Fungible Virtualization Stacks

Refocusing on Optimization of Underlying Resources

A Technical Paper prepared for SCTE/ISBE by

Keith Alan Rothschild
Principal
kar@cox.com

Guy Meador III
Senior Solutions Architect
Guy.Meador@cox.com
Cox Communications
Technology Solutions Engineering,
6305B Peachtree Dunwoody Road
Atlanta, GA 30328

Brian Kahn
VP, Solutions Architecture
Sea Street Technologies
401 Edgewater Place, Suite 570
Wakefield, MA 01880
bkahn@seastreet.com

Table of Contents

List of Figures

Summary

Industry focus on stack providers such as VMware and the various OpenStack implementations has prevented operators from focusing on the core problem: optimization of the underlying resources. Leveraging an integrated Policy-Driven Model-Based Service-Orchestration and Resource-Automation framework, we developed a solution where we can optimize the underlying resources, and where the span of control for any given virtualization stack is what is dynamically managed. This helps us address several of our critical use cases in addition to optimizing raw underlying resources including: (a) power management - turning off un-utilized resources, (b) services of bare-metal requirements for CDN, some real-time processing applications, etc., and (c) managing the needs of specific virtualization stacks (including isolation into secure network zones) required for specific VNF and/or IT domains.

Introduction

The systems we are designing now, and will be implementing over the next few years, need to support innovation over the next decade. It is useful to consider potential future scenarios when positing the span of control of each system. Consider two system environments: the consumer (home or business) and the data center.

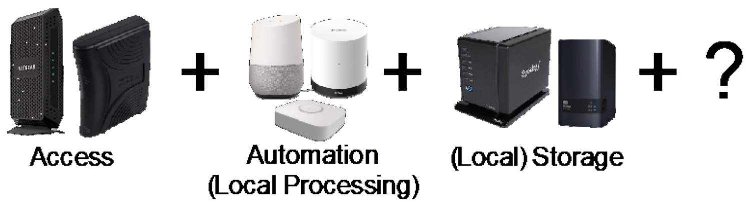

Figure 1 - Universal CPE

There are competing formulations of what makes CPE "universal", with many aspects depending on viewpoint. Consumers may want it to be a retail-available next-generation version of the cable modem, bringing WAN and LAN connectivity, and to allow for any number of additional features such as connected storage, compute, and/or to be converged with their home automation hub. The operator wants to minimize the number of devices deployed at a given customer location to support all of the operator's products at that location – even to the point of deploying a single device - so that the operator neither maintains multiple versions of CPE to support any given customer, nor maintains multiple profiles of CPE across broad customer segments. When determining the functions to be performed on the CPE vs. elsewhere in the operator's span of control, issues related to latency and the importance of continuity of service during loss of WAN connectivity may be important.

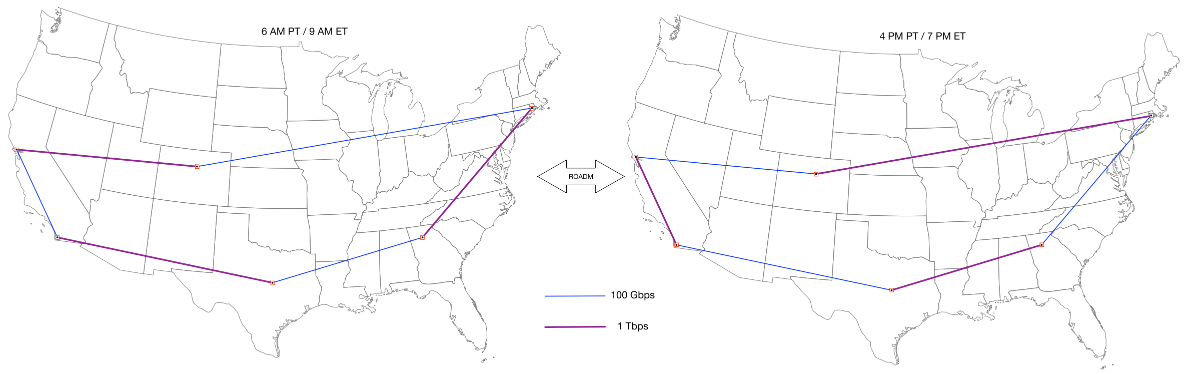

Figure 2 - Time-of-Day Network Reconfiguration

For the data-center environment, the ability to get the most benefit at the lowest cost is highly desirable. This may mean rebalancing optical links between data centers using Reconfigurable Optical Add-Drop Multiplexers (ROADM) to minimize the quantity of the most expensive optics required. To the extent that demand drives electricity and cooling costs, managing demand at operator locations throughout the day may become an important factor in reducing overall operating costs. Demand-shifting strategies may become important, such as making processing payloads "follow-the-sun" (to make use of solar/photovoltaic energy while addressing peak demands). Another use case is reduction of number of compute centers, for example, using excess capacity in the West or East to serve Central needs based on differences in consumption during different times of day, as shown in Figure 2.

With these scenarios in-mind, we need to identify the most important design principles, the universe of resources involved, and the optimal way to manage those resources over the long-term.

1. First Principles

1.1. Service Exposure Pattern and Approach

It is recommended to adopt a top-down, service-centric approach. Specifically, do not define offerings by the way they are technologically constructed and deployed (ex.: resources, stacks or physical location), but by service functionality, characteristics, interfaces, and consumption model. The service exposure pattern has usually been the domain of "cloud" orchestrators and software-only services. It is not identical to SOA, but shares many related concepts applied in a more generalized way, including constituent hiding, functional service interfaces, well defined attributes, and interfaces to control and automate services. It is applicable not only to software-based services, but should include hardware, network, and software. The key result: all services exposed through the pattern having the same functional interfaces and essential service attribute values are equivalent regardless of composition.

1.2. Composites and Composability Across Domains

A ***Composite*** is defined as an entity that contains a number of parts, called ***constituents***, that are used by the entity as a whole. At any particular time, a constituent may be integral and dedicated for use by the whole or, in some cases, may be shared between composites. For this paper, composites and constituents are technological resources or services within, or across, one or more domains.

Composites can be assembled with one of two high-level approaches: static assembly or dynamic assembly. Static assembly implies a tight integration and association between the composite and constituents in a manner that is not intended to be changed over time, except as a result of undesirable events (ex. breakage) or design changes. Dynamic assembly may, indeed, have tight integration and association to the composite, but that association is more changeable in nature and occurs at any time in the normal course of composite operation (not only, for example, in response to constituent breakage).

Virtualized and software-controlled environments make dynamic assembly a more viable strategy as compared to a hardware-centric environment. Moreover, dynamic assembly of composites is essential to flexible, responsive, and agile service delivery. This gives rise to the Principle of Dynamic Composability: The aggregation relationship between a composite and its constituents should be changeable at any time in the composite's life cycle.

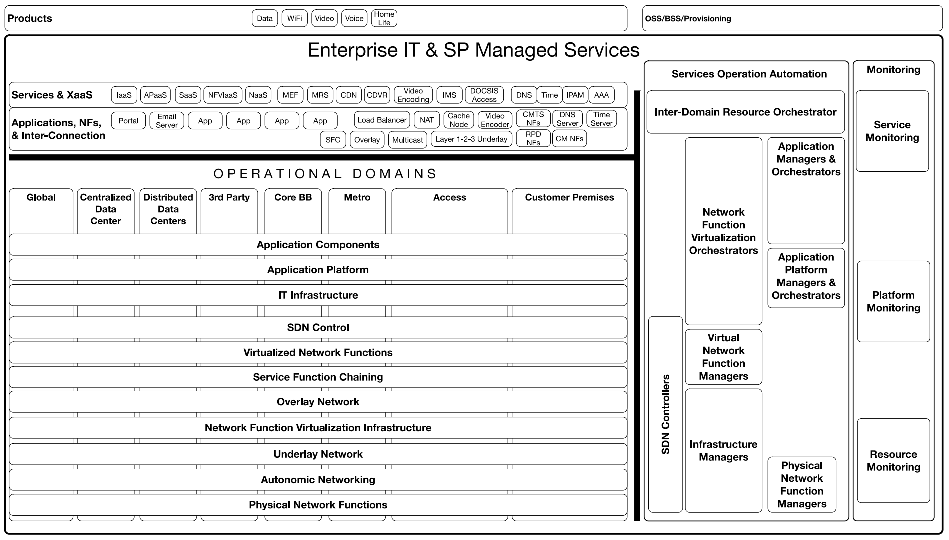

Figure 3 - Domain Diagram

Each domain and the services it provides should be dynamically composed of resources (and, possibly, other services), with such composition being created and changed over time under the control of a higher-level (sovereign) system, such as the Inter-Domain Resource Orchestrator (IDRO) proposed by BT:

"The development of rich NFV business cases depends on agreeing [on] each layer's responsibilities and the application programming interfaces (APIs) and service-level agreements (SLAs) through which each exposes its functionality to others. Otherwise, BT points out, nonsensical scenarios will arise, such as an NFV IaaS provider ceding resource allocation control in its own infrastructure to its customer's NFV Orchestrator (NFV-O). […] Global resource management is parked in the NFV-O alongside service orchestration, but BT and others argue that it should be separated out into a fourth layer: the inter-domain resource orchestration layer." (Chappel, 2015, p.3)

This arrangement enables global, end-to-end, policy-controlled decisions to be centrally determined and to take effect uniformly across and between the domains. Indeed, resources could be provided to one domain for a time and then reassigned to another domain, driven by global decisions and policies arising from, for example, business needs or operational considerations that exceed the scope of any one of the domains.

1.3. Resource Abstraction

Expect resources to change over time as services, technologies and costs change. Focus on a stable, extensible approach to achieve thorough implementation and consistent reuse of policy and business logic. Resources will converge. Policy and business logic will be used to create business differentiation.

Incorporate abstraction and software infrastructure convergence so resource changes are seamless and straightforward. Make certain that any service can call upon, consume and control resources from any-and-all underlying infrastructure or cloud systems (i.e., compute, storage, networking, and future systems) so evolution is modular and straightforward.

Virtualization Stacks should be Composites supporting the Dynamic Composability Principle and Service Exposure Pattern, and should be managed with a layered operations automation approach that composes the resources and services and exposes the desired interfaces and functionality without revealing their composition.

1.4. Optimize for the Business First

Optimize for the business first and the infrastructure second. Expressly enable optimized recurring configurations, even if they are different in different organizations. Optimize infrastructure at the raw resource level (i.e., compute, storage, network links, PNFs/VNFs, stack licenses, etc.), not at the finished configuration level. Enable the business first. Execute, control and assure the full operational lifecycle of services for maximum efficiency, reliability and value. Do not stop at fulfillment.

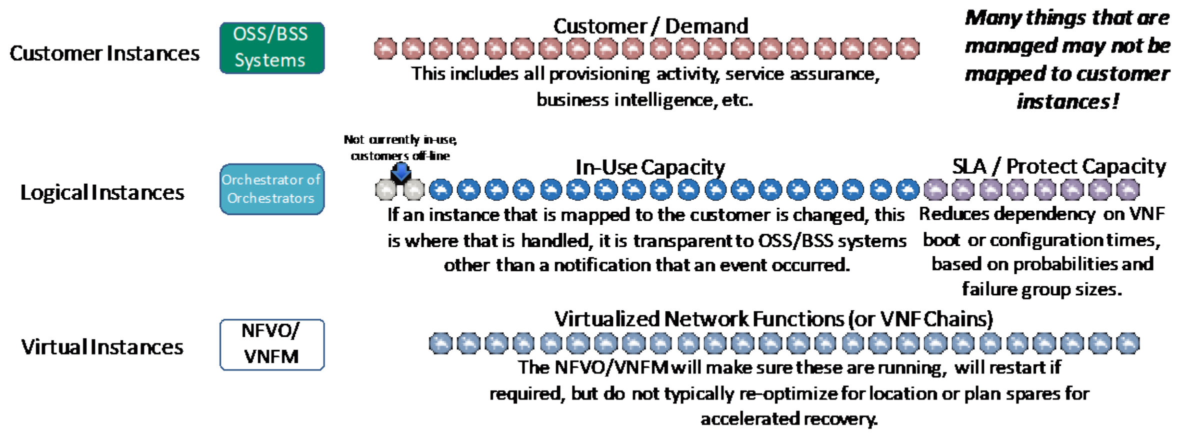
Customer Instances
OSS/BSS Systems
Customer / Demand
This includes all provisioning activity, service assurance, business intelligence, etc.
Many things that are managed may not be mapped to customer instances!
Logical Instances
Orchestrator of Orchestrators
Not currently in-use, customers off-line
In-Use Capacity
If an instance that is mapped to the customer is changed, this is where that is handled, it is transparent to OSS/BSS systems other than a notification that an event occurred.
SLA / Protect Capacity
Reduces dependency on VNF boot or configuration times, based on probabilities and failure group sizes.
Virtual Instances
NFVO/ VNFM
Virtualized Network Functions (or VNF Chains)
The NFVO/VNFM will make sure these are running, will restart if required, but do not typically re-optimize for location or plan spares for accelerated recovery.

Figure 4 - The Relationship Between Virtual Instances and Customer Instances

1.5. Model Based Approach

Use models to operate the lifecycle of services and the resources. Manage the models to manage the service or resource. Encapsulate service lifecycle operations using a consistent, model-based service induction methodology so operations logic is portable and can be understood and simulated alone or in context.

Control operations via design-able, reusable policy models. Encapsulate business logic in policies. Design policies once and reuse them across multiple services, offers and applications as needed. Policies scale in a way that people cannot.

Expect hybrid cloud, inter-cloud, NFVI, and hyper-converged infrastructure models to present and reinvent themselves over time and be ready to include them. Choose a methodology and a platform to allow modular flexibility to progressively add innovative resource types as required.

Perform service level cost modeling before and after implementation to identify and prioritize business, operational, financial and compliance factors. Let the numbers drive the resource and operational decisions. Remember, the cost to automate should be compared against the cost of not automating, including the development and support of M&Ps which will often have a comparable cost to automation independent of how often the M&P might be utilized. An M&P that is rarely used is more likely to introduce uncertainty than the corresponding automation.

2. Resources

For the purpose of this paper[1], we will address two major classes of generalized telecommunications resources, separating them into compute (processing, storage, etc.) and links (networks). Improvements in hardware capabilities have resulted in general purpose hardware being able to displace specialized hardware needed for supporting network functions. The ability to deploy virtualized network functions (VNFs) and the separation of the data plane and control plane in networking enables the creation of physical (underlay) networks and the use of software to control the realized (overlay) networks in a manner that can be flexibly reconfigured as needed. These trends are both in-line with what we would expect from general technology evolution - but are far from where this evolution will end.

Responsible architectural analysis and design will contemplate both the implications of these evolutionary steps, predict the likely vectors of subsequent evolution, and determine if it is worthwhile to predispose solutions to support any specific vector. Optimal placement of compute and link capacity is moving from highly specialized fixed components to components that are generalized and more flexibly configurable, creating a rather complex resource utilization problem.

Placement of general purpose (compute) resources should be contemplated for these variants: locally on customer premises (CoP, Compute on Premises), regionally (as it relates to the provider network), centrally (as it related to the provider), and remotely (third-party). Similarly, (network) links should be

[1] This differs from normal treatment in that what is normally referred to as compute is described as processing, and the term compute is used to describe both processing and storage. Similarly, a third class is often referred to as networks, whereas here, the term links is used instead. Components such as switches or routers are often classified as “network” rather than as compute, however, they are packet processors, and as they become virtualized, may be replaced by generalized compute. As such, the paradigm of “compute, storage, and network” is replaced with “compute and links”, which is more germane to this analysis.

contemplated as they connect the customer premises (access), connect resources (data-center), connect regional data-centers (metro), connect centralized data centers (backbone), and provide for connection to third-parties (interconnects).

Service Provider facilities encompass a variety of configurations and locations into which generalized compute resources are placed. These facilities range from large, centralized data centers to regional data centers, to edge facilities. The evolution of the kinds of generalized compute resources that are placed at each of these facilities, and the mix and scale of compute resources within them will be dynamic and ever-changing (ex.: hybrid cloud, inter-cloud, NFVI, and hyper-converged infrastructure, etc.). In tension with these trends is the goal to select and manage compute resources in a manner that makes them usable by the largest set of services over time without the need to perform manual/physical changes; once put in place, the same compute resources should be usable by any service's software elements as long as the resource meets the minimum capability profile for the software element.

At present, CoP devices are likely to be x86-based, but, at scale, more cost-effective solutions are expected to emerge. Over time, device capability will improve, with the potential for device availability through retail channels and device upgradability (ex. compute capacity). These dynamics will further fragment the uniformity of capabilities across the device population. Independent of capability, these devices will be expected to support a combination of services, as subscribed to by the customer.

As denser alternatives to x86 CoP devices become popular, the desire to benefit from this in the datacenter will begin to gain momentum, and we will see a combination of hybrid virtualization stacks and virtualization stacks dedicated to dense packet processing.

More capable CoP devices may be able to reduce the traffic demands on the access network, and depending on business models and who provides the CoP device, the bandwidth allocation on the access network may need to be dynamically adjusted to accommodate placement of network functions on the CoP or remotely.

The concept of Universal CPE (uCPE) is closely tied to Compute on Premises (CoP). The understanding of what constitutes uCPE may lay within the perspective of the stakeholder, and even then, could be differentiated by the role. For Service Provider (SP) product managers, the universality of uCPE could mean that it will support any combination of services that SP offers, for instance, some combination of video, voice, internet connectivity, home security and/or home automation services. Supply Chain or Field Service stakeholders might also include flexible (possibly modular) support of a multitude of WAN connectivity options. Additionally, we can expect the desire for this to be equipment provided by the customer and available in retail outlets, in order to reduce expected capital outlays. For retail-available components, the relationship between the connectivity aspect and CoP aspect of CPE may be severed, or at least may include the ability to modularly supplement CoP with devices such as NAS and Home Automation Hubs.

The desire for CoP, from the consumer's perspective, goes beyond simple optimization of compute and links, but includes capabilities such as caching of IoT data or the desire to improve performance of automation capabilities by having some of the decision processing occurring on-site. Retail providers of CoP will want to optimize cost at volume, and even if x86-based devices were the preference of the SP community, the OTT and IoT communities would likely move to a more cost effective dense packet processor. As VNFs evolve towards a micro-service orientation, the CoP domain is likely to be a hybrid of SP and customer-provided compute resources; therefore, convergence is expected across SP-services, OTT-services, and IoT device capabilities. The question isn't whether the CoP will evolve to support multiple processing paradigms, but whether the edge-compute and centralized-compute environments will also embrace supporting a hybrid of x86-based and dense-packet-processing-based paradigms. Similarly, we will have to consider management based on the availability of equivalent VNFs for each of the compute paradigms. The need to satisfy multiple compute domains across multiple service offerings that are provided and managed by multiple parties will greatly challenge the practice of treating compute as a resource under the control of the SP NFV architecture.

An implication of the convergence of CoP usage is the provider/consumption model for CoP itself. Consider that businesses and residences are inherently dependent on the managed delivery of services such as power, water, and sewer. In fact, it is the rare circumstance that these services are offered and consumed as anything other than fully-managed services. It may be that within a short amount of time businesses and residences will also become inherently dependent on fully-managed CoP, used as assumed infrastructure. The main aim of such a managed CoP service is to provide available infrastructure positioned between the home network and external networks for use by multiple services and parties.

3. Resource Consumption

Many applications utilized by enterprises, which may be the most common class of applications supported in the IT environment, require a highly available infrastructure. A highly-available infrastructure drives the need for expensive components. Virtualization stacks that offer high-availability and nearly transparent VM migration are likely best suited for such enterprise applications, but, this approach, generally, comes at a cost premium as compared to an approach where the virtualization stack does not offer those capabilities and the application is designed accordingly.

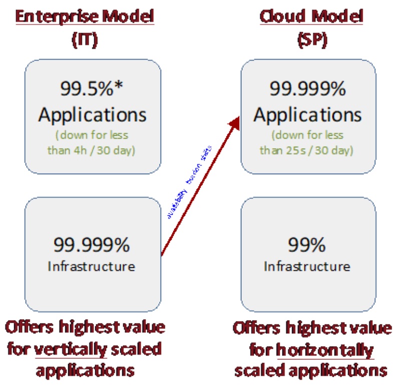
Enterprise Model (IT)
Cloud Model (SP)
99.5%* Applications
(down for less than 4h / 30 day)
99.999% Applications
(down for less than 25s / 30 day)
availability burden shift
99.999% Infrastructure
99% Infrastructure
Offers highest value for vertically scaled applications
Offers highest value for horizontally scaled applications

Figure 5 - Enterprise Application Model vs. Cloud Application Model

The Cloud Model, which is becoming more prevalent, especially in the service provider (SP) space, shifts the burden from the infrastructure to the application (depicted above). The application is designed to account for the fact that a fault may occur in the infrastructure and it must handle these additional failure-case scenarios itself. The virtualization stack in the cloud-model is responsible for making sure that the elastic infrastructure demands of the application, especially those used for self-healing, are handled transparently, and that seemingly immutable infrastructure is supplied to the application on-demand.

Significant application re-design is required to move an application from the Enterprise Model to the Cloud Model, and the expense, which may or may not be capitalized as NRE, may not be warranted for many applications. For those situations, continued use of high-availability virtualization stacks becomes appropriate.

On the other hand, the nature of the Enterprise Model may limit the availability potential of the application to roughly 99.9%, which meets the requirement of many IT applications, but doesn't meet the requirement of many SP applications, which require either three-nines-five (99.95%) or five-nines (99.999%) availability.

In keeping with the 'Business First, Resources Second' optimization principle, we believe it is important to support a reasonable number of environments such as the ones above so that the various constituents gain repeatable, optimized environments that suit their specific needs.

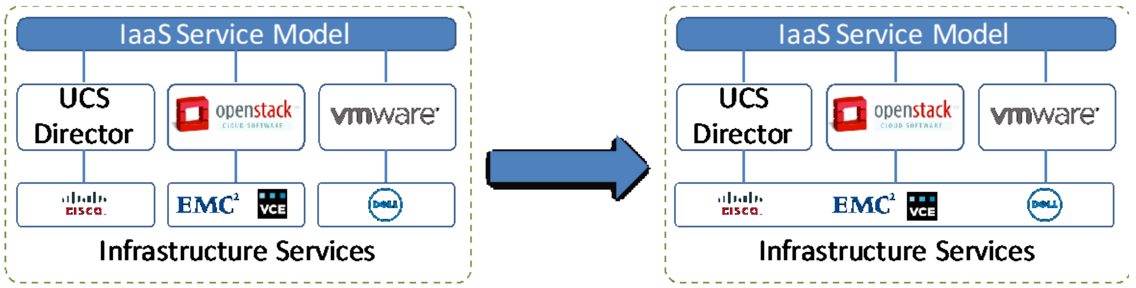

Figure 6 - Desired Relationship of Stacks to Resource Pools

Many virtualization stacks support multi-tenancy, and this is something that should be employed with great caution, especially with service provider applications that leverage and depend on the SDN capabilities integral to these virtualization platforms. We must avoid putting operations in the position where one tenant application requires a specific version of a virtualization stack that conflicts with another tenant application's requirements due to known compatibility issues. Additionally, secure isolation enforced at the network level may be more realistic than performing (ore relying on third-parties to perform) security audits on the virtualization stack.

In this approach, virtualization stacks should be treated as resources, just like compute and storage, that are consumed based on policy and the actual requirements of the service being placed. OpenStack, VMWare, containers, bare metal and future stacks should all be enabled to the degree they are required by the services. Resource optimization will still occur, just at the layer underneath stacks, through the use of common processing and link resources to the degree permitted by the services.

From an equipment standpoint, the goals are to have the smallest possible footprint required to meet the demands, and to maximize utilization and reusability while minimizing 'flavors' of equipment, and, therefore, to minimize CapEx. A related goal is to ensure services are portable across different infrastructure platforms and providers so the SP is free to take advantage of lower cost options as they are available.

4. Automation

Automation, Workflow and Orchestration are utilities that are manually operated or reactively triggered to achieve a specific result. They create leverage for human operators and significantly reduce the manual steps required in a given process. Today there are common tools for deployment, also called fulfillment, of cloud services.

While their names might lead one to believe otherwise, these utilities do not cover the full operational lifecycle of a service. Today, their actions are limited to deployment, single-app scaling and application-based fault tolerance at best. This is not to say they are not useful - only that they aren't as broad as they may seem.

A comparison of scripts, automation, workflow, orchestration and sovereign systems is shown below in Figure 7. The top line in the diagram shows these utilities compared to one another in terms of suitability for single vs. multiple tasks, synchronous (potentially single-threaded) vs. asynchronous (potentially multi-threaded) operations, and their abilities to incorporate rules and policies. The lines below address the levels of information with which they can interact. Finally, the bottom line shows the level of intelligence they can encapsulate.

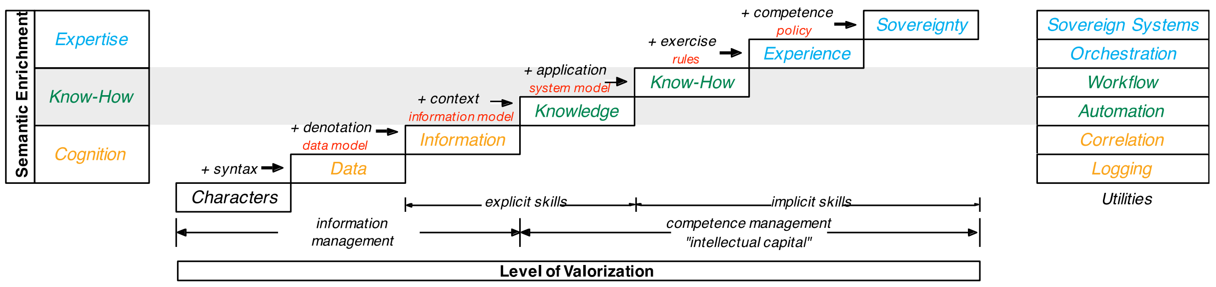

Figure 7 - Comparison of Scripts, Automation, Workflow, Orchestration, and Sovereign Systems

The hierarchy of automation that begins with scripting. Scripting is a series of commands run in a specific sequence, and is kicked off by calling that script. Automation builds upon scripting by explicitly removing the manual need to invoke the script. Automation can be invoked based on some scheduling system or trigger. Scripting and Automation are typically focused on accomplishing a specific task (or closely related set of tasks) in a single- threaded or synchronous manner.

Automation is a utility that exists at many levels in the software and hardware of a cloud. It provides rule-based actions and reactions for a single task. Automation is very useful, but it is generally limited to a set of known actions. Automation can perform poorly in the face of previously unseen conditions (exceptions). Automation systems cannot be used effectively for integration because they only focus on a single task at a time.

Workflow builds upon scripting and automation by executing a series of loosely related tasks, retaining the single-threaded/synchronous nature. It is a utility that provides rule-based actions across multiple tasks. Workflows are commonly used to execute processes. They are single threaded per task. Like automation, workflow systems can perform poorly in the face of exceptions.

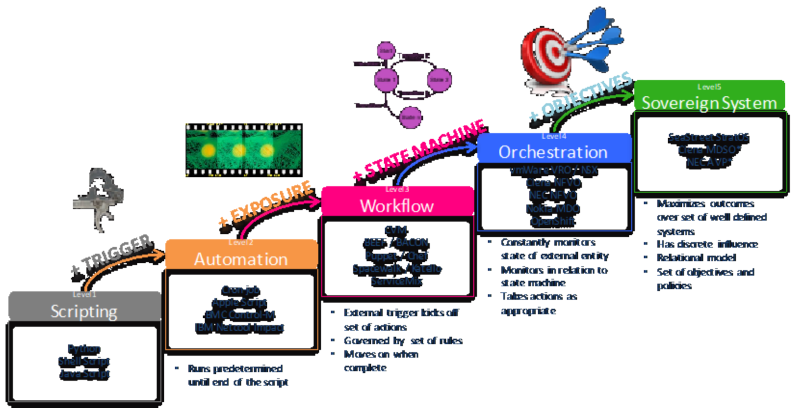

Figure 8 - Evolution between Scripts, Automation, Workflow, Orchestration, and Sovereign Systems

Orchestration deals with the relationship between multiple workflows in a multi-threaded asynchronous manner, and can deal with invocation at multiple points of a process based on rules.

Orchestration systems are commonly used for deployment of applications across cloud resources. They might spawn a thread to acquire compute resources, and deploy and configure software on those resources, while another thread acquired storage and configures it, all for the same application. While orchestration systems typically terminate an activity when fulfillment is complete, there are some that can be called repeatedly to affect scaling. Beyond deployment, initial configuration, scale, and decommissioning, orchestration systems do not effectively address other aspects of the operational lifecycle of a service.

4.1. Resource Management

Model-Based Systems (MBS) are being deployed to close the gaps in automating fragmented manual operations. MBS are replacing Run Book Automation (RBA) systems, which traditionally are fragmented and costly because they rely on manual scripts and templates that represent legacy software. Furthermore, RBA is constrained to only those operations for provisioning, which are often specific to one infrastructure domain (i.e., Compute or Network or Storage and their related configurations). The one-to-one relationship of RBA binds automation to a vendor's technology implementation. Not only is the SP wedded to that vendor, RBA also limits automation to simply those provisioning transactions for a specific infrastructure technology. RBA requires the vendor's customer to maintain multiple teams of operating experts to administer and manage its various systems. These teams are repeated in each technology domain. MBS solve the RBA problem through advanced abstraction by de-coupling and converging automated operations. Through automation, MBS significantly reduces OpEx and improves resiliency, control and availability of those services consuming the underlying infrastructure.

It is typical in the NFV and IaaS/PaaS spaces for solutions to provide an end-to-end ecosystem that can manage almost any task required. The question becomes, even if they can, should they? What is the appropriate set of criteria to make those decisions? One could make the argument that each link-domain and each compute-domain should be managed by an independent domain-specific controller (orchestrator) and that coordination between these domains should occur using a higher-level (sovereign) system. This does not mean that these domain specific systems don't communicate with each other, but there would be no static configuration between them and no clear master among them.

As resources increasingly move to orchestrated virtualization, operators will see the number of resource-oriented orchestration systems increase in their network. For most domains, there are orchestrators that handle that specific domain; but, there is also a need for a higher-level orchestrator which is orchestrating the end-to-end as a whole. Similar to the situation for early automation and workflow-based control systems, directly integrating third party components tends to increase capital burden over time (left-side of diagram below). That approach becomes an integration nightmare and locks in domain orchestrators. By using a higher-level cross-domain orchestrator, policies can be defined generically and then applied down into the domain orchestrators. This preferred approach enables policies to span domain orchestrators, allowing for quicker introduction of new domain orchestrators over time with reduced integration overhead and minimal redesign.

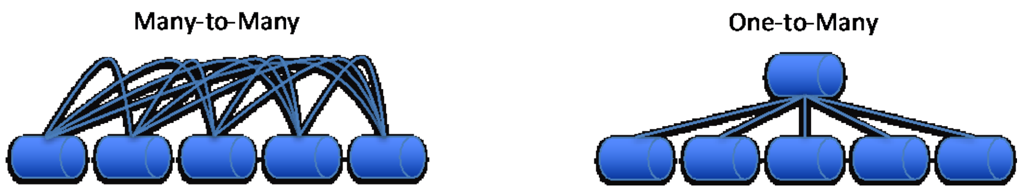

Figure 9 - Integration Platform

The desirable approach is to have an integration platform intermediate between resource orchestrators (right-side of diagram above). It is necessary to avoid use of a simple automation or workflow-based integration platform for this purpose; use, at a minimum, an orchestration-capable platform to act as this integration layer.

As offers become more complicated, a cross-domain orchestrator is needed. Offers like in-home services require a handful of additional service provider services in order for them to work properly; examples include physical network, overlay network, firewall, CoP or CPE device, etc. The ability to coordinate the service across domain orchestrators is critical to understanding the overall SLA for the end customer offer.

Service deployments are becoming more complex and need to be deployed faster in order to keep up with demand. The ability to abstract common functionality into reusable and highly-reliable service models enables deployment to be faster and creates a better service experience. These models should be abstracted from the underlying stacks to make them portable across infrastructure and component services.

The ability to dynamically adjust to real-time telemetry to ensure the SLA and service needs being met at all times is going to become very critical. The number of services that will need to be supported by service provides is increasing quickly and the budget for operations is only going to shrink. Autonomous Operations enables the ability to automated known tasks, scale up repetitive tasks and execute them at computer speed with customer accuracy. The days of having people manually monitor and operate services is going to come to an end given the size and complexity of what is coming.

4.2. Sovereign Systems

Sovereign systems are top-level controllers that operate multiple services together in context with each other and across varied sets of infrastructure. Such systems are almost always model-based and use models to analyze, predict and manage the operation of services on top of ‘real managed things.’

Sovereign systems combine four key elements: (1) stateful awareness, (2) converged declarative telemetry, (3) late-binding policy, and (4) abstraction. A runtime, state-aware process that understands the needs of all of the services, capabilities and capacities of all the resources, and the requirements of the policies simultaneously. Converged and declarative telemetry related to each service. (Note: declarative telemetry means the service declares the telemetry that is important and relative to it and consumes that data specifically. Contrast this with the more typical raw event data that has missing context and requires correlation.) Policy control is applied continuously to manage the automated processes across the service or resource lifecycle. An abstraction layer separates the traditional infrastructure systems and controllers from models that define the service’s operational lifecycle.

Sovereign systems sit above traditional infrastructure systems and controllers (e.g.: scripts, automation, workflow and orchestration) and below traditional OSS/BSS systems and portals. They do not replace these systems. Sovereign systems typically receive an order from an OSS/BSS system or portal for a service and then work through an API fabric to operate infrastructure systems and controllers, to gather telemetry, and to exert control on services and resources to fulfill and lifecycle operate the ordered service, enabling these systems to operate autonomously.

Key differentiators of Sovereign Systems include:

1. They operate continuously, covering the full service-lifecycle.
2. They run many services simultaneously, in context with each other.
3. They operate under policy control and make policies real in the services and infrastructure
4. They are stateful and understand the meaningful condition of each service
5. They understand resources and capacity and know how much capacity remains
6. They contain a real-time ‘as-built’ graph of the cloud components they are responsible for and a real-time dependency map.
7. They understand service level health, and can remediate, migrate, scale, and move services
8. They can handle complex multi-stage activities like upgrade
9. Sovereign systems can be programmed to learn from experience, and services operating under a sovereign system can learn from each other.

Sovereign systems make use of automation and orchestration utilities that exist southbound where appropriate. For instance, it is common for a Sovereign system to use automation and orchestration utilities presented by stacks like VMWare or OpenStack.

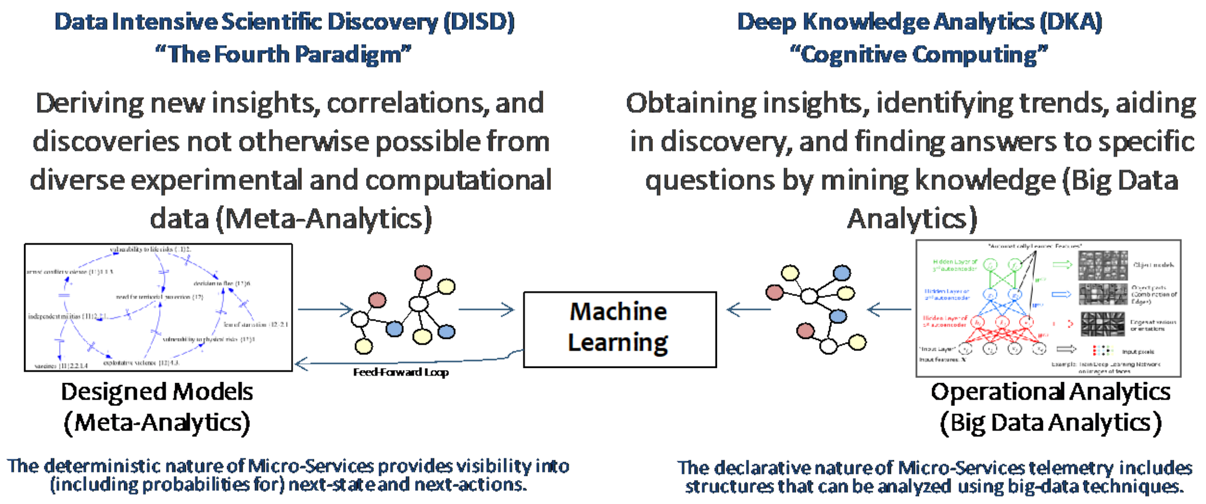
Data Intensive Scientific Discovery (DISD)
"The Fourth Paradigm"
Deriving new insights, correlations, and discoveries not otherwise possible from diverse experimental and computational data (Meta-Analytics)
Deep Knowledge Analytics (DKA)
"Cognitive Computing"
Obtaining insights, identifying trends, aiding in discovery, and finding answers to specific questions by mining knowledge (Big Data Analytics)
Machine Learning
Feed-Forward Loop
Designed Models
(Meta-Analytics)
Operational Analytics
(Big Data Analytics)
The deterministic nature of Micro-Services provides visibility into (including probabilities for) next-state and next-actions.
The declarative nature of Micro-Services telemetry includes structures that can be analyzed using big-data techniques.

Figure 10 - Dimensions of Machine Learning

Through modeling, sovereign systems allow methods and procedures (M&Ps) to be written in software rather than on paper, and they allow these M&Ps to be executed with computer speed and reliability.

Where historic cloud approaches have previously attempted to address deployment and configuration only, a sovereign system addresses the full lifecycle of a service: order, deploy, assure health, assure compliance, remediate faults, failover if required, upgrade, move, port, and destroy/delete. Because they address they the full lifecycle, Sovereign systems have the potential not only to mitigate the additional OpEx that comes with virtualization and cloud, but also to change fundamentally the cost of business by eliminating manual work.

Sovereign systems also need to address very specific concerns related to the potential to accelerate and amplify the effect of failures. Domain orchestrators may not act as expected, either due to malfunction or issues in the how they were modeled. The sovereign system must be able to predict the degree of impact that is expected, monitor the implementation, detect when such an anomaly occurs, and accommodate the anomaly as it can. It should employ circuit-breaker logic with corresponding alarming and notification, rather than enforcing changes that could spiral out of control, or that might result in excessive reconfiguration and possibly reconfiguration loops.

A point of clarification is that Sovereign Systems should not be confused with the concept of data sovereignty. Data Sovereignty is a policy that states where data can be stored or routed or used. For instance, the Australian National Healthcare System has a data sovereignty requirement that the medical information for Australian citizens can only be stored, routed and used within the nation of Australia. You may use a Sovereign System as one way to accomplish this (there are others), but otherwise the concepts are not connected.

5. Proof of Concept

We created 3 different proof-of-concepts (PoCs) with a Sovereign system from Sea Street called StratOS. These PoCs were designed to prove out the concepts described in this paper across different types services, which includes service provider services, customer-facing offerings and implementing/integrating multiple stacks together across domains in the IT operations domain.

For the service provider service we used video encoding. This PoC deployed, configured and managed the operational lifecycle of encoders deployed on bare-metal blade servers. The StratOS system modeled the behaviors of the encoders, video streams, video quality monitoring devices and the infrastructure deployed on. StratOS models each component in the system and aggregates these components into higher-level models that provide a combined higher-level view of the end-to-end server. This means that the StratOS system has a complete and fully stateful awareness of the entire encoder system. This includes the encoder location, configuration, mapping of the health of each video stream both upstream and downstream of the encoder. Once deployed, the system collects telemetry from the infrastructure, encoders and video quality devices and converges this data to determine the health of each individual stream, the infrastructure, encoder application, network, etc. Policies are then applied to the converged data to determine what action, if any, is required to be taken. For example, if the output video quality device reports “no video”, the source quality device would be checked to see if the issue is an encoder issue or upstream from the encoder. Given these data points, StratOS would then take actions, as needed, to remediation to solution such as failover to a standby encoder.

For the customer-facing offering we focused on building a managed router solution with virtualized network functions (VNFs). This PoC created the overlay network for a managed router customer utilizing VNFs (i.e. firewall & content filter) from multiple vendors. The StratOS model for this deployment created each VNF individually and provided the required configuration to connect the service chain together. As with the encoder PoC, the StratOS model contains higher-level models that aggregate lower-level models together enabling StratOS to operate the operational lifecycle at both an end-to-end offer level and component level. This late-binding of network services proved the ability for a Sovereign system to be able to create dynamic offers for customers based on individual needs. In the PoC we created multiple types of SLAs that can be applies to each customer to determine the HA configuration of the service chain, determined placement of services and types of VNFs to be used. Once deployed, StratOS collected telemetry from each of the VNFs, network and the CPE devices to determine the health of the end-to-end offer. When a failure in the offer is detected, StratOS was able to failover the customer to a new service chain within seconds and automate the remediation of failed service chain.

For implementing/integrating multiple stacks together we looked at the IT Operations challenges where private local networks use one type network and the operations backbone uses another. For this PoC StratOS modeled the generic requirements of an IT operations system; a set of VMs and network access to NTP, DNS, the Internet and other private networks. StratOS then determine the correct configuration required within in each environment in order to create the proper routes in the multiple networks showing that a Sovereign system can work across and connect multiple environments.

In each of these cases, the StratOS models are designed to be portable across infrastructure through their abstraction layer. In some cases, we ran the same PoC across different infrastructure stacks and vendors without changing the models or policies defined in StratOS.

During the proof-of-concept work we did, we examined the cost savings for two of services: (1) managed router and (2) video encoding. The result of these analyses showed additive operating savings beyond the benefits of virtualization and standardization of 41% and 31%, respectively. These projected savings were gained through the efficiencies of automated failure detection and remediation, fewer human errors and consistent operations engendered from lifecycle automation.

Services that don't scale often and are stable will see closer to a 30% cost savings through automation, while services which are very dynamic, tend to be less stable, or are scaled often will have closer to a 60% cost savings.

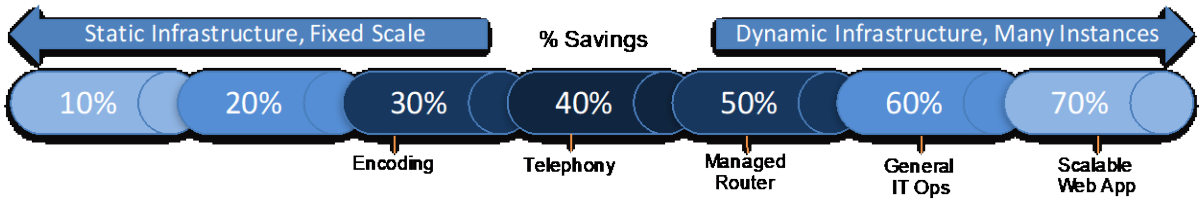

Figure 11 - Factors that Impact Potential for Operational Savings

Automation of the Operational Lifecycle of Services reduces operating costs by:

- Automating manual tasks typically taken on by tier 1 and 2 operations staff, these include typical daily tasks such as monitoring and telemetry convergence, and less frequent tasks such as failure detection, troubleshooting, cause determination and remediation.
- Automating deployment of applications and services, including upgrades and mass configuration changes (e.g., re-IPing a service), through common models that contain instance specific properties.
- Automating security requirements/audits, capacity management, service/datacenter load balancing and other maintenance tasks performed by operations.
- Automating the recovery of a failed offer, application or service by whatever means necessary to get the offer or service up and running again. This includes changes to compute instances, network configuration and more.
- Automating the data flow between the offers, OSS and BSS systems reducing the need for operational staff to enter or translate data to multiple systems.
- Automating the collection of log and other data needed to analyze failure and determine what actions need to be taken to prevent similar failures in the future.

An interesting phenomenon we ran into when evaluating these cost savings relates the relative fallacy of contemplating these as true savings; they may be more accurately described as cost avoidance. Companies typically have a finite pool of resources that they attempt to spend in an optimal fashion, with far more demands than the budget can accommodate. Reducing the cost per action will often allow the company to do more using the same budget, thus a "savings" or "avoidance" of 33% might be more realistically viewed as increasing the operational capacity of the organization by 50% (you can do half again as much as before) and a cost savings of 67% can be viewed as tripling the operational capacity of the organization!

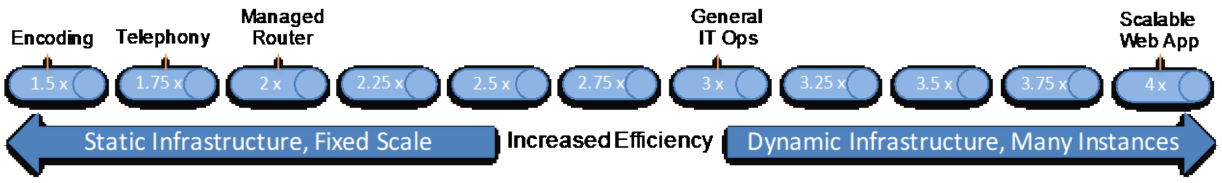

Figure 12 - Factors that Impact Potential for Operational Expense Cost Avoidance

While claims of 33%-75% may be hard to believe. Being able to operate 2x – 10x the amount of equipment (the equivalent of 50%-90% "savings") is easier for people to understand.

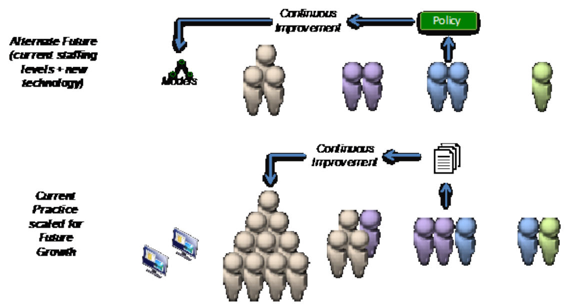
Continuous Improvement
Policy
Alternate Future (current staffing levels + new technology)
Models
Continuous Improvement
Current Practice scaled for Future Growth

Figure 13 - Using Virtualization-Enabled Automation to Empower the Workforce

The reality may lay someone in between, for example, leveraging virtualization-enabled automation may enable a smaller workforce to handle 5x the amount of equipment with half the expense rather than achieving a 90% cost savings.

Conclusion

Although virtualization increases the complexity of solutions, it enables highly flexible automation. Initially, only a few things may leverage this virtualization-enabled automation, however, it will become a pervasive technology paradigm. To prevent virtualization for the sake of virtualization and over-integration of technologies from disparate domains, it is important to have a future-state vision that can be used to guide decisions. Although it is tempting to describe the benefits in terms of cost savings or cost avoidance alone, the theoretical costs would likely have been prohibitive, and an expression of operational efficiencies (do-more-with-less) may be more realistic.

Abbreviations

API	Application Program Interface
BSS	Business Support Systems
CDN	Content Distribution Network
CoP	Compute on Premises
CPE	Customer Premises Equipment
IaaS	Infrastructure as a Service
IDRO	Inter-Domain Resource Orchestrator
IoT	Internet of Things
IT	Information Technology
LAN	Local Area Network
M&P	Method & Procedure
MBS	Model Based System
MBSE	Model Based Systems Engineering
NAS	Network Attached Storage
NFVI	Network Function Virtualization Infrastructure
NFVO	Network Function Virtualization Orchestrator

OSS	Operations Support Systems
OTT	Over-The-Top
PaaS	Platform as a Service
PNF	Physical Network Function
PoC	Proof of Concept
RBA	Run Book Automation
ROADM	Reconfigurable Optical Add-Drop Multiplexer
SLA	Service Level Agreement
SOA	Service Oriented Architecture
SP	Service Provider
uCPE	Universal Customer Premises Equipment
VM	Virtual Machine
VNF	Virtual Network Function
VNFM	Virtual Network Function Manager
WAN	Wide Area Network

Bibliography & References

Chappel, Caroline. (2015) NFV MANO: What's Wrong and How to Fix It. Heavy Reading 13(2).

Chen, Y., Qin, Y., Lambe, M., & Chu, W. (2015, November). Realizing network function virtualization management and orchestration with model based open architecture. In Network and Service Management (CNSM), 2015 11th International Conference on (pp. 410-418). IEEE.

Garcia-Gomez, S., Jimenez-Ganan, M., Taher, Y., Momm, C., Junker, F., Biro, J., ... & Strauch, S. (2012). Challenges for the comprehensive management of Cloud Services in a PaaS framework. Scalable Computing: Practice and Experience, 13(3), 201-214.

Gevorgyan, A., Krob, D., & Spencer, P. (2016, July). Functional Analysis and Design Approach for an Optimal Virtual IP Multimedia Subsystem (IMS) Architecture. In INCOSE International Symposium (Vol. 26, No. 1, pp. 1463-1476).

Ivezic, N., & Srinivasan, V. (2016). On architecting and composing engineering information services to enable smart manufacturing. Journal of computing and information science in engineering, 16(3), 031002.

Ortiz, A. M., Rios, E., Mallouli, W., Iturbe, E., & de Oca, E. M. (2015, September). Self-protecting multi-cloud applications. In Communications and Network Security (CNS), 2015 IEEE Conference on (pp. 643-647). IEEE.