may have to mix different classes of service in a single cluster. Compromise like this makes configuration management essential.

It is a never-ending job as you need to keep up with the versions and product end of life.

Wrong configuration can be costly. There are 6 areas of operations that could be impacted, so choose your trade off carefully.

- Availability
- Performance
- Capacity
- Cost
- Security
- Compliance

In addition, there are VCF product specific settings that need review, such as:

- License.
  Make sure you have the correct amount for each type of license.
- Certificate.
  Many parts of VCF have different expiry dates.
- Configuration Maximum.
  This is especially relevant in a very large deployment where you hit the limit of VCF scalability.
- Users and roles.
  This includes the service account and locally-defined account.
  This includes password management.
- Log.
  ESXi, vCenter, NSX, etc. are configured to send the correct type of logs.
- Compatibility
  As you need to eventually patch and upgrade, this becomes a cycle.

# Purpose-driven Architecture

How do you reflect the business in your private cloud architecture? How does the architecture enable an application-centric operations, where everyone can see the business divisions and their applications?

My answer is a top-down tagging.

## Label or Tag

Label is type of property. It is always external. We attach is to an object. It's added onto the object as part of its creation or during its lifecycle.

Let's take an ESXi host as an example:

- The serial number is not a label.
- The location is a label. It's added by someone, manually or via script.

Tag gives context to various persona running your operations. The context is important. Many customers struggle with tag management as it spans many parts, across software and hardware from all the vendors that you have in your environment.

Label needs to be designed top down so tags are complete and correct.

Categorize your tag, so it's easier to manage them. There are customer-facing tags, and internal-tags.

### Business Tags

These are the tags that application team or your customers care. Have an agreement with them. Note that different department will have different requirements, but you only have 1 set of tags, so have a company wide proposal before meeting each group.

Example tags:

- Cost Center. This enables chargeback and reporting
- Class of Service
- Department. Map this to vCenter folder
- Owner email.
  At the very least, you need the email so system can automatically notify them and ask for approval. In large organization, even a full name may not be unique
- Owner phone.
  For mission critical VM where you need to contact urgently or after office hours.

Balance your tag. Too many tags and it becomes a challenge to update them. For example, if you have 10K VM and each VM is tagged with the direct, actual owner, you can end up with hundreds of names. These people may change department, role, or leave. It's easier to establish a business agreement where you have 1 contact person per department

## IT Tags

These are the internal tags for your infrastructure team.

Example tags:

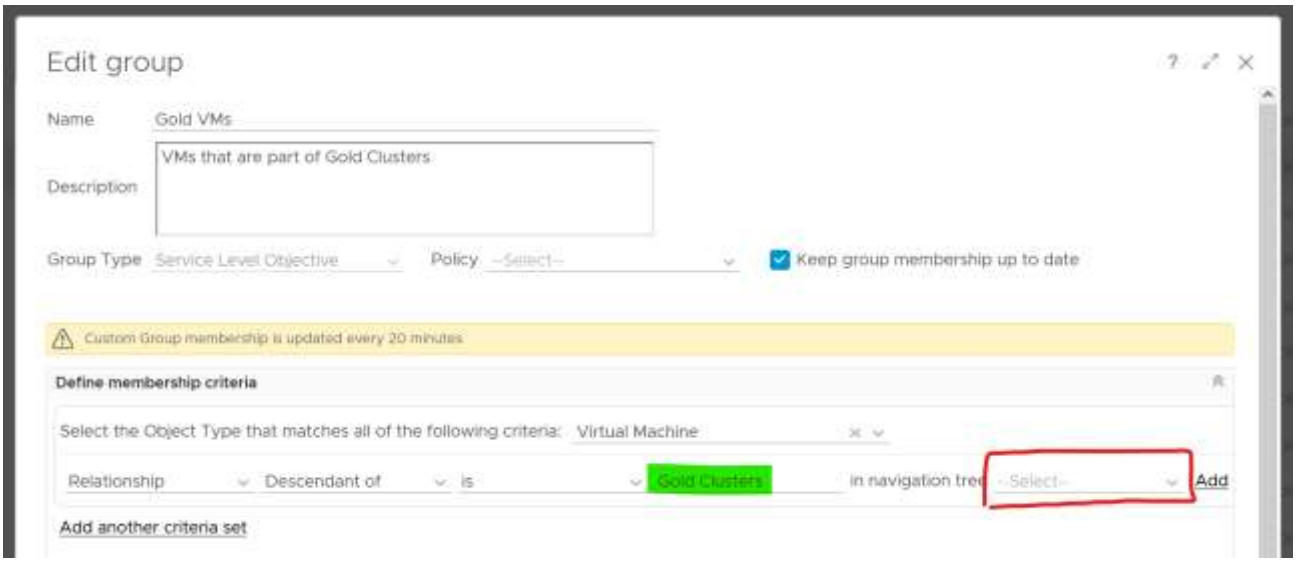
- Refresh Date.
  The date or month or quarter where the hardware is due for a tech refresh. Use the end of warranty as a consideration, and establish a policy on when to upgrade.
- Class of Service.
  For VM, this is inherited from the cluster where the VM is running. It's automatically updated for new VMs or relocated VMs
- Location.
  For hardware, it helps to know the physical location within a DC floor.
- Importance.
  I prefer this terminology over criticality as criticality mixes urgency and importance.
  Give a clear but short name such as high, medium, low.
- Environment.
  Example values: Production, QA, Test, Development.

## Implementation

Use the VCF Operations custom property as opposed to vSphere tags and annotation. These two features were designed much earlier and have limitations, all of which were addressed by VCF Operations. For example:

- Limited to vSphere only. Can't cover Horizon, AWS, Microsoft, business applications, etc.
- Historical data not kept. You don't know what the previous values are, when it changed and who changed it.
- No dynamic membership. Can't automatically assign to objects that meet selected criteria, and membership do not automatically update.
- Values are all strings. You can't do numerical computation on them even if the values are actually numbers.
- The values in vSphere annotations are free-style string, meaning you can't control the consistency of the content. On the other hand, vSphere tags have rigid value. You can't type a value; it has to be chosen from predetermined list. It is not practical with information such as phone numbers, as they're likely unique.
- Tags are per-vCenter, so you need to maintain copies across all vCenter servers.

The following screenshot shows how to dynamically tag all VMs that are in Gold Clusters as Gold VMs. Gold Clusters are in turn a custom group whose members are clusters that provide the gold class of service.




## Naming Convention

Have you ever heard of stories of changes being made by accident to the wrong VM or ESXi host or LUN because their names are similar and hard to read?

Yup. We all have been there 😊

Is the VM naming convention easy to understand yet hard to make human error? How about the Guest OS naming convention?

Does the VM name and Guest OS name map? Is it easy to figure one from the other?

Guidelines for naming convention:

- Make it easy to understand the business context (especially useful for new staff).
- Reduce the chance of making human error (e.g. typo).
- Reflect the criticality of the VM (class of service).
- Provide some context (e.g. name of department owning it).

There is no need to provide the owner information in the name as it's already covered via custom property and vSphere folders.

Design a naming convention for the following:

| Category | Object | |
|---|---|---|
| Consumer | Guest OS | Ideally there is a way to map it to the VM, so keep them fairly similar. For example, WIN for Windows family, and LNX for Linux family. Avoid LIN as that is too close to WIN, both in spelling and in pronunciation. |
| | VM | Typically this would be the same as the OS hostname for manageability reasons.<br>It should have the owner ID, such as the department code. |
| | RDM | Design them as one set. |
| | vVOL | Make sure the name is unique across multiple physical arrays |

| | | |
|---|---|---|
| Provider | Resource Pool | It should have the parent cluster it belongs to |
| | ESXi | it should have reference to the physical location (rack, row, floor) of the box. |
| | Cluster | Class of service |
| | Datastore | |
| | Datastore Cluster | |
| | Distributed Port Group | Designed them as one set. The port name should tell which switch it belongs to. |
| | Distributed Switch | |
| vSphere | Folder | Make sure the 4 types of folders have unique and easy naming convention. It has to be unique across vCenter servers. |
| | Data Center | It should |
| | vCenter | |
| vSAN | Fault Domain | |
| | File Services | |
| NSX | NSX Edge | Avoid including the size as it may change. |
| | NSX Manager | |
| | Logical Switch | |
| | Group | |
| Kubernetes | K8 Container | Designed them as one set.<br>Make sure the names are unique across K8 clusters.<br>It should have the business owner ID. |
| | K8 Pod | |
| | K8 Workload | |

## Example

The name should reflect the object. This is less important in the UI as have you the context. But in code (programming), it helps to know what object you're dealing with to reduce human error.

Here is naming convention I'd use for cluster and datastore name.

| Part | Value | Meaning | Reason |
|---|---|---|---|
| Class of Service | GLD \| SLV \| BRZ | Gold, Silver, or Bronze | I use this as the first part of the name as it's the most important one.<br>I avoid Tier 1, Tier 2, Tier 3 as it could be unclear if Tier 1 is higher than Tier 3. Also, they also differ by 1 character, which is prone to human error. |
| Location | SGP \| SFO | The physical location of the | This does not change. Typically, it's a city.<br>For stretched clusters – clusters spanning two cities or availability zones within a region – suggestions for the location code include: |

| | | | |
|---|---|---|---|
| | | | • Combining both city codes, e.g. SGPSFO.<br>• Using a code for the country/region the stretched cluster is in, e.g. APJ, ANZ, AUS.<br>• Using a generic descriptor like STR or MAZ (Multi-AZ).<br>Different options are suitable for different scenarios depending on your business and where you operate IT infrastructure. |
| Environment | PRD \| DEV | Production, Development | Different treatment or context. |
| Serial No | 1 | No meaning | Just in case you need to have more than 1. Note if you have 3 digit it increases the chance of making a mistake. Think of a way to group them. |

Using the above, you get names like VMW-CLS-GLD-SGP-DEV-1 for the first vSphere gold cluster in Singapore serving development workload, and VMW-CLS-SLV-SFO-PRD-9 for production silver cluster no 9 in San Francisco. I added VMW as you may have non-VMware platforms (such as AWS or mainframe) in your environment.

I'd add LOCL for local datastore, VSAN for vSAN, VMFS for networked VMFS and NFS for NFS type. Raw Device Mapping LUN should have RDM or VMW, whichever easier for your storage team

Don't be hesitant to use dash or underline as they make names easier to read and provide a good delimiter to use in code when managing environments through automation.

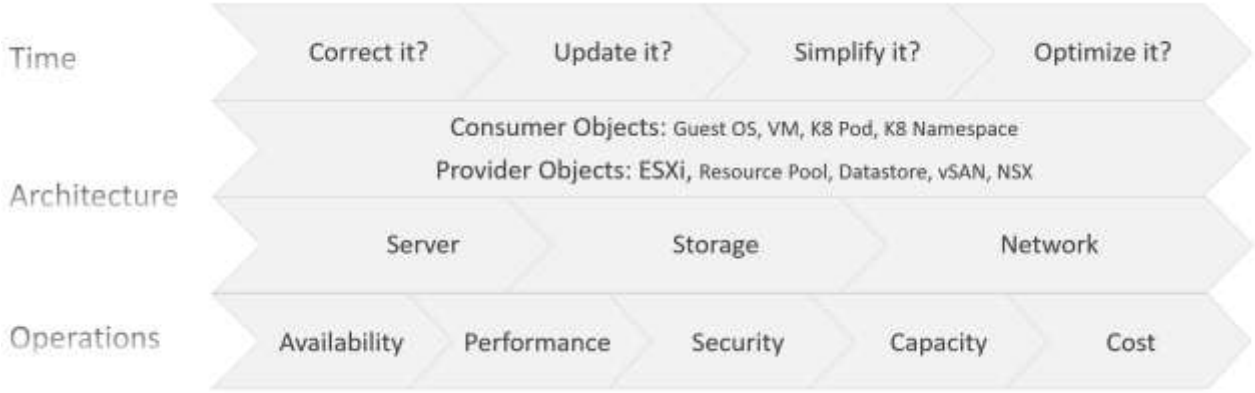
# Review Approach

There are literally thousands of settings that you need to manage. How do you know you cover them all?

Focus on the impact to day-to-day operations a product has, rather than the feature of the product itself. Take the view from Day 2, not Day 1. Products under monitoring, such as vSphere and vSAN, can have features that are related, but have different impact to operations.

VCF Operations takes the principle that there are different impacts to operations, and applies a methodology for looking at configuration. It does not group the settings by features or objects. Rather, it begins with the impact in mind, and prioritize what can be done.

There are 3 dimensions to consider, resulting in 3 ways to review. Use all 3 to ensure completeness.




The first one is based on time. You assess the most pressing issues first. You also check them more frequently. Use a 4-step check, starting from the most urgent.

The second one is based on the SDDC architecture. There are 2 types of objects, consumer and provider.

- Both are equally important.
- Consumer is important as it's what your customers care.
- Provider is important as it can impact many consumers.

As VCF is a type of SDDC, you can review from the 3 types of infrastructure:

- Server or Compute. This covers CPU, GPU, and memory.
- Storage. The stacks start with Guest OS partition → VM virtual disk. From here it covers technology such as ESXi storage subsystem, datastore, RDM, vVOL, and vSAN.
- Network.

The last one is based on the pillar of operations.

- The first to check is availability, as it's the most basic. If the system is down, other problems become irrelevant.
  For hardware, higher availability is achieved via redundant component.
  For software, higher availability is achieved via clustering technology, with 2 load balancers fronting the nodes of the cluster or farm.
- Just because something is up, does not mean it is fast. This is why performance is second. You check for configuration that impacts performance.
- Just because something is fast, does not mean it's secured. This is why a Security and Audit come into play.
- Other areas of pillar operations worth checking is capacity, cost and sustainability.

## Time-based Approach

| | |
|---|---|
| **Step 1** | Address settings that are incorrect, insecure, not following your corporate standards or against best practice. You should correct them as appropriate.<br>This is typically the most urgent step. |
| **Step 2** | The settings are correct, but on older version. It's hard to keep up with all the vendors releases, so you should prioritize those oldest versions, especially those no longer supported.<br>As part of operations best practices, keep the infrastructure up to date. Running outdated components that are too far behind the latest version, can cause support |

| | |
|---|---|
| | problems or upgrade problems. It is common that the fix for the problem is only available in the later versions. Outdated hardware can also result in higher operating costs. Outdated hardware might cost more data center footprint, such as rack space, cooling, and UPS. Refreshing your technology and consolidation are two common techniques to optimize cost.<br>A typical SDDC or EUC architecture spans many components. While each can run the latest version, they may not be compatible or supported. |
| **Step 3** | The settings are correct and up to date, but they complicate your IaaS operations. Since you are unlikely to eliminate them all, establish policy that minimize them as part of simplifying your operations.<br>Complexity can come from many things, lack of standard being one of them. Think of the standards in your SDDC architecture or EUC architecture, and customize the configuration dashboard to shine light on the issue.<br>I think having flexible VM vCPU and memory size do not create complexity, as the permutation is not something you manage. |
| **Step 4** | The last step is about cost and capacity, as there is nothing wrong already. You want to maximize the usage of your resources while minimizing your cost. It's a balancing act! |

# The 4 Types of Check

Checks are performed to ensure you comply with best practices.

You can categorizes check into 4:

- Value Check, where you compare a single item against a desired value. The desired value can be static or universal, or relative depending on the context.
- "Multi Object" check, where you compare multiple items against one another. You can check them for consistency, for compatibility, and for contrast (read: variance)

## Value Check

You compare its value against the desired value. The desired value depends on the type of check:

| | |
|---|---|
| **Against Standard** | Naming convention. Compare the name against your corporate naming standard. |
| | Logging standard. This applies to system such as Windows, Linux, ESXi, VCF Appliances, NSX Edge.<br>For critical VMs, consider logging to capture errors that do not surface as metrics. These errors typically appear as events in the log files, or Event database it the case of Microsoft Windows.<br>Use Log Insight to parse Windows events into log entries that can be analyzed. |
| **Against Version** | It needs to be minimally supported by the vendor.<br>Compare the product version against your corporate standard. Aim to have N and N-1 only. |

| | |
|---|---|
| **Against Date** | Expiry date, which should cover:<br>• end of warranty<br>• end of support<br>• certificate expiration. |
| **Against Threshold** | Absolute threshold. Example is the configuration maximum. Note this can vary depending on the combination. |
| | Relative size, which depends on the parent "container". For example, a relatively large Linux container in a Kubernetes Node can cause performance issue to other containers and itself. |

## Consistency Check

| | |
|---|---|
| **Against Peer** | This is about consistency among members of a group. For example, all the ESXi hosts in a cluster should have identical configuration. |
| **Against Partner** | This is about consistency with adjacent connection. For example, ESXi MTU setting should match DVS MTU setting. |

## Compatibility Check

While consistency is about identical value, compatibility is not. Most of the time, the values are not the same.

Think of "what works with what when and where".

- You're running hardware and software from multiple vendors, and they don't always work best in every possible permutation. Pay attention to the detail, minor version number too.
- Upgrading one component often requires upgrading adjacent products.
- What is right for development environment may not be appropriate for production. In the same production environment, what was suitable last year may not be suitable this year.

## Variance Check

Too many variants happen because you want to achieve flexibility. There is a cost of complexity as the number of variants increases. The complexity can manifest in security, capacity, compliance, performance, or just day to day operations as your team have to be aware of myriads of things.

- Guest OS
- VM
- ESXi
- Server BIOS
- vSphere Cluster
- Storage
- Network

Take note that some variants do not create operational complexity. VM sizing is one such example. Having a t-shirt sizing costs wastage with little benefit of operational overhead.

# Review List

It's difficult to logically list as a setting typically has multiple sides. Take for example, do you put NFS security setting under Storage, Datastore, or Security?

In this section, I group them based on the object. In the case of NFS, it will appear in both ESXi and datastore objects.

Before you review the settings, review the name first. Ensure they follow the naming conventions. There is a chance it's not adhered to.

# Consumer Objects

Guest OS and VM are separate as they have their own set of requirements. In large organisation, there could be 2 separate team responsible for each.

## Guest OS

| | |
|---|---|
| Type | The actual Guest OS may not match the type specified during VM creation. This could be due to Guest OS upgrade, or reformat. Update the value in vCenter to avoid confusion. Do this by comparing these 2 properties:<br>• Guest OS from vCenter<br>• Guest OS from Tools |
| Version | Are there outdated Windows or Linux version? |
| | Too many versions of Linux distros. Within the same versions, minimize the build numbers and patch level. |
| | Too many versions of Microsoft Windows. Within each edition, too many editions |
| Driver | Is PVSCSI used appropriately? |
| | VMXNET3 or PVRDMA or SR-IOV used appropriately? See this for networking driver best practices. |
| Agents | Are they too many agents? Back up agent, security agent, monitoring agent? What's their total footprint (since they are overhead)? |

| | |
|---|---|
| | If you use agents, such as Telegraf agent and Log Insight agent, ensure they are installed only in the correct VMs, are up to date and collecting properly. |
| Windows Drive | Consistent purpose of each letter. For example c:\ drive should always be for OS only. |
| | Consistent usage of temp drive. |
| | Consistent usage of data drive. Since you do not know how many each OS will have, you can start with Z:\ and go backwards. |
| Application | You want to avoid flying blind. After all you have some responsibility for everything that runs on your platform. Just because the VM name or hostname say webserver does not mean it's running web server. Using tools such as Telegraf or Service Discovery, find out what application (key process) are running and who is it talking to.<br>Since your workloads are sharing resources and are over committed, your operations are easier if you know what is running inside. This helps with monitoring and troubleshooting. It is also required by some ISV for software licensing. |
| | Java Virtual Machine or Database with memory allocation too large relative to Guest OS. Unless you're running other application, the extra memory is not accessible by the application. |
| Settings | Too many configuration setting options in your corporate standard. |
| | Disable screen saver as it's irrelevant for server OS |

## Tools

I single out Tools as it's an essential piece, required when running workload on VCF.

Using VMware Tools has multiple benefits, such as driver and observability. For the list of benefits, refer to KB 340.

For more information about VMware Tools, see the VMware Tools documentation.

| | |
|---|---|
| Existence | Are Tools installed on all VMs?<br>The lack of support from Independent Software Vendor (ISV) owning the application is the most common reason that VMware Tools is not installed in the Guest OS. The ISV vendor may claim that no additional software is installed in their appliance unless they have certified it. |
| Availability | If yes, are they running?<br>If VMware Tools is installed, there might be reasons why the application team disables it. The infrastructure team should inform and educate their application team, and document the technical recommendations on why VMware Tools is needed to run at all times. |
| Version | If yes, are they up to date? Compatible with the VM hardware version? |
| | If you are running older version of Tools, you may not have the following Guest OS performance metrics: CPU Run Queue, CPU Context Switch, and Disk Queue Length, Memory Used |
| | Too many versions of Tools. Newer versions tend to have better observability |

# VM

| | |
|---|---|
| Version | Are there outdated VMX hardware versions? |
| | Too many versions of VM hardware |
| Advanced Setting | Minimize the usage of advanced parameter setting.<br>Ensure modification complies with your standard. |
| Multi Network | Are there non-networking VM spanning >1 network?<br>A VM that has multiple network interfaces can bridge the network, causing security risks or network issues.<br>Take note that a VM that is part of multiple networks can do so with just a single NIC card. A single NIC can be configured to access multiple networks, with each interface having their own IP configuration. |
| Driver | Is it correct? Up to date? |

## Compute-related

| | |
|---|---|
| Relative size | Ensure that the VM size does not exceed the size of the underlying ESXi host. If your ESXi host has CPU hyper-threading, do not count the logical processors. Instead, count the physical cores only. |
| NUMA | If the number of configured vCPUs on a VM is higher than number of cores per socket on the ESXi, the VM can experience NUMA effect. If the ESXi has more than one physical CPU (socket), cross-NUMA access negatively impacts performance. For best performance, keep it within the CPU socket. |
| | Modern CPU die consists of multiple chiplets. For example, a 64 core CPU is made of 8 x 8-core chiplets. Each chiplet has its own cache. In this case, any VM greater than 8 vCPU will spread over multiple NUMA nodes. |
| | vSocket and vCore configuration. Does it match vNUMA best practices? |
| Large VM | The larger the VM, the longer time is required to vMotion, Storage vMotion, and backup. |
| | Are there monster VM?<br>Large VMs that are running hot can impact the performance of other VMs, especially since they are given higher shares by default. Only when the large VM is under-utilized, can the ESXi run other VMs. |
| Hot Add | CPU Hot Add and Memory Hot Add. Are those applied at the correct VMs?<br>CPU Hot Add results in NUMA optimization disabled. So memory is being interleaved between the nodes. Hence, only use Hot Add if the benefit outweighs the cost. |

## Storage-related Configuration

| | |
|---|---|
| Virtual Disk | For disk space, if the disk is thin-provisioned and under-utilized, you can deploy other VMs in the same datastore. Ensure that the snapshot is tracked closely, as the risk of capacity running out is higher for a large virtual disk. |
| | What is the unmapped reclamation opportunity? |

| | |
|---|---|
| | Monitor at virtual disk, not VM. Each virtual disk must be monitored in terms of IOPS, throughput, and latency. Having multiple virtual disks increases the monitoring and troubleshooting need.<br>If the reason for having many virtual disks is performance, identify which counter serves as proof that multiple virtual disks are required. It is possible that the performance required is met by a single virtual disk. |
| Snapshot | Ensure that the snapshot is removed within one day after the change request. If not, it might be forgotten, resulting in a large snapshot and impacting the performance of the VM. |
| Disaster Recovery | Incorrect vSphere replication setting. One reason is the VM Owner never informs infrastructure team on the changes. |
| | Are all the VMs protected by SRM meant to be protected? Some VMs could be no longer use but are still protected. Review the Recovery Plan. How frequent was it run and when was the last run? |
| Fault Tolerant | Any feature, especially something as powerful as this one, comes with its own set of complexity. So make sure the VMs with FT are the correct VMs. |
| vVOL | Is the usage following design best practice? |
| Shared Disk | Are there VMs sharing VMDK or RDM disk? |
| Guest : VM mapping | It is simpler to have a 1:1 mapping between Guest OS partitions and the underlying virtual disk (VMDK or RDM). While you can run logical volume at Windows or Linux level, it creates complexity. |
| VM : Datastore mapping | Minimize VM that spans multiple datastores. This can make performance troubleshooting and capacity planning difficult as you create a M:N relationship between VM and datastore. |
| RDM | Keep the usage minimal as they result in LUN sprawl and unused RDM object in the physical array |
| | Are they excessive usage, where RDM is used when there is no need to? |
| | Are they unused or hardly used RDM? |
| | Are they oversized RDM? |

# Resource Management

I highlighted this separately as its flexibility need to be managed carefully. It's hard to keep the settings fully consistent in a large environment over the years.

vSphere provides powerful control over infrastructure resources. You can apply shares, reservation and limit. As feature, they are closely related, appear in the same dialog box in vCenter client UI and should be mastered as one. However, they impact operations differently. The following table describes that in more details.

| | | | |
|---|---|---|---|
| **VM Limit** | Impact the VM | • Should not be used. Rightsize instead<br>• Result in unpredictable performance of Guest OS | Incorrect Configuration |
| **VM Reservation** | Impact the infrastructure | • Keep the total amount low, relative to the cluster total capacity.<br>• Absolute value. A 2 GHz reservation is a 2 GHz reservation.<br>• Result in suboptimal infra capacity, as overcommit is not possible | Suboptimal Configuration |
| **VM Share** | Impact the infrastructure | • Keep the number of variations to below 3. 1 for each class of service.<br>• Relative value. A 2000 worth of reservation depends on the value of other VMs reservation. Be careful when moving VM to another cluster, as the relative changes.<br>• Result in complex operations. Harder to troubleshoot performance when each VM dynamic entitlements fluctuates more. | Complex Configuration |

| Control | Analysis |
|---|---|
| Limit | The most basic form of control. It sets the upper limit that the consumer can't exceed. It is a hard limit.<br>It is more relevant in container as by default it has no upper limit.<br>With limit, if the underlying shared resource is available, it will go wasted instead of being used. On the other than, this guarantees that consumer will not exceed what they pay for. You can also contain the damage of denial-of-service attack.<br>From the above, you can see it cuts both ways. You need to balance cost and performance. |
| Reservation | Complement limit by setting the floor. This is also another form of guarantee, so your total reservation cannot exceed your total capacity.<br>Take note Disk has no reservation for IOPS and throughput. |
| Share | Share is only effective if there is enough resources left to share. If you have 80% reservation, then you only have 20% left to share.<br>The 20% is only effective if the demand is larger.<br>Share is relative to the configured size. A VM that is 10x bigger is given 10x shares by default. |
| | Imagine a host with a fixed capacity of 10 GHz.<br>Two VMs are running on that host, one with a single vCPU, and one with two vCPUs. The single-vCPU VM will have 1000 shares. The two vCPU VM will have 2000 shares. Both VMs want to consume 10 GHz.<br>Ignoring overhead, you could expect the VM with a single vCPU to get 3.33 GHz of a physical CPU, and the VM with 2 vCPUs to get 6.66 GHz of the physical CPU.<br>On the other hand, imaging both VMs only need 5 GHz. In this case, VM 1 gets 5 GHz and VM 1 gets 5 GHz. |

## Share

Shares values are relative, meaning the value depends on the value of sibling objects such as, resource pool or VM.

Here are the default values for VM:

| | vCPU | Memory |
|---|---|---|
| Low | 500 | 5 per GB |
| Normal | 1000 | 10 per GB |
| High | 2000 | 20 per GB |

Here are the default values for Resource Pool:

| | CPU | Memory |
|---|---|---|
| Low | 2000 | 81,920 |
| Normal | 4000 | 163,840 |
| High | 8000 | 327,680 |

I'm unsure why a resource pool CPU is 4x VM but memory is 16x. This means a VM in a resource pool can potentially get 4x for CPU and 16x for memory, relative to VM outside the resource pool.

A VM 2x larger should have 2x share, all else being equal. If you right size VM manually via vSphere Client UI, it auto adjusts the share. If you do via API, it does not. Check those VMs whose share you manually change, to ensure the number is still valid.

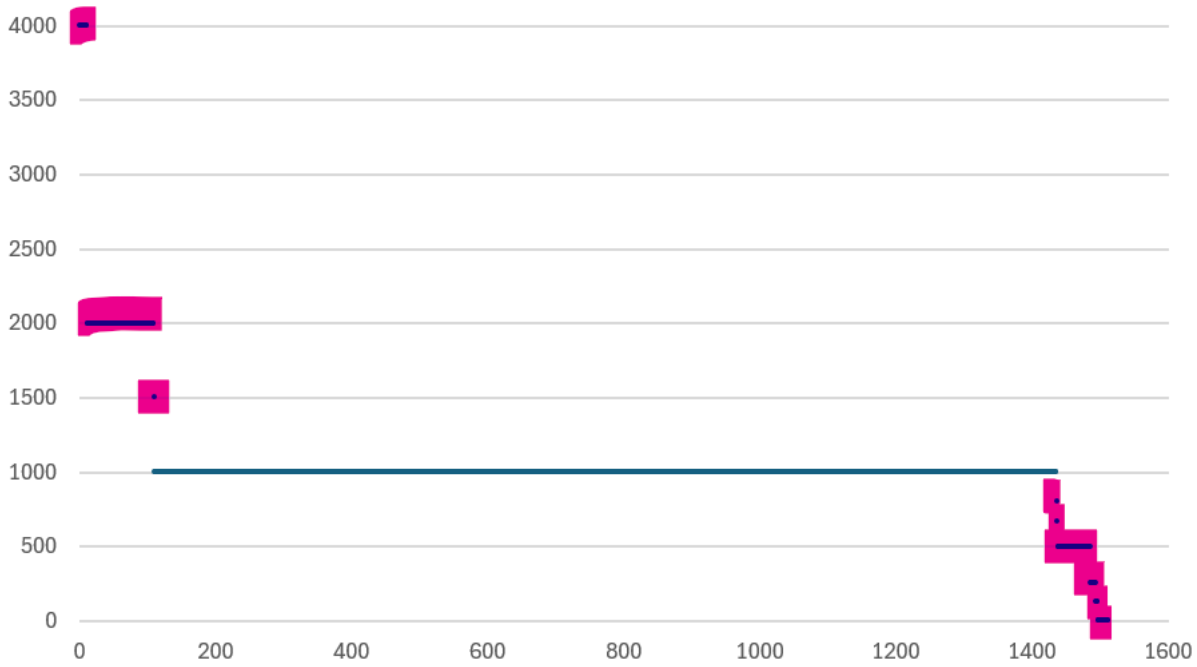
Cluster with many VM Shares (normalized per vCPU and per GB RAM) makes performance troubleshooting harder. Each share should map to exactly one class of service, such as one for Gold and one for silver, as the shares defines the class of service.

If you move VM across clusters, ensure that the values of shares are consistent across clusters to avoid unintended consequences while moving the VM to another cluster.

Check for VMs whose size are not aligned with their share. On a regular basis, ensure the share value per VM vCPU = 1000 and per VM GB of RAM = 10240. I downloaded a list of VMs into a spreadsheet. For a sample size of just 1500 VMs, I see incorrect values such as below.

| CPU Share | vCPU | Memory Share | Memory | Shares per vCPU | Shares per GB RAM |
|---|---|---|---|---|---|
| 4,000 | 1 | 163840 | 1 | 4000 | 160 |
| 4,000 | 1 | 163840 | 2 | 4000 | 80 |
| 4,000 | 1 | 163840 | 2 | 4000 | 80 |
| 4,000 | 1 | 163840 | 2 | 4000 | 80 |
| 4,000 | 1 | 163840 | 2 | 4000 | 80 |
| 4,000 | 1 | 163840 | 2 | 4000 | 80 |
| 4,000 | 1 | 163840 | 2 | 4000 | 80 |
| 8,000 | 2 | 163840 | 3 | 4000 | 53 |
| 8,000 | 2 | 163840 | 4 | 4000 | 40 |
| 16,000 | 4 | 327680 | 21 | 4000 | 15 |
| 8,000 | 2 | 163840 | 14 | 4000 | 11 |
| 4,000 | 2 | 81920 | 2 | 2000 | 40 |
| 4,000 | 2 | 81920 | 2 | 2000 | 40 |
| 4,000 | 2 | 81920 | 2 | 2000 | 40 |
| 4,000 | 2 | 163840 | 4 | 2000 | 40 |

I plotted the values over a scatter chart and see there are both too big and too small.




## Reservation

It limits your ability to overcommit, resulting in less optimal usage. High total reservation, especially both CPU and memory, also complicates the cluster operations as it impacts the HA slot calculation and limits the DRS choice of placement.

Any relatively high amount of VM reservation? If yes, this will impact the HA Slot Size. Large amount of CPU and memory reservation impact the cluster capacity, so use it only for important VMs.

Do you specify it at each VM, or at the parent resource pool?

There are pros and cons.

For VMs that belong to the same application, specifying the reservation at the resource pool is easier.

## Limit

Are there VMs with limit? CPU limit and memory limit can result in unpredictable performance. You should downsize the VM instead.

CPU, Memory and Disk IOPS Limit. There is no network limit.

Avoid using limit as it can result in unpredictable performance. The Guest OS is not aware of this restriction as it is at the hypervisor level. It is recommended that you shrink the VM instead.

Limit has a valid use case in Resource Pool. If the tenants only buys 1 TB of RAM, and the contract does not include burstable capacity, set a limit. This will prevent consumption beyond the sold capacity.

## Resource Pool

Use resource pool sparingly. It should only be used when selling Resource Pool as a Service.

Shares and reservation settings in resource pool and VM need to be looked together, especially in scalable setting.

| | |
|---|---|
| Children | Resource Pools shares do not match the number of VMs. The resource pool value is divided and shared among the VMs. The more the VMs, the lesser the resources allotted to each VM. This can be solved by turning on scalable share. |
| | Ensure the RP setting is not causing the shares or limit issue to children VMs. This can be tricky in large cluster with many hosts as the RP metric is at cluster level, while a VM is running on 1 host at a time. |
| Sibling | Resource Pool with VM as sibling is a common mistake. Once you create the first resource pool, you must create at least another resource pool. A single resource pool with no peers makes no operational sense as it's not a folder. Since no VM should be peer to it, you need at least 2 resource pools. |
| | Compare the relative values across sibling resource pools. If Gold resource pool should have 2x the share of Silver resource pool, then its value has to be 2x. |
| Cascading | Any cascading resource pool? If yes, what's the reason for it since resource pool is often mistaken as folders?<br>Avoid further splitting into sub resource pools. Each layer increases the complexity of performance management. |

# Compute

## ESXi

| | |
|---|---|
| Hardware | Are there too many variations of server hardware vendors, models, and generation? Even within the same model and generation, minimize the specification difference. |
| | Are they outdated hardware? Which model is entering end of support? |
| Version | Are there outdated ESXi versions? |
| | Too many ESXi software versions?<br>For each version, are the update and patch levels consistent? |
| CPU | Older generations.<br>Too many variants. |
| | CPU architecture. Impact NUMA size and type of workload (Intel P-core and E-core) |
| | HT exists but not enabled |

| | |
|---|---|
| | Low frequency. This impacts performance of CPU intensive application. |
| Memory | Are there too many size variants? |
| | Is the CPU: Memory ratio supporting the workload? The general rule of thumb is 1 vCPU gets 4 GB of RAM. So a machine with 64 cores 128 threads get 512 GB of RAM. |
| | If performance is more important than cost, then do not disable large page. |
| BIOS | Are there outdated BIOS versions? |
| | Are the settings consistent, especially within a cluster? |
| Power Management | Unless the workload requires it, BIOS level should be set to OS managed. Pass the control to ESXi, and then set ESXi to balance. |
| Availability | ESXi is in one of these situations:<br>• quarantined<br>• disconnected from vCenter or Cluster<br>• unreachable or unresponsive<br>• Powered off |
| | Standalone ESXi |
| | vMotion disabled |
| Security | Direct Console UI is enabled |
| | SSH is enabled |
| | Shell is enabled |
| | Encryption not enabled |
| Capacity | A small host faces scalability limits in running a larger VM. While a 2-socket, 32-cores, 128 GB memory ESXi can run 30 vCPU, 100 GB memory VMs, the VM experiences a non-uniform memory access (NUMA) effect. |
| | ESXi with Hyper-Threading Disabled. While it provides a more predictable performance to the VM, it comes at a high price. |
| | Low CPU core counts. Aim for 48 to maximise software license.<br>This limits the ability to run larger VM. |
| | Low total memory. Include vSAN ESA and NSX in the sizing. |
| | Insufficient network capacity. Include vSAN ESA and NSX in the sizing. |
| | Too many variants of the above |
| Advanced Setting | Minimize the usage of advanced parameter setting.<br>Ensure modification comply with your standard |
| | Avoid CPU Affinity |
| | Unused device such as DVD-ROM, USB are disabled |

## Management

| | |
|---|---|
| Configuration profiles | Is it used correctly? |

| | |
|---|---|
| Log | Use VCF Operations Logs as it has VCF specific log analysis with extracted fields. The fields are used in out of the box dashboard and alert. |
| ILO | Is lights out management configured?<br>If yes, are they secure? |
| Agent | Any 3rd party agent installed? |
| NTP and DNS | Ensure both Network Time Protocol and DNS are configured.<br>Incorrect time can turn logs from useful to potentially misleading. Logs are a necessary component of operations, and are the main source of information in troubleshooting. While troubleshooting performance across objects, the sequence of logs determines which event is the likely root cause as the oldest event started the chain of events. |

## Storage Related

| | |
|---|---|
| HA | Is the HBA configured for high availability? |
| Paths | Incorrect multi-pathing setting. |
| | Too many storage paths. While redundancy is good, having too many may not bring the result you expect. Compare this with airplane with 2 engines versus 4 engines. |
| | Single path |
| Zoning | FC or iSCSI LUN zones follow best practice, such as single initiator zoning. |

## Network Related

| | |
|---|---|
| HA | Is the HBA configured for high availability? |
| Technology | Actual network speed is lower than configured. This happens due to auto-negotiation, which could happen due to high dropped packets. |
| | Older generations, such as 10 Gb and 1 Gb ethernet |
| Static IP | Any ESXi using dynamic IP address? |
| Consistent Address | Is the network address following easy to understand pattern? Since your ESXi hostname will have some sequential number, make sure this number matches the IP address. This will minimize human error. |
| | Consistency across network, on the same ESXi |
| | Consistency across ESXi hosts, on the same network |
| Capacity | Is the actual throughput the same with the configured speed?<br>Auto-negotiate can result in reduction of speed in order to preserve connectivity. |
| | Any network on 1 Gbps or 10 Gbps?<br>Other than management network, all other networks should be on 25 Gbps. |
| | Is there enough capacity for the purpose?<br>What's your sizing for NSX Edge cluster? It should be greater than the sum of all NSX Edge VMs on the host + vMotion + vSAN. |

| | |
|---|---|
| Kernel Network | Are the following kernel traffic separated:<br>• Management<br>• vMotion<br>• IP storage<br>• vSAN<br>• vSphere Replication<br>Ensure these networks are indeed private, and not used by other purpose. |

## Cluster

A cluster is the smallest logical building block for compute. Consider it as a single computer with physically independent components for high availability. Ensure that it has enough CPU cores, CPU GHz, and Memory. For ESXi in 2024, it is typical to have 1 TB memory. This results in 12 TB of memory for a 12-node cluster, which is enough for DRS to place many VMs as it balances them.

| | |
|---|---|
| HA | HA disabled.<br>Without high availability provided by the infrastructure, each application must protect itself from infrastructure failure. |
| | Clusters with Admission Control disabled. Reservation is respected only when Admission Control is enabled |
| | Cluster HA Failover %. Make sure this number matches your design. |
| | Is there any VM with exception?<br>Overriding the cluster default complicates operations. |
| DRS | DRS disabled.<br>DRS focuses on performance and capacity, while HA focuses on availability. Without DRS, you must build a buffer on every ESXi host to cope with peak demand. |
| | DRS set to manual.<br>This means that DRS initiated vMotion does not take place unless it is manually approved by administrator. Since DRS calculates every five minutes, your quick approval is required to prevent a change of condition. |
| | Cluster with EVC Mode means it's not able to take advantage of the newer capabilities. |
| | Automation Level. Ensure this meets the requirements of that cluster |
| | Is there any VM with exception?<br>Overriding the cluster default complicates operations. |
| DPM | Are the settings matching your expectation? |

### ESXi Consistency

Consistency matters. Check at least the following settings are identical:

- CPU model, generation and frequency.
- Memory size and speed

- BIOS version and ESXi versions.
- BIOS Power Management and ESXi Power Management.
- ESXi Storage Path. Ensure that the number of paths and the path policies are identical.
- All the hardware drivers and firmware

Putting host devices in a consistent bus or slot for a particular type (vendor/model) facilitates automated installation and configuration, and makes administration and troubleshooting easier.

### vSphere Cluster Variation

There are sizable differences in the variants.

| | |
|---|---|
| Sub-cluster | Sub-cluster technique, such as VM to Host affinity creates operational complexity. It can result in performance problem despite sufficient capacity. |
| Small Cluster | A small cluster has a higher HA overhead when compared to a large one. For example, a three-node cluster has 33% overhead while a 10-node cluster has 10%. For vSAN, a low number of hosts limits the availability option. Your choice of FTT is relatively more limited. |
| | High number of small clusters result in silos of resources. |
| Large Cluster | They can have higher redundancy. Instead of 3 clusters with 6+1 each, you can have 1 large cluster with 18+2. You have 1 ESXi host while providing higher redundancy. |
| | Cluster with 32 nodes or more. Large clusters are harder to operate |
| vSAN | It imposes addiitonal consideration, such as a VM will have both compute host and storage host. |

You can combile or integrate multiple clusters. There are 3 possible variants:

| | |
|---|---|
| Multi-Cluster | A group of clusters operating as one large cluster. Each member still has their own capacity remaining calculation, but they can balance between members. |
| Stretched Cluster | Stretched Cluster is more complex than traditional cluster. They provide Disaster Avoidance (DA), not Disaster Recovery (DR). Having a DA does not remove the requirements for DR. |
| SRM Cluster | Cluster integrated with Site Recovery Manager introduces complexity in capacity management as the protected VMs need to be accounted for. |

# Storage

This covers all storage components of VCF, such as datastore, datastore cluster, RDM, vVOL, vSAN, physical storage array, and general storage settings you need to check.

They are put together here to facilitate discussion with storage team.

Too many variants of storage architecture: VMFS, NFS, FC VMFS, iSCSI VMFS, vSAN, vVOL, physical RDM, virtual RDM.

While each architecture has their unique fit for purpose, take note that each requires expertise on performance, security, availability, upgrade. There is a cost as you need to keep up with the technology, and risk that you overlook

## Datastore

| | |
|---|---|
| Version | Are there outdated VMFS or NFS version? |
| Local datastore | Are they any VM running on local datastores? These should be limited to agent VM or VM that do not need vMotion and backup. Backing up from a slow storage can cause performance problem to the VM as the local disk may not be able to serve both.<br>It's not a place you put back up as you can forget. |
| Unused datastore | Datastore with no ESXi. Datastore with no path means there is no ESXi accessing it |
| | Datastore with no VM. |
| Hardly used datastore | Datastore with low VM count. If you have many of them, it results in datastore sprawls and pockets of unused space. |
| | Datastore with low activities. That means the VMs are idle or unused. |
| | Datastore with low usage. |
| Small datastore | Small datastores run a relatively higher risk of being full from snapshot or over-provisioning. |
| Variance | Too many variants in the datastore size |
| Availability | Datastores with too many paths creates complexity |
| | Datastore with single path carries risk as there is no redundancy. |
| Datastore : Cluster mapping | Datastores that are shared by >1 cluster are more complex from both capacity monitoring and performance troubleshooting |
| Datastore Cluster | Inconsistency among the member datastores can result in capacity or performance issue. |

## Others

| | |
|---|---|
| LUN | Unused LUN. That means there is no VMFS or vVOL on top of it.<br>Inconsistent priority. For example, it gets gold priority at the physical array level but silver at vSphere level. |
| NFS | Does the NFS share use up to date security for authentication? |
| vVOL | Are the inventory matching your expectation? |
| Physical Array | Array firmware version need to be compatible with ESXi. |
| Queue depth | Mismatch of queue depths along the various storage stack. Need to calibrate all the way to physical array |
| VAAI | Is it following best practices? |

| | |
|---|---|
| vSAN Max | Are you running VMs in the cluster?<br>While technically this is possible, this is akin to running VM on your physical array. It complicates operations as it adds a server dimension. vSAN Max should be seen as a physical array, with benefits such as simplicity and >2 node service controllers. |

# Network

This covers all component of VCF. They are put together here to facilitate discussion with network team.

This cover distributed switch, distributed port group, NSX, physical switch, and general network settings you need to check.

## Distributed Switch & Port Group

| | |
|---|---|
| Version | Are there outdated versions? |
| NIOC | Is network IO control configured? |
| Port Group | Unused network (distributed port group) is a potential security risk as you may have the tendency of not monitoring it. Network is basically a path or road, so it can be used by unauthorized user. |
| Traffic Shaping | Is it used appropriately? |
| Shares | Are the shares of various network configured correctly? |
| Limit | Is limit configured? Can share be used instead of limit as it's more flexible for contention control? |
| Performance | MTU mismatch |
| Jumbo Frame | Are they configured for the right networks? |
| | Are they configured end to end? For the configuration outside your control, how do you know when they are changed? Use tools such as Cisco Discovery Protocol to probe. |
| Security | Are any of the following networks not set to Reject?<br>• Forged Transmits<br>• Promiscuous Mode<br>• MAC Address Changes |

## Others

| | |
|---|---|
| TOR switch | As a top of rack switch has direct connection to ESXi, ensure the firmware version is compatible. Check the speed does not negotiate down. |
| Firewall | Any ports configured correctly? |

| | |
|---|---|
| | Certain services such as syslog, NTP and vSphere High Availability require ports to be opened. |
| NSX | Version is compatible with relevant components. |
| | Ensure the redundancy for both NSX Controller and Manager match your plan. |
| | NSX Edge configuration not following best practice. |
| Encryption | Are networks such as vMotion encrypted? |
| | Are you using hardware assisted encryption such as Intel AES-NI? |

# Security

There are 2 types of potential security issues:

- Configuration
- Activity

Configuration is easier as it's a setting that can be compared with an expected value.

Activity is harder as it depends on the context. There are 3 subtypes:

- Action by human. Example is someone issueing delete commands on ESXi console.
- Action by system. Example is excessive broadcast packets. Most VMs should not be sending excessive broadcast or multicast packets, as traffic should be unicast. While the monitoring can be done at VM level, the troubleshooting needs to be done inside Windows or Linux as that's the source.
- Action by AI, acting as digital employee. The AI agent is identified by its employee ID, and reports to a human employee.

## Examples

This covers all component of VCF. They are put together here to facilitate discussion with security team.

| | |
|---|---|
| Root | Is the usage minimized? What scenarios are allowed?<br>This applies to both ESXi and VCF Appliances. |
| | Are the commands typed logged and analyzed? Create a Log Insight dashboard that traps undesirable commands such as mv and rm. |
| Time Out | Is the time out configured correctly? Apply it for ESXi Shell and user login. |
| vCenter | Are the access to vCenter properly limited? |

# Value Management

## Part 1 Chapter 5

How value, cost, and price work together, and how to optimize cost so you run the private cloud at half the cost of public cloud, while delivering higher business value.

# Cost versus Price

With hardware becoming commodity and infrastructure becoming invisible, price has become a common denominator among all IaaS providers. Applications can run spanning multiple cloud, so whichever cheaper is likely to get the deal to serve the applications.

“Cluster with plenty of capacity is more expensive as the cost is only shared by few VMs”.

Do you agree with the above?

If you do, you’re mixing cost and price. The cost of the cluster is fixed, regardless of the number of running VM. In fact, you likely pay 5-year cost in advance as you get a discount far higher than bank interest rate. For example, borrowing money costs 10% per year. The vendor salesman offers a 50% discount if you fully paid a 5-year ELA (Enterprise Licence Agreement). Which one do you take, assuming you plan to run for at least 5 years?

Be consistent with the terminology. 1 English word shall give 1 meaning, regardless of the perspective. If you have different interpretation, you get confusion. Since what is price to you is cost to your customer, avoid using the word VM Cost as the definition changes. When you are talking about a specific VM, it has price, not cost.

Cost and Price are often confused as enterprise IT does not charge real money. The IaaS team typically will pass on the cost to the application team. This is where the confusion comes in, as there is no such thing as a VM Cost. Another word, nothing to pass on as VM price and VM cost are 2 independent numbers. The concept of unallocated cost is flawed.

Let’s use an example.




In this initial roll out, the IaaS Team starts with just 5 VM to ensure the platforms are performing as expected. They are proving that the actual performance settings, availability settings, security settings, are working as they should be.

The VM Owners are in fact given a discount to be part of this real-world test.

The cluster total size is 20 ESXi servers, 1000 AMD cores, 16 TB RAM, 40 NVIDIA GPU, and 80 SSD. 5 years support fully paid as part of ELA.

Licenced of MS Windows DC Edition and Broadcom VCF fully paid for 5 years as part of ELA

The total cost for 5 years, including data center facilities, is $12 million.

Financially, it’s approximately $200 K per month.

The VM are not large VMs. They are just 4 vCPU, 16 GB RAM. Together, they cannot even saturate 1 ESXi, let alone 20. Guest how much the total cost to run those 5 VMs for a month?

If your answer is $200 K, you’re right.

Now, will the owner of those VMs be willing to pay that much?

Obviously not. She will compare you with AWS or Google.

She will say public cloud charges $100 per month for such VM. So your price should be just $500 total per month.

Now, the VMs cost is $200 K per month, but you can only price $500 per month. You suffer a big loss, which is common during the early days.

Fast forward…




3 months have passed….

There is a lot of more VMs running, including much larger ones. You make higher profit on these monster VM as you apply a progressive pricing.

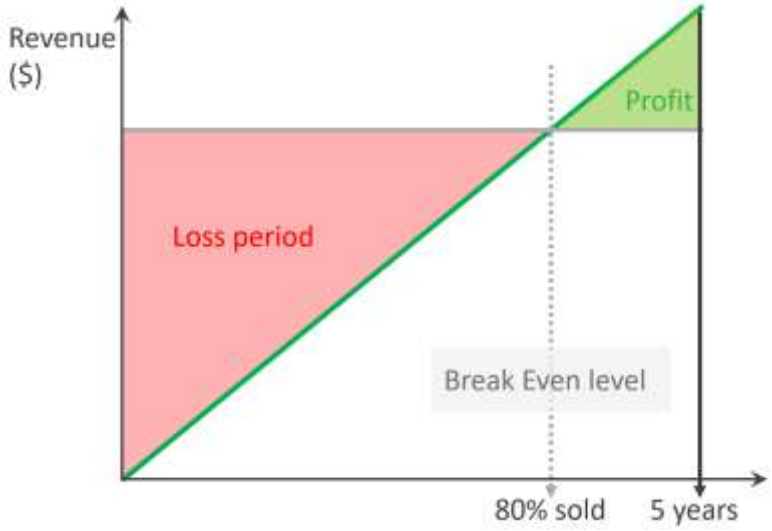
The cost of running all these VMs remain at $200 K per month. However, the total price you earn has now increased drastically, in line with the additional VMs.

As you keep adding VMs, you eventually reach your break-even point.

# Break-Even

When planning your pricing, think of the time required to reach the breakeven point. That period should leave enough time for you recoup your expenses as you likely will make a loss in the early period. It should be way before the depreciation ends.

In the following simplified example, the plan fails to balance cost and price. It assumes a break even that is too close to the end of the deprecation. While it's profitable in the end, the profit is insufficient to cover the loss in the early years.




The break-even point depends on the break-even level. You may not be able to fully sell all the resources at the end. So if your plan is based on 80% sold, then the total VM price of this 80% has to be able to cover the cost of everything.

In the following example, the financial plan correctly balances cost and price. They reach break-even point early enough to recoup the loss.




While there is unsold capacity, and hence less revenue, there is no such thing as unallocated cost.

The preceding examples are simple as they assume linear revenue. Your revenue may follow a slanted S curve, where first few months are low income, follow up rapid sales, but taper off after a few years. Regardless of how fast you make the sales, always ensure you're not making a loss at the end of depreciation period.

# VM Cost vs VM Price

There is no such thing as a VM Cost. So no point trying to figure out the cost per VM or the cost of VM 007.

To prove the point, assume you power off and delete every single VM today.

- How much money do you *save*? The answer is $0. You already have multi-year ELA contracts with various IT vendors, you have to pay regardless of how many VMs are running.
- How much money do you *earn*? Well, the payment from tenants will drop to $0. They will stop paying. Either real money to your company, or funny money to your cost center.

If that's the case, how do you link between cost and price?

The answer is the Unit Cost is based on provider object while the Unit Price is based consumer object. In the case of CPU, the cost is per 1 GHz of ESXi physical core, while the price is per vCPU of VM. The price also depends on the overcommit ratio, so it varies despite the cost being constant.

Unit Cost and Unit Price are not mathematically related. The Unit Cost does not depend on how much you plan you plan to charge. It also does not depend on planned Overcommit Ratio. Let's illustrate with an example:

- You bought 2 identical clusters from the same vendor at the same time. The total cost of both is the same.
- As a result, the unit cost is the same.
- You plan to use Cluster 1 for mission critical, and Cluster 2 for development. Cluster 1 will have no overcommit, while Cluster 2 will have 2x overcommit. As a result, you need to charge 2x for cluster 1 else you will not be able to break even as your cost per core is the same.

Unit Cost is associated with ESXi, not VM. VM is about price, not cost.

# Cost | Capacity

Cost can go down while capacity goes up.

Is old hardware more expensive? It depends on the maintenance cost and cost avoidance.

Cost covers expenses that is beyond capacity. It covers people, process, and architecture. You can reduce these costs by improving

- process effectiveness, typically achieved by business process reengineering exercise.
- process efficiency by automation. E.g. deletion of powered off VM with approval workflow.

# Cost

| | |
|---|---|
| **Cost Savings** | This is an oxymoron. You can't save on what you've already spent.<br>This also includes expenses you're yet to spend but your Finance department has committed the amount in the accounting book. |
| **Cost Avoidance** | You avoid or defer a purchase or spending. You do save cost in the present as you're no longer spending the money. |
| **Cost Optimization** | You reduce future "on-going" cost.<br>For example, instead of $1 billion a year on total IT infrastructure, you will reduce to $800 million starting next year. That's an actual savings that accounting department will register in the book. |
| | I use a fancy word instead of simply "Cost Reduction" as cost may increase fromIT department but reduce in other departments as you automate and replace human with AI. |

# Total Cost

There are 3 main components that make up the entire 5 years cost.

- Product
- Service
- People

Product and Service are always external, meaning you pay a vendor. People can be internal cost or external.

Products are infrastructure hardware and software that you pay vendors. You typically pay a vendor a bundle, as hardware typically comes with support service and management software. To get the cost, simply look at all the sales contract from them.

There are many types of Service for private cloud. The most common ones are:

- Data Center facilities. The unit varies depending on the vendors. For example, you may pay per rack for a data center service, and it comes with certain amount of power and cooling capacity.
- Managed services. Examples are VCF operations management. You typically pay that the vendor upkeeps the VCF environment timely, and make sure it's in healthy condition.

People mean time provided by a human. You pay for his time, not business outcome. The scope of service provided is generally more flexible as you pay for effort and skills, not outcome. It does not matter if the employment contact is part time or full time. You pay monthly salary or daily mandays rate.

People cost is the full loaded cost of the infrastructure team. This includes everyone needed to design, build and manage the private cloud. Yes, this includes portion of CIO salary, reflecting the portion of time he spends on infrastructure.

## Cost Categorization

When you add any cost to the total cost, think of how you will allocate back. Since vSphere VM runs on an ESXi, you want to associate the cost to ESXi. From here, you can always include the HA host and aggregate the cost to the cluster level.

Since you will always have cost component that is not sold per ESXi, how do you associate to all the ESXi? For example, how do you associate the cost of network?

This is tricky as you may pay a lump sum amount to Cisco or Arista as part of your 5-year ELA. Some of these gears may not be used by ESXi hosts only. Some of the network license may not be used in all your ESXi hosts. In this case, the only way is a rough allocation. You simply get the total cost and divide into the number of ESXi.

Let's look at more examples:

| Area | Unit | Apportion | Reasoning |
|---|---|---|---|
| Salary | People | Yes | If the person has other responsibilities, then apportion by estimating the relevant time spent.<br>Example: 10 senior IT managers with average Fully Loaded Cost of $ 1 million and 25% time spent on the private cloud.<br>A rough guide for Fully Loaded Cost for management layer is 1.5x – 2x of annual salary, to account for stocks and perks (e.g. secretary, business trips). |
| DC Facilities | Rack | Yes | Only include the racks used by the private cloud. Within these racks, estimate the portion of the private cloud gears.<br>Example: $100 million for a 10-year lease contract for 100 dedicated racks in a co-location. 70% house the private cloud. Within these 70 racks, the share of private cloud equipments is 80%. |
| Software X | Guest OS | Yes | Assuming the ELA covers both private and public clouds, only includes the Guest OS running in the private cloud.<br>Whether you're just in the planning stage (no VM is deployed) or in the operating stage (most VMs are running),<br>$10 million for 10 thousand OS images in a 5-year ELA. You want to minimize True-Up as your Finance department does not like it, so your plan includes 10% buffer as experience tells you there is always a surprise project, and the new CEO has announced aggressive revenue growth internally.<br>Since the Plan A is based on 9100 VM, do you allocate based on 9.1K VM or 10K VM? Regardless, the $10 million is the minimum commitment you've signed, so it needs to be accounted for.<br>Planning stage: the cost<br>You need to do a rough translation |

# Unit Cost

To calculate unit cost, get all the components so the Total Cost is complete. This must match the depreciation period. Do 5 years instead of 3 years as public cloud has gone up to 6-7 years.

The total cost includes everything. Any expense needed to provide the complete private cloud environment with operations. The workload running on top is not included, so it's basically the infrastructure layer.

| | | | | | |
|---|---|---|---|---|---|
| Product | Hardware | Hyper-Converged Hardware | Total Cost | CPU Share | 40 % |
| | | Server | | Memory Share | 40 % |
| | | Storage | | Disk Space Share | 30 % |
| | | Network | | Others Share | 0 % |
| | Software | Broadcom VCF | | | |
| | | Windows, Linux | | | |
| | | Security Software | | | |
| | | Infrastructure Software | | | |
| Services | Datacenter Facility | Cooling | | | |
| | | Power | | | |
| | | Space & Rack | | | |
| | | Internet | | | |
| People | Infrastructure Team (insourced or outsourced) | Network Team | | | |
| | | Server Storage Team | | | |
| | | Other Infrastructure Team | | | |
| | | IT Management | | | |

Infrastructure software includes softwares such as security and back up.

People including full time, part time and outsourced contractors. the salary of full-time employee and IT leadership.

Dividing the total cost into CPU, memory and disk is not easy. For examples:

- VMware Cloud Foundation contains storage components.
- Network hardware is also used by storage if the storage array is Ethernet based.

Why is the total cost only split into 3 components? The share of others is 0%.

The reason is this is not cost, not price. The Unit Cost has to be based on provider layer. The Unit Cost can not be allocated to consumer (VM or K8 Pod) as we do not know how many they are.

## Unit Cost

Unit Cost depends on hardware and software. New cluster should cost less due to bigger hardware.

For the CPU portion, it is expressed in physical core, not vCPU nor GHz. Notice how software vendors also charge their product by core. How much you price that core depends on the overcommit ratio and SLA.

## Complication with CPU

The above works well with memory and disk space. Both have only 1 unit.

CPU has 2 units. One for speed, one for space.

As you're dealing with capacity, choose the one for space, which is in physical threads.

An ESXi with 10 cores, 20 threads, always have 20 threads of capacity, regardless of the power management settings. Unit Cost is derived by dividing against 20 threads. There is no Class of Service in cost, as the overcommit ratio is always set to 1:1.

## Cost Allocation

What if you need to pass the cost back to your internal consumers? Your IT department is not a profit center, so you just need to show other departments the cost you spend and then divide among the consumers.

The answer is "the request is invalid".

- The reason is your cost is actually price to them.
- Likely, you don't enough demand in the first year. Can you pass this total cost among few tenants? You can't, as they will not pay higher than what AWS or Azure discloses publicly.
- In the last year, where you have full demand, do you intend to pass on the savings? If yes, how will you recover the loss of the initial years?

Based on the above, it's clear you should not allocate cost to each running VM.

Let's use an example from service industry where the practice is matured.

An airline plane carries 300 passengers and fly 500 times a year on multiple routes. There are potentially 150 K seats passengers can buy. The airline has a price for every one of those seats, and they will manage and adjust the price dynamically. There are days they make a loss on a flight, there are days they make a profit.

Do they have 150 K costs for each seat? No. There is no need to calculate the profitability of every single seat.

All they need is the standard or general unit cost.

Same thing with VM as a Service. If you insist on allocating cost to every running VM, you likely end up with incorrect calculation. Show the price instead.

# Cost Avoidance

Since the savings is always in the future, you should not use historical or past value as the value of money changes over time.

Let's take a simple example:

- You spent $3 million on a hyper-converged infrastructure (HCI) solution 5 years ago.

- It has been used well, and capacity remaining is now 0%, so you need to buy a new HCI. This will cost you only $1 million as the cost of HCI solution has gone down by 2/3 in the last 5 years.
- Via a diligent and arduous reclamation process, you manage to free up capacity. As a result, you do not need to spend the $1 million. You can defer this purchase to the next fiscal year.
- What's your cost savings from this reclamation: $3 millions or $1 million?

Accounting wise, it's $1 million only as that's the amount you defer to spend. While that HCI cost you $3 millions 5 years ago, a brand new set with equivalent capacity costs a lot less. In accounting fundamental, you should not mix numbers from different date, let alone from different fiscal years.

BTW, depreciation is not relevant here as you're talking about new system, and not the old one. Depreciation applies to the old storage.

The $1 million cost avoidance is certainly an *estimation*. The actual cost to be avoided or to be spent depends on vendor quotation, and your negotiation skills. Take note that the actual cost typically is more than the HCI cost. Additional costs can exceed the hardware cost. You need to include the full loaded cost, such as data center facility, implementation service, back up storage, administration service, software licence, management, etc.

Reclamation alone does *not* save cost. How much do you save when you delete files in your notebook?

Right. Zero.

Only when it helps you defer buying a new drive that the reclamation becomes real. But it's a cost avoidance, not cost saving.

How about service? We like to cite productivity improvement as a cost saving. While this delivers business value, it is not a hard cost savings. It is a soft benefit with no accounting value. The hard savings only happen when the need to buy additional resource/headcount is deferred, or reduction in Managed Services contract value.

# Cost Avoidance

| Area | Action | Calculation |
|---|---|---|
| Storage | Delete orphaned files and powered-off VMs.<br>Delete unused files from Guest OS | GB of disk deleted files * Cost per GB of new capacity.<br>E.g. 10 TB files * $10 per TB = $100 *future* expense avoided. |
| Compute | Power off idle VMs<br>Reduce oversized VM | Total GB of Consumed RAM saved.<br>Total vCPU saved.<br>Express the above in terms of No of ESXi, multiply by the average cost of *new* ESXi. |
| Network | Consolidate and power off hardware | Total physical network ports saved. |
| Facility | Power off equipment.<br>Change DC Provider to cheaper one. | |

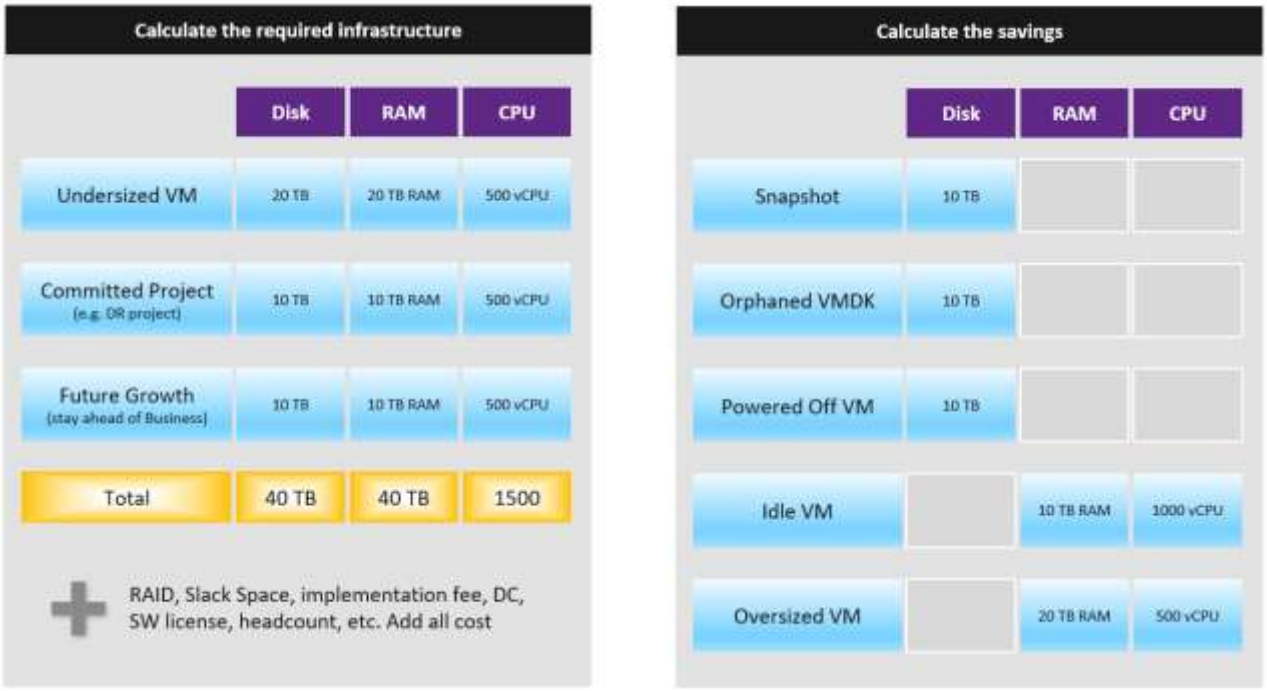
| | | |
|---|---|---|
| Service | Move from cloud to on premises for long term workload. Cloud gets expensive with long term contract.<br>Optimize cost so it does not grow as fast as business demand. | Get the actual monthly bill from your cloud provider. Use 1 year of cost and exclude the highest 20% bill as that is likely a momentary spike. |

IT needs to be ahead of business. When calculating the cost avoidance, includes committed projects and future growth. It's common for projects to not have enough capital to buy from vendor. Reusing existing assets can go along way.

You should also take into account undersized VMs, as the application team may demand that they are upsized.

Calculate CPU, RAM and Disk. If possible, include network too. It is harder to calculate, as by nature it's just interconnect. For each of these three IaaS resources, calculate both the demand and the reclamation. For the demand, don't forget to include the full cost. When a VM needs 100 GB, it translates into a lot more as you factor is DR, cyber recovery, back up, snapshot, etc.

The following table provides an example.




You need to prepare the above table per physical location. Just because you have 10 TB RAM in Singapore does not mean the VMs in Armenia can use it.

# Cost Optimization

For organizations with a large infrastructure footprint, tech refresh is a great way to reduce cost. Going down from 100 racks to 50 racks will reduce both capital and operating cost if you can reduce software licensing cost. Nowadays the software, especially business application software and not infrastructure software, costs more than the hardware.

| Area | Action |
|---|---|
| Vendor | Vendor consolidation.<br>Some vendors prefer you commit long term and will give you lower unit cost. |
| Hardware | Technology refresh. Moving from FC SAN + Array to ethernet based HCI should deliver lower cost. |
| | Virtualize. While server virtualization is widely practiced, there is much more work to be done for storage and network virtualization. Use vSAN as it's already part of VCF. |
| Software | Technology refresh that standardizes or reduces license count. |
| | Removal of overlapping software |
| People | Do away with less employees. In general, many organisations are both top heavy and laden with overlapping departments. A classic example is a centralized Project Management Office and the middle management layers.<br>Be careful with accounting engineering where headcount is saved but the work is outsourced. |
| Facility | Power off equipment and reduce rack footprint<br>Change DC Provider to a cheaper one. |

# Optimized Cost

The above exercise will help in optimizing cost. There are certainly other avenues to optimize cost, as cost covers more than just capacity. It covers People, Process, Architecture.

| Level | | What |
|---|---|---|
| Consumer | Process<br>Guest OS<br>Container<br>VM | Remove wastage, such as Orphaned VMDK, Snapshots, Powered off VM and Idle VM.<br>Reclaim by shrinking oversized VM. Only useful for allocation based from cost viewpoint.<br>Automate, with approval workflow & audit trail. |
| Provider | ESXi<br>Cluster<br>Datastore & DS Cluster<br>Switch and Port Group<br>Hardware | Hardware tech refresh. Newer hardware have more capacity and faster performance, at the same price. You can also save expensive software license (e.g. database, middleware).<br>Virtualize storage and network, not just compute.<br>Consolidate. Small clusters have higher HA overhead, smaller datastore have higher overhead. Optimize cost by consolidating them.<br>Increase utilization of clusters and datastores, without compromising performance.<br>Reduce overhead. Review if the applications truly justify active/passive, resulting in 50% overhead.<br>Standardize the architecture. This reduces complexity, not cost. |

# Complexity

Complexity has cost.

- A public cloud outage, despite not being your fault as you're just a consumer, can cost your company reputation, business and regulatory fine.
- Some cost such as company reputation is priceless.
- Complexity is hard to quantity. Human error can be costly but you need to resort into probability theory to quantify that.
- Reduction of complexity typically increases cost. Simplifying operations, such as not mixing VMs with different class of services in the same cluster, will reduce complexity. But it also comes at a cost of larger infrastructure.
- Standardization will reduce chance of human error. But this also means less flexible configuration, which tends to increase cost. One way to reduce is automation as that reduces the human cost.

# Price

Price is what your customers care as that's what they pay. Since you compete with public cloud, your VM price is largely set. Yes, it's a commodity market after all.

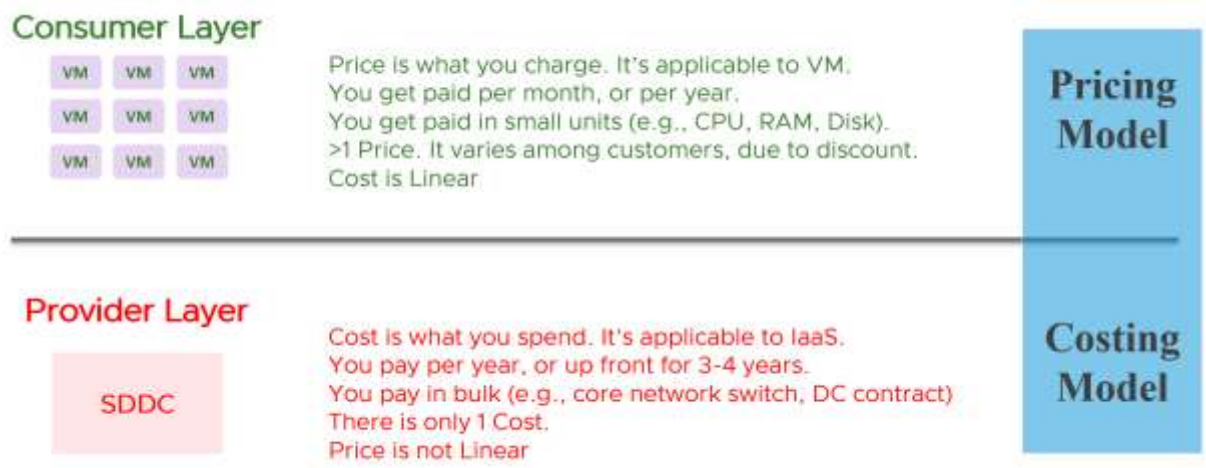
VM Price is not a function of VM cost adjusted with margin, discount and penalty. Price is determined by the value perceived by the paying customer. The cost is actually irrelevant as customers do not and should not have to care about how you manage your profit and loss.




Unit Price should remain the same within the same class of service. Using the airline industry example, the ticket price does not depend on the plane generation. Singapore Airlines has multiple generations of business class seats across different size of planes, yet you never see them price based on that.

Overcommit Ratio is the way you justify a higher price, hence it's imperative to disclose upfront to your customers.

You need to develop both the Pricing Model and the Cost Model together.




There are 2 types of pricing model

- Allocation
- Utilization

Allocation is best when the consumption has a limit. That's why it fits VM well, as a VM always has fixed CPU and Memory configuration. A 64 vCPU VM is charged more than a 8 vCPU VM, regardless of utilization. Both can be idle for days, and the bigger VM will be charged more.

Utilization is the only choice when there is no limit. A container without limit means we don't know what amount to allocate as it does not have predefined size.

# Usage-based Pricing

What exactly is "usage"?

The answer actually varies, depending on the service.

Let's say you provide desktop as a service. If you truly charge based on usage, then if a user does not log in, then there is no usage. The problem here is you need to provide the desktop available 24 x 7, as you do not when the user will login. If you have 1000 users, you need to make sure you can cater for the peak demand, else there will be complaint from users who do not get a desktop when they ask for it.

A good comparison is NetFlix or Internet Broadband. You pay a fix fee, regardless of your usage. Whether you are watching 24 x 7 or just occasionally, you are paying the same rate. The same with your home broadband.

In Kubernetes, pods are by default unlimited. It can grow the size of the node. As the pod has no fixed size, you can't charge based on allocation.

In this case, here are the steps:

1. Work out the total cost.
   This means the sum of everything that you're spending. Do it over 5 years, or whatever the depreciation time you use.
2. Normalize the above to hourly.
   This gives you the hourly total cost.
3. Figure out your capacity.
   Usable, not Total. This is your sellable capacity.

4. For K8 or vSphere, allocate into Compute and Storage.
   Since K8 uses core instead of MHz, you charge using core or milicore as the unit. Do not charge per MHz as that complicates the formula.
5. Project your expected utilization. If you only project 40%, then your price has to be at least 2.5x of the cost.

To translate into price, consider the peak hour. For example, you may mark up during office hours and provide discounts for after office hours. Some workloads need not be run immediately. By having "peak hour" pricing, you spread out the demand to weekend.

What about storage IOPS and throughput? One way to prevent abuse of the shared environment is using cost. This is tricky though, as the usage comes at 2 levels (Guest OS and VM). For example, do you charge for the IOPS caused by back up and snapshots? That's why it's important to keep the pricing model simple.

# Progressive Pricing

As an internal cloud provider, what business problems do you want to solve with pricing? Use price to drive the right behaviour and encourage adoption.

Oversized VM is a problem that is best solved *before* the VM hits production. So design your pricing model to encourage the right size from the beginning. Right size, right from the start. Create a progressive pricing and apply discounts for smaller VM sizes. The following diagram shows an example of tiered pricing. Premium pricing is applied on VMs larger than 16 vCPU, while discounted pricing is applied on VMs smaller than 8 vCPU.



# Class of Service

How do you apply the progressive pricing above into different Classes of Service? How much of a premium should you put on the big VM? How deep should you discount the small VM? The multiplier effect (the progressive tax) cannot be too high because public cloud does not have such tax. They follow a linear pricing. If you use a high multiplier, your price will be too high, or you will absorb a deep loss.

The following table provides an example of multiplier.

| Class of Service | Multiplier for each VM CPU range | | | | |
|---|---|---|---|---|---|
| | 1 – 4 | 5 - 8 | 9 - 16 | 17-24 | >24 |
| Tier 1 (Gold) | 0.8 | 0.9 | 1.0 | 1.1 | 1.2 |
| Tier 2 (Silver) | 0.8 | 0.9 | 1.0 | n/a | n/a |
| Tier 3 (Bronze) | 0.8 | 0.9 | n/a | n/a | n/a |
| Tier 4 (Free) | Capped at 2 | n/a | n/a | n/a | n/a |

We apply the same principle for RAM.

| Class of Service | Multiplier for each VM RAM range | | | | |
|---|---|---|---|---|---|
| | 1 – 16 | 17 – 32 | 33 - 64 | 65 – 96 | >96 |
| Tier 1 (Gold) | 0.8 | 0.9 | 1.0 | 1.1 | 1.2 |
| Tier 2 (Silver) | 0.8 | 0.9 | 1.0 | n/a | n/a |
| Tier 3 (Bronze) | 0.8 | 0.9 | n/a | n/a | n/a |
| Tier 4 (Free) | Capped at 4 | n/a | n/a | n/a | n/a |

Keep your pricing model simple. The more complex your bill, the more you have to explain. The following table provides a suggestion of what to charge and what to bundle. Bundling means you include it in your overall cost but not charge explicitly for it. You are certainly trading off accuracy with simplicity.

| | What You Charge | What You Don't Charge |
|---|---|---|
| Compute | vCPU, vRAM | Usage, Guest OS, Patching, etc. |
| Storage | vDisk in GB | IOPS, Snapshot, Backup, DR, etc. |
| Network | None. All included | Bandwidth, IP Address, FW, LB |
| Security | None. All included | AV, IDS, IPS, Compliance, etc. |
| Service | None. All included | Monitoring, Troubleshooting, Management, Portal |
| Datacenter | None. All included | Space, Power, Cooling, UPS, etc. |

Overly simplified pricing could be unfair to customers, but that's common in other industries. Take the airline industry, where my favourite airline is Singapore Airlines. I notice they have at least 4 generations of planes. The new plane is more efficient, costs less to operate and is more enjoyable to customers. On the other hand, if you take into account depreciation, the old plane is already fully depreciated. And yet, the price is the same across all generations.

# Private Cloud | Public Cloud

In the cloud era, application teams are provided with more choices of infrastructure. All they need is a credit card and application development can start. No need to deal with internal red tape just to get a bunch of hardware, since infrastructure is all available as an on-demand service.

The public cloud providers are competitors to internal on-prem cloud. These vendors are happy to migrate all your workload to their cloud, and replace the infrastructure team with their own staff.

In reality, public cloud and private cloud are complimentary. They have their pros and cons. See this for comparison.

As an internal cloud provider, you need to turn the public cloud providers into allies. That requires a shift in your business model, from infrastructure provider to a multi-cloud service broker. You broker the request for infrastructure with the most appropriate provider. You evaluate, choose and deliver multiple clouds if the on-prem cloud does not meet the business needs. Even the on-prem cloud can be a service that you procure (meaning you do not own the hardware and software), if that fits your business requirements better.

It is indeed possible for a small and no-frill internal IT infrastructure team to complement a much larger cloud provider. Being small, especially since you are on-site and work in the same company, enables you to offer a better service. Nobody likes dealing with the *bureaucrazy*, pun intended, of a large corporation's support organization. You can get lost in the mountain of policies.

You also need to do an apple-to-apple comparison. List the entire components of the service. The following table provides an example, where you add your private cloud alongside externally hosted cloud. You should complement this table with another table comparing the SLA and price. I've provided a sample of SLA table in the Capacity Management section.

| Component | AWS | VMware on AWS | On-Prem Cloud | Remarks |
|---|---|---|---|---|
| Server Hardware | Identical | Identical | Identical | Comparable hardware spec |
| Storage | Excluded | Included (vSAN) | Included (vSAN) | Cost under Storage |
| Backup & DR | Excluded | Excluded | Excluded | Cost under Storage |
| MS Windows | Included | Included | Included | |
| Hypervisor | AWS | vSphere | vSphere | |
| Management Tool | Excluded | VCF Operations | VCF Operations | |

| | | | | |
|---|---|---|---|---|
| Support | Excluded | Included (remote) | Included (on-site) | IT provide full time on-site support |
| Sys Admin | Included | Included | Included | |
| Security | Excluded | Included | Included | AV, Firewall, IDP, IDS, etc. |
| Network: Bandwidth | Excluded | Excluded | Included | |
| Network: Core | Add it | Included | Included | NSX + physical + People<br>LBaaS, FWaaS |
| DC Facility | Included | Included | Included | |

The comparison can be done in 2 ways:

| | |
|---|---|
| Consumer | You compare a single unit of consumption, and over a short time.<br>For example, you compare a VM with 4 vCPU 16 GB RAM 100 GB disk. You take the daily cost, not the 3-year cost. |
| Provider | You compare the whole infrastructure to support all the consumers, and over a long time.<br>For example, you compare the entire private cloud over 5 years. The cloud may have 10K VM on it. |

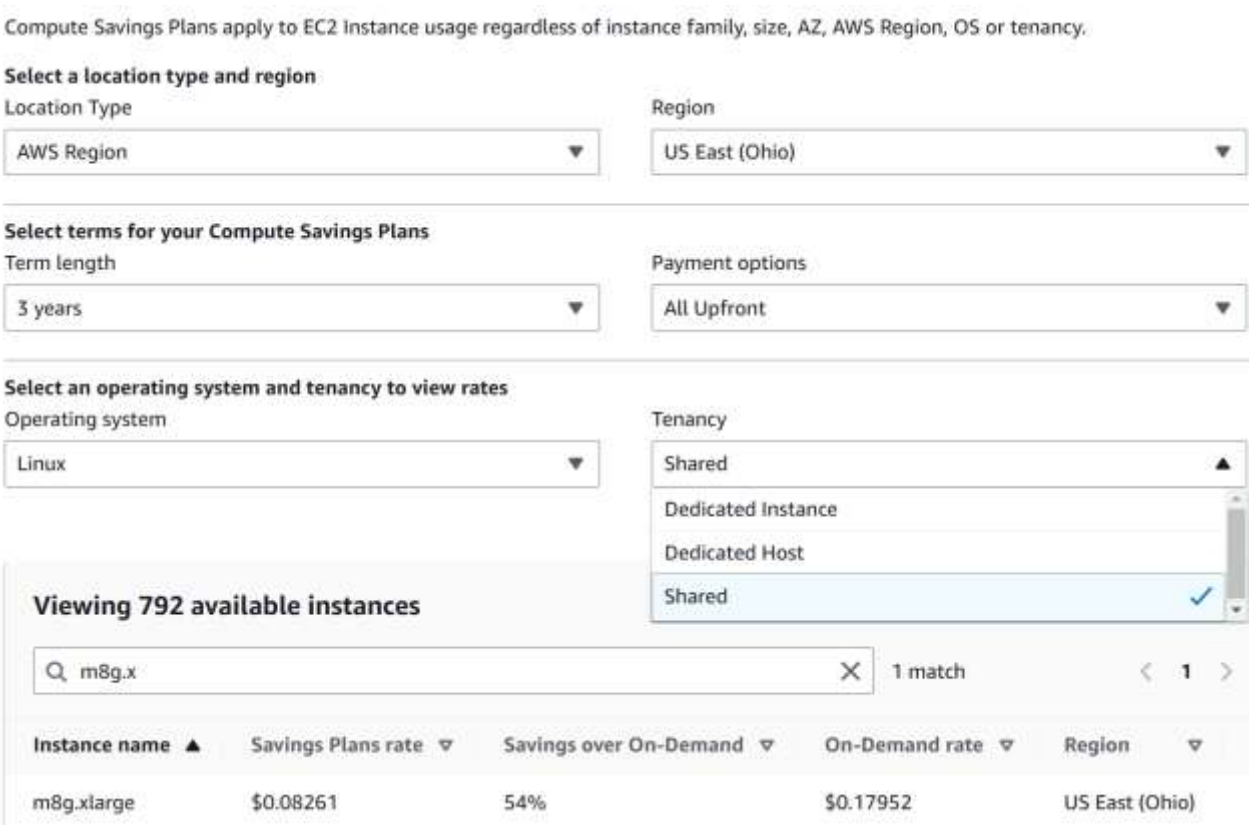
# Sample Comparison

In this example, we take the lowest cost from AWS. Start with EC2 as that's the most popular services that matches what you provide. Take the lowest possible cost. In this case, it is 3 years commit with full upfront payment. Also, take Linux as opposed to Windows.

For CPU, we take AWS Graviton, as it's cheaper than Intel Xeon and AMD EPYC.

For tenancy, we take Shared as it's cheaper. This means your private cloud has an advantage as it's not shared with other companies.

For VM size, we take 4 vCPU and 16 GB RAM as that's the most popular size.




Since you commit this for 3 years regardless of usage, you pay US$ 2171 in advance for a 4 vCPU, 16 GB RAM, 1 TB SSD disk VM without Guest OS.

*END OF PART 1*

*Okay… now that the concept makes sense to you, let's move into how to apply and consume them.*