The Report Committee for Jacob Jason Rosales
Certifies that this is the approved version of the following report:

Mobile Computing in a Clouded Environment

APPROVED BY
SUPERVISING COMMITTEE:

Supervisor: _____
Christine Julien

_____
William Bard

Mobile Computing in a Clouded Environment

by

Jacob Jason Rosales, B.S.

Report

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

Master of Science in Engineering

The University of Texas at Austin

December, 2009

Dedication

I dedicated this report to Dionna, Alyssa, and Angelyna for their love, support, inspiration, and understanding. I love you all and thank you for standing by me these last couple of years. It was your belief and encouragement in me that allowed me to complete this report.

Acknowledgements

I would like to thank Professor Christine Julian and Professor William Bard for their guidance and constructive criticism they provided for this report. Without their domain expertise and influence this paper would not have been possible.

I would like to thank my friend and mentor Sandra Ellet-Salmoran, IBM Second Line Manager, for encouraging me to continue my education and apply to the Executive Master's Program at the University of Texas at Austin. Her encouragement has allowed me to increase my educational background and bring new and innovative ideas to IBM. A special thanks to my employer, IBM, for providing the financial support for my Masters education.

I would also like to thank my friend Margie Carrillo for her assistance in reading each revision of this report and providing valuable feedback and suggestions.

Last but not least, I would like to thank my family for believing in me and giving me inspiration to continue with my graduate education. Without their support, this would not have been possible. Thank you all for your love and support.

Abstract

Mobile Computing in a Clouded Environment

Jacob Jason Rosales, M.S.E.

The University of Texas at Austin, 2009

Supervisor: Christine Julien

Cloud Computing has started to become a viable option for computing centers and mobile consumers seeking to reduce cost overhead, power consumption, and increase software services available within their platform. For instance distributed memory constrained mobile devices can expand their ability to share real time data by utilizing virtual memory located within the cloud. Cloud memory services can be configured to restrict read and write access to the shared memory pool on a partner by partner basis. Utilization of such resources in turn reduces hardware requirements on mobile devices while lessening power consumption for each physical resource.

Within the Cloud Computing paradigm, computing resources are provisioned to consumers on demand and guaranteed through service level agreements. Although the idea of a computing utility is not new, its realization has come to pass as researchers and corporate companies embark on a journey of implementing highly scalable cloud environments. As new solutions and architectures are proposed, additional use cases and consumer concerns have been revealed. These issues range from consumer security, adequate service level agreements and vendor interoperability, to cloud technology standardizations. Further, the current state of the art does not adequately address these needs for mobile consumers, where services need to be guaranteed even as consumers dynamically change locations.

Due to the rapid adoption of virtualization stacks and the dramatic increase of mobile computing devices, cloud providers must be able to handle logical and physical mobility of consumers. As consumers move throughout geographical regions, there exists the probability that a consumer's new locale may hinder a producer's ability to uphold service level agreements. This inability is due to the fact that a producer may not have physical resources located relatively close to a mobile consumer's new locale. As a

consequence, producers must either continue to provide degraded resource consumption or migrate workloads to third party producers in order to ensure service level agreements are maintained.

The goal of this report is to research existing architectures that provide the ability to adequately uphold service level agreements as mobile consumers move from locale to locale. Further we propose an architecture that can be implemented along with existing solutions in order to ensure consumers receive adequate service levels regardless of locality. We believe this architecture will lead to increased cloud interoperability and decreased consumer to producer platform coupling.

Table of Contents

List of Figures

1.0 Introduction

Cloud Computing is neither a new idea nor a new technology; rather it is a combination of existing technologies applied in a specific environment to provide remote services to consumers. However, the current state of the art can be considered in its infancy as new requirements and concerns have emerged and consumers begin to investigate the value of remote resource consumption. If Cloud Computing is indeed, as we believe, a viable option for consumers to migrate physical resources to remote cloud environments, then legacy and emerging software stacks must be able to adapt to these ever-evolving requirements. We believe that one of the most notable constraints placed on cloud vendors is the non functional requirement that cloud producers shall provide a defined amount of resources throughout a given time interval, without degrading consumer performance. In existing software stacks, this requirement can easily be achieved as consumers maintain and manage the physical resources required by workloads. In a clouded environment, consumer resources are managed by cloud producers and can be provisioned across thousands of physical computing resources. As producers attempt to balance workload requests from multiple consumers without exceeding performance and resource thresholds as defined within the consumer service level agreement (SLA), periods may exist in which producers cannot satisfy SLA agreements. Further, these environments should not assume that the physical fabric between consumer and producer is static due to the increasing ability for consumers to dynamically change locales. Changes within the underlying fabric and location of a consumer directly impact and influence a producer's ability to remain within a given set of boundaries as defined within an SLA

Today, consumers are no longer restricted to a single geographical location due to increased usage of physical and logical mobile technologies [9, 17]. Mobile Computing has introduced new technologies that have dramatically changed the way existing infrastructures are designed and managed. Mobile devices can be expected to migrate in and out of network coverage areas, as well as dynamically appear without warning in different regions of the world. This type of mobility is commonly categorized as physical mobility. Logical mobility describes the movement of logical computing resources and software workloads to new physical environments. This category of mobility has increased as numerous virtualization stacks have evolved to provide consumers the ability to migrate workloads to new geographical regions with almost zero application downtime. As consumers continue to adapt virtualization stacks and increase usage of mobile devices, producers should extend cloud environments to dynamically relocate consumer workloads to physical computing resources that are located in a relatively close proximity to consumers. Doing so may increase the likelihood that a producer will be able to maintain the terms of the SLA.

It is our belief that Cloud environments should be flexible enough to transparently migrate Cloud workloads to either producer or third party resources located relatively close to a consumer's locale. As research within Cloud computing has begun to take off, we believe that the issue of Cloud interoperability can increase the likelihood that a producer does not violate an SLA, while also increasing the number of options mobile consumers have available to host workloads. The remainder of this paper has the following structure: in Section 2 we provide motivating examples supporting why cloud environments must become consumer locale aware. In Section 3 we discuss existing architectures that can be harnessed to support consumer mobility. In Section 4 we take a

look at the current state of the art of Cloud Computing. In Section 5 we investigate the current state of the art for mobile technologies. In Section 6 we provide an in-depth look at uses cases for mobility within cloud environments. In Section 7 we propose an architecture that can be implemented in order to decrease the probability that cloud producers will violate SLA agreements.

2.0 Motivation

Technology has increased the amount of resources and services consumers previously were unable to utilize. Today, it is common for these consumers to request services within one region and then migrate to a new region while still intending to utilize the same service(s) with almost no degradation. Mobile technologies and Cloud Computing can be utilized to decrease the probability of drastically degraded services. This section provides a motivating example to help illustrate the need for cloud producers to become consumer mobility aware, in order to continue providing highly available resources regardless of consumer locality.

Consider a mobile device that provides users the ability to practice Telemedicine in urban and remote regions. The device could provide services that allow for the uploading of real time emergency field data to a service provider's local repository, which in turn is retrieved by emergency technicians to process and assess. After an assessment, a mobile device would receive real time responses and act accordingly. As the device moves in and out of network range, it can be expected that the time it takes to transmit and receive data will degrade, depending on the relative distance to an access point. As depicted within **Figure 1**, when consumers move out of service range, network connectivity is lost, which could have drastic ramifications depending on the type of emergency. On the other hand, when consumers remain within range of a producer's coverage area, then services will continue to be provided accordingly. This model has two drawbacks. First, consumers are restricted to a limited range before losing services, and secondly consumers do not have the ability to switch service producers in cases where it may potentially be conceivable to do so.

Figure 1: Traditional Infrastructure

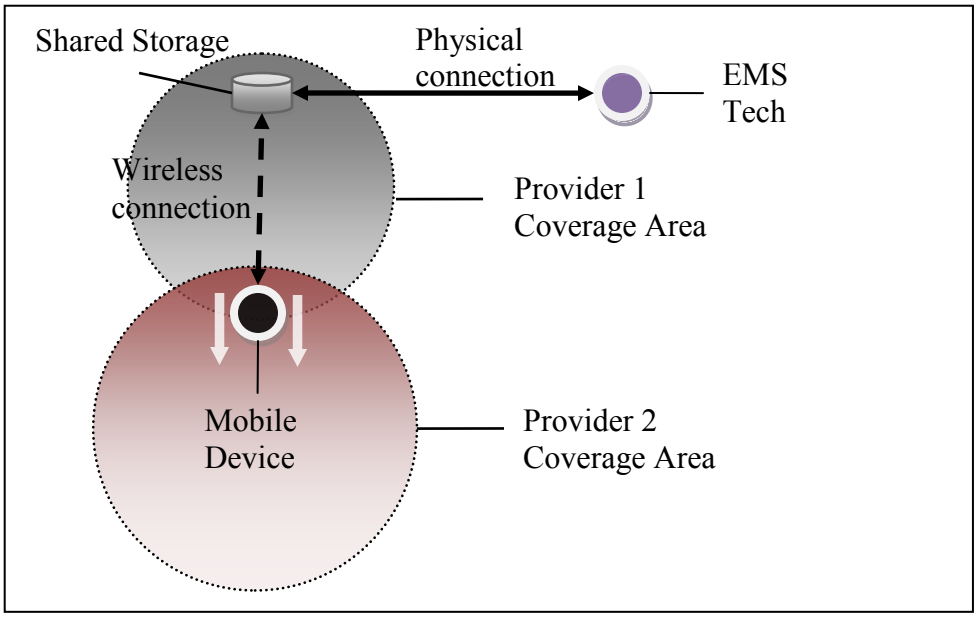
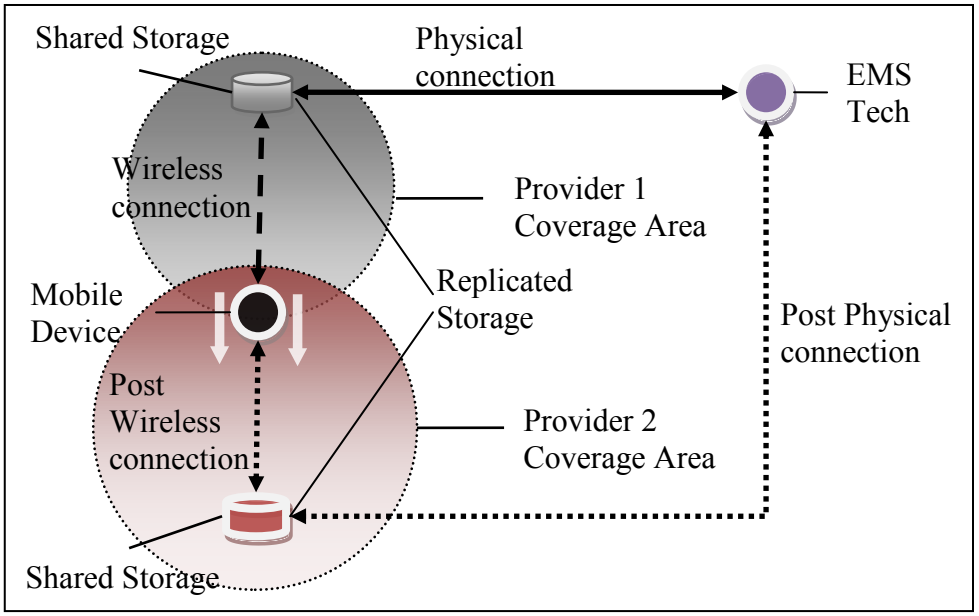


Figure 2: Desired Infrastructure



One plausible solution to this problem is depicted in **Figure 2**. Consumers would utilize application and storage services provided by cloud producers to send and receive real time medical data. Physical mobility would allow the device to report information as it moves into new contexts that can be utilized by cloud providers to initiate a logical migration of the consumer's resources. A migration could either be initiated to an additional cloud owned by the cloud producer or to a third party cloud environment, depending on the availability of producers within the region. In the event that a migration is initiated to a third party cloud, the mobile consumer should not be impacted due to the underlying infrastructure utilized by the new producer. It should still be technically feasible to maintain the same semantics as previously used with an initial producer, thus decreasing consumer to producer platform coupling [5]. Further a consumer's logical resources within the cloud may be dependent on external clouds, thus a migration may in fact cause degradation of these resources. To address this, resources can be replicated across a federation of clouds or mirrored in order to ensure real time synchronization to resources. Regardless of the methodology employed, autonomic procedures should be implemented to reduce the amount of internal and external resource degradation caused by the relocation of a consumer.

Although a plurality of solutions exists for scenarios such as this, the above embodiment depicts how Cloud and Mobile technologies can be harnessed to maintain service level agreements for mobile consumers. In the remainder of this paper we discuss the current state of the art and describe an architecture that can be implemented within cloud environments with minimal impact to mobile consumers.

3.0 Related Work

The current state of the art has provided valuable research in areas ranging from distributed computing, mobility, and resource provisioning. As Cloud Computing is a collection of existing and new technologies, within this section we discuss some of the relevant and innovative strategies that may be extended in order to implement the architecture defined within Section 7.0.

Within [7], the authors describe Cloud Computing as the 5th utility. The concept stems from the notion that users request the use of services through an on-demand basis and only be charged for the resources that they consumed and the duration of their use. This follows how current utility providers operate today. Consumers request the use of water or electricity, and in most cases do not know where the resources are being delivered or generated from. The computing utility idea operates in a similar manner, as consumers utilize cloud services to gain access to computing resources. The authors propose an approach to create a market-based cloud environment where multiple cloud venders can compete for consumer contracts by bidding on service requests. In order to create a market oriented cloud architecture, the authors rely on three critical actors, consumer, broker, and vendor. Consumers request compute resources. Brokers act as the middle man and attempt to locate the requested resources within the given constraints specified by the consumer. Vendors provide services and bid for jobs from both brokers and consumers. The architecture utilizes a SLA Resource Allocator to facilitate the communication between the consumer/broker and the cloud vendor(s). With this model, consumers are able to gain access to large amounts of resources for a fraction of the cost it would require for them to own and maintain the computing resources within their data center. Brokers are able to bring in income by negotiating lower utility prices from

vendors and providing this to consumers of a slightly higher cost. Vendors are able to generate income by leasing out resources within their cloud to brokers and consumers.

The MetaCDN project proposes to build a single namespace or point of entry for consumers to utilize Software-as-a-Service interfaces from within cloud networks [6]. Specifically, MetaCDN provides an interface that allows content owners to publish documents to cloud storage producers. This interface allows consumers to develop applications that are able to communicate through one interface, yet; have access to numerous supported cloud services. MetaCDN does this by providing connectors that utilize proprietary protocols to transfer content to and from the storage clouds. Further MetaCDN proposes to provide the ability to manage QoS for consumers wishing to stay within a specific price range and ensure they are receiving the amount of service as specified within their SLA. Consumers submit a URL to the MetaCDN gateway, which in turn attempts to locate a best match for the URL based on the current location of the consumer and the closest cloud producer. MetaCDN will then return the data back to the consumer where they can put/get information from the current cloud they are connected to. Although MetaCDN provides a mechanism where consumers are not highly coupled to specific cloud producers, it still introduces a single point of failure for clients and currently does not have any failover or disaster recovery mechanisms in place.

The authors of [11] propose Community Cloud Computing (C3). In order to create a community oriented environment, the underlying architecture utilizes concepts from Grid computing, EcoSystem computing, and Green computing. To minimize vendor lock-in, the authors propose C3 which allows for the public community to participate in creating a social compute cloud. To address the issues regarding green computing, the authors argue that although utilizing cloud resources does allow consumers to gain access

to large compute resources; it however does not minimize the cloud's carbon footprint. As adoption of cloud computing technologies increases, so will the amount of physical resources required by cloud producers, which will in turn increase the carbon footprint. C3 would allow for a reduction in the carbon footprint by allowing the cloud to dynamically grow and shrink as needed in order to provide adequate resources to accommodate consumer demands. In order to provide for such elasticity, the authors propose the idea of a community currency. As nodes provide resources, they are compensated by the community. When nodes require personal compute resources, they can then redeem the currency by requesting for the community to process their requests.

The authors of Mobile Service Clouds (MSC) propose the use of overlay networks in order to provide adequate services for mobile consumers [15]. Nodes within an overlay network can dynamically reconfigure and instantiate new services with minimal or no impact to consumers. Such networks provide the ability to locate the current fastest path to a given resource by using nodes within the network to identify the total time each known path will take to get data from the mobile consumer, through the network, and finally to the cloud resources. This same process is handled when the cloud provider is sending information back. Further the network implements high availability by requiring redundant adjacent nodes in cases of node outages. As disruptions are identified, the network dynamically instantiates new nodes to replace lost nodes. Although the benefits of an overlay network can potentially decrease SLA violations, the underlying infrastructure must be supported in all areas a consumer can migrate too. In most scenarios, such a requirement is not practical.

The Eucalyptus project is a Cloud Computing architecture implemented to give researchers and the community access to an open source cloud environment [13].

Eucalyptus is modeled after Amazon's EC2 [1] offering. Due to this dependency, the project currently offers only IaaS cloud services. One of the driving requirements for the project is to decrease vendor lock in, however, as researchers and enterprises deploy Eucalyptus within their data center, they in turn are potentially coupling their platforms and technology to a single cloud provider

The RESERVOIR project proposes an architecture that enables communication between heterogeneous clouds [14]. The authors believe that cloud federations provide more efficient use of resources and have the ability to ensure customer SLAs are not violated. This is achieved by offloading current consumer workloads to neighbor clouds. Further this decreases vendor lock in since a consumers workload can be deployed to heterogeneous clouds and still receive the required resources needed to satisfy their initial SLAs. RESEVOIR addresses these issues by proposing a three tiered architecture where each tier is loosely coupled to the others and the native technologies supporting the cloud environment. This loose coupling allows for workloads to be migrated and scaled transparently to local or remote nodes within the federated cloud.

Within [5], the authors discuss the notion of allowing clouds to be self manageable, provide a solution for resource submission, and finally discuss the ability for service negotiation bootstrapping. The authors propose the use of an Autonomic Manger that is the main entity within the architecture that allows for self management within a cloud. Within this component, a closed loop of control is implemented using monitoring, analysis, planning, and execution functions (MAPE). Each function is used to sense and identify changes within the cloud environment in order to successfully react adequately to change. Within the architecture the authors define a lifecycle used to ensure a self manageable cloud. The lifecycle starts with Meta-Negotiation, which consists of

negotiating with consumers the services and SLAs that need to be agreed upon before jobs can be submitted. Once SLAs have been agreed to, the execution phase can be initiated. Within the execution phase, the cloud processes requests for new submissions and potentially migrates existing jobs to other resources in order to ensure SLAs are not impacted due to the new job additions. The last phase is the post process phase. This phase takes care of restoring resources to the pool. Further the authors provide the idea of negotiation bootstrapping. This concept involves consumers submitting job requests to a central repository in order to locate providers that can host their applications within the specified constraints. As all consumer terminology cannot be expected to be similar, the central repository is used to analyze job requests. During the analysis phase, the component attempts to map the consumer's job request language to the provider's language. If the mapping is successful, a new document is returned to the user indicating the interfaces that are necessary in order to utilize the provider's service. This approach allows for consumers to have the ability to deploy workloads on different clouds without having to worry about vendor lock in. However, if the consumer is not fluent with regards to the set of returned interfaces, then resource utilization is prevented.

This SAGA project attempts to provide a set of interfaces that can be used to increase application interoperability between clouds and grids [12]. SAGA is a programming model that provides a rich set of abstract APIs that allow applications to interoperate with different clouds and grid environments. The goal is to allow applications to run unchanged regardless of which environment the application is deployed on. This is done by providing adapters at the back end of the API interfaces that are context aware of the current runtime environment. These adapters provide the required resources needed in order to function with the native stacks. Overall this

approach allows for applications to interoperate with various clouds that are currently supported.

Although these technologies and projects provide a basis for our goals regarding cloud interoperability, autonomous resource management, mobile awareness, and platform decoupling; to the best of our knowledge there currently does not exist a cohesive platform that implements these core requirements. Further, the presented innovative solutions do not take into consideration the ability for cloud producers to autonomously analyze and migrate mobile consumer workloads without violating SLA guarantees.

4.0 Cloud Technologies

As with all new paradigms in their infancy, Cloud Computing is currently in an evolving stage where research institutes and corporate vendors continuously attempt to define a globally accepted definition of Cloud Computing and what constitutes cloud environments [16]. Rather than provide a globally accepted definition, we propose that cloud computing describes the utilization of heterogeneous remote resources provisioned and guaranteed through the use of SLAs by third party cloud producers. The Cloud model utilizes existing concepts and technologies used within Grid and Utility computing, however, the application and guarantees provided by producers distinguishes Cloud Computing from existing paradigms.

Within Grid Computing, shared resources are provisioned and apportioned across a cluster of heterogeneous distributed systems to be utilized by specific compute-rich scientific applications [10]. Each node within the grid is allocated small portions of the global problem space. As nodes complete tasks, results are collectively combined to structure an overall solution for the given problem. Once acceptable results are obtained, the Grid can be reorganized and resources redistributed as needed.

Utility Computing describes the ability for consumers to purchase or lease computing resources on a pay-as-needed basis. For instance, a company may purchase a virtual server configured with 16 processors and 16 GB of memory, while only paying for 4 processors and 8 GB of memory. As workload demands increase, consumers have the ability to purchase a time based license which will enable additional computing resources on demand. As peak times subside, resource usage can be relinquished or revoked due to the terms of a lease. This model allows companies to provision smaller resources during off-peak hours and increase workload resource utilization during peak

usage times dynamically. The effect of this allows companies to reduce technology costs by decreasing compute resource waste and increasing workload scalability.

As workloads and consumer requirements rapidly change, there exists the need for an environment that provides the ability to quickly deploy and provide scalable computing resources on demand. Cloud computing provides such an environment with the use of service and virtualization stacks. Similar to the Grid computing paradigm, resources within a Cloud are shared across a set of computing environments. Management of these resources is performed by logically provisioning physical resources to one or more virtual environments running within the Cloud. As resource demand is increased, a resource monitor attempts to locate and allocate additional physical resources that can temporarily be utilized by a consumer. Once resources are no longer needed, the resource monitor returns the physical resources back to the available resource pool. Services within the cloud allow consumers to utilize a plurality of compute services, on a pay-as-needed basis. This provides the ability for compute utilities to finally become available to consumers regardless of their locale. These services range from low level storage services to large application infrastructure services. Regardless of the type of service, consumers are guaranteed of the reliability and availability of services. Throughout the remainder of this section, the three main concepts currently available within Cloud environments will be discussed.

4.1 Infrastructure as a Service

Infrastructure as a Service (IaaS) describes the ability to utilize remote hardware resources on a pay-as-needed basis. The resources can either be physical resources or

logical resources provided by the cloud's virtualization stack. Consumers build operating system image instances preconfigured with specific software stacks needed to host scalable applications. Once image construction has completed, consumers can deploy one or more virtual instances which will utilize the physical or virtual resources provided within the SLA. A common scenario involves virtual machines being created and deployed within a cloud. For example, Amazon's Elastic Compute Cloud (EC2)[1] provides infrastructure services to consumers. Within this cloud, consumers can quickly deploy virtual servers based on template or customized software stacks. Once an image is created, consumers can easily configure and scale the number of running virtual OS instances as needed. IaaS provides consumers the ability to quickly deploy and test solutions without having to purchase physical resources. Consumers seeking to test new ideas or concepts can do so with minimal upfront resource costs.

Figure 3: **Typical Cloud Software Stack Hierarchy**

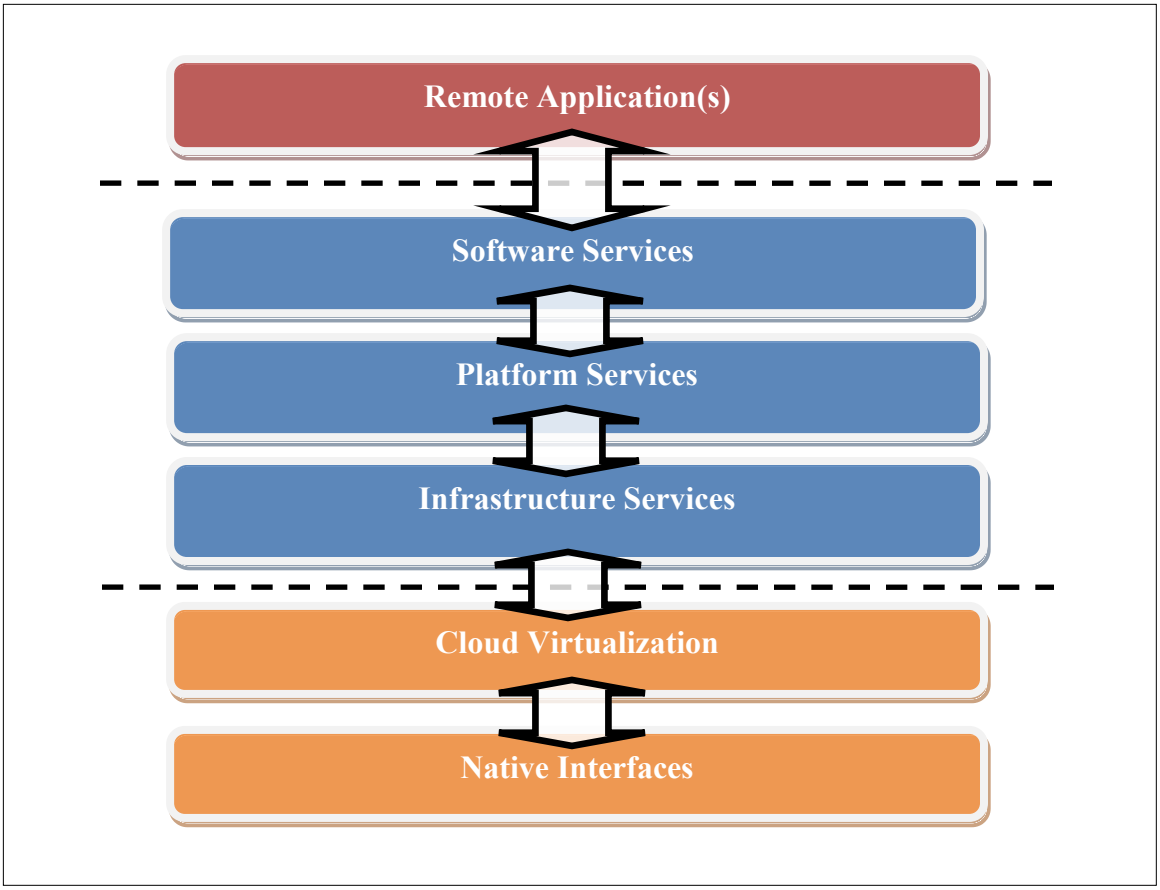


Figure 3 depicts a typical cloud stack hierarchy. The lower down the stack a service resides, the more flexibility it provides to consumers. However, along with flexibility, there exists the need to provide more consumer management of resources. For instance, with IaaS services, consumers are responsible for scaling images up or down as needed. Although this management can be automated, currently it is responsibility of the consumer to implement.

4.2 Platform as a Service

Platform as a Service (PaaS) describes the ability for consumers to develop and deploy dynamically scalable software workloads from within a well-defined cloud compute container. Applications created within this paradigm can take advantage of a rich set of well-defined internal services such as storage access, distributed database access, autonomic scalability, load balancing, fault tolerance, and security. Consumers perform all configuration, operation, and management tasks within the tools provided by the cloud. From a consumer's perspective, each application is self-contained through the use of virtual boundaries. These boundaries allow each workload to run within their sandbox simultaneously along with hundreds of other virtualized sandboxes. As depicted in **Figure 3**, PaaS components are architected on top of virtualization stacks which provide the ability to securely segregate virtual OS instances and increase resource utilization.

One of the most notable PaaS platforms is Google's AppEngine [3]. AppEngine provides consumers the ability to utilize Google's Web Services and internal SDKs. Load balancing and scalability are automatically handled by Google's infrastructure, and consumers are only billed for the amount of resources each application consumes.

4.3 Software as a Service

Software as a Service (SaaS) describes the ability to deliver software solutions over a remote interconnect to consumers [8]. Within this new paradigm, consumers are no longer required to purchase "*shrink-wrapped"* software packages; rather they can incorporate remote software solutions into client side applications through the use of

standardized communication protocols or utilize full-fledged applications through a web browser. Typically the former solutions are architected based on Service Oriented Architectures (SOA), which allow producers to provide a semantic interface that consumers are able to use to integrate within client side applications. The latter provides consumers the ability to access applications without regard to geographic locations, as long as there exists an available interconnect to the application. When SaaS solutions are incorporated within a cloud environment, consumers benefit from the same scalability, load balancing, and fault tolerant features as provided within PaaS services.

Due to the popularity and benefits of SOA architecture, SaaS solutions pre-date the current Cloud paradigm. SaaS inclusion within Cloud environments allows for a logical evolution of the current state of the art to make available software solutions on a broader scale. Producers can deploy custom services within a third party cloud to take advantage of cloud benefits, while at the same time increasing its commercial exposure. Companies such as IBM, Google, and Salesforce.com, have begun to transition existing *"shrink-wrapped"* solutions and services into their respective cloud service counterpart. As mobile device technology continues to increase, usage of these services will increase as many only require a minimal set of resources in order to run efficiently.

5.0 Mobile Technologies

We believe mobile technologies will fuel the adoption, evolution, and advancement of Cloud Computing. Currently there exist two types of mobile technologies, physical device mobility and logical resource mobility. Both technologies have advanced to a state where the needs for physical resources are no longer a concern in order to provide adequate compute capabilities for consumers. As each technology continues to mature, new requirements and constraints will necessitate new advancements for cloud producers. Most notably is the ability for a physical and logical compute resource to rapidly change physical locations at any given instance of time. Advantages in the physical mobility context allow consumers to roam freely across regions and communicate over ad-hoc networks. On the other hand, logical mobility provides the logical movement of software stacks at a fine grained level. In the remainder of this section we discuss these two concepts and their impacts on Cloud Computing

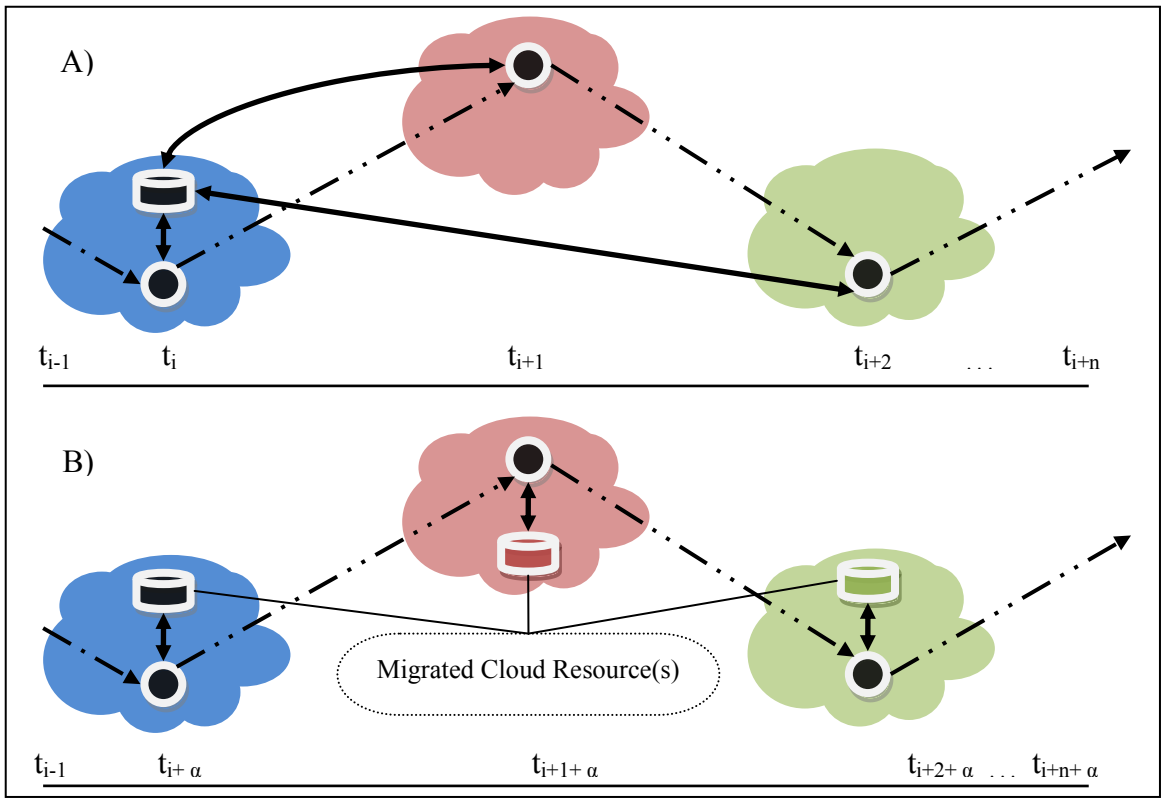
5.1 Physical Mobility

Physical device mobility describes the ability for computing devices to roam unconstrained throughout geographical regions. Although devices have the ability to roam while in an unpowered state, we only focus on "*live"* physical mobility. This type of mobility entails the consumption of computing resources while physical entities still contain sufficient power to adequately provide services to consumers. As mobile consumers roam throughout regions, the probability for communication latency to diminish or enhance, increase or decrease depends on the proximity to local access points. As devices move out of range, remote compute resources become unavailable

until a device is able to reengage network communication. Further there exists the possibility devices may communicate through several access points at one time. This ability provides potential for a best path route to be defined in order to improve communication latency and remote resource access. Much work has been performed within the area of best-path communications within mobile networks; however to our knowledge no research currently exists in determining the best cloud service provider given a mobile device's current contextual information.

As device and sensor technology continue to advance, so do the software stacks that are deployed on them. Today, it is common for mobile devices to have GPS tracking, accelerometers, audio recorders, etc. In the near future, it is conceivable that these capabilities can be utilized to harvest contextual information regarding a consumer's current locale, which in turn can be transparently submitted to one or more cloud producers. The context granularity may range from latitude and longitude coordinates to the sounds of a thunderstorm on the horizon. Regardless of the amount of data being captured, the information can be analyzed by cloud producers to determine if migration of resources to a new cloud closer in proximity to a consumer's current location is beneficial for both parties.

Figure 4: Cloud Consumer Scenarios



A) Dipicts a mobile consumer accessing the same cloud resources from three sperate providers.
B) Depicts a mobile consumer accessing the same migrating cloud resource permiting it is in an intransiant state for some α time interval

Within Figure 4, a mobile device is within range of three different cloud providers at three different time intervals. As a device moves in and out of range of each provider, there exists the possibility that overall resource downtime is decreased in the event that workloads can be transparently migrated to each node along the path, or access to the resources is available through one or more providers. In its simplest case, a cloud producer can maintain SLAs with mobile consumers only if each device can utilize communication services from third party clouds along the path. This follows the more traditional solution of reconnecting to existing infrastructures to continue remote resource

usage. However, in most cases, assumptions are made regarding the network fabric that all networks are created equal. In practice this assumption is not always true and may cause network latency degradation. When a mobile device does not become transient, there may exist periods where movement of an entire workload or resource to a third party cloud may increase network latency. In Section 6.0, we provide further details of motivating use cases for the need to increase mobile consumer awareness within clouds.

5.2 Logical Mobility

Logical mobility describes the utilization of technologies that support and provide the ability of logical computing resources to be transparently migrated to new physical computing environments. These logical resources can be either entire workloads or fine-grained portions of a workload that can perceivably increase the benefit for consumers by being executed on an external host system [17]. Today, logical mobility of operating systems and workload applications has seen a dramatic increase in usage and benefit as companies continue to adopt virtualization stacks. This adoption is due to the ability to migrate resources to new physical hosts when there is a need for a scheduled maintenance on the source host. Rather then force an outage for all consumers, the logical resources can be migrated with almost zero downtime without diminishing the workload's availability. In most scenarios, consumers are unaware of this process and continue on without having to restart applications consuming resources from the workloads.

Within the mobile context, mobile code allows for resources to almost leap off a device onto passing devices or latch onto remote servers for processing, while eventually migrating back to the original mobile device. Upon return, the application that spawned

the mobile code can extract computation results and incorporate them into the system as needed. This ability allows for highly intense computational software to be executed in a cloud to decrease the amount of processing time required to compute and retrieve large amounts of data.

Whether migrating monolithic workloads or finely grained mobile agents, cloud producers must be able to handle logical mobility of these consumers as they rapidly change locations. Unlike mobile devices, logical resources have the ability to move vast distances within short timeframes. OS instances and workloads can be moved either within a datacenter or to cross country datacenters depending on the needs of the consumers. Mobile code fragments can potentially move miles away from their mobile counterpart by quickly hopping from node to node within a distributed environment. As each resource moves, context data should be obtained and sent back to cloud producers for analysis on how cloud resources should be apportioned. Within these scenarios, cloud resources and workloads should be migrated to neighbor clouds within the federation in order to satisfy all parties.

6.0 Use Case Analysis

As Cloud Computing continues to evolve, new use cases and requirements will arise from consumers. Influence from the enterprise to mobile consumers will have a dramatic effect on how producers architect and implement cloud environments. Mobile Computing device usage and advancement will continue to grow, while they in turn will require efficient, secure, and scalable platforms to provide services for mobile consumers. Within the enterprise, corporations will begin to adopt Cloud technologies in order to reduce IT costs and improve resource utilization.

Within the near future, it is conceivable to imagine mobile device deployment will become such a norm that you can actually purchase aerosol canisters containing micro devices which can literally be sprayed into any environment [18]. Although small in stature, these devices will still require the ability to transmit data to central repositories so that data can be analyzed. On the other spectrum, consumer consumption of smart phones will increase traffic congestion within wireless networks due to lack of infrastructure. The speed at which mobile deice adoption is occurring is outpacing the enterprise's ability to support such demand. Today, mobile devices utilizing web enabled operating systems such as Google's Android [2] or Palm's WebOS [4], are now pushing workloads to the cloud instead of remaining located on the physical device. As these devices push and pull more data from the network, communication traffic begins to become congested. In these scenarios, if there exists the ability to transfer such workloads to other clouds based on a consumer's locale, then such measures should be enacted. As we believe enterprises and mobile devices will be the primary influences on the advancement of Cloud Computing, the remainder of this section we expand on our Telemedicine scenario and provide some key requirements that cloud environments must

implement within their architectures in order to increase SLA guarantees for mobile consumers.

6.1 Telemedicine

Revisiting our telemedicine scenario as described in Section 2.0, the utilization of mobile devices is a key enabler in order for this technology to become prevalent across the globe. Whether it is a mobile robot or smart device, these entities allow for the administration of medical practices to be performed on patients in remote regions. Consumers of these devices require the ability to send and receive real time data in order to adequately perform examinations or surgical procedures. Random drops in wireless network communication are unacceptable and could have drastic repercussions. Further, as time is a critical component within the administration process, it is imperative that the data being transmitted be analyzed efficiently and satisfactory responses be provided to the mobile consumer as quickly as possible. In the event that patients require transportation, the mobile consumer will likely be required to accompany them. As the consumer changes locales, remote resources should not degrade; rather they should continue to be highly available.

Although the movement of cloud resources may be a data intensive operation, the potential for continued service level guarantees regarding remote resources being highly available and reliable has higher precedence. As consumers move closer to the outer regions of a cloud provider's coverage area, and there exists an alternative cloud within range, then remote resources should either be migrated to the new cloud, or communication through the new cloud should enable continued resource access.

Consumers should not be required to make radical changes in order to utilize resources due to these types of events. If a standard protocol is being employed over the transport, then continuation of remote resource consumption requires only the change in access points. In the event the third party cloud does not understand the semantics of the new cloud provider's service protocol, then a protocol correlation mapping between what the consumer and producer comprehend should be utilized in order for the producer to *learn* the consumer's service language [6]. If the semantic mapping of the consumer language is successful, then remote resources should be accessible without any protocol alterations within the consumer's platform. The cloud producer should then execute a negotiation protocol with the consumer to verify resource relocation is acceptable. Once agreed, security primitives should be exchanged so that the consumer can access the resources from the new cloud. If either the semantic mapping or negotiation process fails, then the cloud producer should continue to provide resource access.

As with mobile devices, the same scenario can be applied to logical devices migrating around the globe. Within our Telemedicine example, an EMS technician, doctor, or data analysis application is consuming data provided by the mobile device, which is stored on shared cloud resource. In order to access this data, the application must harvest data from the cloud, analyze it, and then provide a response. In the event a technician is unable to supply a sufficient analysis, the technician's application may be migrated across the country and made available to an expert for investigation. Within this scenario, a checkpoint of the application is initiated where no further modification of the programs state is allowed. The next step is to migrate the application's logical resources, such as memory or CPU state, over the network to the destination physical compute resource. Due to the movement of these logical resources, communication latency may

degrade depending on the relative distance between the cloud resources and the logical resource's new location. To decrease the amount of degradation, cloud resources should either be migrated or replicated across clouds. In the latter approach, techniques should be implemented that allow for all clouds to converge to a satisfactory synchronized state where the same data can be retrieved from either cloud. Regardless of the approach, the mobile consumer should have the ability to inform the producer at the start and end of a migration operation. This will enable producers to begin orchestrating resource provisioning and cloud negotiation in an attempt to reduce SLA violations.

From this use case, we propose that the following key requirements be architected as core functionality within a cloud provider's implementation:

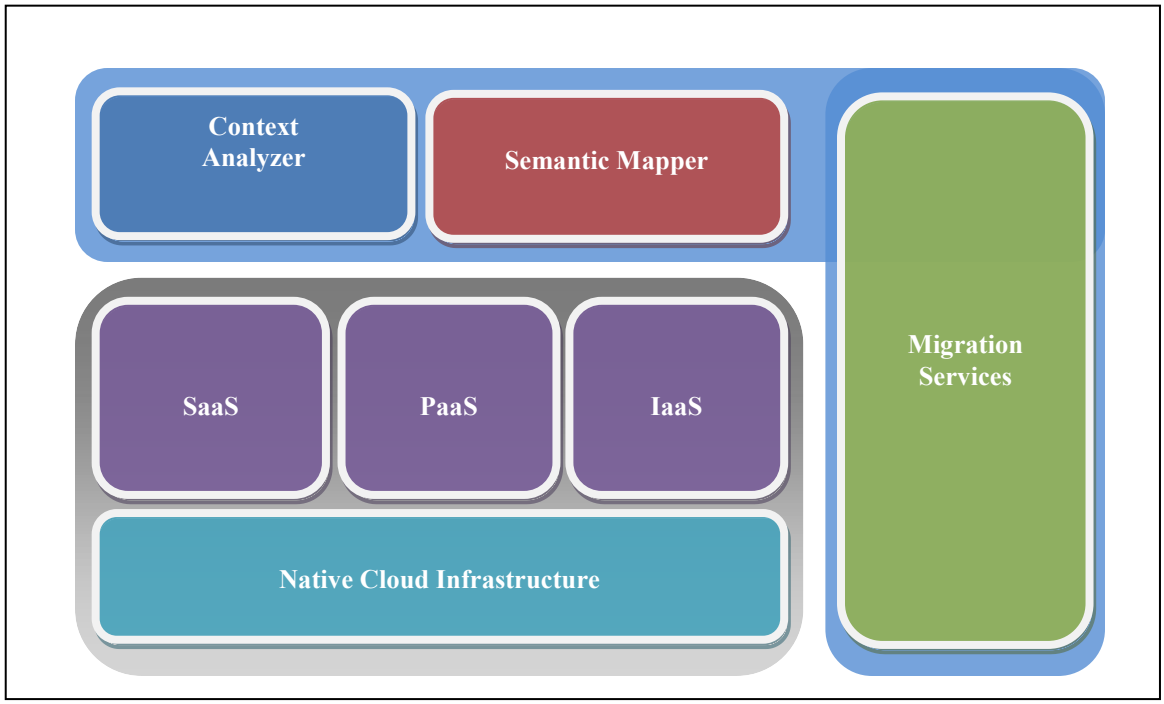
- **Mobile Consumer Awareness** – Cloud producers shall request and analyze consumer contextual information in order to identify potential cloud resource migration candidates.
- **Semantic Protocol Mapping –** Cloud producers shall perform a protocol mapping in order to provide resources to new mobile consumers.
- **Cloud Interoperability -** Cloud producers shall use a standard protocol that allows for federated cloud communication in order to negotiate resource migration.
- **Secure Consumer-to-Producer Resource Migration Negotiation –** Cloud producers shall provide a secure negotiation protocol when consumer resources are migrated.

We believe having these key requirements as core functionality within a cloud can increase SLA guarantees, decrease consumer to producer coupling, and increase cloud interoperability.

7.0 Architecture

In order to provide consumer location awareness within clouds, producer infrastructures should be architected with core consumer context aware services. As new location context metadata is received and analyzed, producers should outsource workloads to neighbor clouds with the consensus of consumers. In order to reduce platform coupling while increasing consumer infrastructure interoperability, cloud infrastructures should provide protocol semantic mappings in order to ensure continuation of services once migration operations have completed. Figure 5 provides an architecture blueprint that can be implemented to provide such services on along with existing cloud environments.

Figure 5: Core Cloud Architecture



Whether workloads are outsourced due to consumer context or cloud workload rebalancing, target producers benefit by either charging consumers or source producers for resource consumption. In essence a cloud economy is formalized as small and large cloud vendors accept and refer consumers to new resource outlets [11, 14]. In the remainder of this section conceptual design points regarding the extended service set are provided.

7.1 Context Analyzer

The context analyzer component is responsible for requesting, receiving and analyzing consumer context information. Receive and request transactions allow the component to maintain an up-to-date context repository for all consumers currently utilizing resources. Analysis capabilities allow the component to make real time migration decisions as needed in order to maintain consumer SLA agreements.

Metadata harvesting can be performed either proactively or reactively. In the former approach it is the responsibility of the component to begin issuing context discovery requests to consumers. As each response is received, the component updates its metadata repositories. In the latter approach, consumers push context data to the cloud to provide assistance in ensuring SLA violations are minimized, as it is also in a consumer's best interest to reduce resource outages as much as possible. Due to consumer platform heterogeneity, utilization of standard semantic information or description languages such as XML or OWL should be employed in order to increase the cloud's consumer base and vocabulary. As new consumers are encountered, the context component can learn new

vocabularies and automatically perform resource migration operations as needed based on this new data.

Within the analysis phase, the component either periodically or instantaneously performs analysis on the content repository. The former methodology is required as it allows for the component to periodically scan the content repositories in search of potential consumer workloads to be migrated. Instantaneous analysis is performed in the event alerts are fired when SLA agreements are in jeopardy of being breached, a consumer has moved to a closer distance to a neighbor cloud, or the cloud is currently experiencing high stress volumes. As analysis is performed, each consumer is placed within a tier to indicate its potential for resource migration. Workloads that have been starved or require immediate migration are placed into high priority tiers. Medium tiers contain workloads owned by consumers who have moved further from the cloud. Low priority tiers are used to house workloads not in high violation risk or currently being deployed within the cloud.

As workloads are placed within the high priority tiers, the component will begin interacting with the migration layer in order to identify potential neighbor clouds that can accommodate new workloads as specified by a consumer's SLA. To prevent consumers from causing migration *thunderstorms*, each consumer is assigned a set of last service timestamps within the repository. If a consumer has been seen to routinely be migrating in and out frequently within the cloud, then the workload is moved to a lower tier in order to minimize overutilization of physical resources. Such scenarios allow for a first line of defense from preventing workloads to cause havoc within the cloud such as Denial-of-Service attacks.

Finally, each consumer is given a reputation profile that is stored within the cloud repository. Within this profile, the consumer is assigned a reputation level along with a list of past cloud violations. The analyzer will utilize this information along with context metadata when placing each workload in its respective tier. Consumers placed within high priority tiers but assigned a poor reputation may still be migrated; however they will be placed at the tail of a migration queue. As a consumer's reputation increases, tier placement and migration placement may be relaxed to allow the workloads to migrate frequently.

7.2 Semantic Mapper

As workloads become candidates to migrate to and from legacy or third party clouds, the service protocols utilized from the consumer's platform should not be required to change. Such constraints not only prohibit mobile consumers from discovering new cloud environments to deploy workloads within, but also increase consumer-to-producer platform coupling. In order to provide for a loose coupling, a semantic mapping between the consumer's SLA [5] and current protocol service interfaces is required. This ability allows for consumers to utilize resources without requiring infrastructure domain knowledge for the new cloud. The semantic mapping component provides such capabilities.

When a consumer initially establishes connection with a cloud, a snapshot of the service interfaces utilized is taken and saved within the cloud repository. In order to increase the potential that a consumer's workload can be moved to a new cloud without hindering the total number of services available, the interface description should be saved

in a semantic or descriptive markup language. This structuring of data in this format allows for both the source and target neighbor cloud to exchange domain interface knowledge. Knowledge repositories managed either on both clouds or an external broker can be leveraged in order to increase the chances that such a mapping can be successful.

In order to provide consumers continued resource utilization even in a limited fashion, the results of the semantic mapping will be sent to consumers for consensus. If consumers are in agreement with the amount of services that can be accessed within the destination cloud, then an acknowledgment response will be sent. At this point the component will inform the migration layer to initiate workload check pointing. In the event the consumer does not consent to the migration, a new neighbor cloud is searched for that can provide a larger set of services. If all discovery options have been exhausted, the cloud will continue providing resources and periodically attempt to locate a new optimal host for the consumer's workload.

7.3 Migration Services

In order for consumer resources to have the ability to migrate to neighbor clouds, each cloud involved in the migration must understand the same migration protocols. As commercial clouds utilize proprietary virtualization technologies, the odds of migrating workloads between heterogeneous neighbor clouds quickly diminish. The RESERVOIR project provides an architecture that can be harnessed in order to overcome these challenges by providing a layer of abstraction on top the native cloud infrastructures [14]. We propose to extend the technology used within RESERVOIR, to allow mobile consumers to securely reinitiate the use of resources once they have been migrated to a

new host. In order to prevent resources from being reserved for prolonged periods of time, the migration reservation service should relinquish resources when mobile consumers fail to initiate service requests. Further, after a migration has been completed, resources should only be exported to authorized consumers.

Due to the heterogeneous nature of cloud environments, a common security methodology should be employed to allow for authentication and authorization amongst disparate systems. To address these issues, we propose to include a set of security primitives contained within a manifest file that will allow consumers and producers to authenticate one another in order to successfully reengage resource consumption.

Within RESERVOIR, cloud workloads and infrastructures are deployed within the virtual environments with all resources and constraints defined within the service manifest document. The document contains key information such as virtual resource requirements, elasticity constraints, etc. As this document is used for all configuration and deployment within the cloud, we propose to extend the manifest document to contain a security primitive section that can be used authenticate and authorize mobile consumers before resources are provided. The key elements to the security portion could potentially be the following:

- **Source Consumer Session Identifier** – Identifier assigned to a consumer after a successful authentication and authorization process.
- **Source Cloud Producer Identifier –** Unique identifier of the source cloud provider
- **Destination Cloud Producer Identifier** – Unique identifier of the destination cloud provider.

- **Migration Temporal Consumer Token** – Unique temporal identifier generated for a consumer for the duration of a migration process. The generation of the token uses input from both source and destination cloud producers.

Once a mobile consumer initiates service requests, it is issued a session identifier. This identifier is used to allow the consumer to continue utilizing services for a predefined period of time before being requested to re-authenticate. When a consumer's workload is eligible for migration, the migration component will begin negotiation deliberations with the destination cloud. Once it has been determined that migration should be initiated, the source cloud will store the source consumer session identifier within the manifest file. This token should be generated in such a manner as the destination producer can identify where it originated from. To assist in this process, the source cloud producer's public identifier is stored within the document. The source cloud producer will also store the destination cloud identifier so that the consumer can extract the information in order to verify it is communicating with the correct cloud provider after a migration. The last primitive is used to provide a migration temporal token. This token is used to indicate the maximum duration the destination cloud will reserve the resource before releasing them in the event the consumer fails to initiate service requests within this time interval. Once the manifest document has been persisted, two versions should be digitally signed and encrypted by the source cloud. Each version will be secured in a fashion where each respective recipient is able to decipher the manifest document.

When a consumer begins to request resources from the new cloud, both parties should verify they are communicating with the intended entities by verifying information contained in their versions of the manifest file. On a successful verification, the

consumer's resources are made available. In the event a consumer fails to initiate service requests, the destination cloud producer should notify the original source cloud of the timeout. This notification allows for resources within the source cloud to become available in scenarios where the mobile consumer has moved out of range of the destination cloud and back within range of the source cloud.

8.0 Conclusion

As Cloud and Mobile technologies continue to evolve, so do the requirements elicited from consumers. In order to account for dynamic changes, cloud environments must provide a rich set of core functional features that can be utilized by mobile consumers. Today, the ability for consumers to physically or logically move throughout different geographical regions, places new constraints on cloud producers to ensure service level agreements are maintained. Since service levels provide guarantees that resources will be presented as reliable and highly available, it is in the best interest of cloud producers to interoperate with neighbor clouds. This interoperability allows the formation of a federation where new cloud economies can thrive, generating more revenue and attracting new consumers.

The current state of the art provides promising avenues in the fields of virtualization and mobility. However, existing cloud environments have not been architected with consumer mobility as a key requirement. As the number of mobile devices continues to increase, it is conceivable that clouds will become overburdened and begin to violate SLA agreements. To minimize violations, cloud interoperability allows for workloads to be migrated on demand or based on contextual information harvested from consumers. As workloads are migrated, consumers and producers should have the ability they to verify they are communicating to the expected entity irrespective of the protocol being utilized. This is accomplished through the exchange of security primitives each entity authenticates before resources are authorized and consumed. In order to minimize platform coupling, the use of semantic protocol mappings can be implemented to increase consumer interoperability while decreasing coupling. By incorporating these

key elements within the architecture, cloud providers increase consumer exposure, minimize SLA violations, and reduce platform coupling.

References

[1] Amazon elastic compute cloud. URL http://aws.amazon.com/ec2/.

[2] Android. URL http://developer.android.com/index.html.

[3] Google app engine. URL http://code.google.com/appengine/.

[4] webosdev. URL http://developer.palm.com/.

[5] I. Brandic and A. Vienna. Towards Self-manageable Cloud Services.

[6] J. Broberg, R. Buyya, and Z. Tari. MetaCDN: Harnessing 'Storage Clouds' for high performance content delivery. *Journal of Network and Computer Applications*, 2009.

[7] R. Buyya, C. Yeo, S. Venugopal, J. Broberg, and I. Brandic. Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility. *Future Generation Computer Systems*, 25 (6): 599–616, 2009.

[8] A. Dubey and D. Wagle. Delivering software as a service. *The McKinsey Quarterly*, pages 1–12, 2007.

[9] L. Johnson, A. Levine, R. Smith, and T. Smythe. The 2009 Horizon Report: K. *Austin, Texas: The New Media Consortium. Cover photograph:"Chapped Lips" by Vox_Efx on Flickr (http://www. flickr. com/photos/vox_efx/3186014896/). Creative Commons*, page 3, 2009.

[10] N. Kaushik. *Distributed resource usage: Lightpath allocation, Advance reservations with guarantees, and Co-alloca-tion waiting-time.* PhD thesis, Santa Clara University, 2009.

[11] A. Marinos and G. Briscoe. Community Cloud Computing. *Arxiv preprint arXiv:0907.2485*, 2009.

[12] A. Merzky, K. Stamou, and S. Jha. Application Level Interoperability between Clouds and Grids.

[13] D. Nurmi, R. Wolski, C. Grzegorczyk, G. Obertelli, S. Soman, L. Youseff, and D. Zagorodnov. The eucalyptus open-source cloud-computing system. *Proceedings of Cloud Computing and Its Applications*, 2008.

[14] B. Rochwerger, D. Breitgand, E. Levy, A. Galis, K. Nagin, I. Llorente, R. Montero, Y. Wolfsthal, E. Elmroth, J. Caceres, et al. The reservoir model and architecture for open federated cloud computing. *IBM Systems Journal. Submitted for publication*.

[15] F. Samimi, P. McKinley, and S. Sadjadi. Mobile service clouds: A self-managing infrastructure for autonomic mobile computing services. *Lecture Notes in Computer Science*, 3996: 130, 2006.

[16] L. M. Vaquero, L. Rodero-Merino, J. Caceres, and M. Lindner. A break in the clouds: towards a cloud definition. *SIGCOMM Comput. Commun. Rev.*, 39 (1): 50–55, 2009. ISSN 0146-4833. doi: http://doi.acm.org/10.1145/1496091.1496100.

[17] S. Zachariadis, C. Mascolo, and W. Emmerich. Satin: A component model for mobile self organisation. *LECTURE NOTES IN COMPUTER SCIENCE*, pages 1303–1321, 2004.

[18] F. Zambonelli, M. Gleizes, M. Mamei, and R. Tolksdorf. Spray computers: Explorations in self-organization. *Pervasive and Mobile Computing*, 1 (1): 1–20, 2005.

Vita

Jacob Rosales was born in El Paso, Texas, son of Jose Rosales III and Georgia Rosales. He received a Bachelor of Science in Computer Science from the University of Texas at El Paso in 2006. After graduation Jacob joined IBM Corporation in Austin, Texas as a Software Engineer. At IBM he designs and implements software for IBM's Advanced POWER Virtualization software solution.

This report was typed by Jacob Jason Rosales.