DISSERTATION

CLOUD COMPUTING COST AND ENERGY OPTIMIZATION THROUGH FEDERATED CLOUD SOS

Submitted by

Yahav Biran

College of Engineering

In partial fulfillment of the requirements

For the Degree of Doctor of Philosophy

Colorado State University

Fort Collins, Colorado

Fall 2017

ABSTRACT

CLOUD COMPUTING COST AND ENERGY OPTIMIZATION THROUGH FEDERATED CLOUD SOS

The two most significant differentiators amongst contemporary Cloud Computing service providers have increased green energy use and datacenter resource utilization. This work addresses these two issues from a system's architectural optimization viewpoint. The proposed approach herein, allows multiple cloud providers to utilize their individual computing resources in three ways by: (1) cutting the number of datacenters needed, (2) scheduling available datacenter grid energy via aggregators to reduce costs and power outages, and lastly by (3) utilizing, where appropriate, more renewable and carbon-free energy sources. Altogether our proposed approach creates an alternative paradigm for a Federated Cloud SoS approach. The proposed paradigm employs a novel control methodology that is tuned to obtain both financial and environmental advantages. It also supports dynamic expansion and contraction of computing capabilities for handling sudden variations in service demand as well as for maximizing usage of time varying green energy supplies. Herein we analyze the core SoS requirements, concept synthesis, and functional architecture with an eye on avoiding inadvertent cascading conditions. We suggest a physical architecture that diminishes unwanted outcomes while encouraging desirable results. Finally, in our approach, the constituent cloud services retain their independent ownership, objectives, funding, and sustainability means.

This work analyzes the core SoS requirements, concept synthesis, and functional architecture. It suggests a physical structure that simulates the primary SoS emergent behavior to diminish unwanted outcomes while encouraging desirable results. The report will analyze optimal computing generation methods, optimal energy utilization for computing generation as well as a procedure for building optimal datacenters using a unique hardware computing system design based on the open-Compute community as an illustrative collaboration platform. Finally, the research concludes with security features cloud federation requires to support to protect its constituents, its constituents tenants and itself from security risks.

ACKNOWLEDGEMENTS

I would like to thank the Department of Systems Engineering for supporting this project. I would also like to thank Dr. Joel Dubow for his support and ensuring I followed through with this project to completion. I would like to thank Prof. Collins and Prof. Pasricha, who guided me throughout the research. Finally, I would like to thank the committee members for trusting me and providing a constructive feedback.

DEDICATION

I would like to dedicate this thesis to my wife Karen Biran who supported me in this journy.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

Organizations that serve and process a large number of simultaneous users along with large quantities of data are termed “cloud computing services." These services enable convenient, on-demand network access to a shared pool of configurable computing resources. Cloud computing provides a rapidly growing share of IT resources. IT related spending toward workload processing increased 32.8 percent in 2014, 29 percent in 2015 and 29 percent during 2016[1]. Also, grid energy sources, powered by hydrocarbons, is increasing undesired CO2 emissions, which are growing from a 2011 level of 21.3 MtCO2e and are expected to rise to 39.1 MtCO2e by 2020[2]. This carbon footprint, makes cloud computing one of the largest emerging sources of undesired CO2 emissions.

There is a need to create an efficient and transparent eco-system that allows cloud clients to match its IT expenses with its planned cost structure as well as use more green energy to power the cloud datacenters. The author proposes a new paradigm that will enable multiple cloud providers to utilize both more efficient computing and increased use of green energy resources by forming a proposed cloud federation. From the Service Provider (SP) customer’s perspective, their organization’s IT total-cost-of-ownership is expected to shift from a capital-expense-based organization, e.g., on-premise deployments, to operational-expense-based organization, e.g., cloud service subscriptions.

The author proposes a new paradigm that allows multiple cloud providers to utilize computing resources optimally. This will be possible by: (1) lowering the required number of datacenters deployed per Cloud service provider (CSP); (2) scheduling available energy needs via aggregators, and lastly by (3) employing, where appropriate, more renewable and carbon-free green energies. In our research herein our approach is quantified and standardized, considering the efforts at the datacenters within a single cloud provider as the baseline. The Federated Cloud demonstrates

[1]Gartner Says Worldwide Cloud Infrastructure-as-a-Service Spending to Grow 32.8% in 2015

[2]GeSI SMARTer2020 The Role of ICT in Driving a Sustainable Future. 2015. GeSI

the utility of a software container-based paradigm for achieving dense and elastic computing load management technologies. Federated Cloud is centered on several discreet Linux Software Containers, managed by a Kubernetes resource management system[3] that acts as the governance engine. Moreover, the proposed solution scales and optimizes cross-datacenters' deployments and cross-regional deployments by suggesting a cross-cloud provider's resource sharing collaboration via a cloud aggregator. Furthermore, it suggests that operating datacenters employ maximum green energy sources.

In the proposed cross-cloud coordinator service, the proposed deployment of computing containers sought to reduce the datacenters' projected carbon emissions growth, and quantify improvements in CPU core allocation, network bandwidth, and digital storage. This Cloud Federation proposal is a novel method to aggregate cloud-computing offerings. Cloud Federation will, for the first time, offer a new paradigm, under one umbrella, addressing the core operational objectives of the current public cloud providers and their consumers. This work describes a cloud computing environment that is seen to be both financially attractive and elastic. The simulations validate the concept through detailed cross-regional tests. The ability to aggregate allows the federated cloud to act as one cohesive cloud and provides the required capacity and maximum variable green energy to its users and service aggregators.

Below we outlined future needs and problems, we will deal with and established the groundwork for commercial development and deployment of a proposed embodiment of a federated cloud service as a complex adaptive system(CAS) [22, 77]. In short, we outlined the CAS needs analysis, the concept of operation, the design principles, the essential requirements, and some proposed metrics for operational benchmarking.

The rest of the work is organized as follows. Chapter 2 discusses the relevant terminology used in the paper. Chapters 3-6 assesses the benefits of a Cloud Federation and whether there is a practical approach to satisfy such needs. Chapter 7 investigate emergent behavior by the constituent systems through simulations that were used to optimize of costs and resource utilization. Chapter 8

[3]http://kubernetes.io

furthers the research about the computing paradigm to be adopted as well as the anticipated impact on the computing hardware equipment industry (Chapter 9). We will also extend the investigation of clean energy use for cloud federation workloads (Chapter 10-12). Finally, we will review the cyber-security aspects of the cloud federation and suggests a tool which predicts and detects the anomalous behaviors based on the resource consumption (Chapter 13).

1.1 Terminology

The following section briefly defines the important terms used in this paper.

CAS(Complex Adaptive Systems): This is a generic term of extensively distributed systems. Sage and Cuppan (2001) describe System of Systems or Federation of Systems as a type of CAS. We use CAS as an abstraction of both systems types as we analyze the characteristics of both systems types.

SP (Service-Provider): SP, the customer, usually an organization with end-users who require processing of IT workloads.

CSP (Cloud-Service-Provider): CSP, one of the CAS constituents. offers computing resources, digital storage and network bandwidth to its customers and the Cloud Federation to process its workloads. Also, it provides and the software that provision and manage cloud service.

Public cloud. It offers computing resources, e.g. network broadband, computing, storage, and infrastructure applications over the public Internet. The organization, who chooses to run their workloads on the public cloud is considered as a cloud tenant. Public Cloud providers adhere to service level agreement for service availability and security.

Private cloud: It can include the public cloud offerings, excluding the multi-tenancy property. However, multi-tenancy can be implemented within the enterprise that operates the private cloud. Therefore, the implementation might be customized to adhere to specific enterprise needs.

Hybrid cloud: The hybrid cloud aggregates several public and private clouds to run heterogeneous workloads that might span across different geographical locations and enterprises.

Workloads. These are the organization's IT needs to serve and process both users' IT services and data. Cloud workloads are broadly of two types: online system, and offline system. The former provides low-latency, read/write access to data. For example, a web user requests a web page to load online and serve within a fraction of a second. The latter provides batch-like computing tasks that process the data offline, which is reported later to users by the systems servers; for example, the search results based on a pre-calculated index. Production offline workloads usually comprise mainly unstructured data sets, such as click stream, web graph, and sensors data. The service level objectives (SLO) for online jobs span a fraction of a second, and those for offline job goals hours, days and, sometimes weeks.

Control Plane: This is the software that automatically controls the operations of software-based systems. It is a rule-based system that accepts signals from various systems components and acts, based on a pre-defined policy.

SLA (Service Level Agreement): This is an agreement between the Cloud-Service Provider and its customers, the Service-Providers. It often includes guaranteed levels of availability, network latency, and numerous other provisions.

ODM (Original Device Manufacturer): This is an emerging manufacturing paradigm. In terms of computing equipment, a manufacturer no longer manufactures equipment, exclusively for name-brand vendors. This means that cloud-service providers operating at scale could buy equipment from an ODM supplier and run any software on it. Also, service providers can work directly with multiple merchant silicon chip vendors and have full access to chipset programming.

COTS Equipment. This refers to commercial off-the-shelf (COTS) equipment, software modules, support components, etc. These are the already-built products offered by commercial vendors. Within the context of this paper, a customized solution must be designed if a piece of COTS equipment cannot be used.

JBOD (Just a bunch of disks) refers to multiple hard disk drives linearly aggregated and managed as a single logical volume with no redundant array of independent disks (RAID) functionality. Thus, JBOD does not feature any redundancy, and the failure of a single hard drive causes data corruption. Therefore, JBOD deployments require an additional software solution that partitions the data across independent JBOD instances.

MoE, MoP. Measures of effectiveness and performance are represented by the acronyms MoE and MoP, respectively.

Chapter 2

Needs Analysis for Cloud Federation SoS

The first step in system development process is needs analysis [44]. Chapters 1-7 assesses the benefits of a Cloud Federation and whether there is a practical approach to satisfy such needs. The primary requirements for providing these benefits by a Cloud Federation are twofold: the first accrues to the CSP and the latter to the SP (the customer) who wishes to offload its workloads to the cloud and meet surge and disaster recovery contingencies in a cost-effective manner. The following section explains both these perspectives. Finally, chapters 8-13 investigate critical federated cloud characteristics.

2.0.1 The Cloud-Service-Provider

Accommodating variable demands for computing resources requires an immense capacity, as it calls for providing for the maximum demand within the SLA requirements. For a single CSP that controls the entire computing resources, the only way to provide enough leeway to deal with sudden variations in the demand for computing resources is by providing for significant overcapacity. This is seldom energetically or economically optimal. Moreover, single CSPs tend to monopolize the market, and will not ensure fair market competition, optimal customer satisfaction, or optimized service rates.

In the case of multiple available CSPs, such as Amazon Web Service, Google Compute Engine, etc., there are sufficient numbers of providers to allow for a competitive marketplace. However, from the service provider's perspective, optimal utilization is challenging, as every CSP builds computing capacities based on its load and market share projections. In some cases, this drives them to underutilization of massive datacenter deployments. In other situations, the CSPs suffer over-utilization because of a miss in the market share, load and reliability projections. Both cases lead to sub-optimal utilization. In the first case, the operational cost will be higher for the generated revenue; in the latter, SLA might be violated and may prove detrimental to the customer's (SP) trust in the service.

One of the important goals of Cloud Federation is to balance the processing resource pools and optimize utilization. Also, distributing the workload across several providers creates surge capability and utilization enhancement that allows reasonable rates for the customer SPs who wish to use general computing services.

2.0.2 The Service Provider

While operating in a cloud environment, the SPs are most interested in three aspects: (1)Availability: Reliable service conditions that make its services available to the users it serves. Reliability is defined by SLA and is measured by the allowed unavailability, aka, downtime. Such measurements are done with the help of independent third party service .e.g., Gartner's CloudHarmony[4].

(2)Latency: Some of the SPs workloads are sensitive to network latency, which is defined by the time the service takes to respond to a user or other sub-system request. In the case of a service that runs in a geo-location, which is different from that of the end-user or other sub-systems, the network latency can impact the overall service performance. Therefore, SPs who run latency-sensitive workloads prefer to provide their service by maintaining optimal proximity to its end-users or sub-systems in which SPs' workloads interoperate with.

(3)Adaptability: The Vendor lock-in risk is one of the core business risks that every enterprise, who wishes to offload its workloads to the cloud, faces. Current IT practices rely on common standards and protocols that allow organizations to switch components or elements in their IT operations. However, cloud computing disrupts most of these practices. Onboarding into a single CSP introduces a risk vector that locks the SP to use the CSP's platform, API's and tools. Adopting a single CSP requires an operational adaptation to CSP's methods. New needs on the SP side or changes in the CSP service terms might sub-optimize the operations of the SP. A Federated Cloud

[4]Research and compare cloud providers and services https://cloudharmony.com/status

removes that risk vector by creating a CSP agnostic apparatus that allows the SP to adapt when the vendor lock-in plays a critical role in the migration decision.

2.0.3 Resulting Operational Objectives

The overall objectives of the federated cloud are to provide a financially attractive, multi-tenant, elastic computing environments that support dynamic expansion or contraction of computing capabilities for handling sudden variations in service demands and energy supplies. More specifically, the objective is to provide a service of immense computing capacity that is agile enough to adhere to local regulatory and compliance rules, as well as offer flexible pricing options. Figure 2.1 depicts the first step in defining the operational objectives of this SOS Cloud Federation. The SoS functions denoted by the leaf nodes in Figure 2.1 will be fulfilled through the SoS constituents.

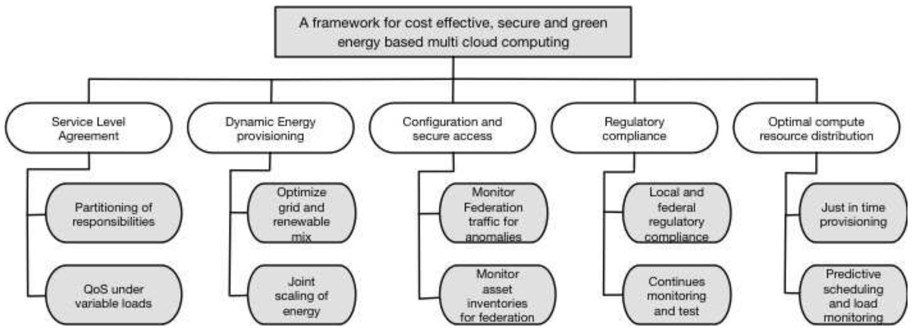

A framework for cost effective, secure and green energy based multi cloud computing
Service Level Agreement
Dynamic Energy provisioning
Configuration and secure access
Regulatory compliance
Optimal compute resource distribution
Partitioning of responsibilities
QoS under variable loads
Optimize grid and renewable mix
Joint scaling of energy
Monitor Federation traffic for anomalies
Monitor asset inventories for federation
Local and federal regulatory compliance
Continues monitoring and test
Just in time provisioning
Predictive scheduling and load monitoring

Figure 2.1: Initial operational objectives analysis. The tree head denotes the overarching objective, a financially attractive and elastic computing environment. The primary branches depict the Cloud Federation SoS primary objectives that distill into functions.

2.1 What kind of System is a Cloud Federation?

The following section analyzes the characteristics of Cloud Federation system. The analysis is based on the five characteristics proposed by [71], as also by [56], whose suggestions are meant for complex adaptive systems, such as Cloud Federation.

Operational Independence of the Individual Systems: A Cloud federation is composed of many CSPs that are independent and useful individually. Amazon Web Service, Google Cloud Platform, and IBM Bluemix are Cloud Federation constituent candidates that provide cloud computing services outside of a Cloud Federation. The Cloud Federation customers, the SPs, are independent business units that can choose to process their workloads with one or more constituent CSPs, as well as through the Cloud Federation. Finally, the Cloud Federation's Clouds-Broker and Clouds-Coordinator are partially independent, regarding of the functionality and services they provide to their users, the CSPs, and SPs.

Managerial Independence of the Systems: A CSP candidate must be operated and managed independently of the other CSP candidates to allow a fair market for the Cloud Federation customers. Clouds-Broker and Clouds-Coordinator also have to be managed separately from the rest of the CSP constituents to avoid the possible conflicts of interest in assigning workloads among the participating CSPs.

Geographic Distribution: Some of SP workloads are attuned to a geographic location. e.g., network latency is the canonical example of the essential properties Cloud Federation will optimize based upon the way individual CSP's currently address it. Cloud federation aggregates various CSPs' services, deployed across different geographic locations so that its workloads are processed with the required proximity to its customers. For example, live video streaming, hosted by an SP, needs to be streamed to its users, maintaining proximity from its targeted users. Such workloads currently require processing by a CSP that operates a datacenter in that geographic location. Also, when attempting to employ green energy, the CSP will have to function in the form of a datacenter with sub-power stations in different geographic locations.

Emergent Behavior: The Cloud Federation attempts to optimize the operation cost of its constituent CSPs by aggregating different CSP datacenter deployments. Such service might impact the price for computing services offered to customers (SPs) by CSPs. Also, a CSP's operating costs might be affected when combining green energy with the cost of energy provided by the grid. The simulation described in Section 6.4 (Cloud Federation Emergent Behavior) will discuss a few of the major emergent behaviors expected of Cloud Federation.

Evolutionary Development: A Cloud federation is never considered fully-formed or complete. CSPs of all sorts of shapes can onboard to the Cloud Federation and process SP workloads. SPs can demand new services through new interfaces offered by Clouds-Broker and Clouds-Coordinator. Such changes, reconciled with the CSP constituents through improved management processes, contribute to Cloud Federation development and benefits.

This particular Cloud Federation is best understood as a System-of-Systems (SoS). This new paradigm, which impacts the constituent cloud services, is based on newly available Service Level Agreements between the federated cloud and the constituent cloud services. However, the constituent cloud services need to retain independent ownership, objectives, funding, and customer relationships and business models. Thus, the primary goal of a Cloud Federation is to provide flexible pricing options, maximize and distribute computing load utilization, and minimize energy use. Since our Federated Cloud paradigm spans different regulatory sovereignties, it must also be tailored to adhere to local regulatory and compliance rules. Federating containerized loads leads to the improved use of computing resources and resiliency of system. In a later section, another type of Federated Cloud architecture for complex adaptive systems, the Federation of Systems (FoS), will be explored and we will show that FoS attributes do not fit Cloud Federation, as contended by [71].

2.2 Concept Exploration of Cloud Federation

The core operational objectives, depicted in Figure 2.1, will serve as a reference for exploring the operational concepts and constraints of the Cloud Federation. First, the projected workload prescribed by the SPs to process using the Cloud Federation will be defined. Next, the computing resources and systems that handle the workloads and the resource management strategy will be synthesized. This is followed by the pricing model, hardware, deployment and security aspects. These allow an SP to deploy the software that integrates with the Cloud Federation control systems. This will be discussed in section 6 on Concept Architecture.

SP Workloads

SP Workload types are twofold: online and offline. The online workloads can be classified into two types: (1) low-latency (10%) and (2) streaming-oriented (90%). The first type of online workload refers to such user's queries as those for a web page. Such a call is expected to return within 100's of milliseconds. The second type relates to video streaming scenarios such as those of Netflix or YouTube. Streaming-oriented calls also need to be returned to the user within the same range as that of low-latency requests of the first content chunks, but further requests can be served with latencies in order of minutes.

The offline workloads are batch-like computing tasks that process the data independently of online connected systems, often with long lead times, and which are later served by other systems, e.g., search results based on a pre-calculated index. Offline production workloads comprise mainly of unstructured data sets, such as click stream, web graph, and sensors data. The service level objectives (SLO) for offline jobs range between hours, and sometimes days. Further, workloads are processed and managed by software tools, which comprise software binaries that are deployed on the CSP and the Clouds-Broker platforms.

The Computing Resources

A cloud computing system comprises two interleaved core elements: hardware, and software. The hardware components include servers, racks, power and cooling system; the software components use the hardware to process the computing workloads. CSP systems comprise one or more datacenters. The datacenter includes clusters of racks, chassis and computing servers that store information in digital format and connect networking equipment. Also, the datacenter includes both peripheral mechanical and electrical elements. A typical datacenter electrical system comprises a power substation, diesel-based generator, main switchboards (MSB), and a backup battery system. The mechanical elements incorporate cooling systems, fire sprinkler systems, and the construction that hosts the datacenter elements [9].

Computing resources are used to operate the software that allows the service, i.e., the control plane, to process the SP workloads, i.e., the data plane. Each and every resource is measured using standard quantifications, e.g., the bit rate for network bandwidth, CPU time for computing, and bytes for digital storage. Also, on the software level, the computing resources are containerized to provide allocation density, elasticity that yields to optimal computing resource utilization and enhanced security [23].

Pricing

Pricing of the Cloud Federation services offers financially attractive opportunities for SPs who wish to offload some of their workloads into a generic cloud computing environment. Also, Cloud Federation pricing enhances the CSP's traditional business model by leasing computing resources to the Cloud Federation customers and increasing its resource utilization.

One of the core functional requirements of the Federation is handling the various pricing models offered by the SoS to CSPs. The pricing scheme should provide an elastic model that allows the customers to allocate and deallocate resources automatically through a call to the Cloud Federation control plane. The following section presents a brief description of the three canonical pricing plans offered today by CSPs: On demand, Spot-price, and Subscription.

On-demand: This is the most expensive option for the SP customer as it offers no guarantees to the CSP. Thus, the CSP's pricing structure is higher for demands of dynamic nature.

Spot-price: As in the smart grid marketplace, customers can bid for resources per demand. Although spot-price allows for low cost, the computing tasks might include a universal support for pause-and-resume, whenever the spot price exceeds the bid price.

Subscription: This allows the CSP to commit its available capacity with a guaranteed income. Subscription also allows the SP to control and manage its operational expenses. But, subscription deprives the SP subscriber of elasticity by charging the SP for times the leased resources run idle. A subscription-based scheme is an optimal option for customers with predicted usage patterns.

The authors add here another, fourth, pricing scheme, namely the performance based price [39]. The performance-based price defines the service rates by a set of performance-related parameters, e.g., work volume, average load, peak load, or deadline parameters. An SP can place a bid on minimum or maximum capacity for a particular set of performance metrics. Performance-based pricing becomes attractive when the cloud service scale is big enough to allow a sufficient resource pool to enable granularity in the sets of possible performance attributes.

Computing Hardware

As cloud computing becomes the primary method to run information and communication services, IT-related spending on datacenter workload processing must also increase, as has been the case during the last three years. A substantial part of the expenditure is presently dedicated to building datacenter deployments across the globe. The computing hardware equipment forms the core of the datacenters; its specification is currently dictated by name-brand proprietary vendors with commodity products that encompass the vendors data center offerings. The datacenter operators are thus required to adjust their architecture to the offerings of various equipment vendors to best run their services. This approach tends to be over specified and otherwise suboptimal, because most data center needs might require only general purpose computing features. As a consequence of this some of the system components remain entirely unused or underutilized.

[9] propose a model that mimics the open source software paradigm and provides a metric for scientific measurement of the hardware computing system design. For this, it uses the Open Compute community framework as an illustrative collaboration platform. The Open Compute community allows for continuous improvement in the equipment specification process, based on both customer's (SP) and operator's (datacenter) needs, along with the evolving vendor's constraints. The community eliminates the dependency on proprietary design and allows the design to be modified organically, not only by the systems' vendors but also the systems' operators, as well as cloud providers.

Continuous Deployment and Integration

One of the Cloud Federation's goals is to enable the SPs with IT agility in their application and workloads to run efficiently within the Cloud Federation. IT-agility is also required in the software that runs the Federation control plane, as it helps the SPs in innovating at a faster pace by building a cloud-native software in a more fault-tolerant and safe deployment environment.

Deployment and integration capabilities require visibility in service status. Also, they need support in detecting and isolating faults to limit the risk associated with a failure or security incident, besides enhancing robustness and resilience. Continuous integration refers to the processes comprising service life cycles (such as those described in ITIL 3, IBM's SOMA (Service Oriented Modeling and Architecture) and various other industry standards), commencing from the moment a code is ready for a test and ending when it handles the production workloads. Continuous integration includes automated deployment, rollback, test, staging, and deployment of production services for both SP, CSP, and the Federation control plane. Moreover, the full software development lifecycle will have to be implemented and managed for Federated Clouds, consistent with the standards and best practices for system lifecycle processes.

Security and Compliance

From the security viewpoint, the SoS Cloud Federation may be considered as a cloud of clouds. In implementing the controls, security operations and monitoring, the CSPs are required to ensure Confidentiality, Availability, and Integrity of each Cloud system and the data contained therein. On inheriting the security measures, the Cloud Federation will include in them supervisory controls, because, while dynamically balancing resources amongst the CSPs, the federation needs to ensure that malware or data loss doesn't occur even when there is the slightest exposure to data storage architecture. Moreover, resource allocations occur dynamically and automatically for a workload that needs to be optimized, and this requires that computing, communication, and data sharing are carried out securely. This aspect has been treated in a separate dedicated paper.

2.3 Concept Architecture phase in Cloud Federation

The following section explores the characteristics of a Cloud Federation SoS and its constituents, i.e., their autonomy, interfaces, interactions, and control methods. The concept synthesis section defines the functional architecture, and finally, a linear model is proposed, which allows initial analysis of dynamic behavior of the Cloud Federation as a SoS. The model will simulate the important aspects of the CSPs, their elements, and their interaction with the Federated Cloud SoS control systems. The Cloud Federation architecture comprises multiple CSPs, Clouds-Coordinator, and Clouds-Broker systems.

Clouds-Coordinator: The coordinator acts as an information registry that stores the CSPs pricing offers and demand patterns. Clouds-Coordinator regularly updates the CSPs availability and offering prices. Also, the Clouds-Coordinator will help in employing, where appropriate, more renewable and carbon-free energies [13].

Clouds-Broker: Clouds-Broker will manage the membership of the constituent CSPs. Both CSPs and SPs will use the Clouds-Broker to onboard to the Cloud Federation. Also, the Clouds-Broker will act on behalf of the SP for resource allocation and provisioning requests. Clouds-Broker also ensure continuous deployment of SP's software, configuration, and data to one or more CSPs' assets, thus helping the SoS to achieve its IT agility goal.

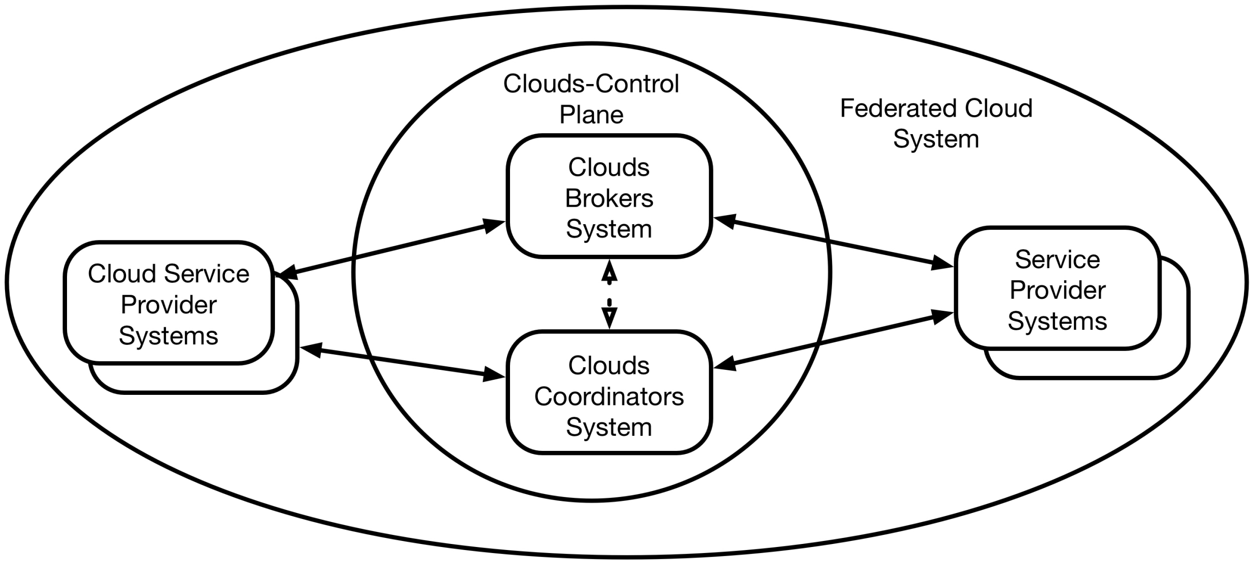


Figure 2.2: Proposed Cloud Federation Systems of System comprises CSPs and SP that are managed by Clouds-Broker and Clouds-Coordinator

2.3.1 Interfaces and Interactions

SPs, CSPs, Clouds-Broker and the Clouds-Coordinator will expose interfaces that allow necessary and more optimal interactions as a system than as a set of individual component systems. This section describes the interface approaches chosen by the authors for Federated SoS constituents' interactions. The communication approach is biased towards loosely coupled messages. However, for, “on-behalf" interactions, agent-based methods are suggested.

By sending loosely coupled message communication, the sender does not expect an immediate response. As the tasks proceed asynchronously, a message queue might build up amongst the interacting components. Therefore, the interface between the sender and the message queue should include a positive or negative acknowledgment indicator through which the sender can convey the logical expectation of an action to happen from potential recipients; for example, SP places a bid for computing resources. The Clouds-Broker process other requests while acknowledging the SP bid request.

As a principle, a mere signal is not enough of an indication that the message has been processed by the destination service. It merely indicates that it has safely arrived at the message queue component and that the sender would like to notify the entire front-end service about the semantics of a record that has been stored in their cache. Moreover, the recipient side will have to support an additionally acknowledge interface that allows the message queue to clear up the read messages. In the unlikely event of the recipient not consuming the message because of an issue, the interface will enforce a timeout duration associated with a message that can trigger a negative acknowledgment back to the sender. It is up to the message originator to decide how to handle such a negative acknowledgment.

The interaction between the federated cloud customers and the cloud providers will be a brokered-based communication. Brokered communication is probably the most popular method for interfacing between components in distributed Internet-based systems [81]. Thus, the interactions between the Clouds-Coordinator and the Clouds-Broker have negotiated communications so that the SPs and CSPs can cooperatively make decisions for lease or release of computing resources.

This negotiation paradigm raises the issue of whether to employ software agents.Agents can act on behalf of the SP and make a proposal to a server agent. The server agent attempts to satisfy the client's proposal, which might involve communication with other services. Having determined the available options, the server agent then offers the client agent one or more options that closely matches the original proposal of the client agent. The client agent may then choose one of the options, request for more options, or reject the offer. If the server agent can satisfy the client agent's request, then it accepts the request; otherwise, it rejects. The canonical example for such a scenario is an SP seeking a spot or performance price through the Clouds-Broker system.

2.3.2 The SoS Functional Architecture

Much of the interest in Cloud Federation, as a SoS, is focused on the desire to integrate existing CSPs systems for achieving new capabilities that are not available with a single CSP. Specifically, it mitigates vendor lock-in to the enterprise, optimize resources utilization and reduced carbon footprint by a CSP's datacenters. The current environment requires that participating CSPs are operationally and managerially independent, evolutionarily developed with emergent behavior, and are geographically distributed [69].

In our Cloud Federation SoS, the CSPs interact voluntarily to fulfill the agreed upon beneficial collective purposes. The primary enablers, the Clouds-Coordinator and the Clouds-Broker, collectively decide how to provide or deny service, thereby providing means for enforcing and maintaining the required standards and compliance [25]. Also, the Cloud Federation allows autonomy to its CSPs' constituents to decide how to fulfill the purpose of the SoS and its business goals. The following section discusses the SoS management, implementation, engineering, and design considerations [24].

Management and Oversight

The core constituents of the Cloud Federation are the SP and the CSP. Both have their owners, stakeholders, users, and business processes, which lead to overlapping authorities with the other participating CSPs and SPs. Further, the lack of common powers and funding pose challenges to the Cloud Federation management and governance. One of the solutions proposed for similar SoS leadership dilemmas is well documented in SEBoK [4]. The solution supports a community approach that establishes SoS principles. The community seeks consensus when one of the Federaton principles comes into question. This is a proven approach, as most of the Internet-based communities are managed with the same modus operandi [70].

The community-based cooperation between CSPs will help to maintain a reliable service by forming joint control systems such as the Clouds Coordinator that will increase the individual CSP's resource utilization and reduce cost by aggregating demand. One might refer to such cooperation as organic regulation. The SPs are interested not only in reliable service but also in an optimized price for computing resources. Therefore, part of the agreement should include deregulation of prices and alignment of the computing capacity deployment with the SP demand. The authors believe that the price for computing resource, offered by the Cloud Federation, will be one of the important and interesting emergent behaviors of the SoS. They are exploring, through simulation and with real workloads, how the cost of computing resources is affected through deregulation of the SoS.

Engineering Design and Implementation Consideration

As in management processes, technical engineering considerations are implemented at two levels: first at the CSP level and the second at the federation level. In case the two levels do not overlap, the two models co-exists well. However, in the event of a conflict, the CSP will seek to reconcile its engineering or operational requirements with the federation and vice versa; for example, the CSP is required to deploy a datacenter in a region to accommodate its customer's needs, unfederated SP. In such cases, the CSP might deploy the datacenter, but the CSP will not allow placing the datacenter in the federation pool; therefore, the action taken does not drive down the computing prices of the already planned resources pool provided by the Cloud Federation.

For adoption to Cloud Federation, the SP requires standard tools that allow deployment and integration of its software, and data for processing a workload. In most cases, the CSP offers such platform to its customers. However, the Clouds-Coordinator and the Clouds-Broker are required to offering a generic method to the SP for deployment and integration with the Cloud Federation. Therefore, the CSP has to modify its systems' boundaries and interfaces by exposing generic interfaces that do not exist otherwise. For example, the CSP might implement tightly-coupled interactions between its resource managers and computing resources.

Testing, validation, and learning also pose challenges when implemented within the Federated Cloud. Joint, end-to-end testing turns into a complicated technical management process. Introducing systems enhancements across life-cycles of multiple systems requires different interaction methods than those used within the same CSP and SP. It is particularly so when the systems have to support both new and legacy systems. Thus, automated and asynchronous testing and evaluation processes that span across the SoS can alleviate the technical process complexity [83].

Metrics Measurement

The Cloud Federation requires the definition of two core sets of metrics: (1) project metrics and (2) process metrics. The project metrics measure the success of the SP and CSP projects i.e., CSP wishes to introduce a change to some of the Cloud Federation constituents that might impact interfaces with the Clouds-Broker or Clouds-Coordinator as well as the SPs. The Cloud Federation needs to establish a standard for change management through a set of metrics, relating to the stability of requirements, the quality of project planning, adherence to project schedules, the verification, validation, and documentation of proposed changes, and quality of project reviews [44].

Process metrics will serve as the operational measurement standard between the Cloud Federation, and its constituents (CSPs) and customers (SPs). SP workloads are often handled by mission-critical business processes, and hence the ability to track and measure the workload status is crucial to the success of an SP when it is running in the cloud. Defining a cohesive set of metrics that is agreeable to CSPs is challenging, as it might conflict with the semantics of some of the CSP metrics. Also, some of the semantics might require detailed metrics that CSPs wish not to expose outside of its system boundaries. Therefore, in the set of parameters to be defined by the Cloud Federation, the focus will be on the quantitative measures that will be used to assess, uncover problems and provide a basis for improving the SP workload, hosted in the Cloud Federation. Figure 2.3 describes the possible set of states of SP workload. The authors propose that the Cloud Federation, through its Clouds-Coordinator, will expose a set of metrics for SPs that are based on the SP workload status. SPs can define a set of measurements that include the SP workload semantics.

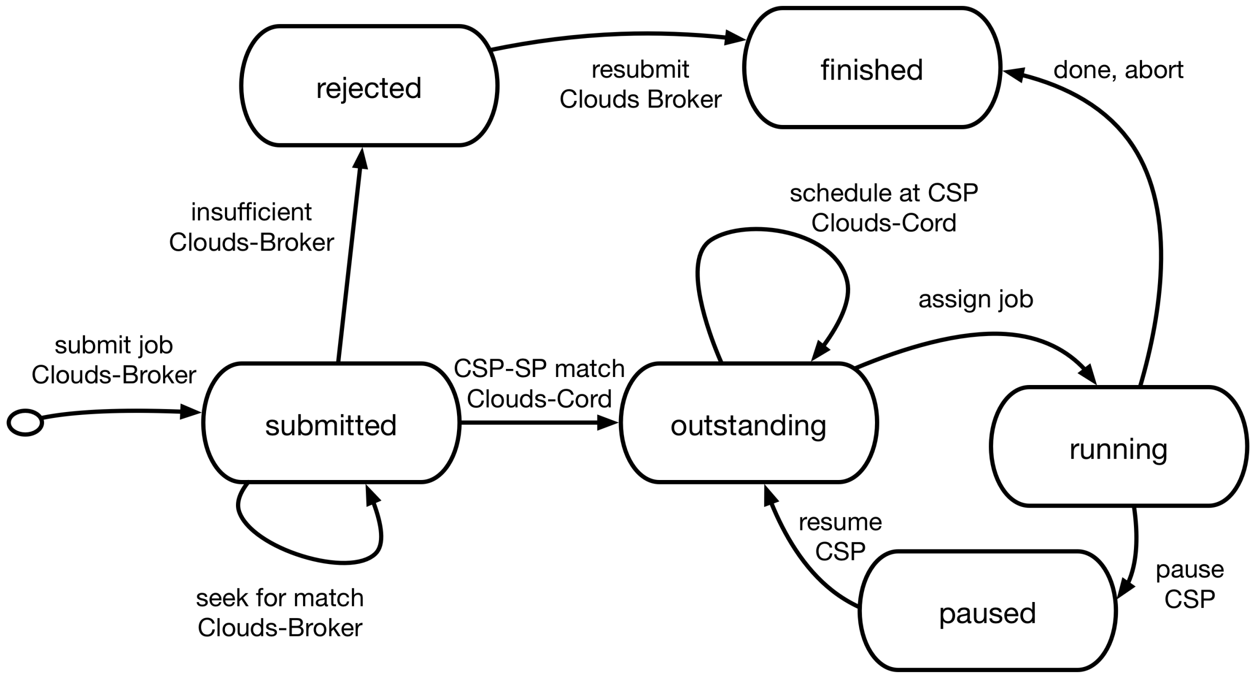


Figure 2.3: Proposed Cloud Federation Systems of System comprises CSPs and SP that are managed by Clouds-Broker and Clouds-Coordinator

2.3.3 Cloud Federation - FoS or SoS?

[71] suggest two types of relevant Federated System complex adaptive systems: System of Systems (SoS) or Federation of Systems (FoS). FoS can be considered as an SoS with a very limited amount of centralized control and authority. In other cases, FoS creates a coalition of its constituent systems, which together form a decentralized power and authority with potentially different new perspective behaviors. Some of the essential requirements from the Cloud Federation side are centralized command and control functionalities, such as cross CSP cyber-security services, and objective computing metering that is decoupled from the CSP metering systems. Also, the Cloud Federation will have to ensure fair market to its customers and the SP by ensuring that there will be no price-fixing across CSPs.

According to [71], SoS property and subsidiarity are crucial to FoS. Subsidiarity claims that power belongs to the lowest possible point among the SoS constituents. In Cloud Federation, neither the CSP nor the SP possesses any systemic power, except for the choice of participating or not participating in the federation. Moreover, the Clouds-Broker and Clouds-Coordinator have the power to decide about SP's workload assignment to CSP, which can dictate computing price, without consulting CSP or SP.

On the other hand, obtaining and maintaining a systems engineering ecology is vital to achieving a sustainable system, which is manifested by forming virtual organizations that foster cross system collaboration [36], an example of which is the Open Compute Project[5]. Also, “separation of power" is an FoS property where the commands and control systems do not reside within the boundaries of the same system. In Cloud Federation, the Clouds-Broker oversees SP workload matches, and Clouds-Coordinator the workload assignments. Both of them impact two of the important emergent behaviors, namely Computing price for SP and resource utilization for CSP. [71] and [36] suggest no set of CAS properties that defines whether Cloud Federation is an FoS or an SoS. However, based on the set of FoS properties that Cloud Federation lacks, [24] considers Cloud Federations need to be a collaborative SoS.

2.3.4 Cloud Federation Emergent Behavior

The following paragraph describes few of the emergent behaviors anticipated by the Cloud Federation.

Computing Price

A core objective of Cloud Federation is to provide for its customer's fair pricing for general computing service. Failing to offer a more cost-effective computing service than that of a CSP directly might invalidate the need for the Cloud Federation service to its customers. The Section on Concept Exploration proposes several price schemes offered by the Cloud Federation. As a deregulated system, computing prices will be determined by CSP's supply and SP's demand. The

[5]http://www.opencompute.org

Federation Clouds-broker is the system that determines the price offered to the SPs. The initial impact of the Cloud Federation on the computing price will be examined in the simulation section.

Computing Resource Utilization

Another core objective of Cloud Federation is optimizing CSP resource utilization. Most CSPs seek more market share in competition with other CSPs. An outcome of such competition is the ever-growing infrastructure in the form of new datacenters across the globe, with no countervailing forces to meet user demand more efficiently and to satisfy societal, environmental and energy requirements. This sub-optimum use of infrastructure increases the carbon footprint, attributable to cloud computing services and also drives up the costs to CSPs. The simulation section explains the possible utilization of computing resource by the Cloud Federation in processing SP workloads.

Carbon Footprint by Computing Resources

Meeting the Federally mandated approach of maximizing the use of green energy in operating datacenters (for a government with recommendations for private sector use as well) requires an energy source-demand coupling scheme that ensures SLO levels of power availability, but with structural bias towards green energy sources over hydrocarbon fueled energy sources. The system that can accomplish this will have to provide seamless failover, in case of sudden interruption in green energy, to grid energy sources or vice versa.

SP workloads require different service level requirements. Serving systems' workloads comprise interactive sessions that pivot on minimum latency. However, low latency is less critical in analytical on-demand streaming, because application clients use buffering techniques to mitigate long latency effects. Therefore, on-demand streaming workloads fit, more tightly than the interactive workloads do, with the observed intermittent and varying green energy availability characteristics.

Green energy supply is unpredictable and requires a complex, adaptable, resource allocation system to provide SP services with steady energy supplies, ensuring concurrently minimal carbon footprint. Such dynamic green power resources can be available in a smart grid only with real-time communications of both short term and predictive energy needs from cloud service providers to green energy providers. The green energy providers will have to disclose the availability of energy dynamically to CSPs, who, in turn, publish their changing energy demands for near-term computing. The SPs can then better maximize the use of green energy for on-demand streaming processes.

[13] and [14] simulate an opportunistic power-mix approach for processing analytical workloads. Their approach demonstrates that potential carbon-footprint savings can be achieved with opportunistic methods. The experiment simulated the usage of green 708 *kWh* out of the total 3,252 *kWh* required for *analytical* systems workload processing i.e. 22% less carbon emission. Fifty-percent of the total workload consumption, i.e., 1822 *kWh*, was processed by brown energy because of false-positive events, i.e., the coordinator assigned a job with no sufficient green recourses to process the job. The authors believe that optimizing the coordinator algorithm can improve the footprint reduction up to 50% for offline workloads and 30% for general workloads[6].

Figure 5.9 shows the job-placement ranges, denoted by the dotted line in both the PV and the Wind plots. Any value above the zero level is indicative of potential benefit. However, such cases are subject to false positive events that can occur because of an unpredicted drop in availability.

Computing Hardware Equipment Price

The cloud computing hardware elements include servers, racks, power and cooling systems. The software elements use this equipment to process the computing workloads. The dominant software paradigm is open, and the source code is freely available and modified according to an open source community. The computing hardware system has neither an open standard, nor has it evolved best open practices for designing a warehouse scale computing system. It is presently dictated by a few name-brand proprietary vendors, such as IBM, Hewlett-Packard, and Dell.

These vendors often offer a tailored business solution in which the equipment design interaction methodologies are derived, based on that single path perspective. Cloud federation is projected to

[6]Cloud Computing, Server Utilization, & the Environment, https://aws.amazon.com/blogs/aws/cloud-computing-server-utilization-the-environment/

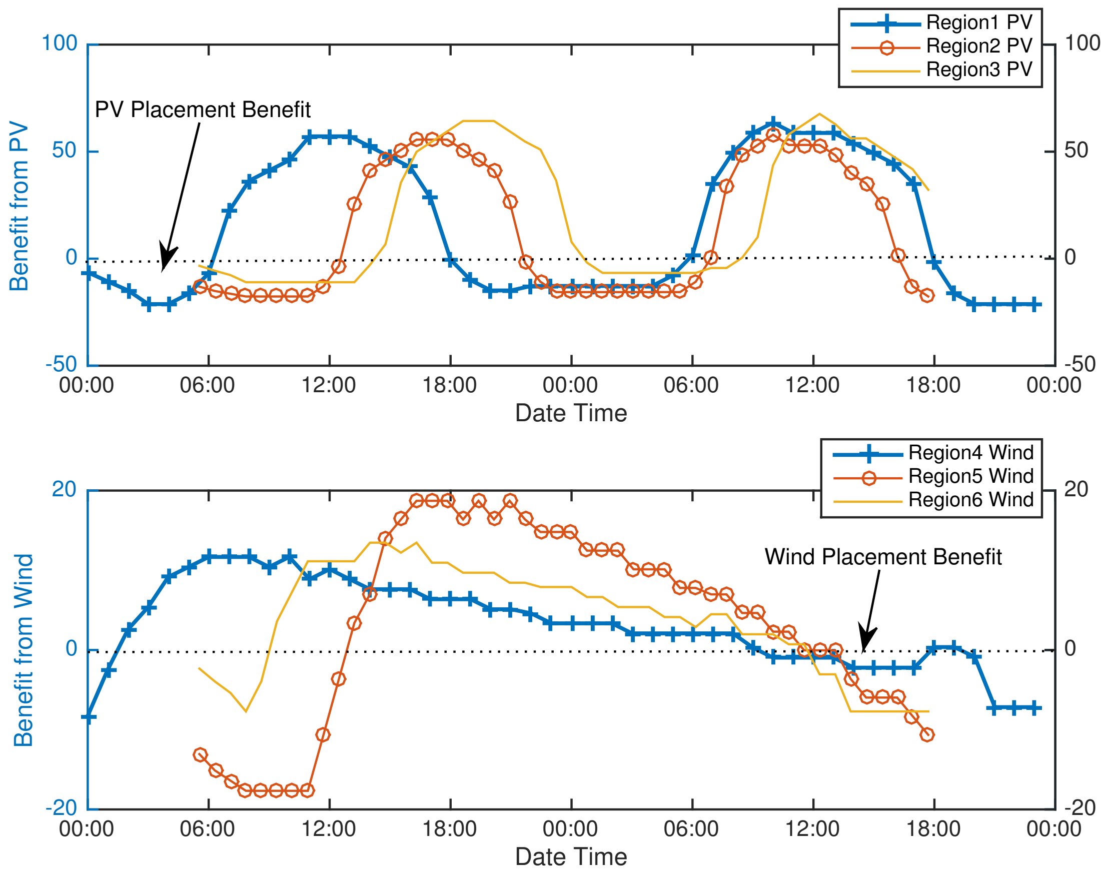


Figure 2.4: Job placement ranges denoted by the doted line in both the PV and the Wind plots. Any values above the zero levels indicate on potential benefits. However, assigned jobs might not be able to fully processed when unpredicted drop in the availability.

change the space of hardware by disaggregation, and the evolvements of the ODM market opened up the equipment hardware design to the collaboration of more contributors. Every contributor brings vital data to the design process, and the systems engineers will help to compare various alternatives by providing a framework that converts individual data to a standard measure.

2.4 Modeling Approach and Simulation of Primary Scenarios

The authors propose here a resource allocation model that is based on best industry practices [37, 38]. In general, SP wishes to run a collection of tasks that define the *job*. The tasks include resource allocation, data partitioning, fetching data from a source location, its processing, and aggregating the results. The author envisions a semantics between SPs and Clouds-Broker that allows the job to be submitted and controlled by the Clouds-Coordinator at the CSP's resources. A job includes five main states: submitted, outstanding, running, paused, and finished. Figure 2.3 depicts the job state transition. Job submission through outstanding is controlled by the Clouds-Broker, and pending through finished states are controlled by the Clouds-Coordinator. Practical examples may, however, cover more states and sub-states. For simplicity, the authors prefer to use the five states that articulate the interactions between the SoS constituents.

The authors have used one-month of Google's cluster-usage traces that include mixed cloud workload. They cover 650K jobs running across 12000 machines in a single datacenter [84]. The goal of the traces is to simulate realistic demand patterns generated by SP that can later help in assessing the SoS emergent behavior, such as the price for computing resource, datacenter utilization, and security risks.

2.4.1 SoS Modeling and Simulation

The core value proposition of cloud-federation is twofold: fair computing price for SPs and optimized energy utilization by the CSPs' datacenters. The following section models the interaction between various systems of Cloud Federation that reflects the SoS emergent behavior when attempting to achieve these two objectives.

For this purpose, the author used SoS engineering analysis developed by Osmundson et al. [64]. The methodology includes a sequence of analysis, transformation, model building, and simulations. The scenario development has been done in the Needs Analysis section, and the Concept Exploration section defined the SoS elements and threads. The following paragraph attempts to represent the operational architecture through simulation.

The goal of this section is to understand some aspects of the SoS emergent behavior and their impact on its constituents. First, SoS's impact on the computing price offered to the SPs by the CSPs through the federation is discussed and then the impact on the CSP's datacenter utilization.

The simulation shows that the Cloud Federation might help CSP in improving its use of resources and reduce the overall CSP carbon-footprint.

Simulation - SoS Cost Impact

The objective of simulation is to depict the possible SoS emergent behavior, and its impact on the computing price to SP. Simulink was used as the simulation tool. The Simulink model mimics the general model shown in Figure 2.2. The model includes a set of CSP instances, $CSP_1...CSP_3$, that generate supply messages to Clouds-Broker system, using uniform random objects, and four SP instances, $SP_1..SP_4$ that create computing demands messages to a dedicated Clouds-Broker interface. Three CSPs were chosen to mimic the current market CSP: Amazon Web Services, Google Cloud Platform and IBM Bluemix.

The simulation product was depicted by the Price Scope object. For simplicity, the price generated was for enterprise-grade CPU performance, for example, Intel Xeon E5 processors that are suitable for production workloads, such as moderate-traffic websites. The simulated Clouds-Broker system included static summation and transfer functions that mimic electronic-commerce system, which implements fair marketplace.

The simulation spanned across 80 days, during which period, the CSPs could onboard the Cloud Federation and the SPs could submit workload jobs to the Clouds-Broker that were later coordinated through Clouds-Control system. The simulation duration included one CSP only, processing four SPs workload requests. During $t_7 - t_{17}$, two more CSP were added to the SoS pool. During $t_{18} - t_{69}$, three CSPs processed the intermittent SP workloads generated. In t_{65}, two CSPs stopped accepting new workload requests by the Clouds-Broker, while the SPs reduced the workload requests to be processed by the SoS.

The cloud-federation price impact simulation (see Figure 2.6) shows one of the important values that cloud-federation brings to its constituent SPs. It shows price stabilization phase throughout the period the CSP was reporting on computing resources availability. The emergent behavior of the Cloud Federation can be analyzed by comparing the price offered outside the Cloud Federation, in the absence of SoS trading agents. The current simulation exhibits a limited design that represents deregulated SoS operations, which optimize the cost for SP. Future studies should include regulated SoS, as the one that occurred in the Power Grid SoS [18].

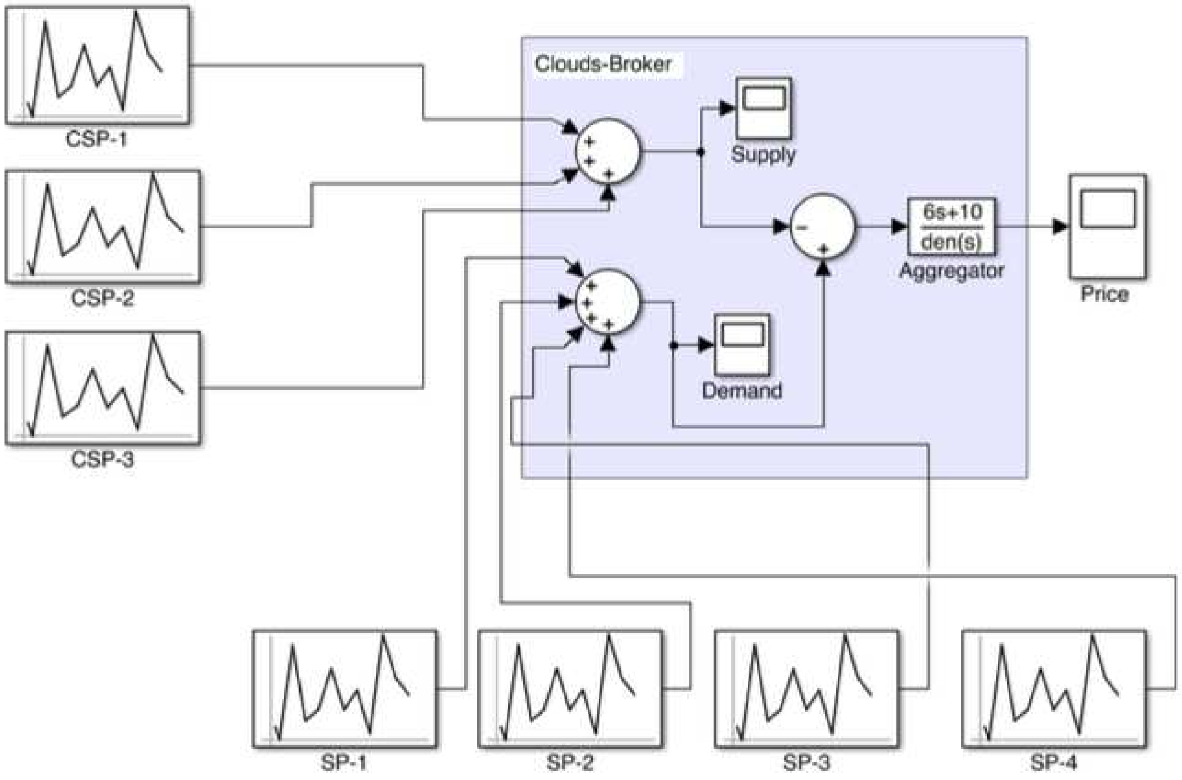


Figure 2.5: Cloud Federation simulation architecture attempt to assess the impact of the federation on computing prices offered for SP and CSP's datacenter utilization. The Simulink simulation includes three CSP instances, $CSP_1...CSP_3$, that generates supply messages to Clouds-Broker system using uniform random objects and four SP instances, $SP_1..SP_4$, that generates computing demands messages to a dedicated Clouds-Broker interface.

Simulation - SoS Computing Utilization Impact

Another cloud-federation emergent behavior handles CSP resource utilization. The simulation assesses the impact on datacenter resources utilization by applying known optimization solvers through the clouds-coordinator. The authors used linear programming solver to find a job-assignment strategy, minimizing thereby the datacenter cost of operations, while obeying a set of datacenter's operational constraints. The job simulation shown in Figure 2.7 under $SP_1..SP_4$

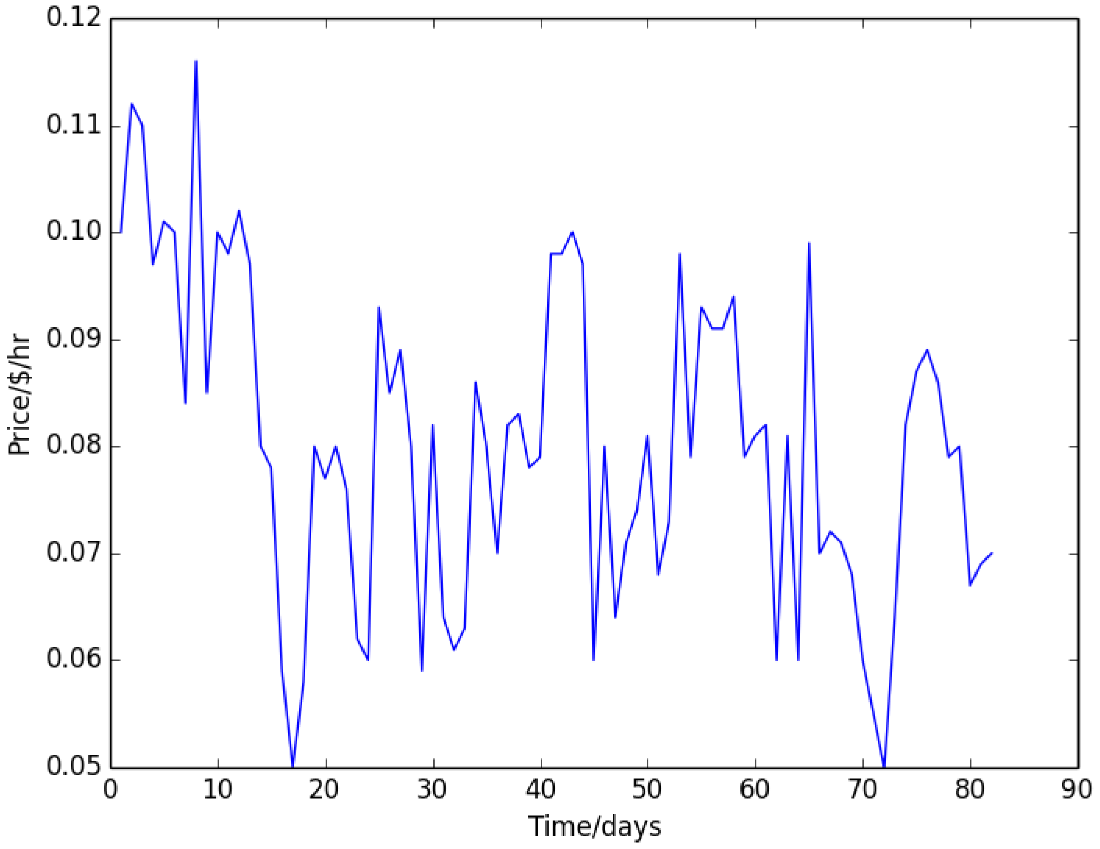


Figure 2.6: Cloud Federation impact on price simulation shows one of the important values that Cloud Federation brings to SPs. It shows the computing price stabilization process when more CSPs reports on computing resources availability.

depicts the job creation rate across 24 hours span, across 30 minutes for each period [84]. The simulation shows CSPs utilization, based on optimal job assignment.

The core objective of optimization function is allocation of virtual-CPU (vCPU) for job execution. The authors simplified the model by optimizing a single resource i.e. CPU. The model can be extended to more practical scenarios that include digital storage requirements and network broadband. Both are proxy to vCPU and hence the simplification. Let $G(g, p)$ denote the cost of vCPU assignment function, where g denotes the cost job assignment of some vCPU. p means the pool of demand for g measured by the cost of $\frac{\#vCPU}{hour}$. Let $C(g, f_c)$ denote the cost function for generating $G(g, p)$, in which f_c denotes the power cost measured in $vCPU/J$. Finally, let I_c denotes the initial cost for running a computing cluster in a CSP facility.

The simulation objective is to maximize

$$f = G(g,p) - C(g,f_c) - I_c$$

where f denotes the CSP optimized utilization.

$G = \sum_{i=1}^{n} g_i \cdot p_i$ and $C = \sum_{i=1}^{n} poweruse_i \cdot f_{ci}$

The utilization upper-bound constraint does not allow CSP utilization rates, $f(CSP_i)$, of more than 80%, consequent to which 20% of the computing resources is left unutilized for unexpected bursts. Another cross-CSP constraint ensures that running optimized $f(CSP_j)$ does not deviates from other $f(CSP_i)$ by more than 10%. The authors applied Matlab `linprog` with the linear inequality constraints specified as the A matrix required by `linprog`. The `linprog` execution yielded the job-assignment shown in Figure 2.7 across 48 periods, each of 30 minutes. The simulation shows that the requests for job originates by $SP_1 ... SP_4$ and the job assignments by the clouds-coordinator of the available $CSP1.CSP_3$. The job assignments in periods 12, 20, 25 were done prior to the actual executions, because of execution latency caused by I_c. Finally, the simulation exhibit the Cloud Federation fairness in the workload balancing across the constituents CSPs. Finally, it optimizes the utilization of each of the CSP resources.

Simulation - SoS Carbon Footprint Impact

From another simulation reported in green cloud coordination paper [13], the authors estimated the impact on the carbon footprint projected to be reduced or, in some cases, eliminated by cloud computing. The utilization of green energy is strongly coupled with the workload type. The authors presented the results of two simulations of two different types of workload. The power-mix approach adopted in both the experiments was an opportunistic match of possible SP workloads and available wind and green solar energy.

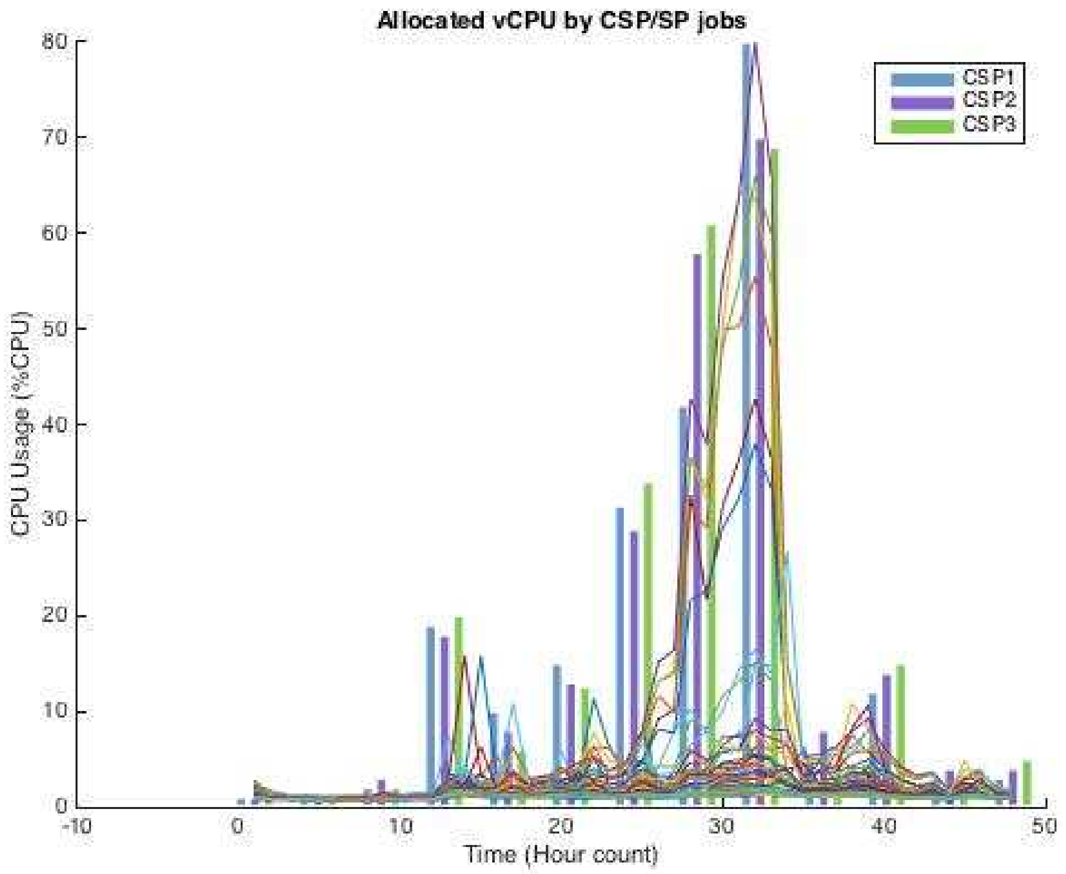


Figure 2.7: Cloud-federation impact on CSP's datacenter utilization simulation. It shows how CSP utilization is distributed homogeneously among $CSP_1...CSP_3$, thus (1) providing fair business platform for CSPs and (2) optimizing resources utilization that reduces redundant carbon footprint originated by under-utilized datacenters

Initial Conclusions and Outstanding Investigations Areas

We investigated emergent behavior by the constituent systems through simulations that were used to optimize of costs and resource utilization (Chapter 8). The following chapters will further research the computing paradigm to adopted as well as the anticipated impact on the computing hardware equipment industry (Chapter 9). We will also extend the investigation of clean energy use for cloud federation workloads (Chapter 10-12). Finally, we will review the cyber-security aspects of the cloud federation and suggests a tool which predicts and detects the anomalous behaviors based on the resource consumption (Chapter 13).

Chapter 3

Cloud Federation Computing Paradigm

Containers is a technology that better enables the service providers who seek simultaneously both scale and elasticity. Each job in the service provider business comes with a service-level agreement (SLA) between the customers and the cloud service provider. SLA expresses the contractual maximum latency for allocating the computer resources to execute a specific job requirement. Immediate response requires short duration SLA agreements, for example, via live video conferencing. In contrast financial-batch calculations possess asynchronous nature, and require SLA's of hours to weeks to detail the assurances and expectations. The overriding goal is to optimally allocate resources to jobs based on a specific customer's needs, and further to reclaim these resources once the job is satisfactorily done. *Density* is the ability to pack maximum number of occupied resources on a single physical compute unit and a crucial factor in operations efficiency as the cost of operating a physical compute unit is known e.g. electricity, cooling, etc. The more tenants running in a single physical unit, the more revenue generated by the baseline compute resources. The ability to migrate running jobs within SLA agreements that allow the shared resource pool to fulfill the density property is key. Shifting running tasks across the resource pool enables enhanced *elasticity* and guarantees safe resource over-provisioning when needed.

The exponential increase in demand and complexity of the present networks, threatens to violate both cost and capacity aspects. Although, the prevalent technology, hypervisor-based virtualization, supports multi-tenancy, its density and elasticity are not optimal and often too slow for the ever-growing demand for the varied needs of compute, storage and network resources. Hypervisor based virtualization offers a full-blown operating system but requires additional compute capacity to accommodate the technology.

3.0.1 Why not Hypervisor-based VM?

In short we judge to fulfill multi-tenancy it is best done with a hypervisor-based architecture with container-based virtualization. In short the combination has better performance as we show herein.

There are two core paradigms to fulfill multi-tenancy in a full blown compute resource that is, hypervisor-based with container-based virtualization. Hypervisor virtualization technology is mature and has an excellent track record for security and compatibility. However, hypervisors have performance characteristics that restrict and rule out their use in some scenarios that requires both density and elasticity. Hypervisor-based virtualization is based on emulating virtual HW. In short, the hypervisor itself emulates system HW that includes: virtual CPU and I/O resources controls and virtual machine monitor (VMM), on top of that emulation a kernel (OS) is being booted so that one can run processes on that OS.

Containers in contrast are not based on that paradigm. The container based paradigm relies on shared OS. i.e. any new containers are most likely to share a similar kernel with other containers. The overall OS resources allocation depends on the way the containers are configured. In hypervisor paradigm, every instance is totally separated from the other virtual machines. That is, different OS instances are not sharing any configuration or code. In containers, every resource, configuration, or code can be shared with other container instances.

Containers-based virtualization still suffers from several operational weaknesses such as security and heterogeneously. Therefore, it is impossible to run both Windows and Linux containers on the same host, as they are not sharing the same kernel while hypervisor-based paradigm allows this duality. Security wise, containers are not as secured as hypervisors because they share the kernel. There were few kernel exploit cases of unauthorized access to /proc/pid/mem [28] were deemed sufficiently risky.

Figure 3.1a depict the classic hypervisor case; the hypervisor kernel emulates virtual HW for the virtual machines that run its appropriate kernel that runs a full operating system up to the application level. Figure 3.1b describes the container added case where the two virtual instances

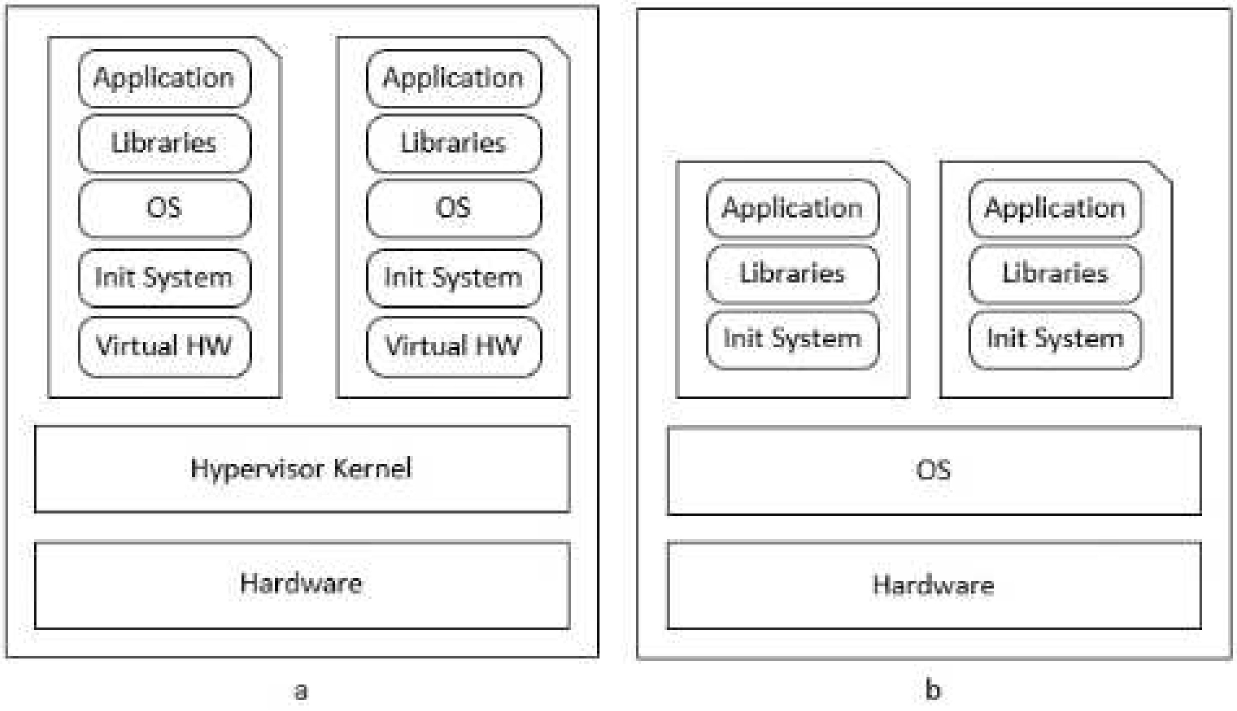


Figure 3.1: Hypervisor and Containers Paradigm - Block Diagram.

run a distinct optimized set of libraries and the combination runs an application. Containers are faster in the initialization phase because it is not required to initialize a virtual HW and it also avoids the interaction with the HW devices. In some cases, containers can share the same init system and runtime libraries, so the application layer is the only isolated layer [58]. With such containers boot time can be as fast as it takes a process to get going within the operation system. Therefore, the start time of a container-based VM can be in the order of milliseconds as oppose to minutes for hypervisor-based VM. The fast initiation time is one of the elasticity property that is vital in utility computing.

The storage footprint analysis of each paradigm shows a significant difference. If we compare the paradigm side by side, the stack of containers is a lot thinner as oppose to the hypervisor-based stack. Below we will show that a typical hypervisor-based stack requires an order of gigabytes of storage while an application container requires only megabytes. Therefore, the optimized storage footprints we propose also support the elasticity property because it can be moved and scaled efficiently and avoid service interruption. The benchmark detailed below will show that only the containers lightness makes them far more dense and elastic as oppose to the existing "plain vanilla" hypervisor paradigm.

Because containers can operate on the Linux kernel, we will show that vertical scaling can happen instantaneously but is effectively altering a resource limit within a process. As for horizontal scaling, their size makes them easily transferable to other machines and increase capacity. Memory scale-up in hypervisor-based technology is rapid as it only requires a resource allocation modification. However, a memory scale-down event in hypervisor requires a kernel boot or doing a balloon inflation [72] or both.

The proxy experiment detailed below will show that the horizontal scaling allows containers to be 3-100 times more elastic than hypervisor-based VM. This is one of the benefits that utility computing gets out of containers because the containers are more elastic than the hypervisors. However, containers convey only a uniform platform per vertical host. The main container's disadvantage is the homogenous nature of the kernel e.g. Windows and Linux instances cannot co-exist with the same container host. However, utility computing is homogeneous by nature as its value proposition comprise of elasticity and density rather than a heterogeneous platform.

3.0.2 Brief Virtualization Technologies Survey

Hypervisors. represents an execution platform based on shared but isolated OS instances. It allows a multi-tenant secure system without using dedicated hardware (HW) because hypervisor runs the guest OS instances in the form of virtual machines. A hypervisor is a layer that fully emulates the required HW and firmware for running an OS instance. Hypervisors existed since 1960s in many forms. VMware is probably the first company that enabled hypervisors for enterprises of all sizes. Later on, Microsoft Hyper-V, XEN and KVM offered more advanced hypervisors.

Para-virtualization. Introduced the Zen paradigm that is the theory that hypervisor-based operating systems cannot run virtualized applications and kernels as fast as native HW. Therefore, both the hypervisor and guest OS needs to alter their kernel physically to make it run as fast as bare metal. The argument that supported para-virtualization was that HW cannot run fast enough using the emulation paradigm. Later on, Intel and AMD added an extra feature for the hypervisor and closed the gap between fully virtualized and para-virtualized systems [66].

Unix Containers Solaris Zones and AIX LPAR and WPAR are shared-kernel-based virtualization technologies that use similar principles as Linux containers e.g. Namespace and CGroups. Its core value proposition to its customers is the ability to run hybrid proprietary Unix multi-tenant applications. e.g. the ability to run Solaris 8, 9, and 10 on a Solaris Zones Guest domain host. [7]

Linux Containers. The first Linux containers were formed in 2005 by open source Virtuoso (OpenVZ). In 2006, CGroups was introduced and used by Google for search engine containers. The first Linux containers implementation, LXC was released in 2008. Later on, OpenVZ, Docker and other solution use the same API with different management wrappers tools. LXC comprise of CGroups and Namespace API.

Windows-based Containers. are collections of normal Windows processes, isolated from the rest of the system so that they don't conflict with each other, plus the ability to create an image with all the libraries and configuration needed to launch an application. Conceptually, the closest equivalent in Windows to cgroups and namespace isolation are Job Objects. However, Job Objects does not allow the same level of quota control for resources such as IO, network or namespace isolation for devices, users and individual processes [89].

3.0.3 Linux Containers

CGroups. controls resource allocation to groups of processes such as CPU, Memory, IO bandwidth, and network bandwidth. It means that it take a group of processes and restrict the amount of resources that they consume. This is how LXC impose resource limit on all Linux containers.

[7]ORACLE SOLARIS 10 (2013) Retrieved from oracle.com/us/products/servers-storage/solaris/solaris-10-overview-ds-075575.pdf

Although there are many resources to control, the experiment below will focus on IO, CPU and Memory allocation per control group.

Namespace. is the isolation method. It separates resources to make them visible only to processes within the Namespace. There are currently six main namespace within the Linux kernel: Network, UNIX Timesharing (UTC), Mount, User, IPC, and process ID[54]. The Mount namespace allows us to have a mount-tree per container. IPC Interprocess communication namespace makes the IPC semaphore unique per container. The Process ID namespace allows containers to maintain its own process ID sequence. The User namespace provides the isolation between users. The Network namespace provides isolation associated with networking e.g. network device, IP protocol stack, IP routing, firewall rules, port numbers, etc. Containers can use all of these combination namespaces or indeed none of them. Namespace allows design secure containers with all the six namespaces as oppose to a lighter container that implements only a few. It depends on use case the container will enable [15].

Security with Containers. There is a contention that containers are not secure as hypervisors. The core security challenges with containers are the ability the potential resources leak between containers. The canonical example for such challenge will be hostile root. The Hostile Root issue discusses the root user of one of the containers accessing resources other than its own container. Since Linux 3.10 the namespace API implementation eliminates any chance of such security leak. It uses the new Linux capabilities. The standard nobody user gets the root privileges within a container. That means that if a user escapes from that container the user is nobody in the host. i.e. root on the container is not root on the host [54].

The Value Proposition

As stated previously, the compute container's value proposition is a tenancy. Allowing as much application tenants to run within a single compute resource will increase the density and better utilize the resource. Enabling multi-tenant application requires a significant investment in application modifications. e.g. adopting the application resource allocation to be shared with identical components such as IP routing tables, IP addresses, ports, filesystem mount points, IPC resources, etc. Moreover, it is coupling between the application and the infrastructure. The virtualization promise helped to address the challenge but in a more expensive way.

Compute containers allow the transformation of a single-tenant application to a multi-tenant application by containerizing it. All it requires is to give it a mount space with private data store with network namespace for a new IP address. Then one can fork the same container *n* times with different namespace for each fork. Because each fork is lightweight it can easily replicate across different multiple machines that reside on a different geo-locations to maintain a fine-grained resource allocation and elastic service model. This is an example of elastic and dense multi-tenant application.

The hypervisors-based technology significantly optimized for the critical computing design principals, e.g., density and elasticity, in the last decade. In fact, containers-based deployments in public cloud today are used in conjunction with hypervisors. We argue that virtualization and containerization symbiotic is an emergent behavior rather than desired system design that attains the required computing properties enumerated herein.

Implementation

Overview. The prototype goal is to exemplify a fine-grained compute resource allocation enabled by the containerized paradigm. It will show the resources allocation ratio between the hypervisor and container virtualization. The prototype includes a vertical cluster of a multi-tier application that spans across different Linux containers and hypervisor instances. The required compute resources for such layers vary, front-end servers focus on managing effective cache layer that provide fast response to the end-user. The backend layer might consume CPU cycles for sorting and compute request originated by the front-end layer. The middle tier comprises of a queue that mediate between the front-end and the backend tiers by creating events based on a policy to be processed by the backend rule engines.

System Architecture. The prototype application comprises of three layers. Presentation layer that operates over `HTTP` protocol and accepts two request types, `/GET` and `/PUT` data. In case of get data request, the logic layer act as a cache layer. It will first attempt to fetch the data from its local cache and alleviate `/GET` calls requests from the database layer. Only requests for records that are not stored in the cache will be fetched from the database layer. In case of a `/PUT` call, the front-end layer will publish an event to the middleware layer that will queue the request for the backend's rule engine servers. The job will be processed according to a policy-based process that later on will be stored in the database layer. In case of a a `/PUT` to an already exist record the cached recored will be invalidated synchronously and notify the entire front-end servers.

The various components uses a simple data structures and open-source based packages. The presentation layer comprises of a Node.js web server. The logic layer comprises of application server, and cache server. The application server uses Python-based library Flask [8]. The in-memory cache store and the data layer is based on Redis key-value cache engine [9]. It allows backend components to subscribe to events generated by the front-end layer for asynchronous processing.

Prototype Workload. The application will serve as a network planning service that accepts routers locations, build a graph and return the shortest path from the start router to the goal vertex. The get request will accept a graph identifier and two vertices. It will return the shortest path between the two points. The put request will accept a graph identifier and at least one coordination and create a new graph or append it to an already exist one.

Goal. The prototype attempts to illustrate the value customers would like to get from a utility-computing service, a fine-grained resource allocation when using Linux-Containers. We will build a complex multi-layer vertical node that can easily scale-out horizontally in case of growing end-user demand. Moreover, scale-in when user demand decreases. We will compare the tradeoffs for deploying and maintaining such architecture in hypervisor virtualization and containerized fashion. The goal is to show how a Linux-container-based application is more efficient and profitable when it fulfills the density and elasticity properties. The success indicator will be the dollar cost

[8]Flask; (2011) Retrieved from flask.pocoo.org

[9]Redis; (2013) Retrieved from redis.io

spent to generate the same allegedly customer requests.

Limitation. As the prototype run on the public cloud environment, scale-in and out events use the cloud provider dedicated mechanisms such as Azure Autoscaling and the Pivotal 'cf scale app' command. Although we used equivalent resources models among the providers, we rely on the correctness of such mechanism and the providers billing fairness.

Experiment Details. We studied the initialization time among the different public cloud providers. The initialization time is measured from the time the start command was executed until the application accepted `HTTP` traffic. The test client we used is an Apache JMeter instance that captures the start timestamp and spawning an HTTP client threads every second. The JMeter log will be the initialization time evidence. To eliminate the network latency we chose the closest region in the public-cloud provider options. i.e. US West. We also validated the latency among the two providers using the *ping* command. The ping tool provides the latency i.e. how long it takes for a single packet to get from the client host to the cloud service. The latency test included 100 (-c 100) pings with 0.1 (-i 0.1) second as the interval between each ping. The latency evaluation was performed against the following DNS entries:

```
ping -c 100 -i 0.1 lxc-cis606.cloudapp.net/ lxc-cis606.cfapps.io
```

where cloudapp.net is a Azure domain and cfapps.io is a Cloud Foundry domain.
The latency value from the JMeter instance to the cloud providers was $150ms$.

The second part of the experiment dedicated to the benefits of a fine-grained resource allocation system. It included vertical and horizontal scale-out and scale-in events that originated by the cloud provider autoscaling mechanisms.

Operational Definitions. The load is generated by an application loader (JMeter script). The loader application measures the various transaction per second(`/GET`,`/PUT`). Let *Throughput* be the set of transaction per seconds the loader able to get from the service. The experiment includes two runs. One against the containers-based cloud environment and a second with just a hypervisor-based environment. Let *Utilization*(%) be the set of CPU and Memory measurements on the application instances. *Allocation* is a function that map transient utilization u_i to number of core allocated. Each core is 2.5GHz with a 1GiB memory. We generated n samples where each Utilization sample denote by u_i and 0 <i$\leq$ n.

$$F_{Allocation}(u_i) = \begin{cases} u_i - 1 & : \text{ if } 0 < u_i \leq 20 \\ u_i & : \text{ if } 20 < u_i \leq 50 \\ u_i + 1 & : \text{ if } 50 < u_i \leq 70 \\ 2 \cdot u_i & : \text{ if } 70 < u_i \end{cases} \tag{3.1}$$

The utilization granularity in Eq. 3.1 was chosen based on previous capacity analysis of multi-tier cloud application [35, 40, 86]. The functions applied through both Pivotal.io and Azure.com developer portals [10]. The JMeter loader will measure throughput generated by two types of calls `/GET` and `/PUT`. There were two types of PUT calls, medium and large. Medium calls included graphs with 50 vertices with an average vertex degree 5. Large calls comprised of graphs with 100 vertices with an average vertex degree 10. The purpose of the two graph types is to test granular data usage while generating high CPU and memory during the shortest-path calculation. The results will include three metric sets, Utilization, Allocation, and Throughput. The desired optimization is defined by Eq. 3.2 as the ratio between the resource allocation using the container and hypervisor virtualization paradigms.

$$Optimization(\%) = \frac{\sum_i^n F_{ContainersAllocation}(u_i)}{\sum_i^n F_{HypervisorAllocation}(u_i)} \tag{3.2}$$

3.0.4 Evaluation

The experiment comprises of two studies: (1) Discrete experiments that proof the container uniqueness as oppose to the bare-metal and hypervisor-based virtualization. (2) A Linux-container-based prototype comprises of three application layers that fulfill both properties, elasticity, and

[10]http://docs.pivotal.io/pivotalcf/customizing/autoscale-configuration.html; http://azure.microsoft.com/en-us/documentation/articles/cloud-services-how-to-scale/

density. The prototype[11] described in the Implementation section will be deployed in both hypervisor and container based public platforms and compared on ground of density and elasticity.

Discrete Performance Study

Resources density depends on the framework ability to migrate running components across the resource pool. A component migration comprise of packaging the component code, replicate it, unpack, and run it in the new hosted environment. We implemented a simple web application that accepts `HTTP POST` requests, logs the call and write to a persistence storage. We also used the *dd* utility to generate local CPU, and I/O load. The goal is to evaluate initialization time and storage overhead when running in a hypervisor and container-based environments. The discrete evaluation was performed on the following platforms:

Hardware. A physical server with four cores `Intel(R) Core(TM) i7-3615QM CPU @ 2.30GHz`. The RAM size was 8GB.

Hypervisor. The hypervisor we used was `Linux CentOS 6 with KVM support`. The guest OS was `Linux CentOS 6`. The virtual hard drive space configured for the server edition is 5GB with minimum 1 GB RAM.

Container. The container platform we used was `LXC-1.1.2` runs on `Linux CentOS 6`

Initialization Time

We first tested the time it takes to initialize a single application instance i.e. guest OS instance in hypervisor-based virtualization and a single Linux container in LXC host. The guest OS init time in Table **??** shows the containers-based virtualization core advantage. The time its takes to spawn a Linux-container instance is in the order to forking native OS process as oppose to an order of minutes in hypervisor-based technology.

[11]Retrieved from github.com/yahavb/MultiLayerApp

Table 3.1: Initialization Time - The hypervisor takes a couple of seconds while LXC is almost instantaneously. Optimized Guest OS init time is in the order of minutes while LXC uses the same kernel i.e. instantaneously. Hypervisor is a separate process than the OS kernel and requires 15% overhead

Category	KVM	LXC
Init Time(sec)	8	1
Guest OS Init Time (seconds)	120	4
CPU Usage - no load (%)	15	3

Storage Footprint and Usage

We measured the storage footprint and usage of each virtualization paradigm during the simulation of read and write operations. Measuring the storage footprint of each virtualization paradigm requires access to the hosting environment. As public cloud providers does not provide access to the hypervisor or the LXC instance, we used a physical server with four cores `Intel(R) Core(TM) i7-3615QM CPU @ 2.30GHz`. We explored two types of scale-in/outs, vertical and horizontal. Vertical scale-out describes the process of provisioning additional resources such as CPU, memory or I/O. Increasing dynamically the memory allocation in hypervisor-based VM known as ballooning. For ballooning the KVM instances, we issue the balloon-inject-host-pressure command. For LXC, we created a blkio cgroup [85] which controls the I/O policies enforced by the LXC process. The test included write operation of 5 GiB. 1MB per request with 5000 requests i.e.

```
dd if=/dev/zero of=cis606-rw-load bs=1M count=5000
```

Figure 3.3 shows minimal throughput impact when CPU, memory and I/O allocations dynamically changed when using Linux Containers. Hypervisor had a severe performance impact while resources modified dynamically. Table 3.2 depicts the advantages containers (LXC) have on hypervisors (KVM) in storage and memory footprints e.g. 30 times less static storage footprint and Guest OS init time. Having a memory footprint in the order of 30 MB allows easy live migration of VM across geo-locations for increasing usage density.

Table 3.2: Storage Footprint - The hypervisor requires its software code and the Guest OS image for each VM. LXC share the same image of the hosted OS. The RSS requires 0.5GiB for hypervisor is it runs a full OS while LXC use 30MB that comprise of container namespace

Category	KVM	LXC
Static Storage footprint (GiB)	9	0.29
Guest OS Init Time(seconds)	120	4
Total Resident Set Size (MB)	500	30

3.0.5 Observation

The pricing model [12] of both services we used in the experiment are based on memory and CPU. Table 3.3 depicts the similarity between the two cloud providers pricing models. Its main purpose is to establish a cross-cloud-services pricing baseline that will help to assess the potential resource allocation optimization.

The experiment included three core sections. The first was a medium and large `/PUT` request with a corresponding `GET` request for utilizing the cache layer. Each new `GET` request consumes more CPU and Memory as it calculates the graph's shortest-path for all vertices and store it in memory. During the cache buildup pages, the throughput, ranges between 7 to 10 `GET` and `PUT` transaction per second. During that phase, the number of 1GiB cores increases until it stabilizes towards 1:20. At that point, the majority of the load comprises of 90% `/GET` requests of already exists graphs and 10% of `/PUT` requests for new graphs. From 1:20 to 1:50 the throughput rises to 24 transactions per seconds while the utilization decreases because most of the GET requests were fetched from the cache memory.

Based on the Optimization Equation 3.2, Figure 3.4 shows more granular CPU cores allocation and deallocation than in Figure 3.4 resulting 22% less resource allocation in container-based technology. Moreover, the two utilization functions are similar enough to conclude the optimization as the standard deviation of both utilization measurements are below 0.133 and average of 35%. Also, the cumulative distribution allocation function 3.6 shows the similarity between the two functions.

[12]http://azure.microsoft.com/en-us/pricing/;http://run.pivotal.io/pricing/

Table 3.3: Pricing Model - The pricing model of Pivotal.io and Azure Pricing Model Comparison. Both services collect $0.03 per hour of 1GiB memory

Cloud Provider	Utilization Model	Utilization Units	Price per Unit
azure.com	CPU & Memory Centric	1 core per 0.76GiB	$0.036
pivotal.io	Memory Centric	1 core per 1GiB	$0.03

3.0.6 Related Work

Prior work leveraged VM consolidation approaches that re-pack VMs into fewer physical machines to increase density [75]. It explored a just-in time approach to VM consolidation by transitioning VMs to an inactive state when idle and activating them on the arrival of client requests [43]. Although the prior work supports greater VM density, it is still limited by the number of VMs that could be hosted in the one inactive state. If idle VMs were hosted in multiple inactive states, VM density can be increased further while ensuring small miss penalties.

3.0.7 Conclusion for Cloud Federation Computing Paradigm

We explored the container-based virtualization technology as a novel method to increase allocated resources density in utility computing. Our numerical evaluation shows that Container-based virtualization enables both lighter storage and memory footprints that help migrate VMs across different locations for maintaining optimized resource allocation. Moreover, it allows an optimized granular resource allocation and deallocation. Its proven capability to dynamically modify its CPU, memory and IO allocations with minimal impact on performance; exemplify the elasticity aspects achieved when applied to utility computing. In summary, container-based virtualization allows cloud providers to increase resource density; that in turn reduces operations costs for providers and cloud tenants through enabling fine-grained allocation of memory and processing units.

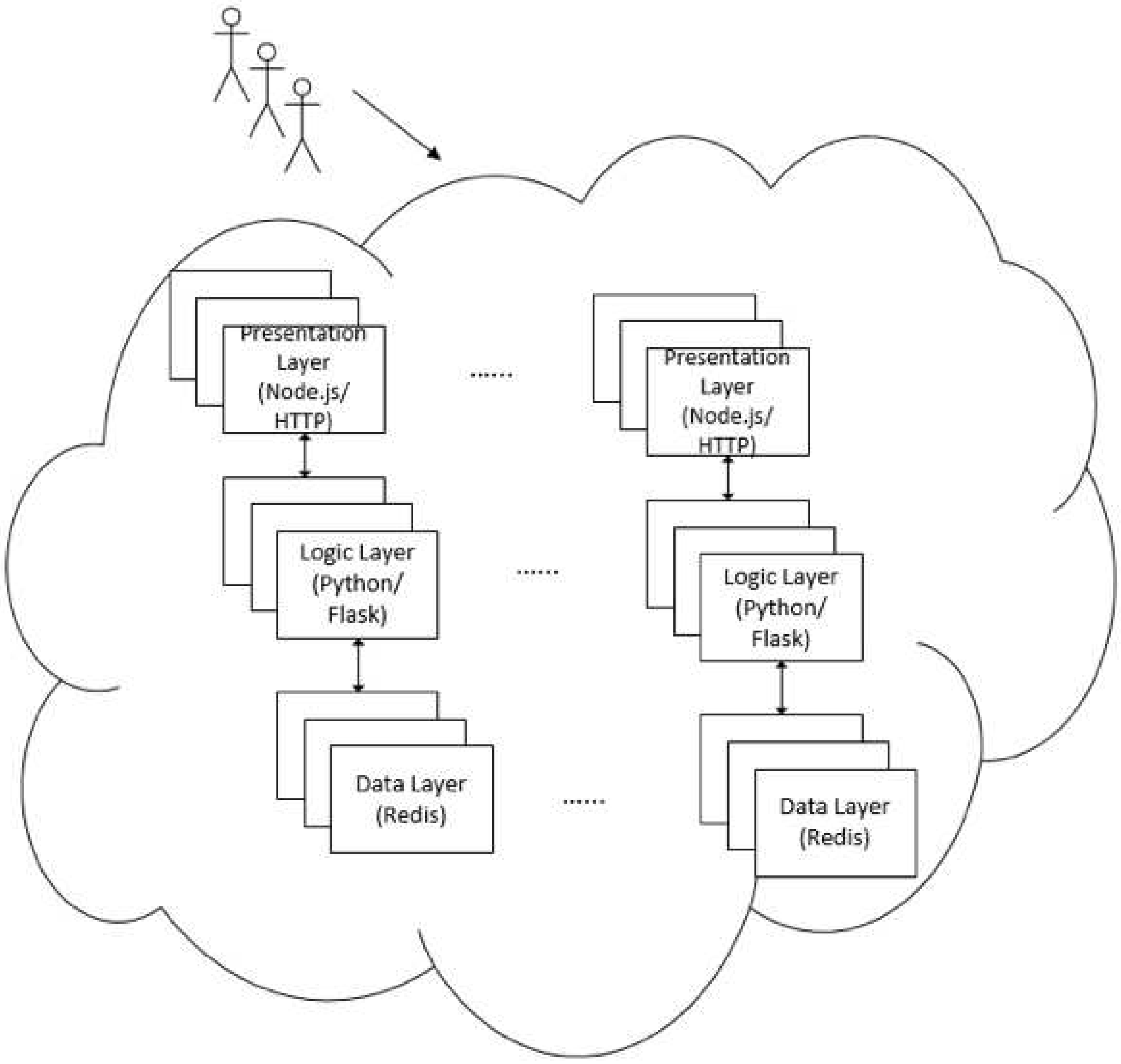

Presentation Layer (Node.js/ HTTP)
......
Presentation Layer (Node.js/ HTTP)
Logic Layer (Python/ Flask)
......
Logic Layer (Python/ Flask)
Data Layer (Redis)
......
Data Layer (Redis)

Figure 3.2: Experimental Setup

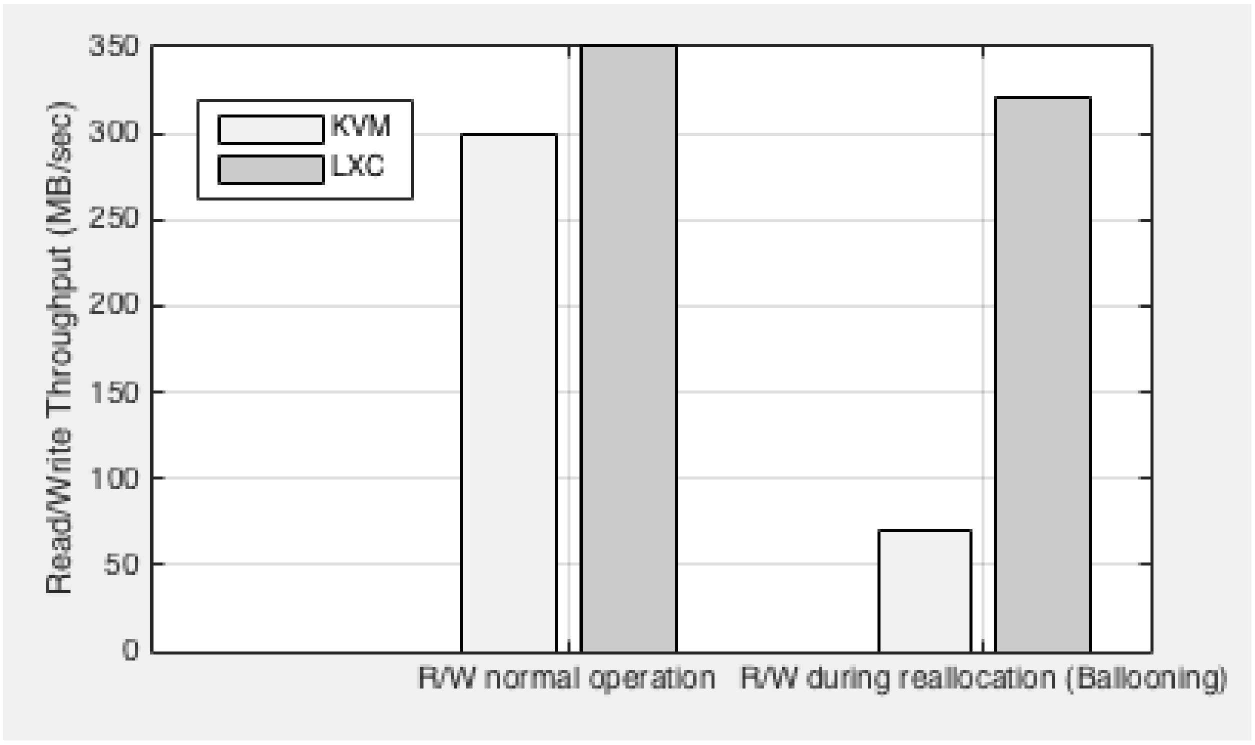


Figure 3.3: LXC provides the ability to modify CPU, Memory, and IO allocations dynamically with minimal performance impact while hypervisor requires significant performance impact

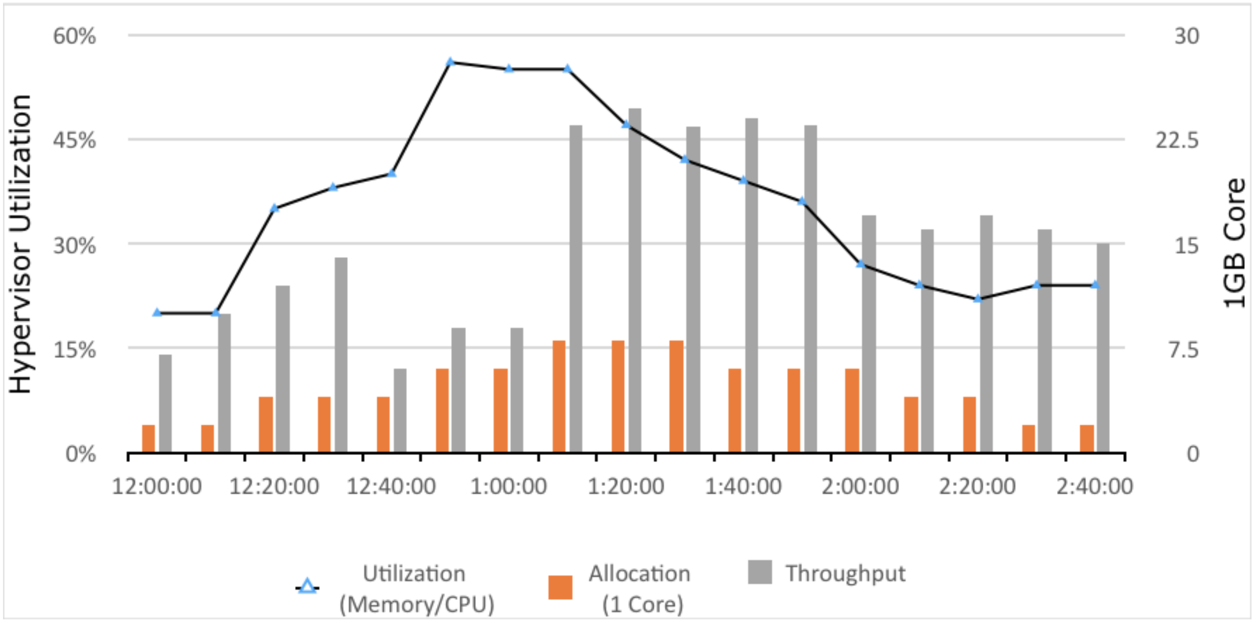


Figure 3.4: Hypervisor- Utilization/Throughput/Cost - Cache buildup during the first 1:20 hours load; Using 44 cores of 1GiB; During the high load segment (1:20-1:50) consumes 28 cores.

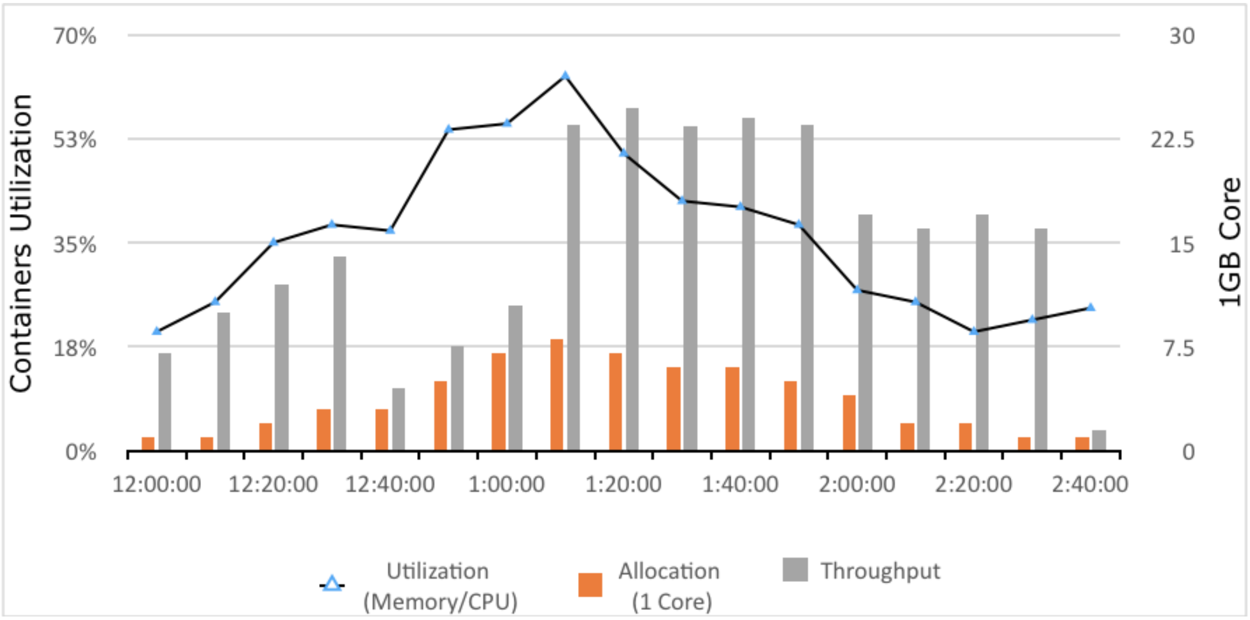


Figure 3.5: Container - Utilization/Throughput/Cost - Similar to Fig. 3.4, however, core allocation/deallocation is done in higher granularity resulting in 22% less allocated CPU core resources

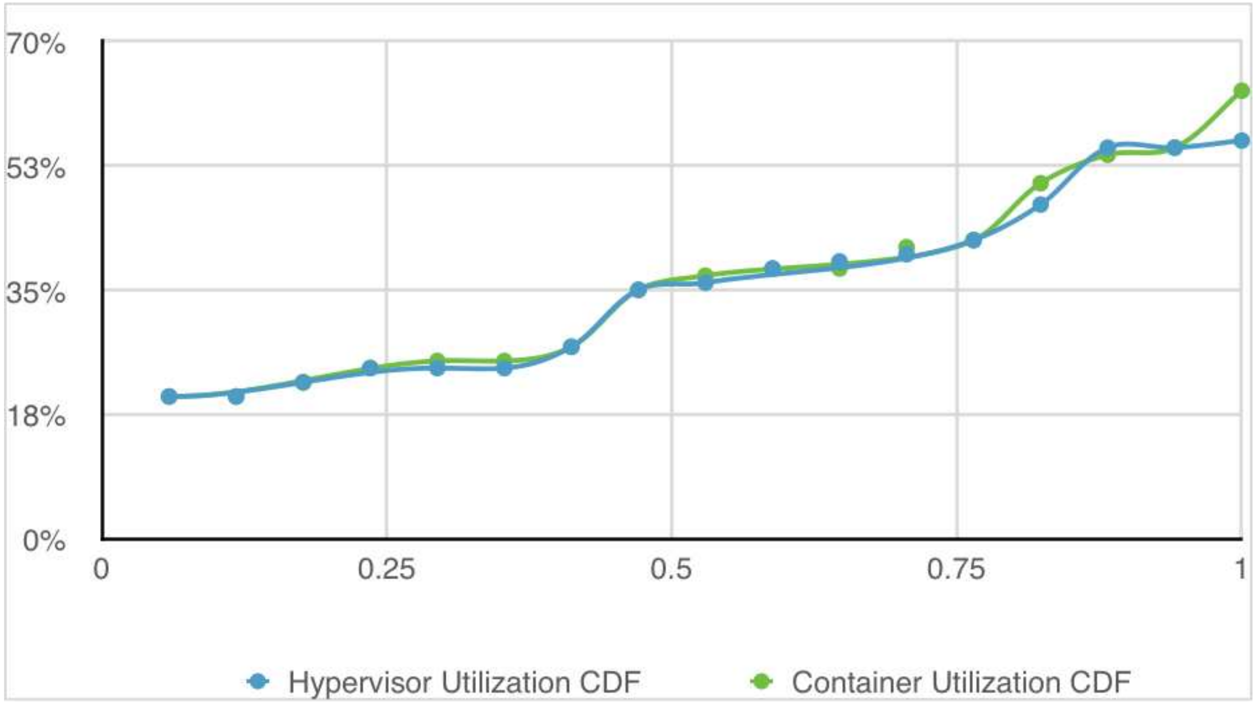


Figure 3.6: Cumulative Distribution Allocation Function - Both distribution functions, shows that the probability of similar allocation for both experiments is high.

Chapter 4

Computing Hardware Equipment

Organizations that serve and process a large number of simultaneous users and an immense amount of user data are termed cloud computing services. These services enable convenient, on-demand network access to a shared pool of configurable computing resources. Cloud computing includes private, public, and hybrid forms. In 2014, IT-related spending toward workload processing increased by 32.8% and grew by a further 29% in the following year; it is expected to have grown by 29.1% during 2016 [60]. This has raised environmental concerns, as hydrocarbon-powered grid energy has become increasingly devoted to cloud computing's growing power needs. Cloud computing is therefore one of the largest energy users in the information spectrum and requires establishing best practices for engineering. This chapter outline a new method that develops an effective conceptual, preliminary, and detailed design for the computing equipment hardware used in cloud computing. Both a reduction to total energy use and the maximization of green grid energy fractions for total energy use will be addressed.

A cloud computing system comprises two interwoven core elements: hardware and software. The hardware element includes servers, racks, and power and cooling systems, while the software element uses this hardware to process workloads. The predominant software paradigm is open with a freely available source code capable of modification by an open-source community. Conversely, compute equipment hardware systems have no open standard, nor have open best practices for designing a warehouse-scale computing system evolved. It is presently dictated by proprietary vendors, such as IBM, Hewlett-Packard, and Dell. These vendors often offer a tailored business solution in which the equipment design interaction methodologies are derived from that single path perspective.

4.0.1 The Business Aspect of Equipment Design and Manufacturing

Previously, proprietary compute equipment vendors have had full control over what software runs best on their fundamental chipset engine. Historically, these vendors offered model specifications that best fit their own products rather than heterogeneous systems, such as a public cloud. Proprietary vendors often embedded management systems into their products that forced cloud providers to use exclusively use that vendor's equipment.

Core goals for cloud computing have been to provide hyper-scale IT agility and rapid resource allocation between computer systems while reducing operation costs. When specific needs are attached to a vendor's product, operators are required to adjust their architecture to the various vendor systems to run their service. This solution is suboptimal as specific needs might include the use of generic computer features in large scales, which can leave some system components nearly or totally unused. Proprietary hardware adds high reliability features on the component level where they are not needed, such as in cold backup systems, and where 99.999% availability is not required. Our model includes a cohesive approach that considers the overall scenario and suggests specifications for discrete components.

Our goal is to provide a new compute equipment design methodology that does not solely rely on proprietary hardware and software vendors. It invites systems operators and cloud computing providers to participate and suggest improvements to our already available IT equipment specifications based on their prior experiences. We suggest a new design for scenarios not used yet in the industry e.g., a dedicated data center for cold storage.

Building a community that includes both IT equipment operators and the suppliers of software and hardware will foster collaboration, which will enable cloud operators to achieve both optimized costs and increased IT agility. This community will eliminate dependency on proprietary design and allow for organic design modification not only by systems vendors but also by systems operators and cloud providers [34]. In this study, we will present a model that mimics the open-source software paradigm and provides a metric for the technical measurement of computer systems' hardware design. Using the Open Compute community as an illustrative collaboration platform, this paper presents a methodology to both gather and release specifications to the professional community. This methodology continuously improves previous specifications based on the needs of both customers and operators along with evolving vendor constraints.

We begin by first discussing the relevant terminology used in this paper. We then analyze the requirements for optimal gathering in a general systems engineering process. This is followed by an exploration of the principles that motivate our design. We then measure system performance and, lastly, present our conclusions.

4.0.2 Systems Analysis and Design

A systems engineering process is driven primarily by system capabilities and customer needs. It ensures the orderly realization of the system configuration, composition, operations, maintenance, support, sustainment, and disposal [30]. In general, the three major phases in systems design are: high level conceptual design, preliminary detailed design, and detailed design. Each phase results in increasingly focused system specifications. Clearly, each phase's system specification relies on legacy specification phases as well.

The *conceptual design* phase yields the system operation requirements. It describes the operational scenarios, e.g., the type of workloads the cloud system will process. The *preliminary design* ensures that the operations requirements conform to the functional analysis and allocation at the subsystem level and the performance requirements of subsystems or the system components. The initial design phase also establishes the product specification Type C and Type D [30].

During the preliminary design phase, we determine whether we are able to use COTS equipment or if we must design a system element from scratch. The initial design phase also defines the specification associated with a process or service, e.g., manufacturing a new element or assembling a group of COTS-based elements into a system component Type D [30]. The preliminary design phase results in product specifications that describe both the quantitative and qualitative measurements to which the detailed design must conform.

We believe that efficient cloud computing service design is strongly coupled with systems engineering practices. Recent advances in the field of hardware and software disaggregation and the evolution of the ODM market have opened up hardware design to more contributors working in collaboration. Every contributor brings vital data to the design process, and systems engineering allows for a comparison of equivalent alternatives by providing a framework that converts subjective data into a common measure.

4.0.3 Computing Equipment Elements

The following section will describe elements of the compute equipment used in cloud computing systems, which later be used as the basic element sets for specification.

Cloud computing system are composed of one or more data centers. A data center includes clusters of *racks*, *chassis* and *servers* that store information in *digital storage* and are connected by *networking equipment*. Data centers also include both peripheral, mechanical, and electrical elements. A typical electric system is made up of a power substation, a diesel-based generator, a main switchboard (MSB), and a backup battery system. The mechanical elements include cooling systems, fire sprinkler systems, and the building that houses the data center itself.

Figure 4.1 depicts the major computer elements of a basic data center, which is composed of computer server clusters that are aggregated by chassis and racks. Racks are usually fed by separate power distribution unit (PDUs). To guarantee a continuous power supply, PDUs are connected to an uninterruptible power supply (UPS) that connects to the primary power source through the automatic transfer switch (ATS). The racks are also connected to the data center backbone network through the top-of-the-rack (ToR) switch; the servers are also connected to the ToR switch.

The core categories of compute equipment are: (1) networking switches and routers, (2) computer servers, (3) digital storage servers, (4) chassis and hardware management, and (5) the power supply. These categories are mapped to design principles and form a measurable effectiveness value in the equipment specification that best fits the user's needs.

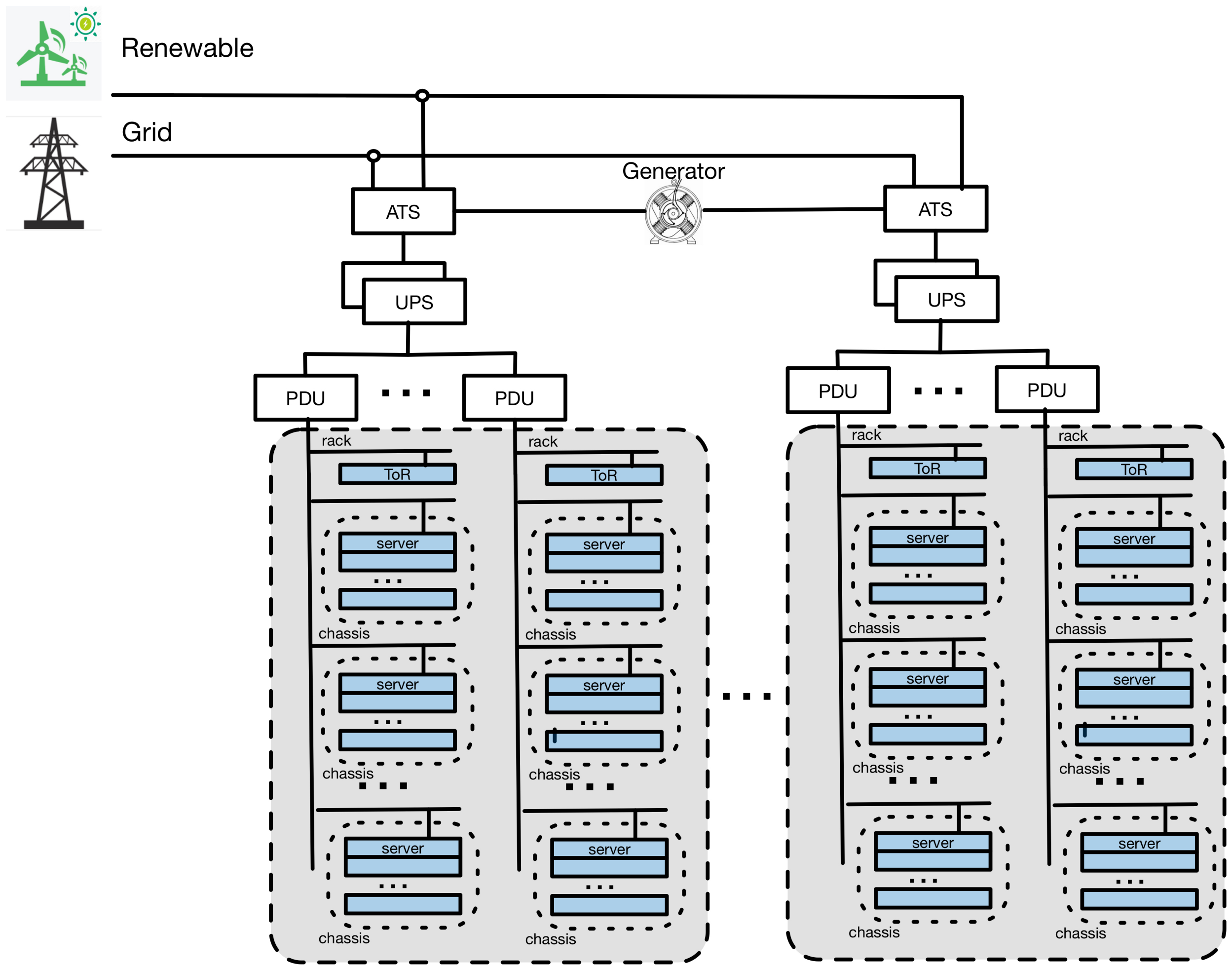


Figure 4.1: Datacenter Architecture - Compute server clusters aggregated by racks and chassis. Racks are fed by separate Power Distribution Unit (PDU) that connects to Uninterruptible Power Supply (UPS) that connects to the primary power source through Automatic Transfer Switch (ATS).

Computer servers include the power supply, motherboards, CPU sockets, gigabit ethernet links, disk drivers, and SSD storage cards that hook into PCI cards. For redundancy, more than one of each element is built into a configuration, e.g., We are going to use two mechanical element types for specification: gear and tray [59]. Here, gear is defined as the equipment that plugs directly into the live power supply, and tray is a gear element that plugs into a chassis or rack. Figure 4.2 illustrates a computer server design composed of elements form different ODMs, which were designed in collaboration with the cloud service provider [59]. The design framework we propose in this study will be exercised through a customized storage computer server designed by the Open Compute community [59].

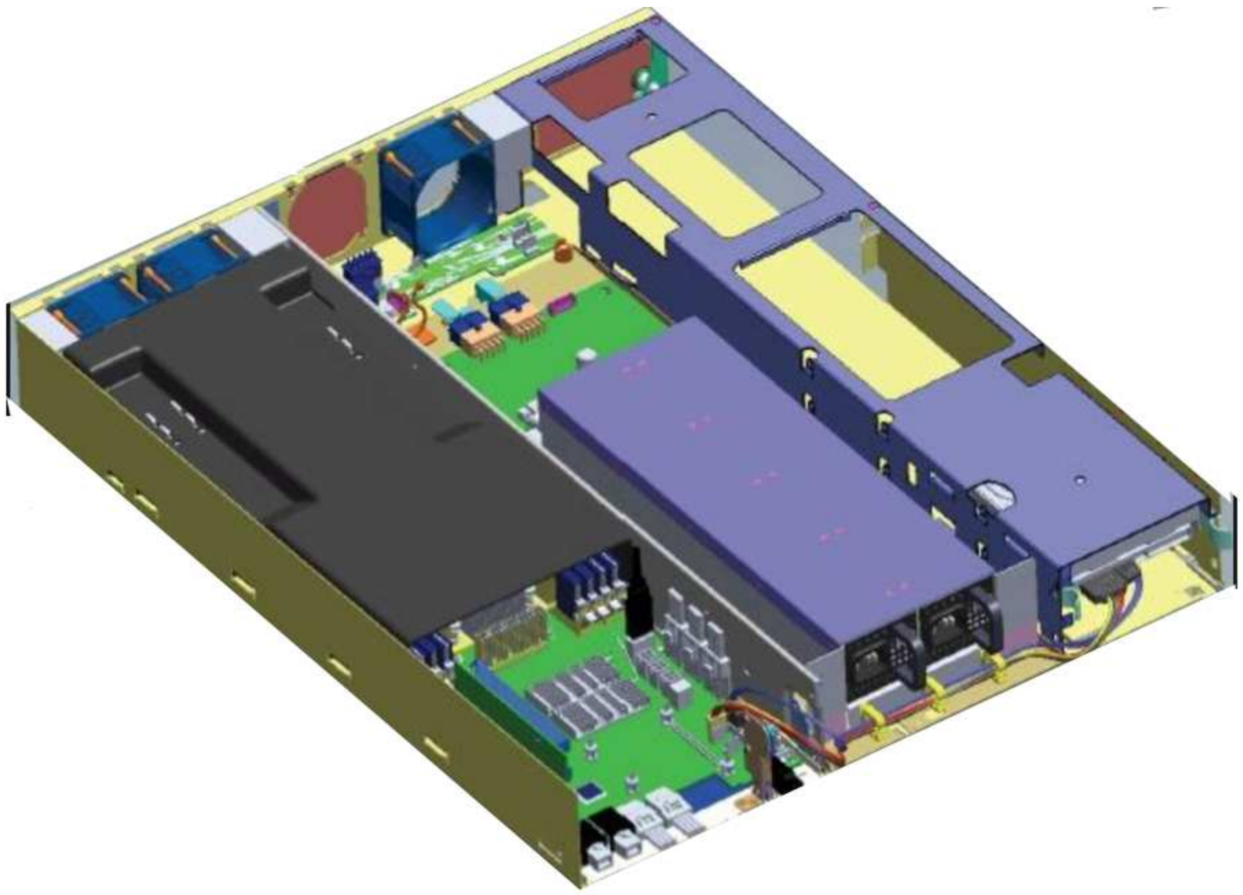


Figure 4.2: Mechanical drawings of Facebook's Dragonstone server. The server has a two sockets server node on the left, redundant power supply in the middle, and then space for 3.5-inch disk drives of SSD storage from Fusion-io in a storage sled on the right. The server is based on Intel Windmill board, redundant power supplies from two different suppliers. Power One and Delta were designed in collaboration with Facebook to fit the middle tray and feed the server node and the storage.

4.0.4 Design Principles for Computing Equipment

Cloud computing has three core characteristics: (1) on-demand computing processing, digital storage availability, and network broadband; (2) multi-tenancy enabled by resource pooling; and (3) elastic resource allocation [31]. The following section will describe the principles to be used in this study for the suggested open system design specification paradigm. The principles goal is to allow effective trade-off analysis when designing a tailored cloud-based system that requires compute equipment.

Design for functionality. All is derived from the technical capability to accomplish the product's intended mission, e.g., the ability to compute operations such as arithmetic or logical operation and store the result in a durable digital storage. Also, for sustaining a measurable workload of such compute operations.

Design for interoperability. This level pertains to the product's capability to operate in a system with external components. Interoperability for computer-related products refers to the ability to share a standard hardware and software interface that can be used to perform a system operation, e.g., various components that are physically interconnected using standard connectors. Suboptimal interoperability may require custom adapters to connect components to each other. Additionally, hardware elements should be integrated directly into the hosting system, e.g., racks should integrate directly into data center air containment solutions [59].

Design for sustainability. This refers to the energy wasted by the product throughout its lifecycle, e.g., the power drained at a constant rate measured in Joules or, more commonly, kilowatt hours (kWh), from the power source to perform the product function. Sustainability is also concerned with the amount of greenhouse gases, toxic substances, and air or water pollution that the product generates. Additionally, it describes the ability to reuse or recycle some of the product's elements upon certain usage time. A sustainable design minimizes non-recyclable components.

Design for reliability. covers the operational failure ratio. Reliability is often measured in terms of *mean time between failure* (MTBF). The goal is to maximize the operational reliability by minimizing the failure ratio and optimizing the system redundancy level.

Design for maintainability. reflects the ease, accuracy, and economy of maintaining the system. It is often measured by the *mean corrective maintenance time* ($\overline{M}ct$). The objective is to minimize the maintenance time and labor hours while maintaining the system with compliance to the manufacturer guidelines. e.g., data cables are located on the front of the rack.

Design for usability and safety. This concerns system interfaces between users and the system equipment or facility. System usability will help to assess the required skill set and training for the operation and maintenance of the system. Safe and usable systems also minimize human error while maximizing productivity and safety.

Design for supportability and serviceability. This centers on increasing the accessibility to elements that requires maintenance. Also, It also includes, the ability to diagnose a component health. e.g., component faults are identifiable from the front of the rack. Routine service procedures do not require tools.

Design for affordability. Often, referred to the economic feasibility of purchasing or leasing a component or even a whole system throughout its life cycle. Affordability is measured with direct relation to the overall project budget.

4.0.5 Developing a Decision Model for Compute Equipment Requirements

Enabling a cloud computing service's computer capacity includes various decision assessment processes. In general, decision evaluation theory relies on three core factors: money invested in the service, money flow modeling, and economic optimization modeling. This section will introduce a framework rooted in decision evaluation theory, which will be based on the multiple criteria enumerated in previous sections as well as a concern for risk and uncertainty, e.g., an unexpected surge or plunge in computer resource demand. Additionally, we will propose a decision evaluation display as a means to present and consider alternatives in the context of multiple compute equipment design principles and limiting factors, e.g., predicated required capacity and workloads.

Comparing Effective Design Alternatives

To compare different compute-equipment design alternatives, it is vital to convert differently sized criteria to a common measure. The conversion allows for an equivalence compression, e.g., transforming MTBF, carbon footprint or $\overline{M}ct$ values that a piece of compute equipment might support into a common measure would allow for an objective assessment. Qualitative measures should also be converted to a similar scale.

A correct conversion model for existing quantitative and qualitative criteria is key to the success of the design process. The conversion process is modeled on existing systems. If the desired system does not yet exist, an analog model will be used through simulation and experimentation. Experimentation includes direct evaluation, where the equipment is subject to manipulation and the results observed, e.g., designing a new server might include modifications to the placement of elements, such as the motherboard, processing elements (the CPU and GPU), and non-volatile memory (the SSD drive). It may also include the construction of a full prototype to be used with its peripheral components, e.g., a new server installed in a previously designed and compatible chassis or rack.

The conversion model we propose is based on prototypes and simulations, which were done on a full set of compute equipment that support common utility computing scenarios. Each solution relates to a specific component, and the scenario conveys the relevant design principles considered in the analysis.

Quantitive Measurement

The most commonly used technical methods for measuring operational and performance properties are the *Measures of Effectiveness* (MoE) and the *Measure of Performance* (MoP) According to [30] the MoE refers to the scenario we wish to implement. A few of the scenarios that we examine include cold storage, high-density I/O, server cooling, and more. Our proposal develops a set of MoPs for each MoE based on a list of design principles. The following section will discuss a solution to a cold storage scenario and quantify the plausible MoP values that will be aggregated based on MoE merits.

Design Example - Cold Storage

Digital information is stored in digital storage devices. Existing storage devices support rigid requirements classifications. For an illustrative example, consider HDD-based, SSD-based and tape-based storage solutions. As specified by the MoE, cold storage devices store data that are almost never read, but which must be fetched for use within seconds when required. Among them, SSD-based solutions respond within milliseconds and fit frequently accessed data. Conversely, response times for tape-based storage devices may last several minutes. Finally, while HDD-based devices can respond within the required time, they require extensive compute resources relative to the frequency of the data's use.

The following tables 4.2, 4.3, and 4.4 captures the result of the initial step in the proposed design framework. Table 4.1 displays the results of the initial analysis in the proposed design framework. The MoP values are determined by previously studied prototypes, experiments conducted for similar products, and feedback received from the Open Compute community. A value of one indicates a poor fit (e.g., tape-based devices are too-slow for cold storage), a value of four indicates a moderate fit (e.g., tape-based devices depend on available COTs; therefore, the interoperability fit is moderate), and a value of seven indicates a nearly optimal fit (e.g., dependency on COTS vendors for HDD-based and SSD based products).

Table 4.1: COTS Cold Storage Measures of Performance

Design principle	MoP Tape	MoP HDD	MoP SSD
Functionality	1	10	10
Interoperability	4	7	7
Sustainability	10	7	8
Reliability	4	7	9
Maintainability	9	9	9
Usability	9	9	9
Supportability	7	7	7
Affordability	9	9	9

Table 4.2: COTS HDD-Based Cold Storage Measures of Performance

Design principle	MoP Value	Premise
Functionality	10	Fit to needs
Interoperability	7	Depends on available COTS
Sustainability	7	Device consumes energy 24x7
Reliability	7	Moderate MTBF
Maintainability	9	Fit to needs
Usability	9	Fit to needs
Supportability	7	Depends on COTS vendor
Affordability	9	Depends on COTS vendor

Figure 4.3 depicts the predicted performance of a design alternative across the design principles for compute equipment. An initial analysis shows that COTS-based products provide suboptimal solutions. In an optimal design, we would see the radar plot spread uniquely across the design principles. Therefore, the design requires additional alternatives, i.e., a custom solution.

Table 4.3: COTS SSD-Based Cold Storage Measures of Performance

Design principle	MoP Value	Premise
Functionality	10	Fit to needs
Interoperability	7	Depends on available COTs
Sustainability	8	Consumes less energy
Reliability	9	Fit to needs
Maintainability	9	Fit to needs
Usability	9	Fit to needs
Supportability	7	Depends on COTS vendor
Affordability	9	Depends on COTS vendor

Table 4.4: COTS Tape-Based Cold Storage Measures of Performance

Design principle	MoP Value	Premise
Functionality	1	Too Slow
Interoperability	4	Depends on available COTs
Sustainability	10	Device consumes energy on-deman
Reliability	4	Susceptible to bits of dust and grit in the data centre air
Maintainability	9	Fit to needs
Usability	9	Fit to needs
Supportability	7	Depends on COTS vendor
Affordability	9	Depends on COTS vendor

A custom storage solution that satisfies the design principles requires a full-stack approach to sustainability and interoperability. This solution has three parts: the facility, the rack, and the server. Our new design will seek to reduce greenhouse gas emissions, allow the various product capabilities to operate with each other, and improve maintainability, usability and safety.

With regard to the facility, reducing greenhouse gas emissions requires the construction of a new building that uses relatively low amounts of power. A large amount of floor space is also needed to accommodate compute equipment that supports up to one exabyte per data hall. By assuming that this dedicated facility will support offline workloads and not live production workloads, we are able to remove redundant electric system, such as UPSs and diesel generators.

For maintainability and usability, the rack component requires easy installation and efficient service operations. Unlike standard racks, data cables should be located in front of the rack. Furthermore, component faults should be identifiable from the rack, and routine service procedures should not require any tools [59].

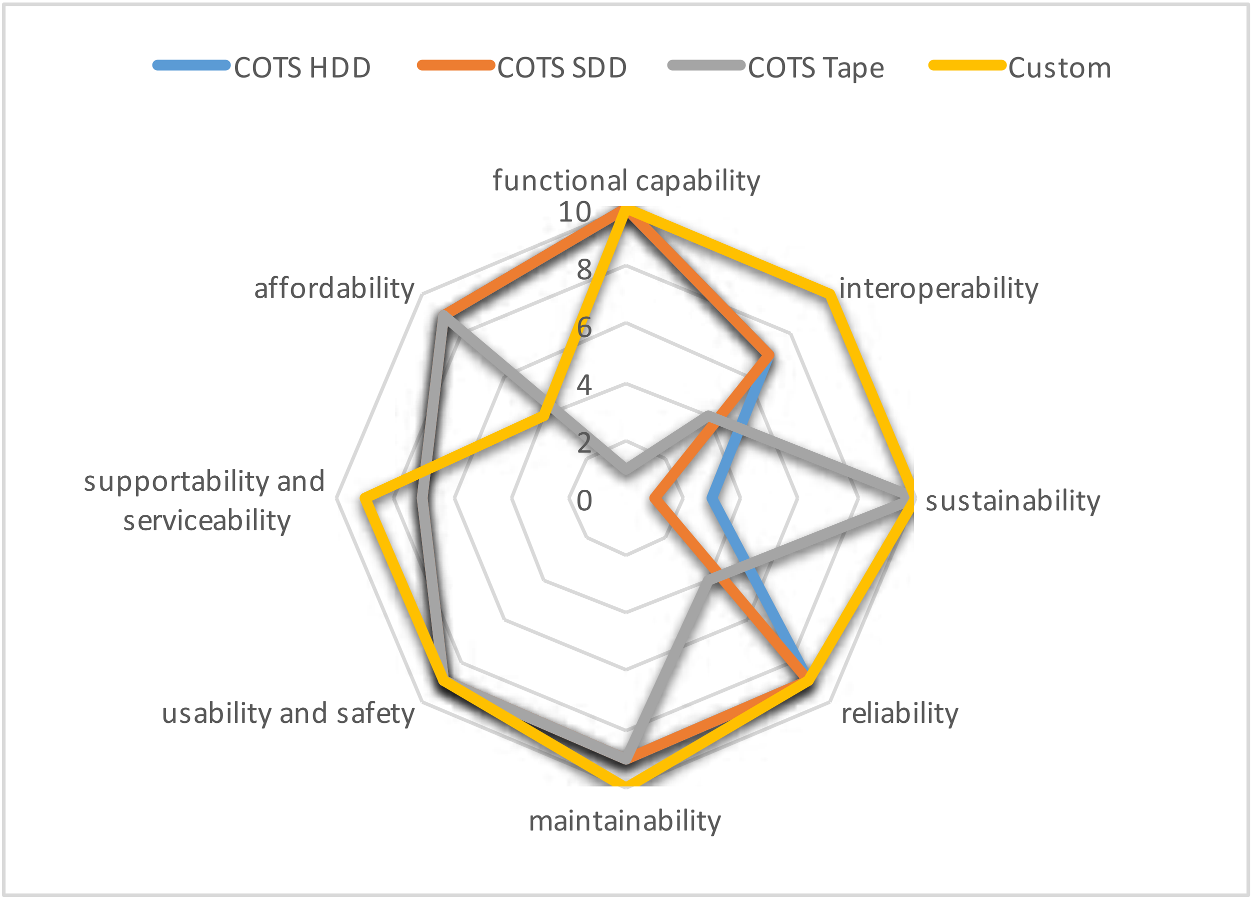


Figure 4.3: This radar plot depicts the anticipated performance of a design alternative across the design principles for compute equipment. An initial analysis shows that COTS-based products provide suboptimal solutions. In an optimal design, we would like to see the radar plot spread uniquely across the design principles. Therefore, the design requires additional alternatives, e.g., a custom solution.

For sustainability, non-recyclable components should be minimized and racks should be integrated directly into the datacenter's air containment solutions with no additional adapters. At the server level, we suggest the use of Open Vault [59] as it is a simple and cost-effective storage solution with a modular I/O topology that is compatible with Open Rack and other rack solutions. Open Vault also offers high disk densities (30 drives in a 2U chassis) and can operate with almost any host server.

Decisions Involving Equipment Criteria

In general, design decisions involve multiple criteria that jointly influence the alternatives under consideration. The following section will present a formal approach to handling multiple criteria that incorporates the MoE and MoP. We will also continue the design for the cold storage scenario and explore more methods to assess the alternatives' effectiveness.

The custom option appears to optimize most of the design criteria. However, it is not yet clear if the design satisfies the affordability criteria (an MoP value of four). Our approach will adopt one of the decision models under uncertainty. At this point, we have a finite set of design alternatives; our next step will be to reduce the MoP assessments in Table 4.1.

To do this, we will first eliminate the alternatives that do not function as needed. The tape-based solution does not comply with the solution?s functionality requirements. Design alternatives with similar values can also be reduced. The remaining alternatives will be analyzed for functionality, interoperability, sustainability, reliability, and supportability.

Interoperability is assessed by examining the impact on costs when operations of a certain scale are deployed across different data centers. With regard to sustainability, if an alternative requires more electrical energy, it will generate a larger carbon footprint, which might result in increases to the carbon taxes paid by the operator. Reliability measures might be less critical for some architectures that can be diminished with optimized supportability: If the product is resilient to sporadic failures (manifested in moderate MTBF values), optimized supportability can be masked from the end-user, e.g., part replacement does not require the tedious troubleshooting process that is usually mandatory under maintenance contracts with the product vendor.

We will now evaluate the costs for the remaining alternatives based on the estimated workloads for the cold storage scenario. This evaluation will serve as a discussion point for decisions approached under uncertainty. This analysis will rely on a set of estimates about the end users and their system-usage patterns to predict the compute workloads that the cold storage service will be required to process.

Equation 4.1 estimates the annual requirements for digital storage capacity. n_u indicates the number of active users, i_s is the average size of image file uploaded by a user. v_s denotes the average video size uploaded by a user. μ denotes the content attenuation coefficient i.e. the time it takes the content to be a candidate to be stored in a cold storage. User behavior patterns indicates negligible number of access to content after 10 days. Finally, $C_{storage}$ denotes the annual digital storage capacity.

$$C_{storage}(n_u, i_s, v_s) = \frac{n_u(i_{daily} + v_{daily}) \cdot n_{days}}{\mu} = \frac{1.00 \cdot 10^9(5 \cdot 30MB + 1 \cdot 250MB) \cdot 360}{20} = 6.5483EB \tag{4.1}$$

Assuming a standard server storage capacity is $S_{storage} = 2 \times 2TB$ disk drives and a standard datacenter has $N_{servers} = 40,000$ servers. Therefore, the required dedicated datacenters for cold storage depicts in Eq.4.2

$$DataCenters = \frac{C_{storage}}{S_{storage} \cdot N_{servers}} = \frac{6.54 \cdot 10^6}{4.29 \cdot 10^1} = 42.92 \tag{4.2}$$

The entire workload is distributed across 40 regions worldwide hence the estimated annual $C_{storage}$ capacity justifies the deployment of a dedicated datacenter for cold storage to address the organization needs. Building dedicated datacenter for particular scenario such cold-storage, allows the designer to assume special conditions in the datacenter such as cooling the aisle only to $77°F$ instead of the required $68°F$ by most COTS vendors. In some cases, custom server-only cooling can compensate on the additional aisle heat.

The next topic we need to address is whether HDD or SSD is the preferred approach for the dedicated datacenters. The predicated usage pattern of the cold-storage service requires some of the storage devices to be idle during most of the time. According to [87], the power consumption of HDD device when idle is 8.84 times than equivalent SSD-based device. However, the cost of SSD disks is double than HDD devices. Therefore, the dedicated datacenter will comprise of both SSD and HDD devices.

An optimal supportability requires the product software to integrate with the software that operates the IT equipment, the firmware. It enables an easy troubleshooting when identifying a faulty component. Also, the integration between the two software stack enable the self healing capability and auto remediation. Such integration with a COTS-based product is more complex and sometime impossible. Finally, such supportability will minimize human intervention when attempting to remediate an issue. When a local outage requires human intervention, the escalation is done after at least one automatic remediation steps. Achieving such remediation and repair capability requires an open interface across the entire software and firmware stack.

Based on the discussion above, we are going to introduce a new set of futures. Future in our case is an business reality that the cold storage service will help to fulfill. Let E denote the set of possible futures in operation cost. E_i denote a cumulative wight of the overall operational and capitalize costs of an alternative i e.g. COTS HDD, COTS SSD, and Custom. The set of futures were assessed by the required scale i.e. the sizing and other design alternatives tradeoffs. For simplicity we will use fibonacci numbers for the cost estimates to remove ambiguity between different futures and design alternatives.

Table 4.5: Operational costs based on business prediction that affects the cold-storage design-estimates are given in Fibonacci numbers to remove ambiguity. e.g. in case slow user-base growth and we use the expensive custom alternative, then the cost will be 1597 the 18th member in Fibonacci sequence. In case of sharp user growth and we used the HDD COTS-based alternative, we will spend more to address the supportability and the sustainability challenges. Therefore, 1597 was assigned for that case.

Future growth	Sharp	Moderate	Slow	Cumulative wight	Hurwicz rule $\alpha = 0.6$
HDD	1597	233	89	38	616
SSD	987	377	144	42	555
Custom	21	987	1597	49	492

The trivial decision approach will be to choose the E_i that complies with $min(E_i)$. However, the proposed alternatives included with certain risks, and uncertainty where no meaningful data are available. Hence, probability risks may occur. Systems research today suggests several decision making under uncertainty [30] (1) Laplace Criterion (2) Maximin and MaxiMax Criteria (3) Hurwicz Criterion. The Laplace Criterion assume a homogenous distribution of risks in which the average predicated cost will mark the optimal future. The Maximin and Maximax criteria suggests extreme risk probabilities. i.e. maximin is based on pessimistic view of the outcome of the set E. Maximax rule express the extreme optimistic of the risks. The two criteria helps to assess the upper and lower bounds of the various design alternatives.

Because previous criteria are based on extreme cases there is a need for another factor that will assess in the decision process. We will introduce another optimism or pessimism factors α among the extremes of previous criteria. This approach called the Hurwicz rule. The optimism index range is $0 \leq \alpha \leq 1$ so $\alpha = 0$ is the most pessimist asseuesnt that will converge with the maximin criteria and $\alpha = 1$ with the maximal criteria. Eq.4.3 depicts the Hurwicz rule we are going to apply on the future set E we gathered in previous design phases.

$$max\{\alpha[maxE_i] + (1-\alpha)[minE_i]\} \tag{4.3}$$

Applying the Hurwicz rule on Table.4.5 along with factoring the cumulative values that we calculated in the previous design phase yields the last column, Hurwicz rule with $\alpha = 0.6$ as our optimism factor. Based on the assumptions and analysis we made, the custom option is the preferred design approach for the cold-storage service.

4.0.6 Conclusions for Cloud Federation Computing Equipment

Cloud computing continues to reshape both large enterprises and small business throughout many industries. It drives the business needs by virtue of greater scale, agility and quicker time to market. Cloud computing creates new demands for more compute hardware equipment of general use. These demands drives transition from closed proprietary platforms to open platforms. The new open platforms will foster collaboration that will enable cloud operators to run optimized costs and with increased IT agility. Also, it removes the dependency in proprietary design and allows the design to be modified organically not only by the systems vendors but also the systems operators as well as the various cloud providers. We present a model that mimics the software open source paradigm and provides a metric for technical measurement of the hardware compute system design. The methodology continuously improves previous specifications based on both customer and operator needs along with evolving vendor constraints using Systems Engineering theory for mitigating certain risks, and uncertainty.

Chapter 5

Clean Energy Use for Cloud Federation Workloads

Over the past decade, cloud-based systems have been required to serve an increasing demand from users work flows and data. Cloud-based systems may be classified into two categories: *serving* systems and *analytical* systems. The former provides low-latency read or write access to data. For example, a user requests a web page to load online or requests video or audio streaming. The latter provides batch-like compute tasks that process the data offline that are later sourced to the serving systems. The service level objectives (SLO) for serving jobs are on the order of fractions of a second, while the SLO for analytical jobs are on the order of hours, sometimes days.

Today, public cloud service providers (CSP) attempt to process both of these workloads with a rich platform that guarantees cost and SLO to their clients. Cloud computing is an emerging infrastructure with limited regulation and compliance requirements [57]. Recently the Office of Management and Budget issued a Federal Data Center Optimization Initiative that promotes increasing use of Green Energy and increased utilization efficiency for all US Federal datacenters [68]. Specific target numbers are set for the end of fiscal year 2018. This publication addresses how those federal requirements may be attained and how federated cloud computing is a key enabler for attaining those performance targets.

Beginning in 2013, the US government initiated a carbon-tax on IT organizations to encourage major CSPs to pursue green energy opportunities for their datacenters operations [73]. US datacenters are projected to consume approximately 73 billion kWh by 2020 [73] with a corresponding increase greenhouse gases. Green energy generation growth is expected to triple by 2040 [65]. However, there is no cohesive system existing to coordinate the rising datacenter energy demand with rising green energy supply. This chapter utilizes Cloud-Federation as a multi-cloud resource coordination system that matches computational resource demands with available energy supply to maximize the utilization of green energy for processing cloud-workloads.

Most CSPs seek more market share in competition with other CSPs. One outcome of such competition is an ever-growing infrastructure in the form of new datacenters across the globe with no countervailing forces to meet user demand more efficiently and satisfy societal environmental and energy requirements. This sub optimum use of infrastructure increases the carbon footprint attributable to cloud computing services and also drives up costs to CSP's. The following sections investigate two types of workloads, serving and analytical systems. This chapter will focus on two workload types, On-demand streaming as a representation of *serving* systems. It will also investigate *analytical* systems and present a unique model that allows optimal clean energy usage.

5.0.1 Enabling Green Content Distribution Network by Cloud Orchestration

On-demand streaming constitutes up to 85% of Internet traffic consumption [46]. On-demand streaming content is managed and distributed by content service providers. It then cached and distributed by Content Delivery Networks(CDN) located at the edges of the Internet network close to the consumers. Because streaming constitutes such a large fraction of Internet resource consumption, this paper will, of necessity, focus on methods to employ green energy to better operate CDN instances of on-demand streaming jobs, which include both video and audio content.

Meeting the Federally mandated approach of maximizing the utilization of green energy to operate CDN instances (for government with recommendations for private sector use as well) requires an energy source-demand coupling scheme that insures SLO levels of power availability but is structurally biased towards green energy sources over hydrocarbon fueled energy sources. A system to accomplish this will have to provide seamless failover in the case of sudden interruption of green energy to grid-energy sources or vice versa i.e., fallback from grid-energy to green energy when surplus green energy is available.

User expectations in on-demand streaming requires different service level requirements than other serving systems workloads. Serving systems workloads are comprised of interactive sessions that pivots on minimum latency. However, low latency is less critical in analytical on-demand streaming since application clients use buffering techniques to mitigate long latency effects. Therefore, on-demand streaming workloads fits, more closely than interactive workloads, with the observed intermittent and varying green energy availability characteristics.

Green energy supply is unpredictable and requires a complex, adaptable, resource allocation system to provide CDN services with steady energy supplies while concurrently seeking minimal carbon footprint. This dynamic availability of green energy resources in a smart grid requires real time communication of both short term and predictive energy needs from cloud service providers to green energy providers. The green energy providers need to disclose availability dynamically to CSPs, who, in turn, disclose their changing energy demands for near term computing. SPs can then better maximize the use of green energy for on-demand streaming processing.

This is a classical resource management and coordination problem [51]. The following approach builds upon prior work that was done in this area [13, 50], specifically that done on alleviating the sudden lack of green energy to meet low-latency workloads. The approach herein employs an application-buffering scheme that better allows for opportunistic, green, on-demand streaming processing. It requires an extended, cohesive, federated system that aggregates supply and demand across multiple geographic locations employing the smart grid command and control infrastructure to achieve an optimal dynamic matching of green energy sources and computing loads.

This chapter proposes an implementation that utilizes a control component in a federated cloud that coordinates and optimizes the resource allocation among the participant CDN providers. It treats the volatile nature of green energy resources as a resource allocation problem, the solution of which is a resource orchestration system that is optimized with the goal to operate increasingly near to the limit of supply by green energy sources constrained by SLO reliability requirements. This system will be demonstrated by modeling a prototype that simulates resource allocation in a micro federated cloud eco-system to achieve an energy supply-computation demand match optimized within seconds.

Since the focus of this work is on green energy utilization in a federated cloud, the scheduling algorithms and resource management issues, while important, are discussed only to the extent necessary to help the reader understand the required architecture for heterogeneous energy compute clustering. This work is meant both as a case study in energy utilization, and a presentation of a novel method of coordinating high-velocity data streams, and extended to a unified orchestration system, to optimize the performance of federated cloud systems.

The chapter starts, with the on-demand streaming economics, increased green energy utilization and anticipated smart grid progressions as applied to on-demand streaming. Then it discusses the green energy utilization problem is analyzed. Finally, we present a cloud coordinator prototype that is built on Kubernetes [17], an open source cluster manger, and extend that prototype to discusses the need for and requirements of a unified system that orchestrates cluster compute resources in a federated cloud.

On-Demand Streaming and CDN

Over the last decade, video and audio traffic became the dominant segment of consumer internet traffic. Cloud service-providers such as Netflix, Amazon Instant and YouTube disrupted the prior linear TV data distribution model. Also, video streams delivered by mobile terminals grew as mobile connectivity improved [62]. Video streaming is expected to constitute up to 85% of Internet consumers traffic [46] within a few years. The US portion of video streaming is 14%[13] and the number of US Unique IPv4 connected addresses is 17% [62]. The streaming workload is comprised of live streaming and on-demand streaming, with the relative fractions of 6% and 94% respectively [46]. Other predictions support similar ratios, 12% live-streaming and 88% for on-demand.

A key driver for the rapid expansion of streaming video was the shift from specialized streaming protocols and infrastructures such as RTSP, and RTMP [55] to a simple HTTP progressive download protocol. This led to a shift from proprietary streaming appliances to commodity servers.

[13]Internet Statistics retrieved from https://www.statista.com/chart/2647/global-internet-usage-by-the-numbers/

In turn, this change removed a barrier for CDN's to process on-demand workloads. Most present day, CDN service providers support a seamless integration with cloud-based object storage that pipelines the digital content from the organization site to the CDN instance that runs at the Internet provider edge[14]. Furthermore, the HTTP chunk-based streaming protocol support in a CDN allows the client application sufficient time to detect the optimal CDN instance to handle user workload. The optimal CDN instance assignment is done by the cloud control plane resource manager. The prototype described below will demonstrate such optimal resource allocation.

We used server utilization and power metrics from [62, 73] to design the prototype. Most of these sources we considered have limited utilization rates and server utilization distribution. Also, utilization and power consumption do not scale linearly [7]. However, for clarity, in the interest of maintaining focus on the larger goal of the paper, CDN resource management systems for green energy utilization, we assumed linear relationships and accepted the risk of loss of accuracy in our estimates. High fidelity simulation accuracy is not critical for the goal of this chapter.

Energy Saving Potential in Operating a Distributed CDN Resource Management System

The approach is to aggregate the required traffic for on-demand workload processing, and use standard compute device specification to assess the electrical energy and carbon footprint that will be required by that workload[15].

The estimated data rate for streaming is given as $S_{total} = 63000PB/mo$ (where PB is Petabytes). Figure 5.1 shows users workload pattern of 9 busy hours in which the workload spans throughout 13 hours a day which yield $126PB/sec/mo$/ The on-demand streaming portion is estimated as 78% across four main US regions denoted by $k = 4$, the number of region used, for the purposes of this paper although k can be varied depending upon the degree of granularity desired in the simulations. $S_{on-demand}$ denote the on-demand portion.

[14]Cloud Front reference retrieved from https://aws.amazon.com/cloudfront/

[15]OpenCompute Project, Servers Specification guide retrieved from http://www.opencompute.org/wiki/Server

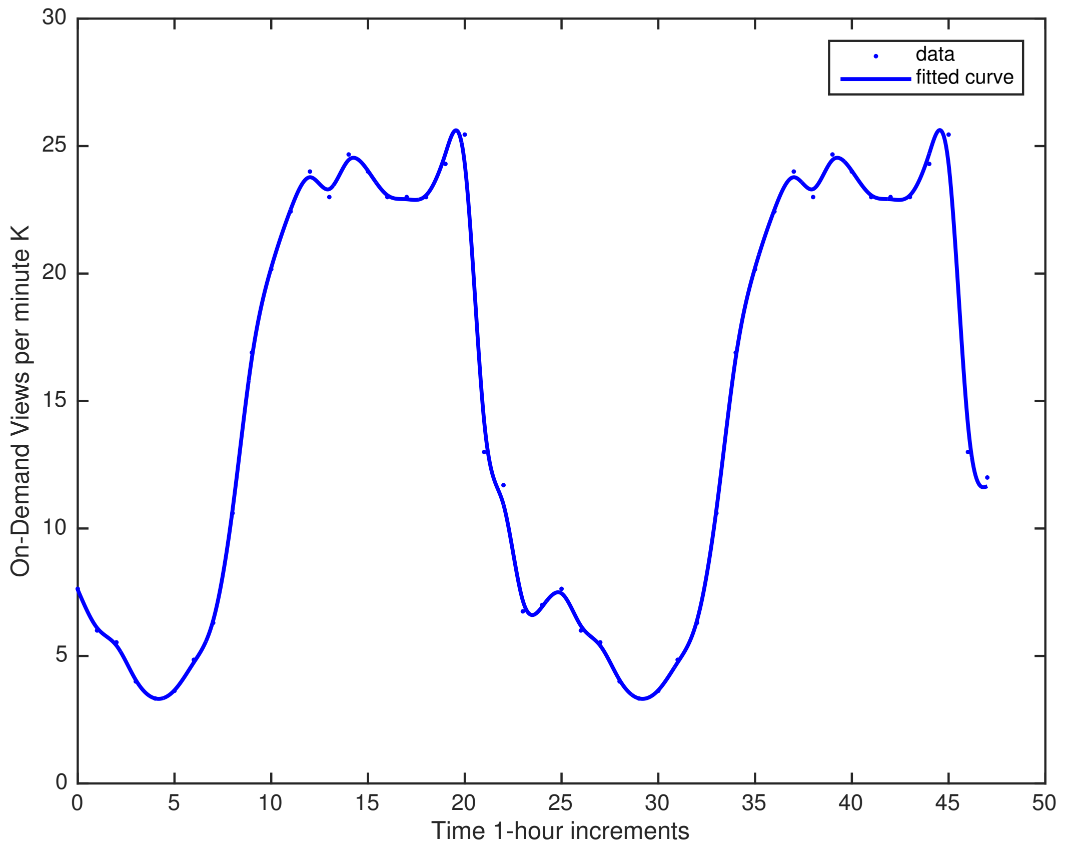


Figure 5.1: On-Demand Video views observed throughout 48 hours with 1-hour increments. Data was fitted with smoothingspline for curve and surface fitting [46]

$$S_{on-demand} = S_{total} \cdot 78\% = 49140PB/mo$$

$n_{hours/day}$ denotes the number of effective hours in a day for streaming.

$$n_{hours/day} = n_{busy} + k = 13$$

$$D_{rate} = \frac{S_{on-demand}}{n_{hours/day}} = 126PB/sec/mo$$

N_{max} denotes the estimated number of required servers in maximum CPU capacity. N_{max} is bounded by the maximum network throughput a single server can ingest. Standard commodity servers can handle up to 8.5Gbps i.e. 1.026GB/sec.

$$N_{max} = \frac{D_{rate}}{T_{max}} = \frac{126 \cdot 10^6 GB/sec}{1.026 GB/sec} = 118588235.3$$

u_{opt} denotes the CPU utilization factor so servers has sufficient capacity to handle management tasks. We estimate 60% utilization factor $u_{opt} = 100/60$.

$$N_{opt} = N_{max} \cdot u_{opt} = 118588235.3 \cdot \frac{100}{60} = 197647058.8$$

E_s denotes the midrange server energy consumption for various server types s. We consider three types of servers:(1) compute server(5kWh/server), (2) digital storage server(1.7kWh/server) and (3) network server(1kWh/server). Storage server acts the digital storage controller. The network server acts as the router and switch. The compute server is the server that processes the on-demand streaming.

$$E_y = N_{opt} \cdot \sum\nolimits_{s \in S} E_s \tag{5.1}$$

$$= N_{opt} \cdot (5kWh + 1.7kWh + 1kWh)$$

$$= 1.521 \cdot 10^6 kWh/mo = 18.26 GWh/y$$

The saving potential from running on-demand video streaming using green energy resources is 18.26GWh a year based on current on-demand consumption and expected to grow 89% by 2019 [16]. i.e. 34.5GWh per year for on-demand streaming. The next sections will explain the challenges in utilizing green energy followed by a method that that addresses some of these challenges and thereby maximizes the utilization of green energy.

5.0.2 Coordinating Green Clouds as Data-Intensive Computing

The following paragraph focuses on *analytical* systems workloads that typically comprise 48% of the cloud workload [3]. Also, some of the workloads patterns can be predicted as recurring

[16]https://www2.deloitte.com/content/dam/Deloitte/in/Documents/technology-media-telecommunications/in-tmt-rise-of-on-demand-content.pdf

jobs [33] and the deadlines for *analytical* jobs are more liberal by an order of magnitude than the *serving* jobs. Finally, one of the ways to handle offline workloads is to process the data streams offline through a highly scalable data-parallel frameworks like MapReduce by submitting jobs to a control plane [26].

This chapter proposes a component in a federated cloud that will optimize the resource allocation and coordination among the participant CSPs. It will also focus on addressing the time and power volatile nature of renewable energy resources as a fast date problem. It will present a resource management system with a goal to increase the utilization of data-centers, and to operate increasingly to the limit of supply by renewable energy sources. Finally, a prototype is built to simulate resources allocation in a micro federated cloud eco-system to achieve a supply demand optimized match within seconds.

Clean energy sources are time and power volatile and require complex resource allocation and coordination systems to maintain highly available data-center service with steady energy at a minimal carbon footprint. Further, availability of clean energy resources in a smart grid can be pushed from a variety of energy sources deployed across the nation as can the pull needs of data-service centers. Thus, CSPs will publish their compute resources availability, and service providers (SP), will publish their changing energy demands for near term computing. These streams of data include both high-volume and high-velocity characteristics termed a fast-data problem [48]. Our proposal uses as a base previous work that was done in this area [16, 50, 53]. However, previous work treated clean energy within a single data-center operated by a CSP. The proposal below suggests an extended cohesive system that aggregates supply and demand across multiple geographic locations employing the smart grid sense command and control to achieve an optimal match.

Optimum Clean Energy Utilization is a Fast-Data Problem!

This study suggests that efficient, clean energy utilization requires three consecutive steps: (1) Clean energy resource availability signals. (2) Exploration and analysis of prior years seasonal solar data and current weather reports and evolving green energy capacity availability planning.

(3) Acting fast enough based on the predictive analysis. The last step is the key component in solving the optimum allocation problem.

We judge that that clean energy utilization is different from the classic big-data problem such as the Hadoop MapReduce Method. Big-data solutions solve the case where the data is at rest, not fully consistent (aka eventual consistency) and require liberal SLO that is later provided as computing trends, and other business intelligence applications. However, our case handles data in motion that requires consistent, real-time aggregation, and transaction processing. That is per-event decision making using both real-time contextual and historical data as dual guides for proper algorithms. Finally, we argue that based on the clean energies' volatile nature, processing streams of compute demand and clean energy supply requires that a need for a compute resources can be addressed with a currently available demand.

5.0.3 Why is Green Energy Utilization Hard?

The following section describes why utilizing green energy for compute purposes, while a justifiable goal, is limited by SLO reliability. It will present a scenario where balancing time-varying energy generation patterns with changing dynamic energy demands of cloud computing sometimes conflict. The green energy time varying generation patterns considered by us focuses on wind and solar generation. Figure 5.4 show historical data on dynamic nature of green energy sources and Figure 5.5 shows the dynamic cloud energy demand.

Frequency Stability in Wind-Power Generation

The daily wind power variation characteristics will be employed as a metric that illustrates the duration and level for a given amount if wind energy availability. The electricity generation process from wind is comprised of a wind turbine extracting a kinetic energy from the air flow. The wind is rarely steady; it is influenced by the weather system and the ground surface conditions, which are often turbulent [27]. Also, the generation process must happen at the same instant it is consumed unless it is stored in grid level battery banks. Unfortunately, grid level energy storage technology is not keeping up with grid level energy generation technology [29].

Sample wind and power generation data were obtained from NREL [29]. We used datasets from 2006-2012 across different regions in the US and aggregated more than 600 observations. Finally, the data were fit using smoothing splines[17]. The usable power generated from a wind turbine is generally described by a Rayleigh distribution [27]. It defines three main points in the wind power generation process: (1) the cut-in is the minimum viable wind speed for electricity generation from a wind turbine. (2) the rated level, describes the point where the power reached its local maximal capacity without adverse effects on the turbine life by too strong a wind. (3) Cut-Off, is the term for the local minimum for the generation cycle. Beneath that speed, there is not enough power for viable electricity generation. Thus, if the wind velocity is too low, the data-server gets no wind energy. Figure 5.2 shows wind generation variations that crudely fit a Rayleigh distribution with $b = 300$ assuming the form of the Rayleigh Probability Distribution Function is:

$$f(x|b) = \frac{x}{b^2} e^{\frac{-x^2}{2b^2}}$$

The measured generation cycles range between 140 and 180 minutes per cycle. Equation 5.2 expresses the generated power by a wind turbine, given a wind velocity. The function g describes a viable electricity generation given a wind power. The wind power availability indications will be generated by a wind turbine and fed into the coordinator database as a potential power source to datacenters in a region. Our prototype will assume wind power availability indications as the wind tuple *{region,cut-in,rated,cut-off}* indications.

$$P_{output}(wind_v) = \begin{cases} 0 & : \text{ if } wind_v \leq rated \\ g(wind_v) & : \text{ if } rated < wind_v \leq c_{out} \\ 0 & : \text{ if } wind_v \leq c_{out} \end{cases} \tag{5.2}$$

[17]Matlab Smoothing Splines retrieved from https://www.mathworks.com/help/curvefit/smoothing-splines.html?requestedDomain=www.mathworks.com

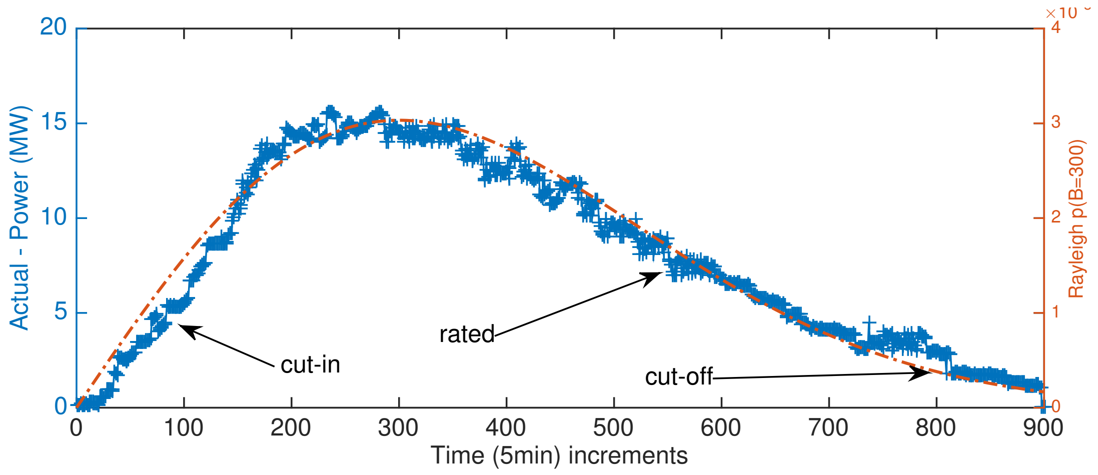


Figure 5.2: Aggregated wind power measurements between 2007-2012 that fits Rayleigh Distribution with b=300

Efficiency and Daily/Hourly Availability in Solar-Power Generation

Photovoltaic solar (PV) energy availability is defined by the solar power intensity denoted by s(*Watts/m2*), which varies with local daylight hours and the clear or cloudy sky conditions [29]. Moreover, the PV cells are most effective at lower temperatures [79]. The PV cells electrical power generation, defined by Equation 5.3, is a function of the solar intensity denoted by η_{solar}. Solar power generation also depends on the PV power efficiency denoted by s. It encapsulates both the predicated temperature,the sky conditions, the solar cell efficiency, and the DC to AC inverter efficiency. The solar cell area denoted by $a(m^2)$.

$$P_{output}(s) = \eta_{solar} \cdot s \cdot a \tag{5.3}$$

Our prototype will assume solar power availability indications as the solar-tuple *{region,power-efficiency}*. Based on the solar generation pattern presented in figure 5.3, the generation prediction utilizes the the local time in a region and the given power-efficiency

Optimum Green Energy Utilization for On-Demand Streaming is a Resource Management Problem!

This study suggests that efficient; green energy utilization for on-demand video streaming workloads has three main requirements: (1) efficient compute resource discovery, (2) efficient

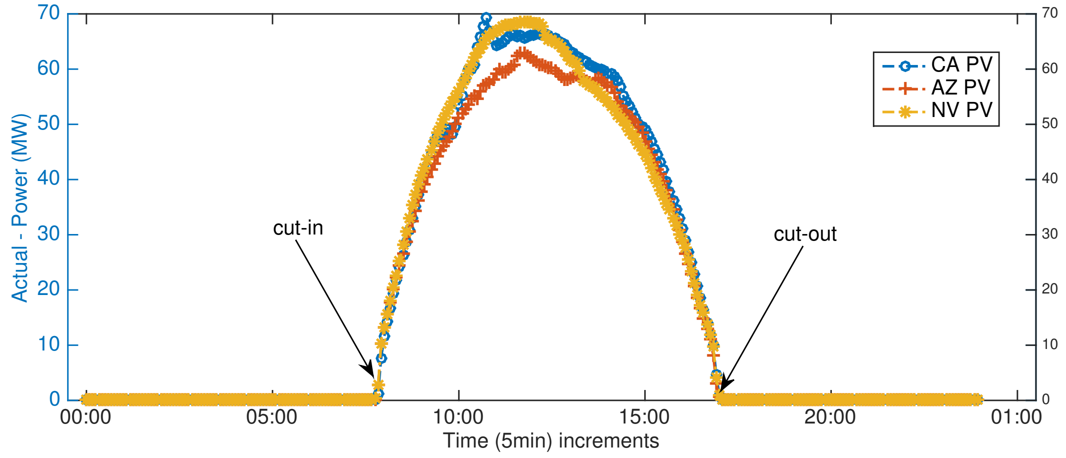


Figure 5.3: Aggregated solar power generation between 2008-2011 taken in Palm Springs CA, Prescott Airport CPV, AZ and Nevada Solar One, NV, indicating on a stable and fixed solar-based power [29]

load balancing among the provisioned compute resources and (3) smart failover mechanism that mask failover events from *green* CDN-edge instance to *grid* CDN-edge instance while end-users stream on-demand video [2]. These are discussed below.

Compute Resources Discovery. This assessment comprises both internal and external discovery. Internal discovery refers to CDN-edge instances that run in compute pods that must be able to be easily discovered and connected to control-plane endpoints consistently regardless of which cloud-service-provider is hosting the CDN-edge. External discovery refers to the ability of end-users discovering CDN-edge instances through DNS services for HTTP(S) on-demand video streaming.

Optimal Load Balancing is the seeking of the "best" CDN-edge, based on optimization criteria, for any given the workload processing. After initial discover and connection, clients should be served by the optimal instances based on proximity from the end-users, current load factor, and the availability of green energy resources. e.g., session requests originated from New Jersey should be served by US East as oppose to US West to avoid latency and signal loss.

Efficient Failover is a main component for on-demand video streaming based on green energy. If the endpoint becomes unavailable, in this case due to a sudden lack of green energy, the system must failover the client to another available endpoint that manages the streamed content.

Also, failover must be completely automatic i.e. the clients end of the connection remains intact, and the end-user oblivious to the failover event, which means that the end-user's client software requires no support handling failover events. Finally, multiple CDN-edge instances co-located in a region should be accessible by end-users through Domain Naming Service (DNS), as most clients-streaming (browsers) software supports DNS resolutions for finding available CDN-edges.

The green energy utilization model for processing on-demand video streaming is different than the classic scheduling problem where classical optimal resource allocation techniques are applied [8, 88]. We argue that based on the green energies' volatile nature and the on-demand video streaming workload characteristics, the optimal resource allocation approach should be opportunistic. It requires an effective resource management system for processing on-demand video streaming workloads. Our prototype will employ a Kubernetes flavor "Ubernetes" that implements the three main requirements above.

5.1 Clean Energy Mix Evaluation for Online On-Demand System

In the following section we evaluate a compute load coordination system component that harmonizes *on-demand streaming* job demands with available compute resources, with priority given to those powered by green energy sources. Such resources will be published to the coordination system through a resource availability tuple *{region,cut-in,rated,cut-off,power-efficiency}*, where *region* indicates the geographic availability region. *power-efficiency* indicates solar or wind based energy power efficiency. *cut-in, rated* and *cut-off* the values appropriate to those energies.

On-demand streaming job demand includes the specific *region, total-job workload, load-factor*, as well as *contract deadline* SLA. The *load-factor* indicates the required number of CPU cores per the *total-job-workload*. The geographic *region* indication will be used to optimize the match between the supply and demand. Also, the *total-job-workload* and the *deadline* will be compared against the *cut-in,rated, cutoff* time for wind or *power-efficiency* for solar, based on the published *load-factor*.

We suggest a hybrid datacenter that does not deviate from the common datacenter architecture. The core difference lies on an automatic transfer switch (ATS) that switches between different available power sources: generator, grid or green energy when available. In both cases the datacenter design does not change and requires incremental changes only by adding green energy power sources to the datacenter's ATS's (Figure 4.1).

We suggest two types of compute clusters, green-clusters powered by green energy and grid-clusters powered by the electrical grid. Figure 4.1 shows a simplified datacenter power distribution that supports green energy sources. In such datacenter, both *serving* and *analytical* systems deployed in grid clusters. Further, for incoming *analytical* workloads, few clusters use green resources when there are a viable green energy and mostly standby. As a mitigation strategy, a compute live migration procedure will be available in case of unpredicted lack of renewable resources during a workload processing which presents a risk for SLO violation.

5.1.1 Experiment Planning

Below is simulation of a cross-regional platform that is comprised of control-plane, workload-plane and coordinating components. This will be embodied in a resource allocation system (Kubernetes). This system will: (1) provision resources to be neared users; (2) optimize utilization by prioritizing the use of underutilized resources; and (3) seamlessly remove malfunctioning hardware from the system. The control-plane will enable an effective compute resource provisioning system that spans across different public cloud providers and regions. The coordinating components will accept user-workload demands as well as green energy availability from various regions and opportunistically seek to process streaming workloads using compute resources provisioned by green energy resources. The workload-plane will be comprised of edge streaming servers that process the end-user on-demand video streaming. It will built of standard Apache HTTP[18] servers that runs on the edge location.

[18]Apache Web Server reference retrieved from https://httpd.apache.org

The control-plane software infrastructure is based on Kubernetes [17], it facilitates internal discovery between CDN instances so instances can connect across different cloud boundaries and regions. Further, end-users can discover the optimal CDN-edges that are (1) nearby, (2) less loaded and (3) healthy. Finally, the Kubernetes automation framework allows the failover mechanism with no dependency upon the end-user client. In particular, we will exploit the *livenessProbe* option that automatically removes green-compute pods, a set of CDN-edge instances, in case of a sudden lack of green energy.

The coordinator component accepts incoming supply and demand traffic, calculates a potential match, within minutes, and notifies back the CSP and the SP for transaction completion. We use Redis[19] as the in-memory data store as the database that stores the system supply and demand calls originated by the end-user workload. The workload is generated by Jmeter instances[20]. The workload generated based on the on-demand video views observed by [62] depicted in Figure 5.1. Green energy availability simulated based on the known regional patterns depicted by Figure 5.2 and Figure 5.3.

We count the number of matches i.e. on-demand video streaming processed by CDN-edge instances operating on green energy. Also, we measure the false-positive cases where a match was suggested but did not met the SLO's deadline due to a violation that caused by a sudden lack of green energy resources. We use the data to extrapolate the possible energy (kWh) that could be generated by the using green CDN-edge instances depicts in Equation 5.1.

5.1.2 Execution - The Preparation

The prototype experiment included the setup of three virtual datacenters deployed in different regions: (1) Central US, (2) West US and (3) East US. The clusters were sized based on US population distribution[21] by regions i.e. 20% for West US, 40% for East US and 40% Central US.

[19]http://redis.io

[20]http://jmeter.apache.org

[21]US Population Distribution retrieved from https://www.census.gov/popclock/data

The cluster sizes for West US, Central US, and East US are 3, 7 and 7 machines respectively. Each machine is standard 2-CPU cores with 7.5GB of memory. Also, the user demand simulation will rely on the US population distribution. Finally, the green energy supply simulation will be based on wind or solar availability observed in the various regions.

The control-plane is comprised of docker API server and controller-manager. The controller coordinator component will need to allocate resources across several geographic regions different cloud providers. The API server will run a new federation namespace dedicated for the experiment in a manner such that resources are provisioned under a single system. Since the single system may expose external IPs it needs to be protected by an appropriate level of asynchronous encryption[22]. For simplicity, we use a single cloud provider, Google Container Engine, as it provides a multi-zone production-grade compute orchestration system. The compute instances that process the user workloads are deployed as Docker containers that run Ubuntu 15 loaded with Apache HTTP server. For simplicity, we avoid content distribution by embedding the video content to be streamed in the Docker image. We run 52 Docker containers that span across the three regions and act as CDN-edges. Green CDN-edge instances differ from grid CDN-edge instances by Kubernetes labeling. The simulation of the hybrid datacenter is depicted in Figure 4.1.

A coordination database system that aggregates green energy, solar or wind, availability, was built in software. When energy sources manifest the cut-in patterns depicted by Equation 5.2 and Equation 5.3, the coordination system starts green CDN edges in the availability regions. Also, when green energy availability reaches cut-off rates, the coordination system turns off green CDN edge instances.

5.1.3 Baseline and Variability of Workloads

The baseline execution included data populations of both green energy availability and user demand for for video streaming. The data population was achieved by the Kubernetes-based Jmeter batch jobs. The loader jobs goal is to populate the coordinator database with green energy supply

[22]Simulation code and data retrieved from https://github.com/yahavb/green-content-delivery-network

based on using a Weibull distribution, which is a generalization of the Rayleigh distribution described above for wind and a normal distribution for solar. Also, the user demand was populated according to the observed empirical patterns depicted by Figure 5.1.

We simulated the availability and unavailability of green energy using Jmeter-based workload plan against the coordination system. Our implementation starts green CDN-edge instances opportunistically upon green energy availability. Once a CDN-edge instance declares its availability it processes live workloads.

We use the Kubernetes *livenessProbe* for communication between CDN-edge instance pool and its load-balancer that divert traffic to its pool members. Finally, another workload Jmeter-based simulator generates on-demand streaming calls. This workload simulates end-user demand. It includes HTTP progressive download calls to pre-deployed video media in the CDN-edges.

5.1.4 Main Execution

In each of the three regional CDN-edge clusters the Kubernetes Jmeter batch jobs that generated green availability traffic to the coordination component were executed. The simulation is comprised of availability indication that are based on Figure 5.2 and Figure 5.3. We randomized solar production by using a factor of $\alpha = 0.2$ based on collected data between 2008-2011 in Palm Springs CA, Prescott Airport CPV, AZ and Nevada Solar One, NV [73]. Also, we randomized the wind production by a factor of $\beta = 0.4$ based on collected data between the years 2007-2012 [29]. The demand simulations included a set of calls to the coordinator component spread across 48 hours. The calls originated from three different timezones. The supply simulations consist of wind and solar-based energy time and power windows.

The experiment executions generated two main data traces that we used for the resulting generation computation. The first trace is the simulators logs. The simulator logs includes the demand and supply records. Demand records stored in the Redis key-value store under the key "DemandEvents" followed by timestamp, region and the required compute capacity. The supply calls were stored in the Redis key-value store under the key "SupplyEvents" follows by timestamp, region and supply phase i.e. cut_{in}, cut_{out} or $rated$. For query simplicity the loader ingested three types of records for each supply and demand the by the keys: (1) supply or demand (2) timestamp, and (3) by region. This approach optimized the coordinator queries by timestamp and regions for green CDN-edge instances allocation.

The second trace is the actual allocation logs. It is generated by the coordination system that invokes the Kuberenetes command for green CDN-edge instances initialization and disposal. This was used to determine the green energy utility translated into energy (kWh) that did not use grid energy sources.

Limitations

Every supply and demand was recorded three times to ease the query process. This approach was used since Redis provided limited query abilities by different keys. This approach might suffer data inconsistency issues where a supply metric was successfully committed to one key recode but missing on other key. Production systems should add extra safety gates when ingesting data. We used Redis because of its popularity in the Kubentese community. However, our approach is not limited to Redis or other database systems for that matter.

When measuring the green energy overall utility, we used the container initialization and disposal as indication that green energy utility was used. Specifically, we used the *'kubectl logs POD'* command based on the assumption that the coordination system invocation commands are tightly coupled with green energy availability. It is likely that collecting the actual video streaming traces through the various Apache access logs of the CDN-edges will be more accurate.

In the case of a sudden lack of green energy while streaming video a failover occurs. Such failover event relies on domain naming services (DNS), the impact of DNS caching was not included since that might cause streaming delay on the user side. Also, when the coordinator algorithm determines there is enough green energy available it will take grid pods down and activate green pods up in a controlled fashion e.g., one at a time so that no requests are lost during the transition phase. For simplicity, the algorithm avoid that.

5.1.5 Analysis

The green energy supply simulation plotted in Figure 5.4 shows the energy generation in MW for both wind and solar sources. The simulated amounts were adjusted to the amount observed in the traces between 2007-2012 [29].

The user workload simulation plotted in Figure 5.4 follows the observed user patterns depicts in Figure 5.1. Also, it shows the aggregated green energy availability for each region. The cloud-coordinator uses these data sets to determine if there is enough green energy available before provisioning green-pods and possibly taking grid-pods down. The utility of the green energy was

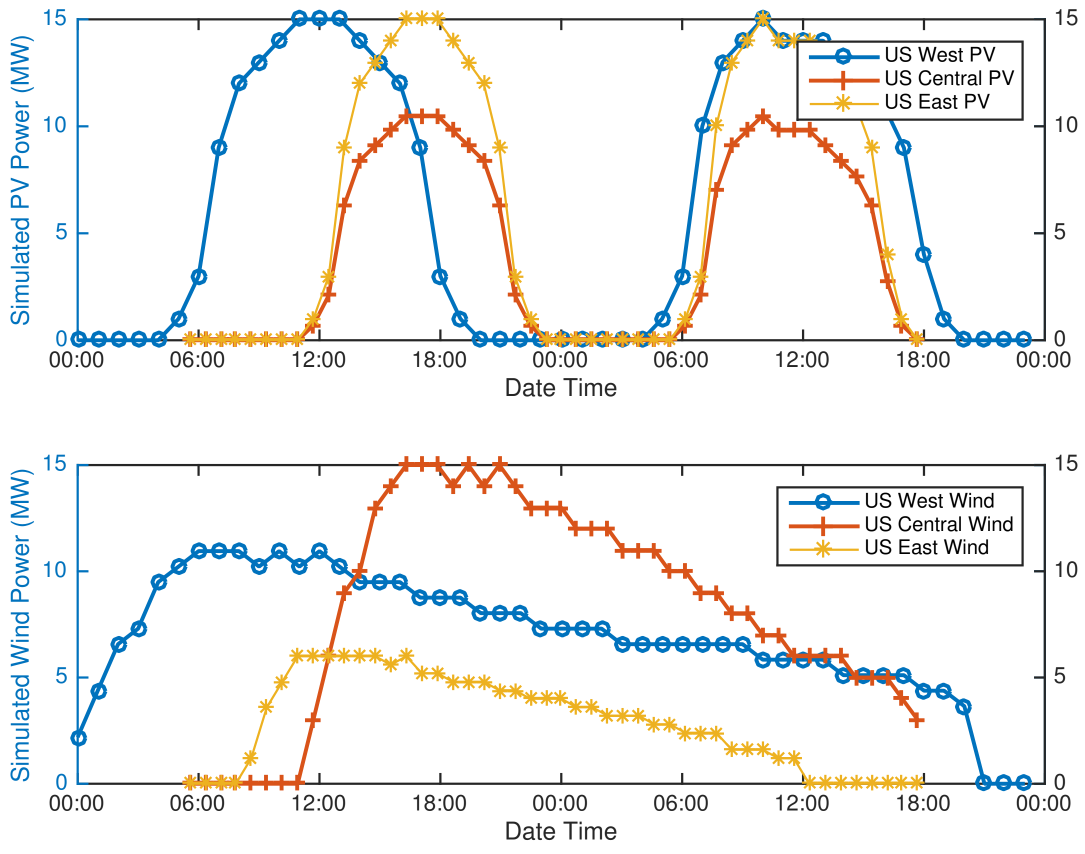


Figure 5.4: Green energy availability simulated in MW across three regions. Amounts are adjusted to NERL measurements i.e., wind generation in West US moderate, Central US outstanding, East US fair. For solar generation in West US strong, Central US moderate-high, and East US low.

calculated based on the cases where sufficient green energy was available to run the green-CDN pods within the same region. Otherwise cross-regional latencies might degrade the on-demand video experience. The measurements in Figure 5.5 were adjusted to the estimates of required energy (kWh) for operating the green compute pods. The case where there was negative green energy available it was considered as a miss in the overall utility reckoning.

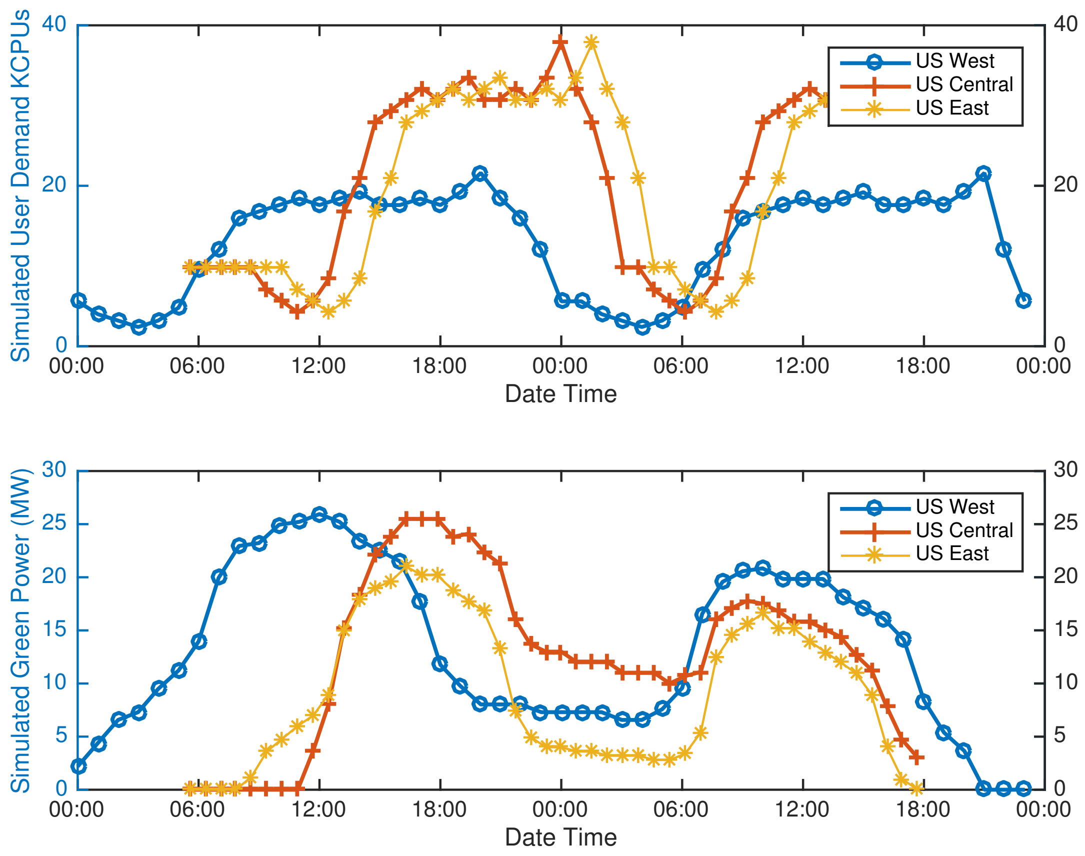


Figure 5.5: Video on-demand user workload per region adjusted with user population opposed to aggregated green energy availability, solar and wind energy.

5.1.6 Discussion on Energy Mix when Processing CDN Workloads

The scenario described above simulated the usage of green 801.3 kWh out of total 3642 kWh to process the video on-demand streaming workload. i.e. 22% by opportunistic matching. When counting the utility per region West US used 42% of the green energy. Central US used 28% of the simulated green energy. East US utilized only 18% as the initial ratio between user demand and green energy availability was relatively low. Although West US reached 40% utilization it contributed nationally only to the 20% portion it contains from the entire experiment test set. By way of comparison, Jeff Barr of AWS noted that their data centers utilize a 28% cleaner power mix[23]. Extrapolating the simulation results to the initial assessment in Equation 5.1 yields to a saving of:

$$(18.26(GWh) \cdot 1.89) \cdot 0.1(\$/kWh) \cdot 22\% = \$759,250.8/year$$

5.2 Clean Energy Mix Evaluation for Offline System

In the following section we evaluate a coordination component that harmonizes *analytics* jobs demands with available compute resources powered by green energy resources. Such resources will be published to the coordination system through a resource availability tuple *{region,cut-in,rated,cut-off,power-efficiency}*, where *region* and *power-efficiency* indicates solar or wind based energy and *region, cut-in, rated* and *cut-off* of those energies.

Analytics job demand includes the specific region, total-job workload, load-factor, as well as contract deadline stermed tuple. The *load-factor* indicates the required number of CPU cores per the *total-job-workload*. The region indication will optimize the match between the supply and demand. Also, the *total-job-workload* and the *deadline* will be checked against the *cut-in-rated, cutoff* time for wind or *power-efficiency* for solar, based on the published *load-factor*.

We will suggest a hybrid data center structure that does not deviate from the common data-center architecture. The core difference lies on an automatic transfer switch (ATS) that switches between different available power sources: generator, grid or clean-energy when available. In both cases the data-center design does not change and requires incremental changes only by adding clean-energy power sources to the datacenter's ATS's (Figure 4.1).

[23]Amazon Web Services Sustainability reference retrieved from https://aws.amazon.com/about-aws/sustainability/

Figure 4.1 shows a simplified data-center power distribution that supports clean energy sources. In a data-center with available clean-energy resources for both *serving* and *analytical* systems deployed in brown clusters. Further, few clusters use green resources when there are a viable clean energy and standby for incoming *analytical* workloads. As a mitigation strategy, a compute live migration procedure will be available in case of unpredicted lack of renewable resources during a workload processing with a risk for SLO violation.

5.2.1 Experiment Planning

We wish to simulate an isolated group of computing resources so it can operate by various energy resources, especially available green ones. We use a group of leased resources from existing cloud providers to form a virtual-data-centers set that operates in a federated scheme. Each virtual-data-center includes with internal arbitrator component that collects and aggregates internal signals about its utilization and availability. The arbitrator then reports to the central coordination system. We use Apache Mesos[24] for the virtual-data-center abstraction.

Also, we build a highly available coordination component that accepts incoming supply and demand traffic, calculates a potential match, within minutes, and notifies back the cloud-service-provider and the service-provider for transaction completion. We use VoltDB[25] as the database and application server for the coordinator component.

Finally, we simulate customer's demand for compute resources through client simulator with a Java-based application that generates pseudo demand traffic to the coordinator service. The coordinator service will run on separate resources pool than the virtual-data-centers and the client simulator.(Figure 5.6)

The data collected includes customer workload for processing and jobs deadline. Also, the number of matches the coordination component found and processes by green clusters without SLO. Finally, we measure the false-positive cases where a match was suggested but not met the

[24]http://mesos.apache.org

[25]http://voltdb.com

SLO's deadline due to a violation that caused by a sudden lack of clean energy resources. We use the data to extrapolate the possible carbon-footprint that could be generated by the used virtual clusters.

The research goal is to show a significant improvement in the carbon emission generation by data-centers. Let $MtCO_2e$ denote the carbon emission. Electrical usage can be consumed by the $Consumers$ set: {cooling, storage, servers, CPU, power} systems and denoted by EU measured in *kWh*. The average regional carbon dioxide emissions measured in lbs/*kWh* and denoted by RE_{co_2}. Therefore, the total electrical usage is:

$$EU_{Total}(kWh) = \sum\nolimits_{c_i \in Consumers} EU(c_i) \tag{5.4}$$

and the carbon footprint generated by the workload is:

$$MtCO_2e = \frac{EU_{Total}(kWh) \cdot RE_{co_2}(lbs/kWh)}{2,204.6(lbs)} \tag{5.5}$$

As the experiment uses virtual data-centers, we do not have access to the power consumption by the cooling, power and storage system. The results capture compute jobs durations measure in server $\frac{CPUcore}{time}$ and use the average Thermal Design Power (TDP) of 200W per core (0.2kW). c_i and t_i denotes the number of cores and time used per job respectively. The electric usage by server ($EU_{servers}$) will be the sum of the electric utilization of the executed jobs that was matched by the coordinator component (Equation 5.6)

$$EU_{servers}(kWh) = \sum_{i=1}^{jobs} c_i \cdot 0.2 \cdot t_i \tag{5.6}$$

We assume a linear relation between the power utilization of the cooling, power and storage systems with the servers power utilization. (Equation 5.4)

5.2.2 Execution - The Preparation

The Preparation for the execution included the setup of the three virtual data-centers each with 6 machines D4 series with 8 Intel(R) Xeon(R) CPU E5-2660 0 @ 2.20GHz cores, 10Gbps NICs, 28 GB memory and run Ubuntu 15.0. Each virtual data-center is located in a different geo-location. The virtual data-centers were simulated by Mesosphere cluster with three jobtracker(Mesos masters) and three tasktracker(Mesos slaves). We installed Hadoop Cloudera CDH 4.2.1-MR1 on the takstrackers. The Coordinator Database run on a separate resource pool with 1 machine D series similar to the virtual data-centers specification. Finally, the loaders run on an 3 D2 series machines with 2 cores Intel(R) Xeon(R) CPU E5-2660 0 @ 2.20GHz cores, 10Gbps NICs, 7 GB memory and run Ubuntu 15.0. Apache Hadoop ships with a pre-built sample app, the ubiquitous WordCount example. The input data file was created using $/dev/urandom$ on the takstrackers hosts[26]. The input data file was copied to the HDFS directory that was created as part of the Hadoop preparation (/user/foo/data).

Anticipated Required Deviations from the original job plan. The original plan was to simulate the scenario where the customer keeps his data at a different location than it might be processed. We plan on using a job migration scheme that was originally designed for workload migration across different geo-location[16]. The suggested method optimizes the bandwidth costs of transferring application state and data over the wide-area network. Our experiment generated the data file at the loader host and did not include the job migration. We believe that including the job migration aspect could impact the presented results. However, the job migrations proven efficiency and later studies minimize that deviation.

5.2.3 Baseline and Variability of Workloads

Baseline. The execution baseline included a load that runs without the coordinator component i.e. loaders generated load to the virtual data-centers resources for 48 hours. The load scenario included a single file generation that was submitted to one of the tasktrackers. The output of each

[26]https://github.com/yahavb/GreenCloudCoordination/

executed jobs included the CPU time spent for each execution. The data collection included the execution log of the command: Hadoop jar loader.jar wordcount /user/foo/data /user/foo/out.

Variability of Workloads. The workload comprises of data files with words that need to be counted using the Hadoop WordCount. The load complexity depends on two factors, the file randomness level, and its size. We rely on the native operating system randomness, and our virtual data-centers and loaders are homogeneous. Therefore, the size is the remaining factor for differentiating workload types. We evaluated the federated-cloud coordinator by generating three load types simultaneously. The three types intend to cover the following cases (1) A match was found between workload and sufficient green energy resources. (2) A match was found, but there was not enough power to complete the job with no SLO deadline violation. (3) Like the former but with SLO deadline violation. The three types will be uniquely distributed across wind and solar based virtual data-centers.

5.2.4 Main Execution Issues

In each jobtracker host in a virtual data-center, we executed a simulator that generated green availability traffic to the coordination component. The simulation comprises of availability indication that are based on figure 5.2 and figure 5.3. We randomized solar production by using a missing factor of $\alpha = 0.2$ based on collected data between 2008-2011 in Palm Springs CA, Prescott Airport CPV, AZ and Nevada Solar One, NV [29]. Also, we randomized the wind production by a missing factor of $\beta = 0.4$ based on collected data between the years 2007-2012 [29]. The demand simulations included a set of calls to the coordinator component spread across 48 hours triggered by a Monte Carlo simulation inspired by [32]. We built a Rayleigh-based distribution model in the Monte Carlo simulation using Matlab Statistics and Machine Learning Toolbox[27]. The calls originated from three different timezones. Each call comprises of the tuple {region,total-job-workload, load-factor, contract deadlines}. The supply simulators are comprised of the wind and solar-based energy time and power windows. Supply call to the coordinator includes the tuple {region,cut-

[27]https://www.mathworks.com/products/statistics.html

in,rated,cut-off,power-efficiency}. The coordinator runs a temporal stored procedure that find a match between the supply and demand data. When a match found, the coordinator generates an assignment call to the jobtracker in the corresponded virtual data-center to execute the jobs. The execution will fetch the data to be processed and report temporal statuses to the coordinator. Every status call generates a check against the current demand levels in the particular region. If the demand changed, and the request cannot be fulfilled, the job considered as false-positive. If the jobs are completed successfully, the request counted as success along with the Core/hour saved.

5.2.5 Analysis

The experiment executions generated three core data logs that we used for the result generation. The first data set is the simulators logs. The simulator logs comprise of the demand records. Demand records stored in the Demand VoltDB table. The supply calls were stored in the Supply VoltDB table. The match transactions stored in the Jobs VoltDB table with status and the number of cores/hour used for the job. Status values can be Success or False-Positive. Figure 5.8 shows the two supply indications. Both signals were generated based on Equation 5.2 and Equation 5.3. The demand data was generated based on known usage patterns that are spread across three timezones.

5.2.6 Discussion on Energy Mix when Processing Offline Workloads

The experiment simulated the usage of green 708 *kWh* out of total required 3,252 *kWh* for *analytical* systems workload processing i.e. 22% less carbon emission (Equation 5.5). 1822 *kWh*, 50% of the total workload consumption, was processed by brown energy because of false-positive events i.e. the coordinator assigned a job with no sufficient green recourses to process the job. We believe that optimizing the coordinator algorithm can improve the footprint reduction up to 50%.

Figure 5.9 shows the job placement ranges denoted by the dotted line in both the PV and the Wind plots. Any values above the zero levels indicate on potential benefits. However, such cases are subject to false positive events that can occur when a unpredicted drop in the availability. Further, such cases utilize the hybrid data center power scheme describes in figure 4.1.

5.2.7 Conclusions for Clean Energy Use in Federated Cloud

The future growth of cloud computing will increase its energy consumption as a fraction of grid power and will cause a significant addition to the ever growing carbon emission since 70% of US power is generated by hydrocarbon fired power plants. Using rapidly emerging green energy for processing cloud computing workloads can limit the anticipated carbon emission growth. However, balancing time varying green energy utilization with time varying energy demands of cloud computing is a complex task that requires sophisticated command and control prediction algorithms beyond the scope of this paper but are emerging in the form of a smart grid system of systems [10]. Our study shows that green energy utilization for on-demand streaming workload is best described as a resource management problem. The solution presented demonstrates real time balance of green resource supply and cloud computing workload demand and utilizes Ubernetes an open source container cluster manager. The results approximate within 21% those observed in a single cloud instance in the field. Our study also shows that green energy utilization for offline workload processing is a fast data problem. It's solution best utilizes VoltDB, an in-memory database, to allow near-time response for green resources supply and workload demand. Our future work will focus on optimizing the false-positive ratio, to further reducing the cloud computing carbon footprint by up to 50%.

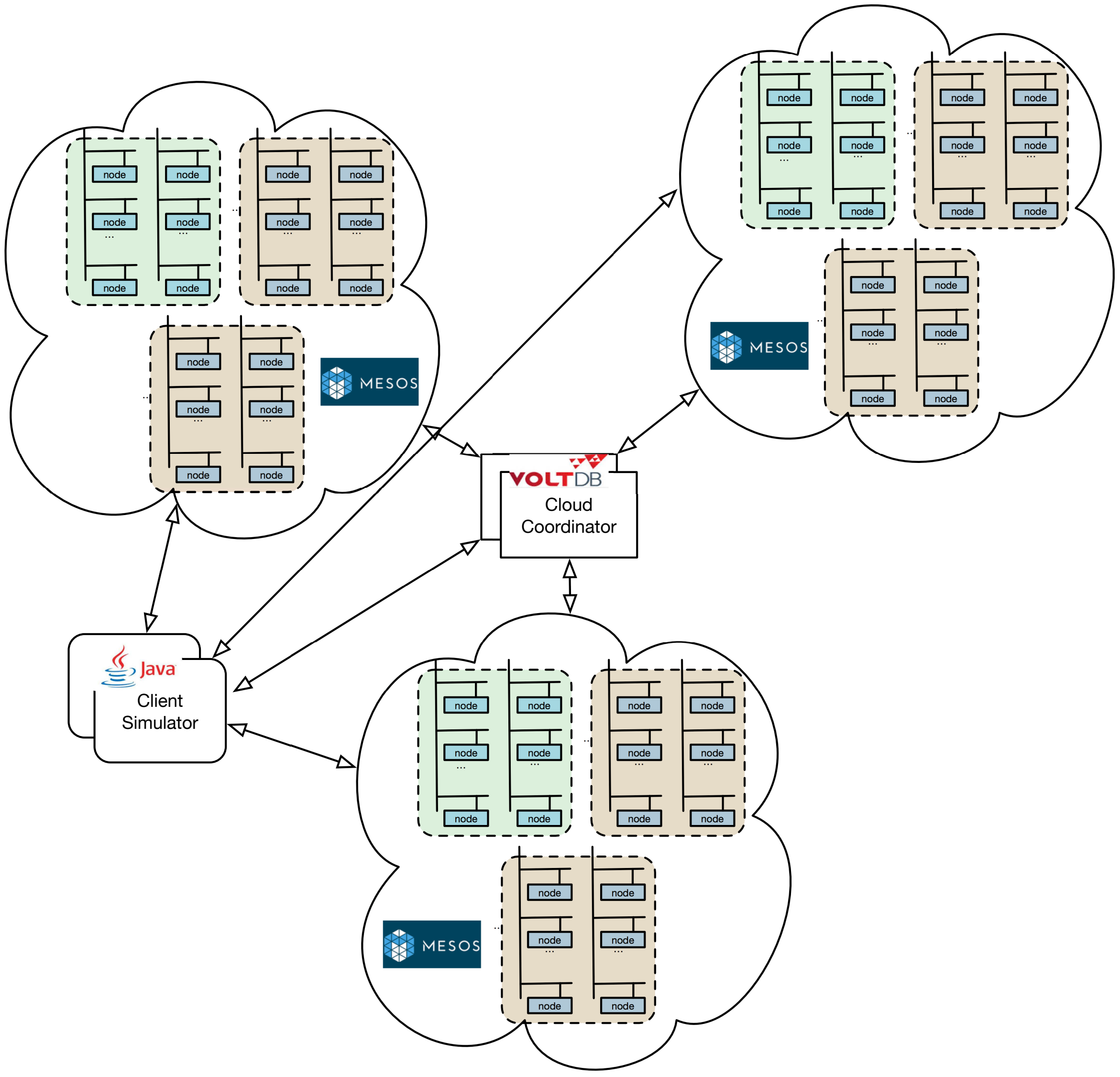


Figure 5.6: Experiment Architecture - Three virtual clusters deployed on a public cloud services located in a different regions grouped by Apache Mesos. A single VoltDB instance as the coordinator service. Finally, Client simulator that generates pseudo demand that sends data to process

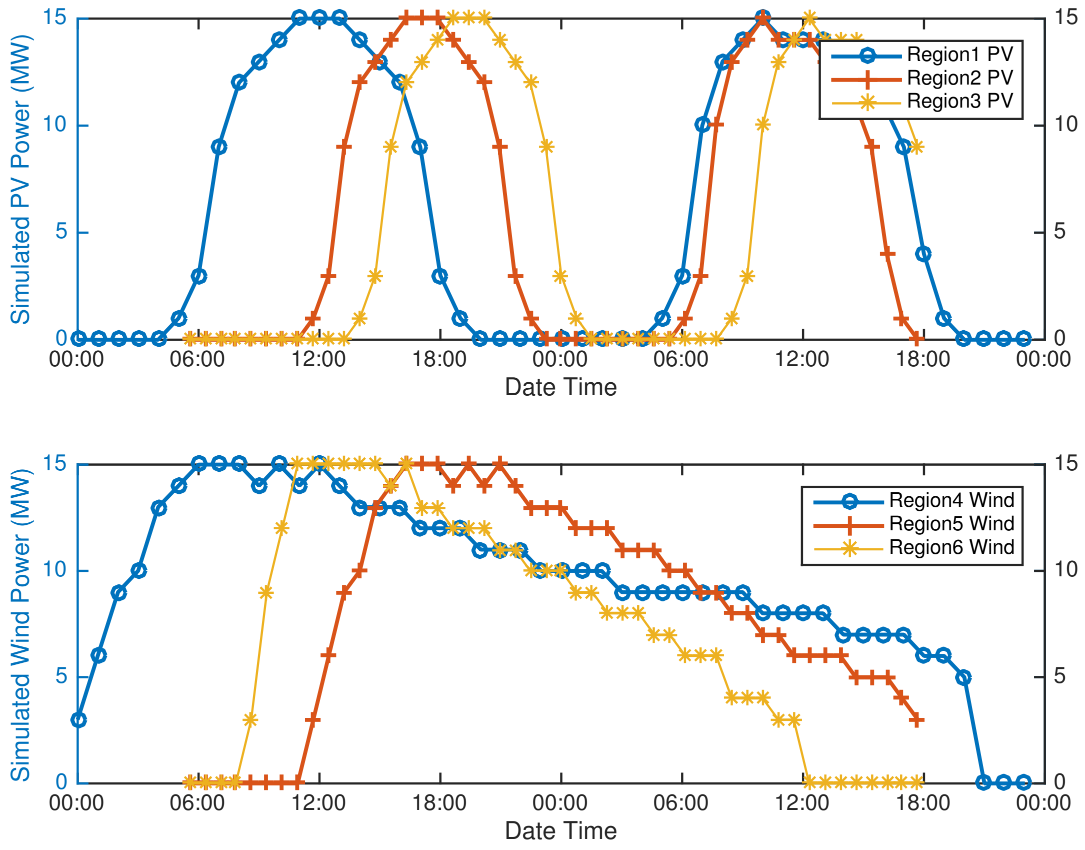


Figure 5.7: Green Energy availability simulated across six different timezones, three for PV power and the bottom three for wind. These indications are fed, in near-time, to the coordinator database that runs a temporal stored procedure that seeks for match with pending registered jobs.

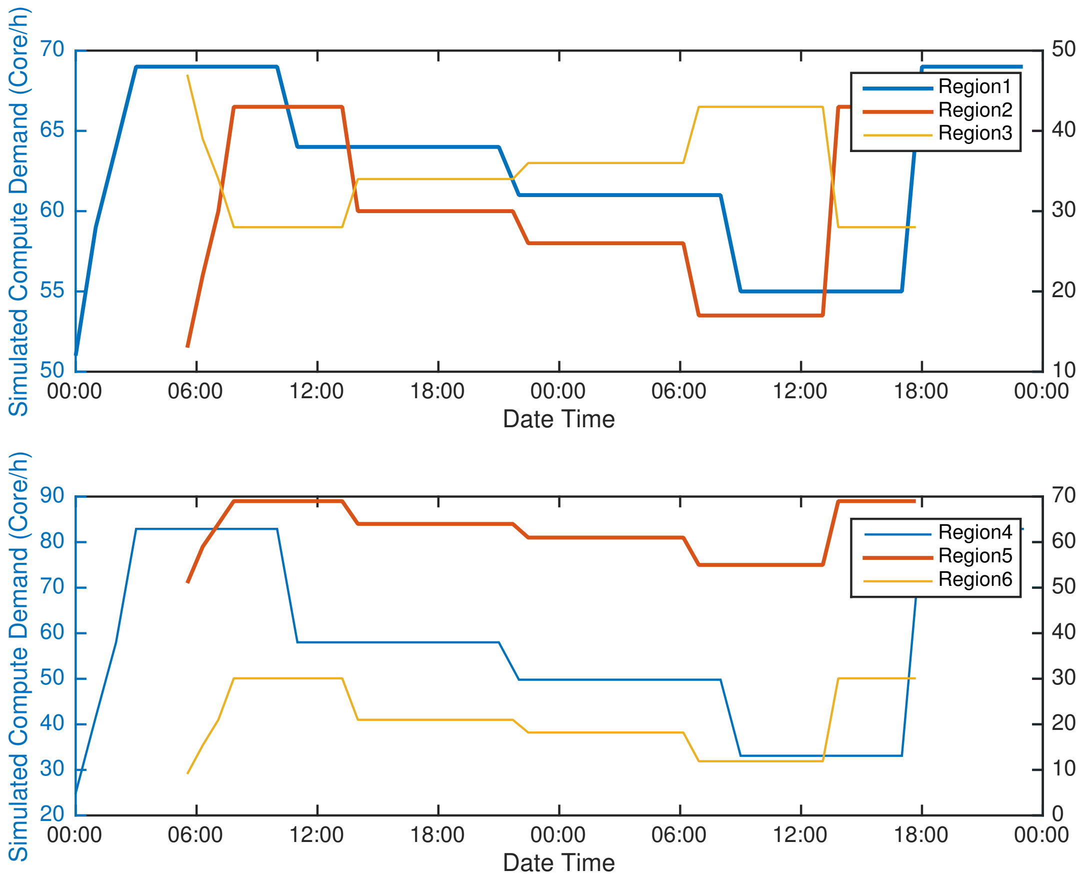


Figure 5.8: A Monte-Carlo-based simulated compute demand across six different timezones, three for PV power and the bottom three for wind. These indications are fed, in near-time, to the coordinator database that runs a temporal stored procedure that seeks for match with available green energy.

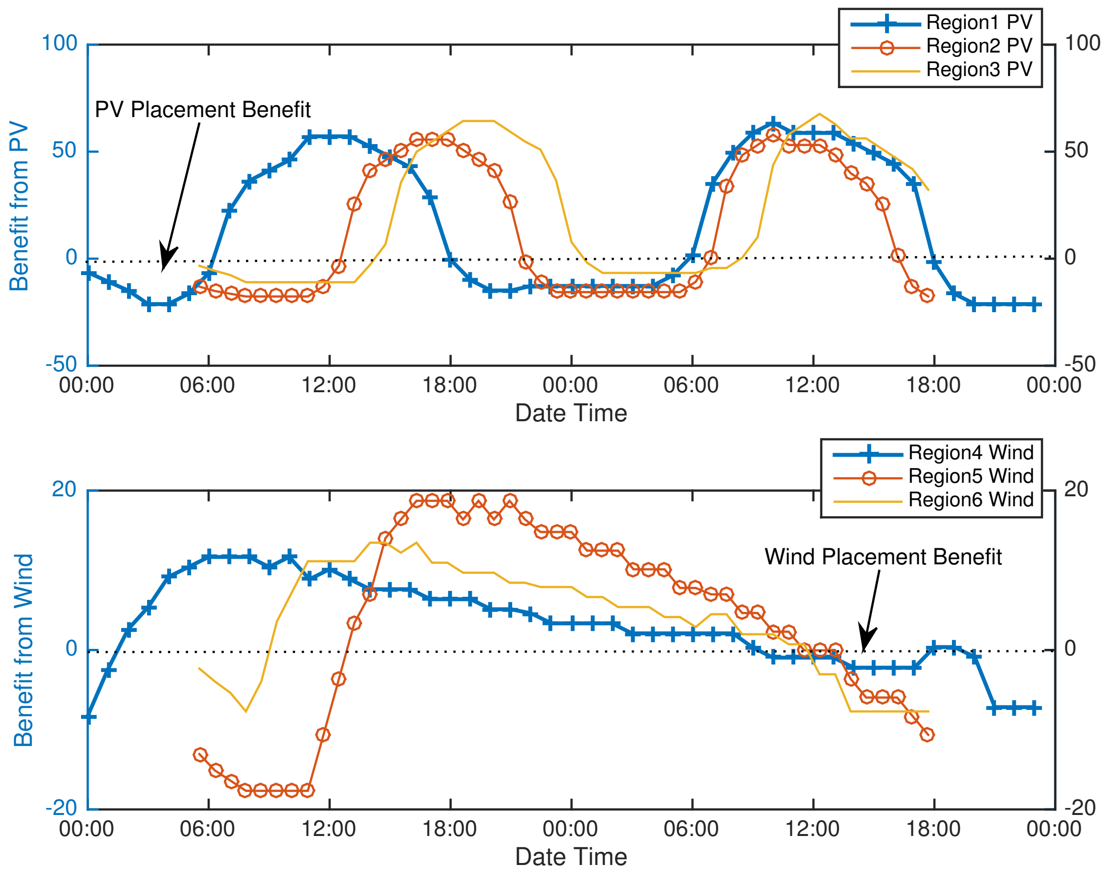


Figure 5.9: Job placement ranges denoted by the doted line in both the PV and the Wind plots. Any values above the zero levels indicate on potential benefits. However, assigned jobs might not be able to fully processed when unpredicted drop in the availability.

Chapter 6

Semantic-less Breach Detection of Polymorphic Malware in Federated Cloud

As an increasing fraction of computing services move to the Cloud, there will be a proliferation of software characteristics, service models, and deployment options. Many organizations are moving into hybrid cloud/hosted computing models. A Cloud Service Provider (CSP) goal is to maximize its market share among the potential Service Providers (SP). Accommodating variable demands for computing resources requires an immense capacity, as it calls for providing for the maximum demand. In some cases, this drives them to underutilization of massive datacenter deployments. In other situations, the CSPs suffer over-utilization because of a miss in the market share, load and reliability projections. Both cases lead to sub-optimal utilization.

From the SPs perspective, they are most interested in availability and adaptability. The former refers to reliable service conditions that make its services available to the users it serves. The latter relates to the Vendor lock-in risk[49]. Single CSP provides a sub-optimal solution to the SP thus multi-cloud become an attractive solution. However, multi-cloud solution implemented by the SP requires expensive adaptations to the CSP's tools and service constructs that may vary among different CSPs.

Cloud Federation is a new paradigm that allows many CSPs to utilize computing resources optimally [11, 13]. Also, it allows SP to avoid the Vendor lock-in risk and provide service availability that can not be provided by a single CSP. No matter what the architecture, there is a need for to ensure the security and information assurance to users. Cloud Federation is an advantageous structure for aggregating cloud based services under a single umbrella to share resources and responsibilities for the benefit of the member cloud service providers. Federation is useful not only for sharing resources amongst cloud service providers but also for providing enclaves for interactions to perform domain-specific missions such as electrical grids and supply chains.

The Federation will need to assure that data transfers amongst the Federation's CSPs are secure. The Federation will, above all, need to detect any anomalous behavior occurring in transactions and resource sharing. In addition to the growing number of security tools, there is a need to log and identify security issues requiring attention early on in the process. In particular, breach detection in inter-cloud data transfer and communications is a particularly serious issue because of the possibility of an attacker potentially gaining access to more than one CSP federation member.

It is the purpose of this paper to describe security best practices for Cloud Federation. The paper also describes a tool and technique for detecting anomalous behavior in resource usage across the federation participants. Specifically, this method is developed for Cloud Federations since they have to deal with a heterogeneous multi-platform environment with a diverse mixture of data and security log schema, and it has to do this in real time. This Semantic-less tool is described below after a description of the context of the issue.

The reminder of sections is organized as follows, Section 13.1 (Cyber Security Challenges in Federated Cloud) describes the core challenges of such inter-cloud system in a multi-layer model; Section 13.2 (Semantic-less Breach Detection) discuss the tool we suggest for detecting the behavior of the anomalous system that runs in the Cloud Federation; Section 13.3 (Evaluation) discuss the breach-detection tool prototyped; finally, Section 13.4-13.6 (Execution and Analysis) analyze and present the prototype results.

6.0.1 Cyber Security Challenges in Federated Cloud

The Cloud Federation has a global scale software and hardware infrastructure. We describe a progressive layers security model starting from the physical security of data centers, progressing to the hardware and software that underlies the infrastructure, and the constraints and processes to support the Cloud Federation operational security. The following section describes the Cloud Federation cyber security design throughout the data processing life cycle at a Cloud Federation. e.g., enables secure communication with tenants (SP) and its customers or control plane communication including CSP, Cloud-Brokers, and Clouds-Coordinator.

Figure 6.1 describes the cyber security layers offered by the Cloud Federation. The following paragraph briefly describes the security elements corresponded with each layer[28]. Our extended cyber security model will emphasize the operational security with unique breach detection methodology. This was done since the operational security corresponds to the perimeter security of an enterprise system and the interface to the Federation members. Also, it will suggest a system for encryption of both inter micro-services communication with emphasis on cross-CSP for tenants' workloads.

Infrastructure Security

The required baseline security level needed for cloud federation constituent's systems is referenced in Figure 2.2. It includes deployed facilities and computer systems managed by the CSPs or the Federation. The larger CSP's often exceed these baselines.

Datacenter Premises. CSPs design and build its data centers based on its expected computing capacities and service reliability manifested by their SLA and the redundancy levels of sub systems[42, 61]. The datacenter incorporates various components of physical security protections. Access to such facilities is governed by the CSP security operations. It uses technologies such as biometric identification, metal detection, metal detectors, and CCTV solutions [63].

Hardware Design. CSPs data centers run computing server machines fed by power distribution units and connected to a local network that is all connected to the edge of the wide network. The computing, digital storage, and networking equipment require a standard that ensures the required audit and validation of the security properties by components [9, 47], e.g., hardware security chip [76].

Machine Identity. confirms that any participating computing server in Cloud Federation can be authenticated to its CSP machine pool throughout a low-level management services[76]

Secure Start-Up. Ensures that CSPs servers are booting the correct software stack. Securing underlining components such as Linux boot loaders, OS system images and BIOS by cryptographic

[28]We extrapolate Google Cloud security model from https://cloud.google.com/security/security-design/

signatures can prevent an already compromised server from being continuously compromised by an ephemeral malware.

Operational Security

Operational security comprises the business flows between the SP with the cloud federation and the CSP it uses for processing workloads. The following section briefly discusses the required cybersecurity measures needed for SP and CSP business scenarios in a cloud federation.

Cross-SP Access Management: SP workloads are manifest in two workload types, (1) short-lived workloads. i.e., jobs that are terminate upon completion, and (2) long-lived workloads. i.e. services. The former workload might require connectivity to external services during its processing. The latter might expose serving endpoints to other services. e.g., short-lived jobs might require persistent storage to write its job results hence connecting to BigTable[29] storage server provisioned by other CSPs, which, in turn, require access management that uses credentials and certificates stored within the Cloud Federation.

SP Front End Service Discovery: Long-lived workloads might expose public facing endpoints for serving other workloads or end-user requests. SP front-end services require publishing endpoints to allow other workloads within or external to the cloud Federation to discover their public facing entry point and this requires service discovery capabilities. Service discovery endpoints, and the actual service endpoints, are prone to risks such as Denial of Service attacks or intrusions originated by an attacker. We argue that current solutions offered by individual CSP's are sub-optimal because of the target scope of the intrusion. i.e., assuming an attack probability for a given CSP, running several CSPs reduces the risk by a factor of the number of CSPs. Later sections will formulate the risk function and show how cloud-federation minimizes those challenges by using the semantic-less breach detection system and show how most risks originate by crossing machine boundaries.

Secure Continues Deployment: Continuous Deployment (CD) is the function that allows cloud-

[29]https://cloud.google.com/bigtable/

native applications to get updated through an automated pipeline that is initiated by a new or updated code submission, compiled, tested through various quality gates until it is certified for deployment of the production systems and deployed seamlessly. Continues deployment enables cloud applications to innovate faster and safer no matter what number of machines are in the service pool. A secure continuous deployment service requires secured SP code and a configuration repository that authenticates to the target computing resource regardless of the CSP network segmentation. Traditional network segmentation, or fire walling, is a secondary security mechanism that is used for ingress and egress filtering at various points in the local network segment to prevent IP spoofing[6, 67]

Authentication and Authorization. In a federated cloud architecture, deployed workloads might require access to other services deployed by the federation. The canonical example will be an end user request service deployed in the Federation that triggers another micro-service within the SP architecture. Such cascading requests require multilayered authentication and authorization processes. i.e., a micro-service calls another micro-service and authenticate on behalf of the end user for audit trails supported by the end-user authentication token and the cascading micro-service tokens generated throughout the end user request. Figure 6.2 depicts the data flow during a call initiated by SP Micro-Service that runs in one of the federation's CSP denoted by CSP_i and CSP_j. A call initiated from SP_n that was provisioned in the federation as ms_n. The call destination runs on a different and sometimes the same SP. Let SP_m denote the destination SP. The call payload is encrypted by SP_n private key. The call arrived at an SP_m endpoint and checked for admission. SP_m admission control decrypts the call payload using SP_n public key that was submitted throughout the on-boarding process to the Cloud Federation. It is verified for authenticity and authorization of allowed call-sets. If admitted, SP_m calls and process the get_data() call and sends back the response to the originating SP, SP_n.

Breach Detection: The Cloud federation comprises various workload types that are owned by different autonomous organizations. Breach detection includes a complex data processing pipeline that integrates system signals originated from specific users of a CSP service as well as the potential cloud federation tenants. System signals are comprised of network devices as well as signals from infrastructure services. Only in recent years, after the growing numbers of data breaches and liabilities arising from losses,[5, 45, 74] have organizations started to incorporate business related metrics for breach detection[52]. Both data pipelines need to generate operational security warnings of potential incidents. The output of such warnings usually alerts security operations staff to potential incidents that require the relevant team's triage and response as appropriate.

Such methods are sub-optimal in a Federated Cloud for two main reasons: (1) different data sets are owned by different organization departments that are not integrated physically, schematically or semantically, (2) Lack of unification of both data sets as accomplished by fusion requires a complex transformation of both data sets semantics into a single data set. The above situation exacerbated when migrating the workload to the cloud as it introduces another orthogonal data set that contributes to complexity. The following sections propose a method for breach detection that collapses the three silos into a cohesive semantic-less data set that will enhance the Cloud Federation services detection breaches to an extent limited by available data and their investment in detection .i.e. allowing methods to the tenants to incorporate more data about their workload for more automatic detection.

6.0.2 Semantic-less Breach Detection

Malware infected cloud-computing-workloads introduces three core risks for organizations (1) Service unavailability, (2) Data breach , and (3) Data corruption. There is a need for breach detection system that helps to determine whether a workload is infected as well as the type of exploited risk type as enumerated above. Breach detection system effectiveness is influenced by a number of factors. We focused on the human social factor and the emergent public cloud offering. The following paragraph describes the important factors required for optimized breach detection. This mode of breach detection has to span the heterogeneous schema employed by the various federation members.

The Human Social Factor

Enterprise IT is typically organized into silos. e.g., IT operations, network operations, database administration, and product engineering. The silos goal is to allow field-based ownership. Usually, silo teams are governed by different management hierarchies, communication styles, and vocabularies i.e. *semantics*. As as far as cyber security goes, semantics manifest by a particular interpretation of intrusion or breach. e.g., malware sending data to C&C might not impact the normal operation of a workload. Thus, product owners are oblivious to that risk while network operations detect unusual egress or ingress traffic usage patterns.

Enterprise IT workloads deployed as SP workload requires adopting unified cyber security best practices that overcome the different management hierarchies, communication styles, system security pans, data scheme, semantics and, vocabularies. The next paragraph shows how Federated Cloud helps enterprise IT improve its cyber security resiliency by offering a prediction tool that allows SP to apply proactive policies to mitigate potential threats.

The Cloud Federation Factor

Public Cloud services exacerbate the organization's human factor risk by introducing an additional silo that is often separated from the organization it serves. Public cloud operations are agnostic to its tenant's workload semantics by definition. CSPs configure their multi-tenancy to allow business with conflict of interest to run its workload on the same platform. Such practices and policies, augment the lack of cohesive view required for optimized malware detection.

Workloads deployed in public cloud services are not limited to known machine boundaries as traditional on-premise models offer. Although CSPs feature cyber security mechanisms that attempt mimicking the traditional computing workload hosting, workloads artifacts are under the CSP control. As such, the cloud client workloads might be compromised. Thus, there is a need for another cyber security dimension for the SP workload that overcomes the lack of control when running in the cloud.

We proposed a unique self-learning methodology that removes the need for tenant information that streamlines semantic-less information from the various software stacks of the Cloud Federation, including both tenant metrics and control-plane metrics. Also, it streamlines training data of security incidents shared in collaborative platforms outside the Cloud Federation. We also argue that a Cloud Federation optimizes such collaboration and self-learning process. We prototyped a system that implements such self-learning system that resulted in up to 87% True-Positive rate with 93% True-Negative.

Workload data and usage patterns form a critical path for the SP business success. The leakage of some of the workload data and usage patterns impose a threat to the SP business. This challenge represents a new threat of organizational espionage as well as attacks on the SP service that impacts SP business continuity. Therefore, sharing semantics breaks the isolation between the two systems and might hold the hosting system accountable for security attacks in CSP or Cloud Federation platforms. Also, transforming every workload semantics into a coherent model that aggregates numerous SP workloads requires a significant amount of investment. SPs will be reluctant to make such an investment, especially since it doesn't produce income. Therefore, this method has a low likelihood of being implemented. Therefore, enabling a method that eliminates SP investment and business risks is a key for the breach detection system success. Finally, a Cloud-Federation provides a centralized view of cross-CSP operations. Such centralized view allows SP workload deployment to different CSPs to gather a rich data set that will be available for malware identification and later, for predictive analytics. We suggest a method that captures computing resources usage and intra federation traffic and infers potential breach or disruption to proactively alerts CSP security stakeholders about suspicious cyber instances.

From Workload Semantics to Semantic-less

Cloud workloads are broadly composed of two types: online system, and offline system. The former provides low-latency, read/write access to data. For example, a web user requests a web page to load online and serve within a fraction of a second. The latter provides batch-like computing tasks that process the data offline, which is reported later to users by the system servers; for example, the search results based on a pre-calculated index. Offline production workloads are usually comprised of mainly unstructured data sets, such as click stream, web graph, and sensors data[11, 13].

The semantic-less detection will address the polymorphic malware case as its data stream are abstracted from computing activity. More specifically, a tenant's workload in a federated cloud manifested by software containers that are limited to not more than (1) namespace per tenant for isolation and (2) limited to a resources control groups(aka cgroup)[30] Control groups are the mechanism for limiting computing server host CPU, Memory, Disk I/O and Network I/O usage per namespace. That is the foundation of Linux Containers, which alludes to the existing methods of measurements of the metrics set, CPU, memory and I/O usage. We call this set the behavioral attributes set. Access to cgroup and namespace configuration and control is available on the host level i.e. the host OS that runs the multi-tenant workloads i.e. a control-plane component.

Data Collection

Both Cyber Security leaders and national agencies agree that addressing emerging cyber risks require sharing cyber attacks retrospects and their historical behavior, and discovered vulnerability reports as a foundation for collaboration, predictive time series analysis, risk quantification and risk allocations all leading to safer cyber services [21, 41]. Incidents are often documented in unstructured reports that require a manual analysis to identify trends[80].

To assess whether or not a system was breached, it is required to establish malicious system behavior patterns and then decompose those patterns into generic computing system metrics that can later be classified as harmful or safe. The following paragraph includes the source datasets we chose to assess the initial malicious patterns and their detectability by our method. We continued by decomposing the data and removing the tenant semantics. That allows a generic pattern of malicious activity dataset that can be used as a training data for the supervised model.

[30]https://www.kernel.org/doc/Documentation/cgroup-v1/cgroups.txt

Source Datasets

We choose the National Vulnerability Database (NVD)[19] and the Vocabulary for Event Recording and Incident Sharing (VERIS) [78]. Both datasets included thousands of reported incidents spanning across various categories. Our model focuses on (1) Unauthorized access attempts, (2) Suspicious Denial of Service, and (3) Data Stealing Malicious Code, including ransomware instances. We filtered the incidents that conform to the categories and performed a qualitative assessment of the identified breach impacts. Lastly, for simplicity, we appllied an additional category that distinguishes the target component reported, service-based or client-based. We included only the service-based incidents. i.e., reported incidents that clearly targeted desktops and workstations were not included in defining tenant semantic structures.

We applied filters for training data accuracy. Filters for VERIS dataset included server workloads as indicated in Section 4.2.1, i.e., Authentication Server, Backup Server, Database Server, DHCP Server, Directory Server(LDAP, AD), Distributed control system, Domain Name Server, File Server, Mail Server, Mainframe Server, Web Application Server, and Virtual Machine Server[78]. Assets operating systems were filtered to Linux and Unix as such operating systems are more prevalent in servers than Windows, MacOSX, and mobile device operating systems.

VERIS dataset includes incident actions. We filtered the action types that fit the paper focus workloads. i.e. Brute Force, Cache Poisoning, Cryptanalysis, Fuzzing, and HTTP Request Smuggling attacks. We excluded Buffer overflow cases as such attacks can be prevented in deterministic methods and common in Windows-based operating systems[82]. The dataset size following the refinement is 5015 incidents. Table 6.1 summarizes the dataset we used for the training data.

Table 6.1: Summary of datasets used

Malware Category	modus operandi	Number of Incidents	
Brute Force	Exhaustive effort of data encryption	946	
Cache Poisoning	Corrupt data is inserted into the cache database e.g., DNS	894	
Cryptanalysis	Exhaustive effort of data encryption	750	
Fuzzing	Injects random bad data into an application to break it	946	
HTTP Request Smuggling	Exhausting a proxy cache by sending HTTP requests	639	
Data stealing malware	Data transmitting across unencrypted network	840	

Removing the Tenant Semantics

Our approach attempts to detect anomalies in both control-plane and tenant activities that conform to suspicious patterns. In Section 13.2.4 Data Collection, we defined a categorical dataset that adheres to real incident data. This data applies to potential breaches for server-based workloads. We stipulate, for the purposes of this paper that such server-based workload will obey similar suspicious patterns when deployed in the cloud.

In this paragraph, we transformed the categorical dataset into a multivariate time series data that can be used for supervised anomaly detection. The multivariate set is comprised of general operating system observations that do not include any workload semantics but could be used for contextual anomaly detection. The contextual attributes are used to determine the position of an instance on the entire series. We showed that, based on collected incident data, the conversion of behavior patterns to multivariate time-series satisfies effective breach detection of any malware, conventional or polymorphic.

We gathered the operations reported in the incident reports (Table 6.1) and inferred about the operating system resources consumed during the malware lifespan. Table 6.2 depicts the relationship between the malware characteristics and operating system usage. Figure 6.3 describes a workload sample, video-on-demand. It shows the common pattern of the operating system resources usage that will be used as multivariate time series data sequences. The Evaluation section describes in more details the nature of the data and how it translates into meaningful time series data.

Table 6.2: Dataset Classification

Malware Category	OS Resources Patterns
Brute Force	Extensive CPU, Memory, I/O to disk or network
Cache Poisoning	Extensive I/O to disk or network
Cryptanalysis	Extensive I/O to disk or network
Fuzzing	Network I/O Ingress
HTTP Request Smuggling	Network I/O Egress
Data stealing malware	Network I/O Egress

Prediction Methodology

We used the data gathered in Table 6.2 for formulating the anomaly detection problem of polymorphic malware[20]. The detection approach includes three distinct methods: (1) Detecting anomalous sequences in OS usages time series events, (2) Detecting anomalous subsequences within OS usages time series, and (3) Detecting anomalous OS usages events based on frequency. Let T denote a set of n training sequences based on OS usage generated by CSPs, SPs, and the Federation control plane. Also, S denote a set of m test sequences generated based on Table 6.2, we find the anomaly score $A(S_q)$ for each test sequence $S_q \in S$, with respect to T. T mostly includes normal OS usage sequences, while S includes anomalous sequences.

The semantic-less tool output produces a score for a scanned training sequence T using Regression. i.e., forecast the next observation in the time series, using the statistical model and the time series observed so far, and compare the forecasted observation with the actual observation to determine if an anomaly has occurred[20]. For simplicity, our model uses TensorFlow for regression calculation[1].

6.0.3 Evaluation

We prototyped Cloud Federation system that mimics that properties analyzed in section 2, Cloud Federation. The prototyped system includes the component that is depicted in Figure 2.2. For the scope of the prototype, we enabled semantic-less metrics from both SPs and CSPs to improve correlation efficiency. CSP data sharing limits the effectiveness of any cyber analytical technique and, in practice will represent a compromise between improved cyber security and CSP privacy and confidentiality. With that proviso, in the following section, we evaluate a computer load coordination system component that manages on-demand streaming, generates T, a set of n training sequences based on OS usage generated by CSPs, SPs, and the Federation control plane. We chose on-demand video streaming as Video streaming is expected to constitute up to 85% of Internet consumers traffic within a few years[12]. Also, we showed that video-on-demand streaming follows a pattern of usage that can be monitored for breach detection that can help on-demand SP to seamlessly improve their consumer's privacy and provide their studio's safe e-commerce platform.

Experiment Planning

Below is a simulation of a cross-regional platform that is comprised of control-plane, workload-plane and coordinating components. This will be embodied in a resource allocation system(Kubernetes). This system provisions resources to be a priority of being near, users. The control-plane enables an effective compute resource provisioning system that spans across different public cloud providers and regions. Also, it collects operating systems usages for both the SP workload and control-planeThe coordinating components will accept user-workload demands as well as green energy availability from various regions and opportunistically seek to process streaming workloads using compute resources provisioned by green energy resources. The workload-plane will be comprised of edge streaming servers that process the end-user on-demand video streaming. It will be built on standard Apache HTTP[31] servers that run on the edge location.

The control-plane software infrastructure is based on Kubernetes[32], it facilitates internal discovery between Apache HTTP server instances so instances can connect across different cloud boundaries and regions. This architecture provides an open architecture that enables continuous monitoring. In a real world federation the data load may require several big data nodes and substantial compute capacity. This is a demonstration and proof of concept on a finite scale to permit model and parameter tracking and adjustment.

6.0.4 Execution

The System Preparation

The prototype experiment included the setup of three virtual datacenters deployed in different regions: (1) Central US, (2) West US and (3) East US. The clusters were sized based on US

[31]Apache Web Server reference retrieved from https://httpd.apache.org

[32]Kubernetes reference retrieved from http://kubernetes.io

population distribution[33] by regions i.e. 20% for West US, 40% for East US and 40% Central US. The cluster sizes for West US, Central US, and East US are 3, 7 and 7 machines respectively. Each machine is standard 2-CPU cores with 7.5GB of memory.

The control-plane comprised of Kubernetes API server and controller-manager. The controller coordinator component will need to allocate resources across several geographic regions to different cloud providers. The API server will run a new federation namespace dedicated for the experiment in a manner that such resources are provisioned under a single system. Since the single system may expose external IPs, it needs to be protected by an appropriate level of asynchronous encryption[34].

For simplicity, we use a single cloud provider, Google Container Engine, as it provides a multi-zone production-grade compute orchestration system. The compute instances that process the user workloads are deployed as Docker containers that run Ubuntu 15 loaded with Apache HTTP server. For simplicity, we avoided content distribution by embedding the video content to be streamed in the Docker image. We ran 52 Docker containers that span across the three regions and acted as Content Delivery Network edges.

Baseline and Execution

The baseline execution included data populations for video streaming. The data population was achieved by the Kubernetes Jmeter batch jobs. The loader jobs goal is to generate traffic that obeys the observed empirical patterns depicted in Figure 6.3. The system usage for both control-plane and SP capture, through cAdvisor, a kubernetes resource usage, and performance analysis agent. The agent, from every node in a cluster, populates system usage data to Heapster, a cluster-wide aggregator of monitoring and event data [35].

We labeled the system usage with the semantic-less dimensions, Network egress was measured by thousands of transmitted packets (k-TX), Disk writes per second (k-write/sec) and CPU usage

[33]US Population Distribution retrieved from https://www.census.gov/popclock/data tables.php

[34]Simulation code and data retrieved from https://github.com/yahavb/green-content-delivery-network

[35]https://kubernetes.io/docs/user-guide/monitoring/

per container (%). The Heapster aggregated the data based on the labels that are later pushed to centralized database, influxDB. We also used the influxDB HTTP API to inject randomized system usage data according to the three labels, CPU, network and disk usage. Those considered as the anomalous sequences $S_q \in S$. We used Figure 6.3 as a baseline sequence that randomized using NumPy[36]. The randomization followed the malicious usage patterns described in Table 6.2.

The execution required a TensorFlow session that looped through the dataset multiple times, update the model parameters and obtain the anomaly score $A(S_q)$ for each test sequence $S_q \in S$, on T. The breach and anomaly detection was performed using the following data streams and learning algorithms.

Limitations

We used influxDB because of its seamless integration with Kubernetes Monitoring system. However, our approach is not limited to influxDB or other database systems for that matter. We used TensorFlow for regression and anomaly score calculation. We did not use long training sequences. The maximum duration spanned across 48 hours. Training with longer sequences using long-running jobs and TensorFlow model checkpointing would improve our results. Our test content variety was limited and fixed. That might impact the generated tests sequences stability. Larger content variety would require longer training sequences for optimal detection.

6.0.5 Analysis

The prototype included two core datasets, normal (Figure 6.3) and malicious (Figure 6.4). The CPU usage in the normal dataset fit the viewing patterns at the first half of the run. The second half required less CPU due to the caching mechanism applied in the Apache HTTP server that alleviates the need for the CPU when served through a cache. The Disk write pattern manifested similar content caching schema. The network egress ratio was not impacted by the caching schema.

The malicious dataset used the malware classification table (Table 6.2). Figure 6.4, shows a semantic-less behavior for ransomware malware that attempts to encrypt data while serving work-

[36]Package for scientific computing with Python, numpy.org

load. Suspicious signals denoted by a star and o markers for CPU and disk write respectively. Based on the dataset classification, ransomware requires no network egress but CPU for data encryption and writing back to disk the encrypted payload. Our prototype included similar patterns depicted in Table 6.2 with a similar approach as done for ransomware.

The model attempted to detect anomalous subsequences within S_q. e.g., S_{net} for network, S_{cpu} for CPU, and S_{rw} for disk read and write transactions. We used a scoring based techniques for each of the observed sequences. The score range is between [-10,10] and executed 2000 training epochs. Scores that are closed to 10 indicates on a potential breach. Negative scores indicates normal behavior. The scores in Figure 6.4 is $max(S_{net}, S_{cpu}, S_{rw})$. The maximum-based aggregation was chosen for simplicity as each training sequence plays an equal role in the potential for breach. Other anomalies might use different type of aggregations. According to the anomaly scores shown in the experiment, up to 87% was True-Positive of detected breaches and 93% of the detections where True-Negative.

6.0.6 Conclusions for Semantic-Less Breach Detection in Federated Cloud

Security practices traditionally focus on prevention and tightening perimeter boundaries. However, with the advent of disparate, distributed, large scale, multi-tenant environments such as the proposed Cloud Federation, the traditional perimeter boundaries along with traditional security practices are changing. Defining and securing asset boundaries is more challenging and the system perimeter boundaries are more susceptible to breach. In this paper, we proposed a proactive approach for detecting a breach in a cloud workload. Such method requires no upfront investment from the monitored services. Upfront investments are often one of the main barriers to securing cloud service. Our tool eliminates such need and uses general system usage patterns that help to predict potential breach proactively.

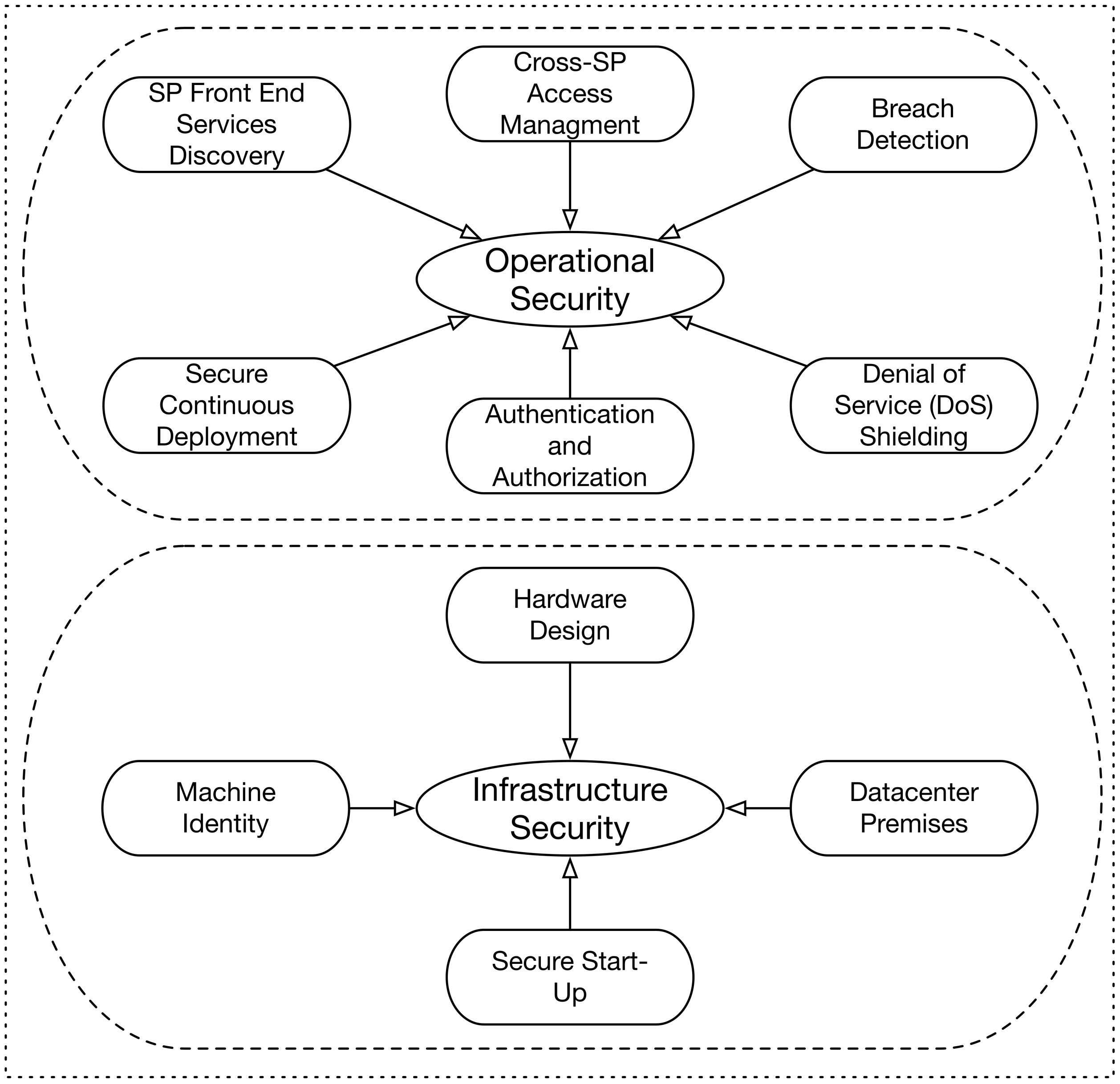

SP Front End Services Discovery
Cross-SP Access Managment
Breach Detection
Operational Security
Secure Continuous Deployment
Authentication and Authorization
Denial of Service (DoS) Shielding
Hardware Design
Machine Identity
Infrastructure Security
Datacenter Premises
Secure Start-Up

Figure 6.1: Cloud Federation Cyber Security Model includes two core layers, infrastructure security and operational security

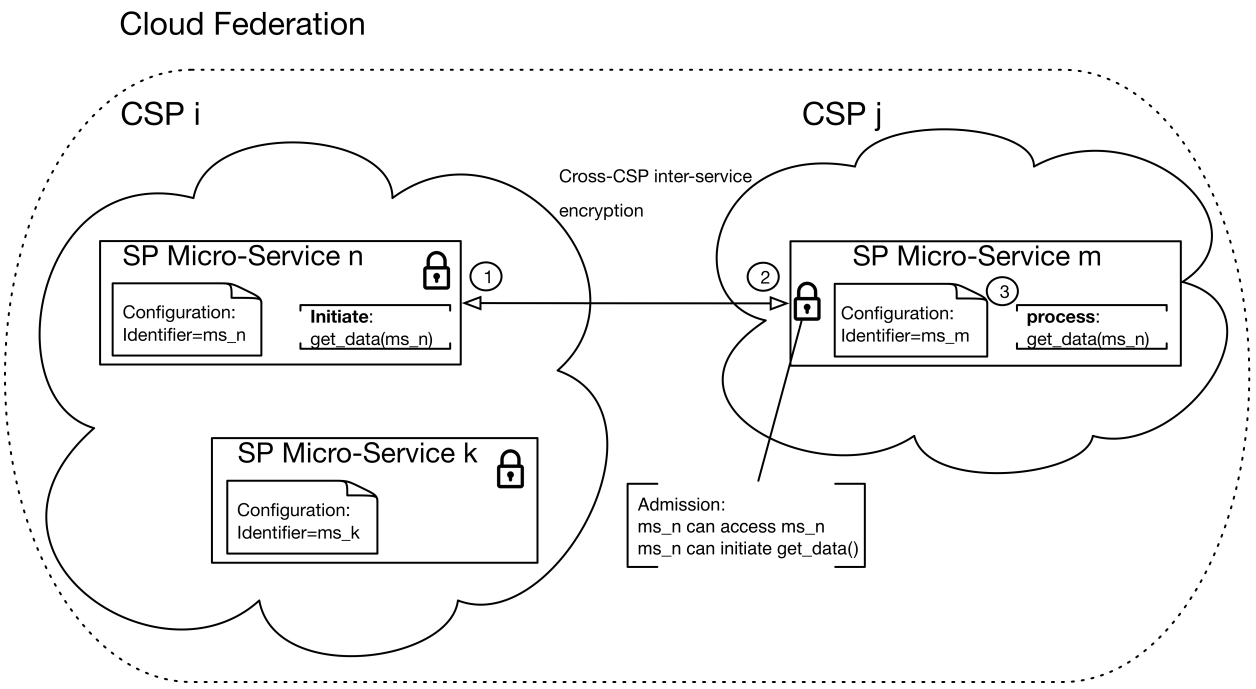


Figure 6.2: Authentication and authorization in Cloud Federation Cross-SP model

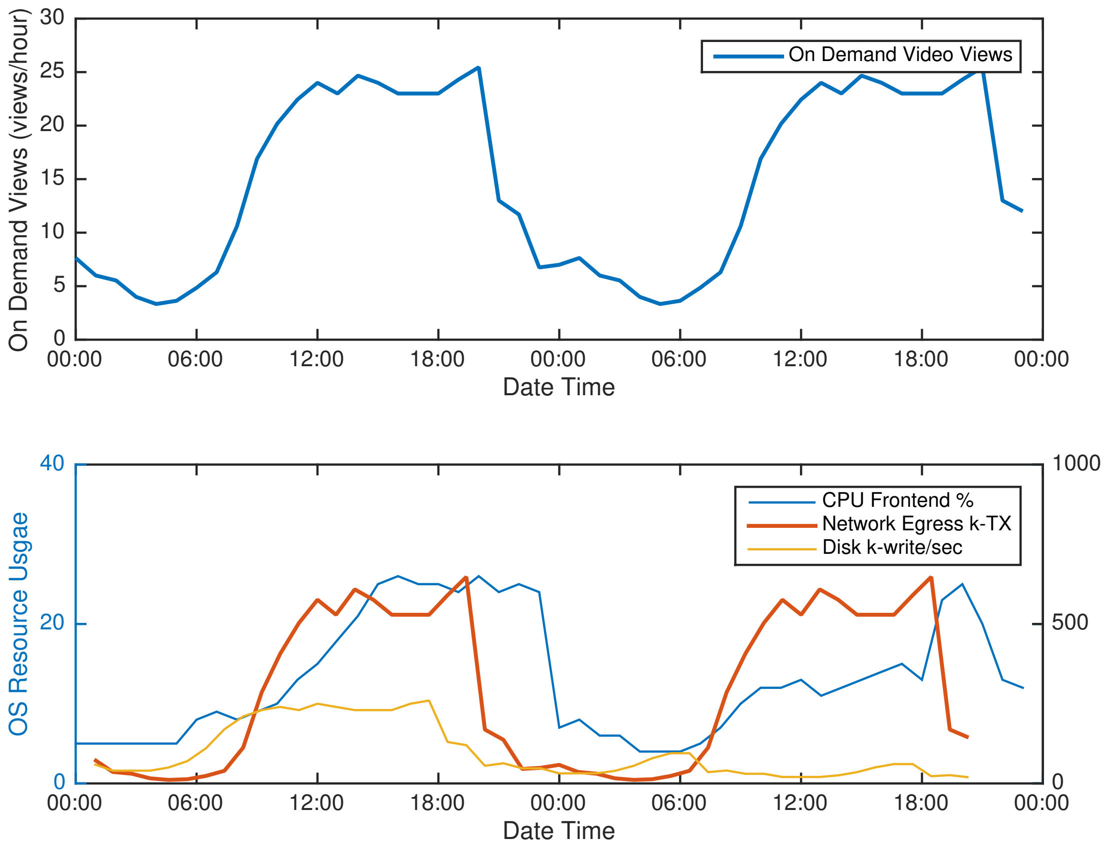
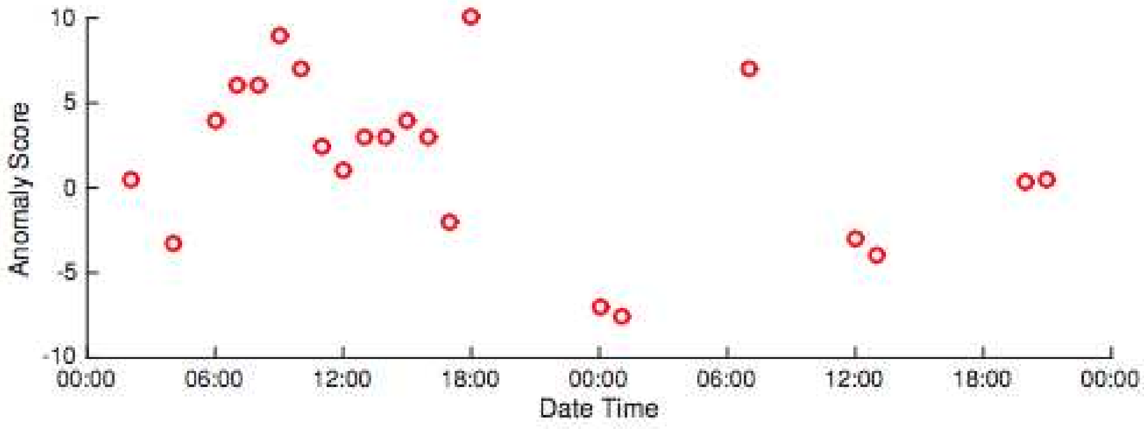


Figure 6.3: Workload semantics sample transformed into semantic-less training sequences



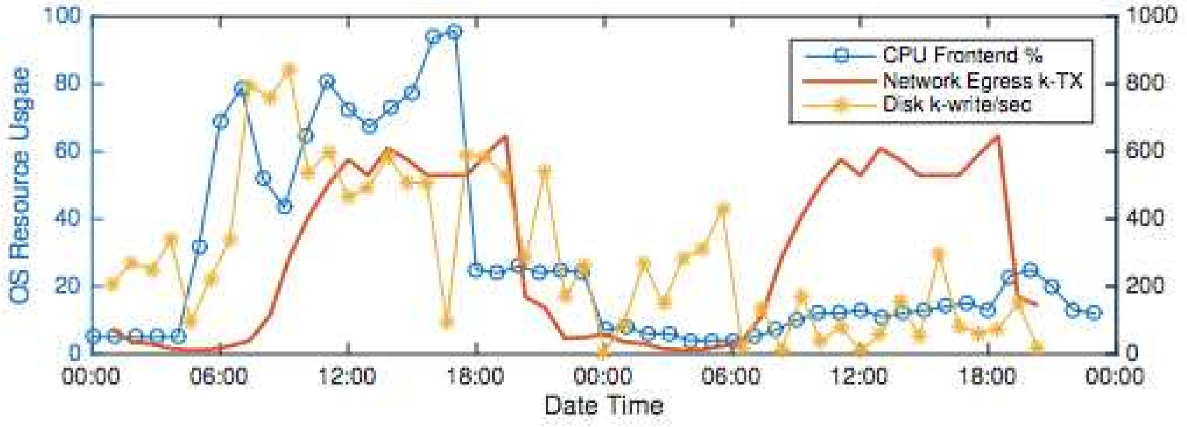


Figure 6.4: A Ransomware Anomalous Workload semantic-less sample

Chapter 7

Conclusion

Cloud computing is reshaping IT for both public sector, the private sector and military enterprises. Cloud computing drives new modes of considering computing needs in terms of scale, agility and marketing speed. Cloud computing creates new demands for more computing hardware equipment of general and specialized use. These demands are what drive the transition from a one-fits-all solution to the distributed Federated SoS paradigm. The new collaborative SoS requires a new control methodology that turns out to be both financially attractive and environmentally significant. The architecture is designed so that the constituent cloud providers retain independent ownership, objectives, funding, and sustainability means. In the authors' paradigm, any changes in the constituent cloud provider's business processes are based on cooperative agreements between the Federation and the cloud providers. Finally, explored emergent behavior by the constituent systems was explored through simulations that were used to optimize of costs and resource utilization.

Bibliography

[1] Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., et al. (2016). Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*.

[2] Abdelbaky, M., Diaz-Montes, J., Parashar, M., Unuvar, M., and Steinder, M. (2015). Docker containers across multiple clouds and data centers. In *Utility and Cloud Computing (UCC), 2015 IEEE/ACM 8th International Conference on*, pages 368–371. IEEE.

[3] Abolfazli, S., Sanaei, Z., Ahmed, E., Gani, A., and Buyya, R. (2014). Cloud-based augmentation for mobile devices: motivation, taxonomies, and open challenges. *IEEE Communications Surveys & Tutorials*, 16(1):337–368.

[4] Abran, A., Moore, J., Bourque, P., Dupuis, R., and Tripp, L. (2004). Software engineering body of knowledge. *IEEE Computer Society, Angela Burgess*.

[5] Agrawal, T., Henry, D., and Finkle, J. (2014). Jpmorgan hack exposed data of 83 million, among biggest breaches in history.

[6] Andersen, D. G. et al. (2003). Mayday: Distributed filtering for internet services. In *USENIX Symposium on Internet Technologies and Systems*, pages 20–30.

[7] Barroso, L. A., Clidaras, J., and Hölzle, U. (2013). The datacenter as a computer: An introduction to the design of warehouse-scale machines. *Synthesis lectures on computer architecture*, 8(3):1–154.

[8] Beloglazov, A. and Abawajy, J. (2012). Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing. *Future generation computer systems*, 28(5):755–768.

[9] Biran, Y. and Collins, G. (2016). Open compute-equipment design specification as a standard for cloud computing. In *Zooming Innovation in Consumer Electronics International Conference (ZINC), 2016*, pages 70–75. IEEE.

[10] Biran, Y., Collins, G., Azam, S., and Dubow, J. (2017a). Federated cloud computing as system of systems. In *Computing, Networking and Communications (ICNC), 2017 International Conference on*, pages 711–718. IEEE.

[11] Biran, Y., Collins, G., Dubow, J., and Azam, S. (2017b). Federated cloud computing as system of systems. In *2017 Workshop on Computing, Networking and Communications (CNC) (CNC'17)*, pages 130–135. IEEE.

[12] Biran, Y., Collins, G., Dubow, J., and Pasricha, S. (2016a). Coordinating green clouds as data-intensive computing. In *2016: 3rd Smart Cloud Networks & Systems Conference (SCNS'16)*, pages 1–8. IEEE.

[13] Biran, Y., Collins, G., and Liberatore, J. (2016b). Coordinating green clouds as data-intensive computing. In *2016 IEEE Green Technologies Conference (GreenTech)*, pages 130–135. IEEE.

[14] Biran, Y., Pasricha, S., Collins, G., and Dubow, J. (2016c). Enabling green content distribution network by cloud orchestration. In *Smart Cloud Networks & Systems (SCNS)*, pages 1–8. IEEE.

[15] Bottomley, J., Emelyanov, P., Kantee, A., Cormack, J., Klein, D., Bester, D., Monroe, M., and Seely, A. (2014). Operating systems. *USENIX Login*.

[16] Buchbinder, N., Jain, N., and Menache, I. (2011). Online job-migration for reducing the electricity bill in the cloud. *NETWORKING 2011*, pages 172–185.

[17] Burns, B., Grant, B., Oppenheimer, D., Brewer, E., and Wilkes, J. (2016). Borg, omega, and kubernetes. *Communications of the ACM*, 59(5):50–57.

[18] Cardell, J. B., Hitt, C. C., and Hogan, W. W. (1997). Market power and strategic interaction in electricity networks. *Resource and energy economics*, 19(1):109–137.

[19] Center, C. C. (2004). Vulnerability notes database. *CERT Coordination Center*.

[20] Chandola, V. (2009). *Anomaly detection for symbolic sequences and time series data*. PhD thesis, University of Minnesota.

[21] Chew, E., Swanson, M., Stine, K., Bartol, N., Brown, A., and Robinson, W. (2008). Performance measurement guide for information security. national institute of standards and technology (nist) special publication 800-55 revision 1.

[22] Cohen, M. D. and Axelrod, R. (2000). *Harnessing complexity: organizational implications of a scientific frontier*. Simon and Schuster.

[23] Collins, G. and Biran, Y. (2015). Multi-tenant utility computing with compute containers. In *Consumer Electronics-Berlin (ICCE-Berlin), 2015 IEEE 5th International Conference on*, pages 213–217. IEEE.

[24] Dahmann, J. (2014). 1.4. 3 system of systems pain points. In *INCOSE International Symposium*, pages 108–121. Wiley Online Library.

[25] Dahmann, J. (2015). The state of systems of systems engineering knowledge sources. In *System of Systems Engineering Conference (SoSE), 2015 10th*, pages 475–479. IEEE.

[26] Dean, J. and Ghemawat, S. (2004). Mapreduce: simplified data processing on large clusters. osdi?04 proceedings of the 6th conference on symposium on opearting systems design and implementation? dalam: International journal of enggineering science invention. *URL: http://static. googleusercontent. com/media/resear ch. google. com (diunduh pada 2015-05-10)*, pages 10–100.

[27] Deshmukh, M. and Deshmukh, S. (2008). Modeling of hybrid renewable energy systems. *Renewable and Sustainable Energy Reviews*, 12(1):235–249.

[28] Donenfeld, J. (2012). Linux local privilege escalation via suid/proc/pid/mem write.

[29] Energy, G. (2010). Western wind and solar integration study. *NREL Subcontract Report*, pages 1–13.

[30] Fabrycky, W. J. and McCrae, E. A. (2005). 6.1. 2 systems engineering degree programs in the united states. In *INCOSE International Symposium*, pages 833–847. Wiley Online Library.

[31] Feehs, R. J. (2013). 8.2. 2 cloud effectiveness model. In *INCOSE International Symposium*, pages 1099–1115. Wiley Online Library.

[32] Feijoo, A. E., Cidras, J., and Dornelas, J. G. (1999). Wind speed simulation in wind farms for steady-state security assessment of electrical power systems. *IEEE transactions on Energy Conversion*, 14(4):1582–1588.

[33] Ferguson, A. D., Bodik, P., Kandula, S., Boutin, E., and Fonseca, R. (2012). Jockey: guaranteed job latency in data parallel clusters. In *Proceedings of the 7th ACM european conference on Computer Systems*, pages 99–112. ACM.

[34] Frachtenberg, E. (2012). Holistic datacenter design in the open compute project. *Computer*, 45(7):0083–85.

[35] Gandhi, A., Harchol-Balter, M., Raghunathan, R., and Kozuch, M. A. (2012). Autoscale: Dynamic, robust capacity management for multi-tier data centers. *ACM Transactions on Computer Systems (TOCS)*, 30(4):14.

[36] Handy, C. (1995). Trust and the virtual organization. *Harvard business review*, 73(3):40.

[37] Huang, Q., Birman, K., van Renesse, R., Lloyd, W., Kumar, S., and Li, H. C. (2013). An analysis of facebook photo caching. In *Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles*, pages 167–181. ACM.

[38] Ihm, S. and Pai, V. S. (2011). Towards understanding modern web traffic. In *Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference*, pages 295–312. ACM.

[39] Jain, N. and Menache, I. (2012). Performance-based pricing for cloud computing. US Patent App. 13/530,399.

[40] Jamshidi, P., Ahmad, A., and Pahl, C. (2014). Autonomic resource provisioning for cloud-based software. In *Proceedings of the 9th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, pages 95–104. ACM.

[41] Jaquith, A. (2007). *Security metrics*. Pearson Education.

[42] Kim, D., Chung, H. R., and Thompson, P. R. (2009). Cloud-based automation of resources. US Patent App. 12/634,050.

[43] Knauth, T. and Fetzer, C. (2014). Dreamserver: Truly on-demand cloud services. In *Proceedings of International Conference on Systems and Storage*, pages 1–11. ACM.

[44] Kossiakoff, A., Sweet, W. N., Seymour, S., and Biemer, S. M. (2011). *Systems engineering principles and practice*, volume 83. John Wiley & Sons.

[45] Krebs, B. (2014). The target breach, by the numbers. *Krebs on Security*, 6.

[46] Krishnan, S. S. and Sitaraman, R. K. (2013). Understanding the effectiveness of video ads: a measurement study. In *Proceedings of the 2013 conference on Internet measurement conference*, pages 149–162. ACM.

[47] Krutz, R. L. and Vines, R. D. (2010). *Cloud security: A comprehensive guide to secure cloud computing*. Wiley Publishing.

[48] Lam, W., Liu, L., Prasad, S., Rajaraman, A., Vacheri, Z., and Doan, A. (2012). Muppet: Mapreduce-style processing of fast data. *Proceedings of the VLDB Endowment*, 5(12):1814–1825.

[49] Leavitt, N. (2009). Is cloud computing really ready for prime time. *Growth*, 27(5):15–20.

[50] Lim, N., Majumdar, S., and Ashwood-Smith, P. (2014). Engineering resource management middleware for optimizing the performance of clouds processing mapreduce jobs with deadlines. In *Proceedings of the 5th ACM/SPEC international conference on Performance engineering*, pages 161–172. ACM.

[51] Liu, X., Dobrian, F., Milner, H., Jiang, J., Sekar, V., Stoica, I., and Zhang, H. (2012). A case for a coordinated internet video control plane. In *Proceedings of the ACM SIGCOMM 2012 conference on Applications, technologies, architectures, and protocols for computer communication*, pages 359–370. ACM.

[52] Liu, Y., Sarabi, A., Zhang, J., Naghizadeh, P., Karir, M., Bailey, M., and Liu, M. (2015). Cloudy with a chance of breach: Forecasting cyber security incidents. In *USENIX Security*, pages 1009–1024.

[53] Liu, Z., Lin, M., Wierman, A., Low, S. H., and Andrew, L. L. (2011). Geographical load balancing with renewables. *ACM SIGMETRICS Performance Evaluation Review*, 39(3):62–66.

[54] Love, R. (2013). *Linux system programming: talking directly to the kernel and C library*. " O'Reilly Media, Inc.".

[55] Maier, G., Feldmann, A., Paxson, V., and Allman, M. (2009). On dominant characteristics of residential broadband internet traffic. In *Proceedings of the 9th ACM SIGCOMM conference on Internet measurement conference*, pages 90–102. ACM.

[56] Maier, M. W. (1996). Architecting principles for systems-of-systems. In *INCOSE International Symposium*, pages 565–573. Wiley Online Library.

[57] Mell, P., Grance, T., et al. (2011). The nist definition of cloud computing. *Computer*.

[58] Merkel, D. (2014). Docker: lightweight linux containers for consistent development and deployment. *Linux Journal*, 2014(239):2.

[59] Mills, S. (2015). Open compute project. open rack version 1.2. 2015.

[60] Moore, S. (2015). Gartner says worldwide cloud infrastructure-as-a-service spending to grow 32.8 percent in 2015.

[61] Nelson, R. (2016). Iot spans edge to data center. *EE-Evaluation Engineering*, 55(7):18–21.

[62] Nygren, E., Sitaraman, R. K., and Sun, J. (2010). The akamai network: a platform for high-performance internet applications. *ACM SIGOPS Operating Systems Review*, 44(3):2–19.

[63] Orr, R. J. and Abowd, G. D. (2000). The smart floor: a mechanism for natural user identification and tracking. In *CHI'00 extended abstracts on Human factors in computing systems*, pages 275–276. ACM.

[64] Osmundson, J., Irvine, N., Schacher, G., Jensen, J., Langford, G., Huynh, T., and Kimmel, R. (2006). Application of system of systems engineering methodology to study of joint military systems interoperability. In *Proceedings from the 2 nd Annual System of Systems Engineering Conference, Ft. Belvoir, VA, sponsored by the National Defense Industrial Association (NDIA) and OUSD AT&L*.

[65] Outlook, A. E. et al. (2010). Energy information administration. *Department of Energy*, 92010(9):1–15.

[66] Park, E., Egger, B., and Lee, J. (2011). Fast and space-efficient virtual machine checkpointing. In *ACM SIGPLAN Notices*, pages 75–86. ACM.

[67] Peng, T., Leckie, C., and Ramamohanarao, K. (2007). Survey of network-based defense mechanisms countering the dos and ddos problems. *ACM Computing Surveys (CSUR)*, 39(1):3.

[68] Rao, L., Liu, X., Xie, L., and Liu, W. (2010). Minimizing electricity cost: Optimization of distributed internet data centers in a multi-electricity-market environment. In *INFOCOM, 2010 Proceedings IEEE*, pages 1–9. IEEE.

[69] Rechtin, E. and Maier, M. W. (2010). *The art of systems architecting*. CRC Press.

[70] Roedler, G. (2012). Harmonization of key systems engineering resources. In *15th Annual NDIA Systems Engineering Conference, San Diego, CA (US)*. Citeseer.

[71] Sage, A. P. and Cuppan, C. D. (2001). On the systems engineering and management of systems of systems and federations of systems. *Information knowledge systems management*, 2(4):325–345.

[72] Salomie, T.-I., Alonso, G., Roscoe, T., and Elphinstone, K. (2013). Application level ballooning for efficient server consolidation. In *Proceedings of the 8th ACM European Conference on Computer Systems*, pages 337–350. ACM.

[73] Shehabi, A., Smith, S., Sartor, D., Brown, R., Herrlin, M., Koomey, J., Masanet, E., Horner, N., Azevedo, I., and Lintner, W. (2016). United states data center energy usage report. *Data Center*.

[74] Sidel, R. (2014). Home depot's 56 million card breach bigger than target's. *Wall Street Journal, Sep*.

[75] Singh, R. P., Brecht, T., and Keshav, S. (2015). Towards vm consolidation using a hierarchy of idle states. In *ACM SIGPLAN Notices*, volume 50, pages 107–119. ACM.

[76] Skorobogatov, S. P. (2005). *Semi-invasive attacks: a new approach to hardware security analysis*. PhD thesis, Citeseer.

[77] Stacey, R. D., Griffin, D., and Shaw, P. (2000). *Complexity and management: fad or radical challenge to systems thinking?* Psychology Press.

[78] Team, V. R. (2015). 2015 data breach investigations report. *Verizon Data Breach Investigations Report (DBIR)*.

[79] Tiwari, A. and Sodha, M. (2006). Performance evaluation of solar pv/t system: an experimental validation. *Solar Energy*, 80(7):751–759.

[80] Tufte, S. E. (2015). Documenting cyber security incidents. *Department of Management Science and Engineering, Stanford University, School of Information, UC Berkeley*.

[81] Vlissides, J., Helm, R., Johnson, R., and Gamma, E. (1995). Design patterns: Elements of reusable object-oriented software. *Reading: Addison-Wesley*, 49(120):11.

[82] Wartell, R., Mohan, V., Hamlen, K. W., and Lin, Z. (2012). Binary stirring: Self-randomizing instruction addresses of legacy x86 binary code. In *Proceedings of the 2012 ACM conference on Computer and communications security*, pages 157–168. ACM.

[83] Wei, Y. and Blake, M. B. (2010). Service-oriented computing and cloud computing: challenges and opportunities. *IEEE Internet Computing*, 14(6):72.

[84] Wilkes, J. (2011). More google cluster data. *Google research blog, Nov*.

[85] Xavier, M. G., Neves, M. V., and De Rose, C. A. F. (2014). A performance comparison of container-based virtualization systems for mapreduce clusters. In *Parallel, Distributed and Network-Based Processing (PDP), 2014 22nd Euromicro International Conference on*, pages 299–306. IEEE.

[86] Xu, X., Lu, Q., Zhu, L., Li, Z., Sakr, S., Wada, H., and Webber, I. (2013). Availability analysis for deployment of in-cloud applications. In *Proceedings of the 4th international ACM Sigsoft symposium on Architecting critical systems*, pages 11–16. ACM.

[87] Yan, M. and Ehlen, J. (2012). Open vault storage hardware v0.5.

[88] Zhang, Q., Zhu, Q., and Boutaba, R. (2011). Dynamic resource allocation for spot markets in cloud computing environments. In *Utility and Cloud Computing (UCC), 2011 Fourth IEEE International Conference on*, pages 178–185. IEEE.

[89] ZHANG, Y. and ZHENG, W. (2009). http://msdn. microsoft. com/en-us/library/ms713492 (vs. 85). aspx http://msdn. microsoft. com/en-us/library/ms713492 (vs. 85). aspx. *IEICE transactions on information and systems*, 92(12):2422–2429.