A Survey on Cloud Interoperability

Zhizhong Zhang
zzzhang@cs.hku.hk

Chuan Wu
cwu@cs.hku.hk

David W.L. Cheung
dcheung@cs.hku.hk

ABSTRACT

Cloud computing describes a new distributed computing paradigm that allows users with different computing demands to access a shared pool of configurable computing resources (e.g., servers, networks, storage, database, applications, and services) that can be rapidly provisioned and released over the Internet with minimal user-management effort or cloud-provider interaction. However, with several commercial cloud providers coming up, each with their own APIs, application description formats, and varying support for SLAs, vendor lock-in has become a serious and inevitable issue for cloud end users. Cloud Interoperability is crucial to enabling sharing of resources from a pool of cloud-service providers in a seamless fashion. The current state-of-the-art involves the concerted efforts from both the research and industry community. In this paper, we have made a comprehensive survey on the interoperability of different cloud vendors and cloud platforms, along with the efforts towards cloud interoperability from both research and industry. We hope to pave way to the forthcoming research on cloud interoperability.

> *Cloud computing has become a lot like the Hotel California: Once you pick a provider you can check out anytime you want — but you can never leave.* [1]

1. INTRODUCTION

Cloud computing [2, 3] can be defined as accessing third party software and services in a pay-as-you-go manner. It facilitates scalability and virtualized resources over Internet as a service providing cost effective and scalable solution to customers.

From the cloud users' point of view, cloud users have a strong motivation to move into the clouds from their existing infrastructure, due to the flexibility, stability and scalability brought by cloud computing. For example, Netflix [4] no longer runs its own data center and shift its applications to Amazon's cloud instead. On the other hand, cloud users will also feel reluctant to stay in the clouds offered by a single provider, because it might not be able to satisfy all the needs of the cloud users, such as geo-location of the data centers, poor SLA, or ridiculously high prices. Moreover, the cloud users are most likely looking for interoperable clouds where they could deploy their cloud applications and migrate seamlessly.

From the cloud providers' point of view, the increasing competition between different vendors in the cloud market, such as Amazon, Google, Microsoft and SalesForce, each of which promotes its own, incompatible Cloud standards and formats. This incompatibility among cloud providers can protect the interest of each provider, but not in the long run. And this incompatibility also indicates that the cloud market is still in its infant and immature stage. Moreover, small cloud providers are more motivated towards federation, rather than monopolies like Amazon, Google, etc, which is witnessed by the fact that most open-source cloud platforms are initiated by relatively small cloud providers (e.g. Rackspace, GoGrid) and late-comers (e.g. Red Hat, Dell, Oracle, etc.). The emerging need for interoperable clouds forces them to agree upon a widely accepted, standardized way to import/export cloud details and specifications.

The importance of cloud interoperability has been highlighted by the European Commission, which aims to coordinate the efforts to "identify by 2013 a detailed map of the necessary standards (inter alia for security, interoperability, data portability and reversibility)". Cloud interoperability is also important to the cloud users, who concern about "How can I keep control over my data?", "Will your cloud platforms help me migrate my existing technology investments to the cloud and how do I use private clouds?"

In order to better define the problem of cloud interoperability, it is important to understand what is interoperability. Typically, it means the ability for different heterogenous systems to be able to function/interact together. For clouds, interoperability could be defined as the ability to understand each others' application formats, service level aggrement (SLA) templates, authentication and authorization token formats and attribute data. In Cloud Computing, the interoperability between two different cloud providers refers to their ability to **cooperate** or else **interoperate**, thus establishing a federation of clouds [5]. Therefore, interoperability is a prerequisite for cooperation.

The industry tries to address the cloud interoperability issues via *standards* and there have been many cloud standards being proposed and even put into use in recent years. Although those standards are not fully compatible with those adopted by giant cloud vendors, they proved to be effective with the help of third-party cloud brokers (i.e. RightScale). However, they will take years to be fully agreed upon and adopted, if ever. Apart from standardization, which is by definition provider-centric, researchers among the academic

community have advocated a user-centric approach that gives users an unprecedented level of control over the virtualization layer.

Taken both the provider-centric and user-centric approaches in mind, we try to take a deep survey on both, but with more emphasis on the provider-centric approach. In the provider-centric approach, we analyze the cloud interoperability issue in the following three angles: *taxonomy*, *standards* and *implementations*. In the user-centric approach, we introduce and summarize some research projects that take advantage of nested virtualiztion [6] (nested virtualization is a technique that allows multiple hypervisors to run on top of an underlying hypervisor, thus enabling a more flexible virtualized infrastructure).

The rest of the paper is organized as follows. Section II discusses the definition and different angles of cloud interoperability. Section III presents the taxonomy of IaaS interoperability. Section IV introduces three mainstream cloud standards and their comparisons. The specific implementations of those standards are presented in section V. Section VI will address the cloud interoperability problem in a different angle: User-Centric approach. Last but not the least, section VII will conclude this paper.

2. OVERVIEW OF CLOUD INTEROPERABILITY

2.1 Definition of Cloud Intereoperability

Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. This cloud model is composed of five essential characteristics (*On-demand self-service*, *Broad network access*, *Resource pooling*, *Rapid eslasticity*, *Measured Service*), three service models (*SaaS*, *PaaS*, *IaaS*), and four deployment models (*Private Cloud*, *Community Cloud*, *Public cloud*, *Hybrid cloud*). [7].

According to the European Commission (EC) [8], interoperability is defined as the ability of Information and Communication Technology (ICT) systems and of the business processes they support to exchange data and to enable the distribution of information and knowledge. According to the literature review, Cloud computing **interoperability**, **compatibility** and **portability** are closely related terms and often confused. Therefore, Cohen clarifies the similarities and the differences among these terms in an attempt to exemplify and differentiate them: *Cloud computing interoperability is the ability for multiple Cloud providers to work together or interoperate. Cloud Compatibility and Portability answer to the question "how?". Cloud Compatibility means application and data that work with the same way regardless of the Cloud provider, whereas Cloud Portability is the ability of data and application components to be easily moved and reused regardless of the provider, operating system, storage, format or API*

2.2 Service Models

The cloud computing community has widely adopted the following three service models to categorize cloud services [9]:

- *Infrastructure as a Service (IaaS)*. The capability provided to the consumer is to provision processing, storage, networks, and other fundamental computing resources where the consumer is able to deploy and run arbitrary software, which can include operating systems and applications. An example of IaaS is Amazon EC2 and Google Compute Engine.

- *Platform as a Service (PaaS)*. The capability provided to the consumer is to deploy onto the cloud infrastructure consumer-created or acquired applications created using programming languages, libraries, services, and tools supported by the provider. An example of this is Google App Engine, which allows developers to run web applications on Google's infrastructure with automatic scalability.

- *Software as a Service (SaaS)*. The capability provided to the consumer is to use the provider's applications running on a cloud infrastructure. Examples are Saleforce CRM and GMail [10].

2.3 Types of Cloud Interoperability

This subsection provides an overview of different Cloud computing interoperability viewpoints [11, 12]. It attempts to illustrate the different interoperability classifications and to explore the characteristics/aspects of each category. The clear understanding and identification of the requirements that ensure interoperability is definitely the first step towards the standardization of Cloud computing platforms, APIs and services.

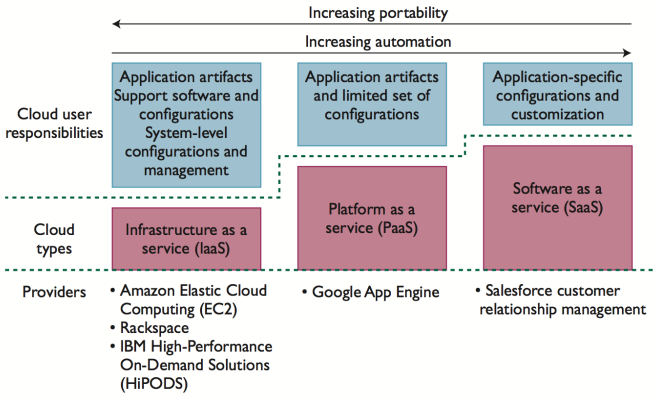

Figure 1: The cloud landscape illustrates the difference in user responsibility for each of the three types of cloud

An semantic approach in delimitating Cloud computing interoperability is presented by Sheth and Ranabahu [13], where Cloud computing interoperability is closely associated with the type of heterogeneity that arises during the interoperation of Clouds. Fig. 1 shows two types of heterogeneities.

The first is vertical heterogeneity, which is within a single silo. This is usually addressed by using middleware to homogenize the APIs and sometimes by enforcing standardization. For example, the Open Virtualization Format (OVF) is used to allow migration of virtual appliances across IaaS clouds, standard email protocols like IMAP and POP enable

the smooth migration of enterprise mail services between different SaaS providers.

The other type is horizontal heterogeneity, which is, in contrast, across silos. Overcoming this fundamentally difficult, because each silo provides different levels of abstraction and services. And since the Cloud user responsibilities in different silos vary (more specifically, decreasing from IaaS to SaaS), Cloud users would choose carefully between different cloud stacks depending on their own needs of portability and automation. For example, PaaS clouds offer faster setup for applications, and many exploit the free hosting opportunities of some PaaS providers (e.g. Google App Engine). When the application grows in scope and criticality, however, an IaaS cloud might be proven cheaper, more reliable and flexible, which leads to a transition across silos (from PaaS to IaaS).

In this paper, we will mainly focus on **IaaS**. **(Might mention PaaS and SaaS later)**

3. TAXONOMY OF IAAS INTEROPERABILITY

Teckelmann and Reich [14, 15] have presented a taxonomy of interoperability for IaaS (Fig. 2). The taxonomy discusses all important issues, such as virtual appliance and access mechanism, to demonstrate the needs and trends in the state-of-the-art development aiming for IaaS interoperability [16].

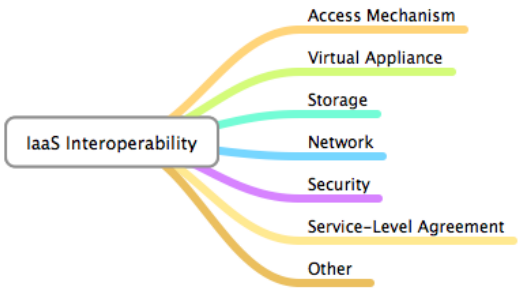

Figure 2: Interoperability of IaaS Taxonomy

- *Access Mechanisms.* Access mechanisms have a major incluence on cloud interoperability since they define how a service can be accessed by developers or end users. The industry considers three common standard types of access mechanisms: *Application Programming Interface (API)*, *Graphical User Interface (GUI)*, and *Command-Line Interface (CLI)*. Main issues: CLI and GUI are important tools for users to access cloud offerings, but they lack interoperability. CLI and GUI are mainly implemented on top of existing APIs, which use proprietary XML, JSON or plain HTTP as data formats. In order to facilitate interoperability, APIs have to expose service discovery functionalities by the combination of REST and HTTP Query service discovery of a certain domain.

- *Virtual Appliance.* The idea of Virtual Appliance is to deliver a service as a complete software stack installed on one or more VMs, as proposed by DMTF [17]. Three issues of Virtual Appliance must be addressed:

  - *Life Cycle*: As defined by DMTF, the five stages of the life cycle of a VA are: *Develop*, *Package and distribute*, *Deploy*, *Manage*, and *Retire*. Due to software upgrades or security fixes, update management are needed in all five stages of the life cycle, which will pose a significant problem to cloud providers. Start from pre-configured VM images, updates are applied across the whole life cycle of a VA carefully so as not to avoid SLA (which will be discussed later in this section) violations that disrupt VMs' uptime and network performance.
  - *Virtualization Platform.* Many different virtualization technologies exist, i.e., *OS-level virtualization*, *Full virtualization*, and *Paravirtualization*, each of which has its own advantages and disadvantages. Furthermore, taken hardware-assisted virtualization into consideration, cloud providers also need to choose between Intel's VT and AMD's AMD-V. In addition, *Host Architecture*, *Host Operating System* and *License* are also important issues.
  - *Virtualization Manager.* A virtualization manager is responsible for managing VAs on physical hosts. For interoperability, an important task is to move the VAs to different hosts and/or cloud providers. Therefore, Migration is an essential task for the managers.

- *Storage.* Both the management and organization of storage must be addressed to prevent incompatibility. There are three topics in storage management: *Backup*, *Replication*, and *Snapshots*. The backup functionality is important to guarantee long-term preservation on a non-volatile storage media. Replication means that data are not only being copied to one location, but they are distributed to several sites. Snapshots are full copies of data objects. In the organization of storage, the schemes of organization and the image format are the major concerns. To avoid compatibility problems, the introduction of an intelligent object storage layer can be a solution. VM images and related data are decoupled and stored on most appropriate storage. Logical connections can be kept using metadata information.

- *Network.* There are two important topics in terms of network interoperability: *Addressing* and *Application-level communication.*

  - *Addressing*: A major problem for cloud computing is the need to have a reliable remote access to applications and their underlying VMs, even when they are being moved between subnets within a cloud or to others.
  - *Application-level communication*: The APIs must be RESTful. There are two major communication protocols in concern: *HTTP* and *XMPP*.

Interoperability in networking is achieved through the dominant position of the IP protocol family. In terms of IaaS interoperability, IP mobility is essential during live migration of virtual machines., otherwise it would directly cause service downtime and therefore violating SLA. Many solutions exist to extend the capability of IPv4, including IPv6 and Software-defined networking.

- *Security.* Authentication, Authorization and Accounting introduce the AAA model into this taxonomy.
  - *Authentication* : A confirmation of a stated identity and an essential security mechanism are critical in clouds and other IT areas. Two kinds of authentication mechanisms are presented: *HTTP Authentication* and *Public Key Infrastructure.*
  - *Authorization* : Cloud providers need to allocate the resources or rights according to the user's credentials after the user has proven to be what he/she stated. Popular authorization models like DAC (Discretionary Access Control) are widely adopted by Amazon and Rackspace.
  - *Accounting* : Logs are central information sources for common system and fault analysis. Interoperability between clouds can be seriously affected if no such information is available. Therefore, in conjunction to security, accounting is necessary for the record of the amounts of events and operations, and the saving of information about them. The current issue is that a well-defined best practise guidelines (not necessarily standard) should be agreed upon and adopted, such that interoperability can be achieved.
  - *Encryption* : Challenges arise when ensuring the security between endpoints through communication encryption. Encryption mechanisms (SSL, TSL, VPN) are some of the widely adopted protocols. [18].
- *Service Level Agreement.* SLAs are ubiquitous in modern IT systems. In terms of cloud interoperability, SLA involves the four important topics: *Architecture*, *Template Format*, *Monitoring* and *SLA Objectives.*
  - *Architecture* : Web Service Agreement Specification (WS-A) is the standard for SLA management architecture in web service environments.
  - *Template Format* : Template formats are electronic representations for the purpose of automated management. In order to achieve interoperability, it is necessary to agree upon a single template format. WS-A seems to be a good choice, but cloud providers have not yet started to offer machine-readable SLAs.
  - *Monitoring* : Monitoring of SLAs is important for both cloud providers and users. A provider wants to ensure that the provision of resources is according to the SLA and no liability arises. On the other hand, a cloud user wants to ensure the adherence of cloud providers as mentioned in the contract. In terms of interoperability, a dangerous scenario arises when high static thresholds of SLOs can not be satisfied by any cloud providers, which leads to permanent SLA violation.
  - *SLA Objectives* : SLOs are the core component of a SLA, because they directly influence the interoperability of cloud systems. In a scenario where a customer wants to change his/her cloud provider, the absence of SLOs can make it difficult to compare the old SLA with a new one.

  APIs are crucial to interoperability for exporting SLA management functionalities. The separation of an SLA managing entity strengthens its interoperability through exchangeability and extensibility.

4. STANDARDIZATION

To start with, we provide a cloud computing standards and interoperability view to show some aspects of interoperability and standardization in cloud computing landscape. When we look across the broad range of concepts and new ideas that people considered in cloud computing, potentially hundreds of standards will be involved. Many of them are just paper work with no further progress. Some of them lack the support from the industry. Below we will discuss three major standards in cloud interoperability, which are widely adopted.

Many organizations are involved in various standardization efforts on the common theme of clouds. Notable among them are the working groups operating within the Open Grid Forum (OGF) umbrella. Other prominent industry consortiums active in cloud standardization effort are Distributed Management Task Force, Inc. (DMTF), and the Storage Networking Industry Association (SNIA). In this section we will summarize key open cloud standards that have emerged and point out the cloud component they try to standardize.

The following open standards do help build bridges towards the goal of achieving user applications and cloud providers interoperability. A significant progress has been made for pivotal elements such as storage, infrastructure management, and application description formats, but there still remains much work to be done to reach the final destination [19]. Below is a simple table summarizing the mapping between cloud interoperability taxonomy and the standards.

	OVF	CDMI	OCCI
Access Mechanism	×	✓	✓
Virtual Appliance	✓	×	×
Storage	×	✓	×
Network	×	×	✓
Security	×	×	✓
SLA	×	×	✓

4.1 OVF

The Open Virtualization Format (OVF) [17] is a packaging standard designed to address the portability and deployment of virtual appliances. It enables simplified and error-free deployment and migration of virtual appliances across multiple virtualization platforms, including IBM, Microsoft, JumpBox, VirtualBox, XenServer, AbiCloud, OpenNode Cloud, SUSE Studio, Morfeo Claudia, and OpenStack. OVF is a critical standard in cloud interoperability because it abstract the virtual appliance as a single atomic unit, which will make the migration of VMs between interoperable clouds easier.

An OVF package contains multiple files in a single directory. The directory always contains an XML file called OVF

descriptor with the VA name, hardware specifications, and references to other files in the single package. In addition, the OVF package typically contains a network description, a list of virtual hardware, virtual disks, certificate files, information about the operating system. DMTF advertises this format as vendor-neutral as it contains no reference to any current vendor-specific information. Written as an XML file, it features descriptions of most of the components of such an appliance:

- VMs' hardware (CPU, Memory...) and contextualisation informations;
- Disks and images used;
- Networking;
- Startup order of the different VMs.

Key features of OVF:

- Enables optimized distribution
- Provides a simple, automated user experience
- Supports both single and multi virtual machine configurations
- Enables portable VM packaging
- Affords vendor and platform independence
- Supports localization
- Offers future extensibility

OVF addresses the issues of *Virtual Appliance* of the cloud interoperability taxonomy:

- *Life Cycle*: With a standard format of VA, the five stages of VA life cycle (*Deploy, Package and distribute, Deploy, Manage, Retire*) can now be handled freely by the developers, due to the flexibility of XML, which contains all the matadata of the VA. Cloud providers are also capable to build pre-configure VM images based on OVF, therefore allowing Cloud users to deploy their services and automate fast.
- *Virtualization Platform*: The extensibility of OVF allows different virtualization platforms to be defined within a VA, whether it be *OS-level virtualization*, *Paravirtualization*, or *HVM virtualization*.
- *Virtualization Manager*: OVF also provides a simple, automated interface for the upper virtualization manager (e.g. *libvirt* [20]).

4.2 CDMI

The Cloud Data Management Interface (CDMI) [21] defines the functional interface that applications will use to create, retrieve, update and delete data elements from the cloud (Fig. 3). It is the storage backbone of cloud interoperability. As part of this interface, the client will be able to discover the capabilities of the cloud storage offering and use this interface to manage containers and the data that is being placed in them. In addition, metadata can be set on containers and their contained data elements through the interface. The features of CDMI include functions that

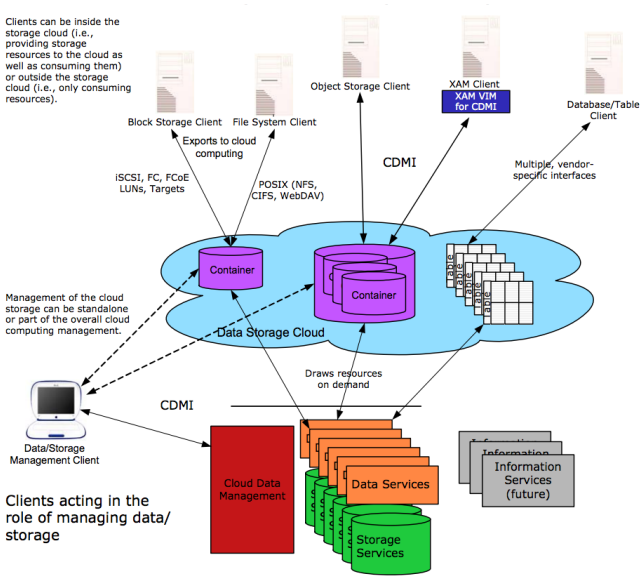

Figure 3: Cloud Storage Reference Model

- allow clients to discover the capabilities available in the cloud storage offering
- manage containers and the data that is placed in them
- allow metadata to be associated with containers and the objects they contain.

By abstracting the storage as containers which contain data objects, CDMI addresses the following issues of *Cloud Storage*:

- *Backup*: Backup is the most common operation for cloud storage, which simplified by CDMI as a simple interface. Cloud users can schedule the backup or perform backup operations manually.
- *Replication*: As seen in the figure, each container will be replicated in the low-level Data Storage Cloud.
- *Snapshots*: CDMI defines snapshot as a point-in-time copy (image) of a container and all of its content, including subcontainers and all data objects and queue objects. A snapshot operation is requested using the container update operation, in which the snapshot field specifies the requested name of the snapshot. A snapshot may be accessed in the same way that any other CDMI object is accessed. An important use of a snapshot is to allow the contents of the Source Container to be restored to their values at a previous point in time using a CDMI copy operation.

4.3 OCCI

Open Cloud Computing Interface (OCCI) [22] is a boundary API that acts as a service front-end to an IaaS provider's interlnal infrastructure management framework. OCCI provides commonly understood semantics, syntax and a means of management in the domain of consumer-to-provider IaaS. OCCI was originally initiated to create a remote management API for IaaS model-based services, allowing for the

development of interoperable tools for common tasks including deployment, autonomic scaling and monitoring. It has since evolved into a flexible API with a strong focus on interoperability while still offering a high degree of extensibility. OCCI provides commonly understood semantics, syntax and a means of management in the domain of consumer-to-provider IaaS. It covers management of the entire life-cycle of OCCI-defined model entities and is compatible with existing standards such as the Open Virtualisation Format (OVF) and the Cloud Data Management Interface (CDMI). Notably, it serves as an integration point for standardization efforts including DMTF, IETF and SNIA. It defines a set of common standard interface of management in cloud interoperability. The OCCI community works in a distributed, open community under the umbrella of the Open Grid Forum (OGF), whose contributing members include:

- **Industry**: Rackspace, Oracle, Platform Computing, GoGrid, Cisco, Flexiscale, ElasticHosts, CloudCentral, RabbitMQ, CohesiveFT.
- **Academic & Research**: SLA@SOI, RESERVOIR, the Claudia Project, OpenStack, OpenNebula, DGSI.

In order to be modular and extensible, the current OCCI specification is released as a suite of complimentary documents, which together form the complete specification. The documents are divided into three categories consisting of the OCCI Core, the OCCI Renderings and the OCCI Extensions.

- The OCCI Core specification consists of a single document defining the OCCI Core Model. The OCCI Core Model can be interacted with renderings (including associated behaviours) and expanded through extensions.
- The OCCI Rendering specifications consist of multiple documents each describing a particular rendering of the OCCI Core Model. Multiple renderings can interact with the same instance of the OCCI Core Model and will automatically support any additions to the model which follow the extension rules defined in OCCI Core.
- The OCCI Extension specifications consist of multiple documents each describing a particular extension of the OCCI Core Model. The extension documents describe additions to the OCCI Core Model defined within the OCCI specification suite. They do not require changes to the HTTP Rendering specifications as of this version of the specification.

OCCI addresses the following issues of *Access Mechanism*, *Network*, *Security* and *SLA*:

- *Access Mechanism*: Access mechanisms are exclusively describe in the OCCI Rendering specification, where RESTful HTTP APIs are defined. Each resource instance within an OCCI system has a unique identifier stored in the *occi.core.id* attribute of the Entity type, such that they can be accessed just like how we visit the websites on the Internet.
- *Network*: The Network within an OCCI system is a L2 networking entity (e.g. a virtual switch). It can be extended using the mixin mechanism to support L3/L4 capabilities such as TCP/IP etc.
- *Security*: Since the OCCI Core and Model specification describes a model, not an interface or protocol, no specific security mechanisms are described as part of the specification. However, elements described by this specification need to implement a proper authorization scheme, which MUST be a part of a concrete OCCI rendering specification, part of an OCCI specification profile, or part of the specific OCCI implementation. Concrete security mechanisms and protection against attacks should be specified by OCCI rendering specification. In any case, OCCI rendering specification must address transport level security and authentication on the protocol level.
- *SLA*: SLA is addressed in the *Billing and Negotiation/Agreement features* part of the OCCI specification, which are currently still in progress.

5. IMPLEMENTATIONS OF IAAS PLATFORMS

Cloud standards could not evolve by themselves without the cooperation of their open source implementations. This section will briefly introduce the fast-growing ecosystem for cloud interoperability and portability. The state-of-the-art implementations of IaaS Cloud management system include *OpenStack*, *OpenNebula*, and *Eucalyptus*. All implementations of IaaS platforms must be constituted of at least three parts: *Compute*, *Storage*, *Image*. Of course, different platforms will vary in terms of implemention, and might additional features.

Apart from full implementations, there have been other effects among industry that try to leverage existing cloud standards by offering APIs. These include *LibCloud* and *δ-Cloud*. LibCloud and δ-Cloud are important and useful. They provide abstractions that help developers write applications without being tied to a specific cloud vendor (or intermediate layer such as OpenStack). While lack of lock-in is a concrete benefit, it comes at a price. The price is that the API shim cannot expose features that differentiate the platforms. This may represents a significant loss of functionality or performance. **plug-ins or hooks may be some kind of suppliments to these open APIs**.

Both approaches have their place and are needed in the market. If the developer needed to write against multiple clouds for portability, then LibCloud is a slam dunk. If the developer needed rich features and a full ecosystem, then OpenStack or Amazon are better choices.

5.1 OpenStack

OpenStack [24] consisted of a trio of "core" services:

- **Object Store ("Swift")** provides object storage. It allows you to store or retrieve files (but not mount directories like a fileserver). Several companies provide commercial storage services based on Swift. These include KT, Rackspace (from which Swift originated) and my company Internap. In fact, the images for this blog post are being served via the Internap Swift implementation.
- **Image ("Glance")** provides a catalog and repository for virtual disk images. These disk images are mostly

commonly used in OpenStack Compute. While this service is technically optional, any cloud of size will require it.

- **Compute ("Nova")** (Fig. 4) provides virtual servers upon demand. Similar to Amazon's EC2 service, it also provides volume services analogous to Elastic Block Services (EBS). Internap provide a commercial compute service built on Nova and it is used internally at Mercado Libre and NASA (where it originated).

In terms of cloud interoperability, the most significant feature of OpenStack is Software-Defined Network, which is named Quantum. Quantum intends to provide "networking as a service" between interface devices (e.g. vNICs) managed by other OpenStack services (e.g. Nova).

- Quantum gives cloud tenants an API to build rich networking topologies, and configure advanced network policies in the cloud. (Examples: create multi-tier web application topology)
- Quantum enables innovation plugins (open and closed source) that introduce advanced network capabilities. (Examples: use L2-in-L3 tunneling to avoid VLAN limits, provide end-to-end QoS guarantees, enable WAN live VM migration, etc.)
- Quantum lets anyone build advanced network services that plug into OpenStack tenant networks. (Examples: LB-aaS, VPN-aaS, FW-aaS, IDS-aaS, DC-interconnect-aaS).

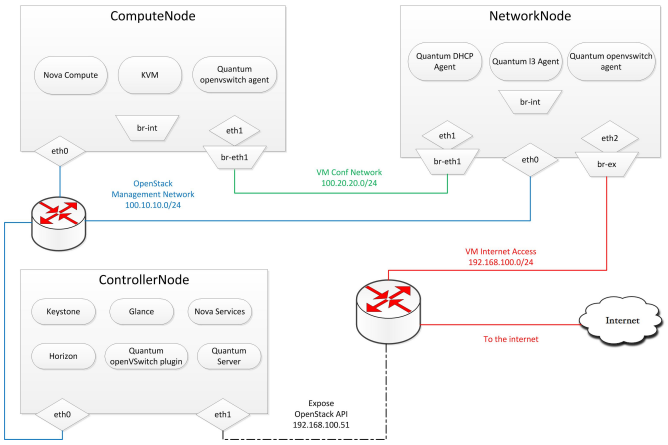

Figure 4: OpenStack (with Quantum)

5.2 Eucalyptus

Eucalyptus [23] is a Linux-based software architecture that creates scalable private and hybrid clouds within your exsiting IT infrastructure. Eucalyptus provides a virtual network overlay that both isolates network traffic of different users and allows two or more clusters to appear to belong to the same Local Area Network (LAN). Eucalyptus also interoperates seamlessly with Amazon's EC2 and S3 public cloud services and thus offers the enterprise a hybrid cloud capability.

The following is an explanation of terminology and concepts used by Eucalyptus (Fig. 5), in terms of cloud interoperability:

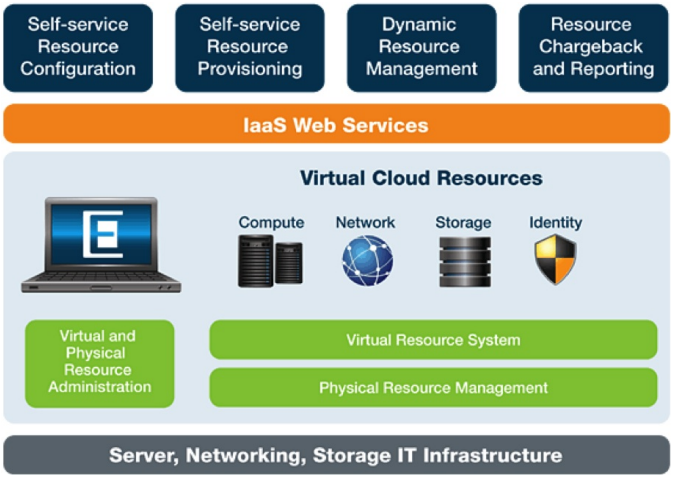

Figure 5: Eucalyptus

- *Images*: An image is an operating system configuration.
- *Instances*: When an image is put into use, it is called an instance. The configuration id dynamically executed at runtime and the cloud controller decides where the image will run, storage and networking is attached to meet resource needs.
- *IP Addressing*: Eucalyptus instances will always have a private IP but can also have a public IP. By default, an instance has both and Eucalyptus also supports elastic IPs.
- *Security*: TCP/IP stack L3 security is achieved using security groups, which share a common set of firewall rules. This is a mechanism to firewall off an instance using IP address and port block/allow functionality. Eucalyptus also support L2 security.
- *Networking*: There are four modes of networking. In Managed Mode, Eucalyptus manages a local network of instances, including security groups and elastic IPs. In System Mode, the LAN that is attached to Eucalyptus manages the network of the Eucalyptus cloud. In Static Mode, Eucalyptus maintains a DHCP server and assigns IP addresses to instances. In Managed-NOVLAN Mode, Eucalyptus takes full advantage of security groups and elastic IPs.
- *Access Control*: A user of Eucalyptus is assigned an identity, and identities can be grouped together for access control.

5.3 OpenNebula

OpenNebula [25] is an open-source industry standard for data center virtualization, offering feature-rich, flexible solution for the comprehensive management of virtualized data centers to enable on-premise infrastructure as a service clouds. It provides many different interfaces that can be used to interact with the functionality offered to manage physical and virtual resources. There are four main different perspectives to interact with OpenNebula (Fig. 6):

- Cloud interfaces for *Cloud Consumers*, like the OCCI and EC2 Query and EBS interfaces, and a simple self-service portal
- Administration interfaces for *Cloud Advanced Users and Operators*, like a Unix-like command-line interface and the powerful Sunstone GUI.
- Extensible low-level APIs for *Cloud Integrators* in Ruby, Java and XMLRPC API
- A marketplace for *Appliance Builders* with a catelog of virtual appliances ready to run in OpenNebula environments.

In terms of cloud interoperability, OpenNebula provides the following subsystems:

- *Multiple Zones*: The OpenNebula Zones component (oZones) allows for the centralized management of multiple instances of OpenNebula, called Zones, for scalability, isolation and multiple-site support.
- *VDCs*: An OpenNebula instance (or Zone) can be further compartmentalized in Virtual Data Centers (VDCs), which offer fully-isolated virtual infrastructure environment where a group of users, under the control of the VDC administrator, can create and manage compute, storage and networking capacity.
- *Hybrid*: OpenNebula gives support to build a hybrid cloud, an extension of a private cloud to combine local resources with resources from remote cloud providers (e.g. Amazon EC2). A whole public cloud provider can be encapsulated as a local resource to be able to use extra computational capacity to satisfy peak demands.

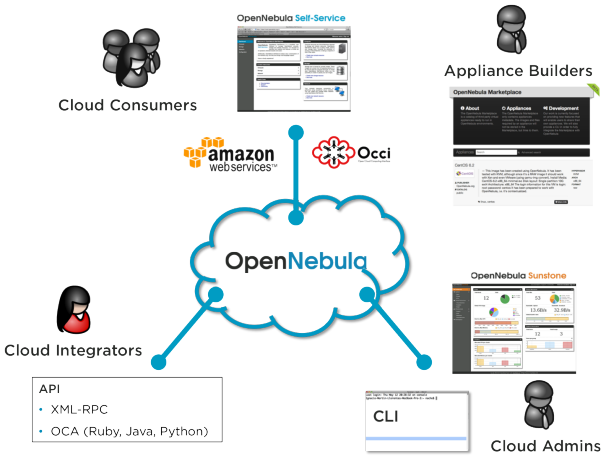

Figure 6: OpenNebula

6. USER-CENTRIC APPROACH

Standardization (along with its implementation), as mentioned in the previous section, aims to address the interoperability issues by building up common standards (OVF, CDMI, OCCI, etc) in nearly every dimension of the issues and challenges listed in the taxonomy. But history has already taught us lessons that the industry might take years or even decades to agree upon such common standards due to selfishness of the giant cloud vendors and even political obstacles. Instead of standardization, which is by definition provider-centric, researchers have advocate a *user-centric* approach that could give users an unprecedented level of control over the virtualization layer. Fundamentally, today's clouds lack the homogeneity necessary for cost-effective multi-cloud deployments. A single VM image cannot be deployed unmodified on any IaaS cloud. Even worse, there is no consistent set of hypervisor-level services across providers. While some progress towards multi-cloud homogeneity is expected through standardization efforts such as the Open Virtualization Format, these provider-centric approaches will likely be limited to simple cloud attributes like image format and take years to be universally adopted.

6.1 Xen Blanket

Based on nested virtualization [6], the Xen-Blanket [26] leverages a second-layer Xen hypervisor completely controlled by the user that utilizes a set of provider-specific Blanket drivers to execute on top of existing clouds without requiring any modifications to the provider. Blanket drivers have been developed for both Xen and KVM based systems, and achieve high performance: network and disk throughput remain within 12% of paravirtualized drivers in a single-level paravirtualized guest. The Xen-Blanket is deployed today on both Xen-based and KVM-based hypervisors, on public and private infrastructures within Amazon EC2, an enterprise cloud, and Cornell University. The Xen-Blanket has successfully homogenized these diverse environments. For example, they have migrated VMs to and from Amazon EC2 with no modifications to the VMs. Furthermore, the user-centric design of the Xen-Blanket affords users the flexibility to oversubscribe resources such as network, memory, and disk. As depicted in Fig. 7, a Blanket layer embodies three important concepts:

- First, the bottom half of the Blanket layer communicates with a variety of underlying hypervisor interfaces. No modifications are expected or required to the underlying hypervisor.
- Second, the top half of the Blanket layer exposes a single VM interface to Blanket (second-layer) guests such that a single guest image can run on any cloud without modifications.
- Third, the Blanket layer is completely under the control of the cloud user, so functionality typically implemented by providers in the hypervisor (such as live VM migration), can be implemented in the Blanket layer.

More specifically, the benefits brought by Xen Blanket are that the cloud users now own full control of a *super* VM, which may contain a cluster of many other traditional VMs. Thus, the Xen Blanket here acts as a middleware between the VM and the underlying hypervisor, which is another application of David Wheeler's theorem: *All problems in computer science can be solved by another level of indirection* [27].

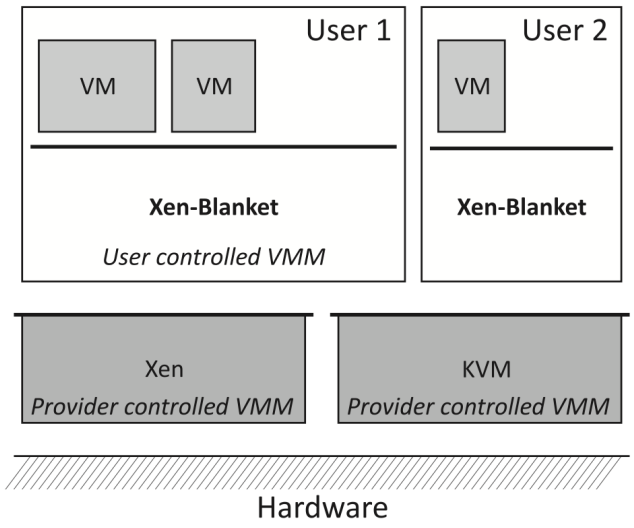

Figure 7: Xen-Blanket

6.2 Self-service Cloud Computing

To address the inflexible control over VMs, Self-service Cloud Computing [28] introduces a new self-service cloud computing model that splits administrative priviledges between a system-wide domain and per-client administrative domains. Each client can manage and perform priviledged system tasks on its own VMs, thereby providing flexibility. Fundamentally, it changes the way in which commodity hypervisors assign privilege to VMs by separating dom0 into *Udom0* and *Sdom0*, thus enabling cloud users to take full control over Udom0, which is almost the same level of flexibility brought by Xen Blanket.

The implementation was done on Xen hypervisor. The demonstration in the paper was successful, but still needs a lot of work to achieve production quality.

6.3 OpenCirrus

OpenCirrus [29] is a federated testbed of distributed clusters that aims to enable cloud targeted system level research deploying a user-centric cloud by allowing access to bare hardware, as in Emulab, in a number of dedicated data centers. Open Cirrus offers a cloud stack consisting of physical and virtual machines, and global services such as sign-on, monitoring, storage, and job submission. By developing the testbed and making it available to the research community, the authors hope to help spur innovation in cloud computing and catalyze the development of an open-source stack for the cloud. However, OpenCirrus is not aimed at applying this ability to existing cloud infrastructures.

7. RESEARCH CHALLENGES

7.1 Pricing Strategy

From a cloud provider's perspective, the elastic resource pool (through either virtualization or multi-tenancy) has made the cost analysis far more complicated than regular data centers, which simply calculates their cost based on power consumptions of static computing. Moreover, an instantiated virtual machine has become the unit of cost analysis rather than the underlying physical server. In a federation of clouds, the VMs are more dynamic, since VMs can easily be migrated between interoperable clouds. If we consider the case of nested-virtualization, it will be even more complicated, because a VM might also be a hypervisor. A promsing pricing strategy needs to incorporate all the above as well as VM associated items such as software licenses, virtual network usage, node and hypervisor management overhead, and so on.

7.2 Service Level Agreement

Although cloud consumers do not have control over the underlying computing resources, they do need to ensure the quality, availability, reliability and performance of these resources when consumers have migrated their core applications onto their entrusted cloud. In other words, it is vital for consumers to obtain guarantees from providers on service delivery. Typically, these are provided through Service Level Agreements (SLAs) negotiated between the providers and consumers. Federation of clouds will also raise a number of problems for SLAs. For example, it is almost inevitable to avoid downtime during live migration between interoperable clouds, and the cloud provider (both the incoming and outgoing) need to possess precise and updated information on the resource usage at any particular time during the migration. Furthermore, advanced SLA mechanisms [30] need to constantly incorporate user feedback and customization features into the SLA evaluation framework.

7.3 WAN Live VM Migration

There would a huge demand of WAN live migration in the environment of federation of clouds. But in traditional frameworks, live VM migration over WAN is not possible. Recent studies have suggested many different approaches to modify the live migration workflow to make it work on a wide area network. But their performance and reliability is still not promising. Recent advances in Software-Defined Networking and IPv6 have made it possible to tackle this problem in another angle: the network layer.

7.4 Moving between clouds

The challenges that will be faced when cloud users decide to move an application between clouds:

- Rebuilding the application and application stack in the target cloud.
- Setting up the network in the target cloud to give the application the support that it had in its original cloud.
- Setting up security to match the capabilities provided by the source cloud.
- Managing the application running in the target cloud.
- Handling data movement and the encryption of data while it is in transit and when it gets to the target cloud.

7.5 Multi-Cloud Deployments

Using tools from Rightscale, a cloud user can create ServerTemplates, which can be deployed on a variety of clouds and utilize unique features of clouds without sacrificing portability. However, users are still unable to homogenize the underlying clouds, particularly hypervisor-level services.

Middleware, such as IBM's Altoculumus system homogenize both IaaS clouds like Amazon EC2 and PaaS clouds like Google App Engine into a PaaS abstraction across multiple clouds. However, without the control at the IaaS level, the amount of flexibility provided to the cloud users is still fundamentally limited. It will be a great challenge to provide such IaaS abstraction across clouds.

8. CONCLUSION

James Gosling[31] made a nice note on the phases that standardization process goes through, and the relationship between the level of technical and political interest in a topic.

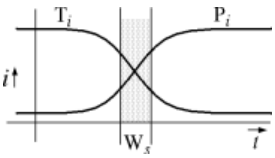

Figure 8: Standardization process

The i axis describes level of interest and the t axis describes time. T_i describes technical interest, and P_i describes political interest. As time passes, technical activity declines as the technology becomes understood. Similarly, generally fueled by economic pressures, the political interest as a technology increases in some period. For a standard to be usefully formed, the technology needs to be understood: technological interest needs to be decreased. But if political interest in a standard becomes too large, the various parties have too much at stake in their own vested interest to be flexible enough to accommodate the unified view that a standard requires.

Applying this model to cloud interoperability and standardization, we are currently in the T_i phase, where technologists are showing interest and trying to understand the technology behind cloud computing. There is still a long way to go.

To conclude the standardization of cloud, we can compare it with the evolution of web. When the web started it was just about sharing static HTML files and nothing more. Then the CGI era came into being and brought dynamic updates to web pages. Next the application servers across platforms like J2EE/.Net etc brought the web applications into context. History repeats itself. Likewise, cloud computing will also witness a similar evolution. But we are not quite there, yet. For example, VM provisioning in the cloud/cross many clouds are just like the CGI era of the web evolution.

9. REFERENCES

[1] Breaking through cloud addiction. [Online]. Available: http://techcrunch.com/2012/12/01/netflixs-amazon-cloud-addiction/

[2] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "A view of cloud computing," *Communications of the ACM*, vol. 53, no. 4, 2010.

[3] A. Fox and R. Griffith, "Above the clouds: A Berkeley view of cloud computing," *Dept. Electrical Eng. and Comput. Sciences, University of California, Berkeley, Rep. UCB/EECS*, vol. 28, 2009.

[4] Netflix. [Online]. Available: http://netflix.com

[5] H. Qi and A. Gani, "Research On Mobile Cloud Computing: Review, Trend, And Perspectives," *arXiv.org*, vol. cs.DC, 2012.

[6] M. Ben-Yehuda, M. D. Day, Z. Dubitzky, M. Factor, N. Har'El, A. Gordon, A. Liguori, O. Wasserman, and B.-A. Yassour, "The turtles project: design and implementation of nested virtualization," in *OSDI'10: Proceedings of the 9th USENIX conference on Operating systems design and implementation*. USENIX Association, 2010.

[7] Peter Mell and Tim Grance, "The NIST Definition of Cloud Computing," http://www.nist.gov/itl/cloud/upload/cloud-def-v15.pdf, 2009.

[8] European interoperability framework v2. [Online]. Available: http://www.informatizacia.sk/ext_dok-european_interoperability_framework/9319c

[9] T. Dillon, C. Wu, and E. Chang, "Cloud Computing: Issues and Challenges," in *Advanced Information Networking and Applications (AINA), 2010 24th IEEE International Conference on*, 2010, pp. 27–33.

[10] C. Shan, C. Heng, and Z. Xianjun, "Inter-Cloud Operations via NGSON," *Ieee Communications Magazine*, vol. 50, no. 1, pp. 82–89, 2012.

[11] A. Parameswaran, "Cloud Interoperability and Standardization," *SETLabs Briefings*, 2009.

[12] G. Machado and D. Hausheer, "Considerations on the Interoperability of and between Cloud Computing Standards," *... Workshop: From Grid to Cloud ...*, 2009.

[13] A. Sheth and A. Ranabahu, "Semantic Modeling for Cloud Computing, Part 1," *Internet Computing, IEEE*, vol. 14, no. 3, pp. 81–83, 2010.

[14] R. Teckelmann, C. Reich, and A. Sulistio, "Mapping of Cloud Standards to the Taxonomy of Interoperability in IaaS," *Cloud Computing Technology and Science (CloudCom), 2011 IEEE Third International Conference on*, pp. 522–526, 2011.

[15] R. Prodan and S. Ostermann, "A survey and taxonomy of infrastructure as a service and web hosting cloud providers," *Grid Computing, 2009 10th IEEE/ACM International Conference on*, pp. 17–25, 2009.

[16] B. Rimal, E. Choi, and I. Lumb, "A taxonomy and survey of cloud computing systems," *INC, IMS and IDC, 2009. NCM'09. Fifth International Joint Conference on*, pp. 44–51, 2009.

[17] Open virtualization format. [Online]. Available: http://www.dmtf.org/standards/ovf

[18] S. Dowell, A. Barreto, J. Michael, and M.-T. Shing, "Cloud to cloud interoperability," in *System of Systems Engineering (SoSE), 2011 6th International Conference on*, 2011, pp. 258–263.

[19] P. Harsh, F. Dudouet, R. G. Cascella, Y. Jégou, and C. Morin, "Using Open Standards for Interoperability - Issues, Solutions, and Challenges facing Cloud Computing," *arXiv.org*, vol. cs.DC, Jul. 2012.

[20] Libvirt. [Online]. Available: http://libvirt.org/
[21] Cloud data management interface by snia. [Online]. Available: http://www.snia.org/cdmi
[22] Open cloud computing interface. [Online]. Available: http://occi-wg.org/
[23] Eucalyptus. [Online]. Available: http://www.eucalyptus.com/
[24] Openstack. [Online]. Available: http://www.openstack.org/
[25] Opennebula. [Online]. Available: http://opennebula.org/
[26] D. Williams, H. Jamjoom, and H. Weatherspoon, "The Xen-Blanket: virtualize once, run everywhere," in *EuroSys '12: Proceedings of the 7th ACM european conference on Computer Systems*. ACM Request Permissions, 2012.
[27] D. Wheeler, *Another level of indirection*.
[28] S. Butt, H. A. Lagar-Cavilla, A. Srivastava, and V. Ganapathy, "Self-service Cloud Computing," *ACM CCS*, 2012.
[29] R. Campbell, I. Gupta, M. Heath, and S. Y. Ko, "Open cirrusTM cloud computing testbed: federated data centers for open source systems and services research," *... in cloud computing*, 2009.
[30] S. Yan, B. S. Lee, and S. Singhal, "A model-based proxy for unified IaaS management," *Systems and Virtualization Management (SVM), 2010 4th International DMTF Academic Alliance Workshop on*, pp. 15–20, 2010.
[31] Phase relationships in the standardization process. [Online]. Available: http://nighthacks.com/roller/jag/resource/StandardsPhases.html