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 system of systems to access a shared pool of configurable computing resources (e.g., networks, servers, storage, data, 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. Interoperability is central to enabling sharing of resources from a pool of cloud-service providers in a seamless fashion. Intensive research and development efforts are devoted to overcome this state of the art. 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 industry and academia . We aim to pave way to the 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 [27, 31] 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. **(more general introduction from cloud computing)**.

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, prevents them from agreeing upon a widely accepted, standardized way to import/export Cloud details and specifications. The need is emerging for interoperability between clouds so that a complex and developed business application on clouds is interoperable.

In order to better define the problem of 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 [2]. 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 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 via 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 *industry practises*. In the user-centric approach, we introduce and summarize some research projects that take advantage of nested-virtualiztion.

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. Architectures of some cloud vendors (including Amazon and Google) is introduced in section IV. Section V introduces three mainstream cloud standards and their comparisons. The specific implementations of those standards are presented in section VI. Section VII will address the cloud interoperability problem in a different angle: User-Centric approach. Last but not the least, section VIII 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 provi-

sioned and released with minimal management effort or service provider interaction. This cloud model is composed of five essential characteristics, three service models, and four deployment models. [3].

According to the European Commission (EC) [4], 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.

2.2 Service Models

The cloud computing community has widely adopted the following three service models to categorize cloud services [5]:

- *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. Another example is Amazon S3, which provides a simple web service interface for users to store and retrieve any amount of data, at any time, from anywhere on the web.

- *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 [25].

2.3 Types of Cloud Interoperability

This subsection provides an overview of different Cloud computing interoperability viewpoints [28, 29]. 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.

An semantic approach in delimitating Cloud computing interoperability is presented by Sheth and Ranabahu [6], where Cloud computing interoperability is closely associated with the type of heterogeneity that arises during the interoperation of Clouds. Fig. 1 shows the different user responsibilities for three types of cloud service models. Clouds interoperate to meet the needs of client applications using infrastructure, platforms or services coming from different Clouds. Sheth and Ranabahu recognize two types of heterogeneity; vertical and horizontal. Using the term "silo", they refer to **IaaS**, **PaaS** or **SaaS** Cloud level.

Vertical heterogeneity emerges within a silo, i.e. when a customer needs to utilize services from different layers of the Cloud stack, but within the same silo. Horizontal heterogeneity depends on the level of the Cloud stack and therefore

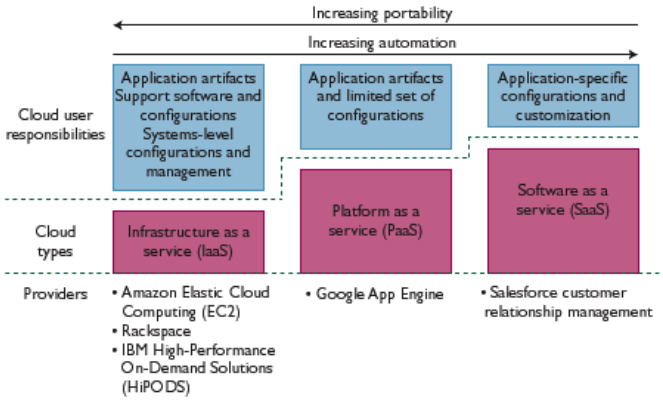

Figure 1: The cloud landscape illustrates the difference in user responsibility for each of the three types of cloud

it is divided in three subcategories, dealing with the interoperability at IaaS, PaaS and SaaS level, respectively. It emerges when a customer opts to use simultaneously more than one services hosted at different providers or to change service provider, while also remaining in the same Cloud stack layer.

In this paper, we will mainly focus on **IaaS**. **(Reasons pending, should include the figure on portability across cloud layers.)**

3. TAXONOMY OF IAAS INTEROPERABILITY

Teckelmann and Reich [7, 32] 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 [30].

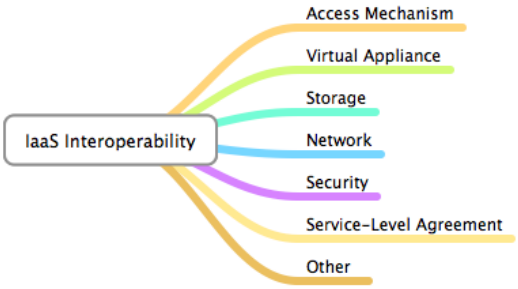

Figure 2: Interoperability of IaaS Taxonomy

- *Access Mechanisms.* Access mechanism defines 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)*.

- *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 [8]. Virtual appliance is not just a virtual machine, but a new computational resource for cloud computing.

- *Storage.* The compatibility of underlying storage resources is still required to allow the migration of data from one location to another. Both the management and organization of storage must be addressed to prevent incompatibility.

- *Network.* 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. Encryption mechanisms are also important for communication [9].

- *Service Level Agreement.* 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. ARCHITECTURES OF DIFFERENT CLOUD VENDORS

4.1 Amazon AWS

Amazon [10] is one of the leading companies at the area of cloud computing platforms. The resource model that underlies the cloud services offered by Amazon is as follows:

1. The image that is a predefined way to instantiate a VM.

2. The instance type that defines the instance characteristics (i.e. CPU, RAM, hard disk) as standard templates.

3. The instance that is the realization of an image using an instance type.

4. The storage that denotes the component where the data are stored.

5. The network that connects many instances.

6. The IP address that is static IP addresses which can be related to any instance.

7. The availability zone that is a distinct location where the cloud infrastructure is located. Each availability zone is insulated from failures in other availability zones.

The Amazon API is widely used not only by Amazon Web Services but also by other open source cloud management tools, such as Eucalyptus, OpenNebula, and OpenStack. The API is based on the Amazon Resource Model presented above. Three storage components are offered, namely EBS, S3 and SimpleDB, that differ in the way they store the data and in their usage (e.g. EBS can be used as an attached disk to an image while S3 and SimpleDB are simply used as independent storage services). Moreover, the API offers the capability to create virtual networks that serve as Virtual Private Clouds (VPC). The protocols used by the API are the SOAP and the REST.

- The user's DNS requests are served by Amazon Route 53, a highly available Domain Name System (DNS) service. Network traffic is routed to infrastructure running in Amazon Web Services.

- HTTP requests are first handled by Elastic Load Balancing, which automatically distributes incoming application traffic across multiple Amazon Elastic Compute Cloud (EC2) instances across Availability Zones (AZs). It enables even greater fault tolerance in your applications, seamlessly providing the amount of load balancing capacity needed in response to incoming application traffic.

- Web servers and application servers are deployed on Amazon EC2 instances. Most organizations will select an Amazon Machine Image (AMI) and then customize it to their needs. This custom AMI will then be used as the starting point for future web development.

- Web servers and application servers are deployed in an Auto Scaling group. Auto Scaling automatically adjusts your capacity up or down according to conditions you define. With Auto Scaling, you can ensure that the number of Amazon EC2 instances you're using increases seamlessly during demand spikes to maintain performance and decreases automatically during demand lulls to minimize costs.

- Resources and static content used by the web application are stored on Amazon Simple Storage Service (S3), a highly durable storage infrastructure designed for mission-critical and primary data storage.

- Static and streaming content is delivered by Amazon CloudFront, a global network of edge locations. Requests are automatically routed to the nearest edge location, so content is delivered with the best possible performance.

- Availability zones (AZs) are distinct geographic locations that are engineered to insulate against failures in other AZs. Multiple AZs are combined into a region. Here, the entire web application is deployed in two different AZs for high availability.

Discussion on interoperability of Amazon AWS: support import and export from/to other cloud platforms in various VM formats.

4.2 Google Compute Engine

Launched on July 2012, Google Compute Engine (GCE) [11] is quite like AWS and Rackspace. But it has very different targets: problems using large compute jobs, batch workloads, or that require high performance real-time calculations. GCE is not for building websites. In the future they plan on adding more features like load balancing.

Datacenters:

- Region: for geography and routing domain.
- Zone: for fault tolerance
- Currently operating 3 US datacenters/zones, located on the East coast of the US.
- Working on adding more datacenters globally and adding more datacenters in the US.

API:

- JSON over HTTP API, REST-inspired, authorization is with OAuth2
- Main resources: projects, instances, networks, firewalls, disks, snapshots, zones
- Actions GET, POST (create), DELETE, custom verbs for updates
- A command line tool (gsutil), a GUI, and a set of standard libraries gives access to the APIs. Experience is like Amazon in that you have an UI and command line tools.
- All Google tools use the API. There is no backdoor. The web UI is built on Google App Engine, for example. App Engine is the web facing application environment and is considered an orchestration system for GCE.
- Partners like RightScale, Puppet, and OpsCode, also use the API to provide higher level services.
- Want people to take their code and run it on their infrastructure. Open API. No backdoors. Can extend that stack at any level.

Virtual Machines:

- A combination of KVMs (Kernel Virtual Machines) and Linux cgroups are used for the underlying hypervisor technology. Linux scheduler and memory manager are reused to handle the scheduling of the machines.
- KVM provides virtualization. Cgroups provides resource isolation. Cgroups was pioneered by Google to keep workloads isolated from each other.
- Internally Google can run virtualized and non-virtualized workloads on the same kernel and on the same machine, which allows them to deploy and test one single kernel. Located in a zone.
- Fast boot times: 2 minutes.

Storage:

- Focused on creating persistent block device that offers performance/throughput so you dont need to push storage local.
- Two block storage devices: Persistent Disk and Local Disk.

Discussion on interoperability of GCE: migration is done via RightScale, Puppet or other third-party cloud brokers.

4.3 Rackspace

Rackspace [12] founded the OpenStack project (which we will discuss later) in conjunction with NASA. So when we are talking about the interoperability of Rackspace clouds, we are actually referring to OpenStack.

4.4 GoGrid

GoGrid [13] is the one of the world's largest pure-play Infrastructure-as-a-Service (IaaS) provider specializing in Cloud Infrastructure solutions. Currently powering thousands of customers globally, we make complex infrastructure easy by enabling businesses to revolutionize their IT environments with the Cloud.

As listed in their documentation, GoGrid provides the following REST API for cloud users:

- **grid.server.list** API function call to list servers in your grid.
- **grid.server.get** API function call to retrieve a single server in your grid.
- **grid.server.add** API function call to add a server to your grid.
- **grid.server.edit** API function call to edit a single server in your grid.
- **grid.server.delete** API function call to delete a server and remove it from your grid.
- **grid.server.power** API function call to start, stop, or restart a server.

Although GoGrid allows cloud users to choose from almost 100 server images from "vanilla" OS distros to pre-built application stacks provided by GoGrid Exchange partners TrendMicro, Snort, Bitnami, and more, it does not support any open VM standard specification.

4.5 Microsoft Azure

Up until now, Microsoft's Platform-as-a-Service (PaaS) stack offered a "virtual machine role" that provided non-persistent compute nodes that were not equivalent to traditional IaaS services offered by providers like Amazon Web Services. That changed with the new Windows Azure [14] Virtual Machines service where persistent Windows and Linux machines can be created, hosted, and managed. Windows Azure users can browse prebuilt Windows and Linux virtual machine templates or bring their own virtual machine (VHD only) to the Windows Azure cloud.

Virtual Machines are durable (meaning anything you install within them persists across reboots) and users can use any OS with them. Their built-in image gallery includes both Windows Server images (including the new Windows Server 2012 RC) as well as Linux images (including Ubuntu,

CentOS, and SUSE distributions). Once the users create a VM instance they can easily Terminal Server or SSH into it in order to configure and customize the VM however they want (and optionally capture their own image snapshot of it to use when creating new VM instances). This provides the cloud users with the flexibility to run pretty much any workload within Windows Azure.

Aparently, the interoperability mechanisms adopted by Microsoft Azure are not compatible with other open standards (e.g. OVF, CDMI, OCCI). That also dictates that it will be difficult for users to import existing VM images onto Azure.

However, Microsoft Azure do provide a very good platform (in the realm of PaaS) for developers to build web applications. It has released SDKs for Node.js MongoDB, Hadoop, Memcached, etc.

5. STANDARDIZATION

Open standards are the main proponents of interoperability. An inclusive standardization process has more milage in getting accepted by the stakeholders than a process that is exclusive. A question may be raised regarding the adoption of cloud standards by well entrenched providers. Why is being interoperable good for them? Critics always point that being able to vendor-lock-in a customer is good for the business as it may reduce customer churn, but in our opinion this may not be true. Brand loyalty can be achieved by providing superior services at attractive prices. Further being interoperable could bring in big government, banks, and health-care providers businesses into clouds thus vastly increasing the customer base.

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 [24].

5.1 OVF

The Open Virtualization Format (OVF) [8] 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.

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

5.2 CDMI

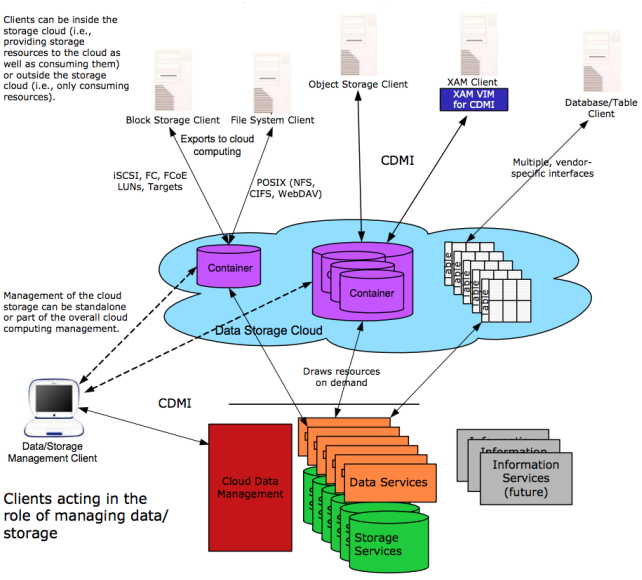

Figure 3: Cloud Storage Reference Model

The Cloud Data Management Interface (CDMI) [15] defines the functional interface that applications will use to create, retrieve, update and delete data elements from the cloud (Fig. 3). 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

- 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.

As of Oct. 16, 2012, The Storage Networking Industry Association (SNIA) announced that CDMI, the first industry-developed open standard for data storage as a service as part of cloud computing, has been designated an International Standard by the Joint Technical Committee 1 (JTC 1), a joint effort of the International Organization for Standardization (ISO) and the International Electrotechnical Commission (IEC) for the purpose of standardization in the field of Information Technology.

5.3 OCCI

The Open Cloud Computing Interface (OCCI) [16] is a RESTful protocol and API for all kinds of management tasks. 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.

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.

6. IMPLEMENTATION AND OPENAPIS

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*.

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.

6.1 Eucalyptus

Eucalyptus [17] 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 Amazons EC2 and S3 public cloud services and thus offers the enterprise a hybrid cloud capability.

Eucalyptus consists of the following components (Fig. 4):

- Cloud Controller (CLC): this component provides EC2 functionality
- Walrus: this component provides S3 functionality
- Cluster Controller (CC): this component provides management service for a cluster in your cloud
- Storage Controller (SC): this component provides EBS functionality
- Node Controller (NC): this component controls virtual machine instances

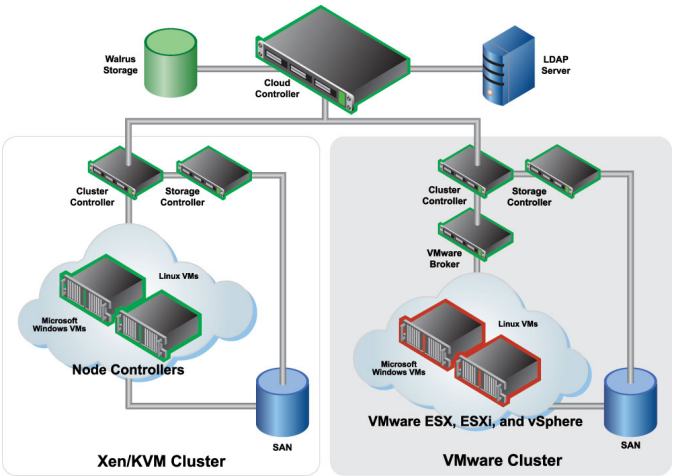

Figure 4: Eucalyptus

6.2 OpenStack

OpenStack [18] 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. 5) 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).

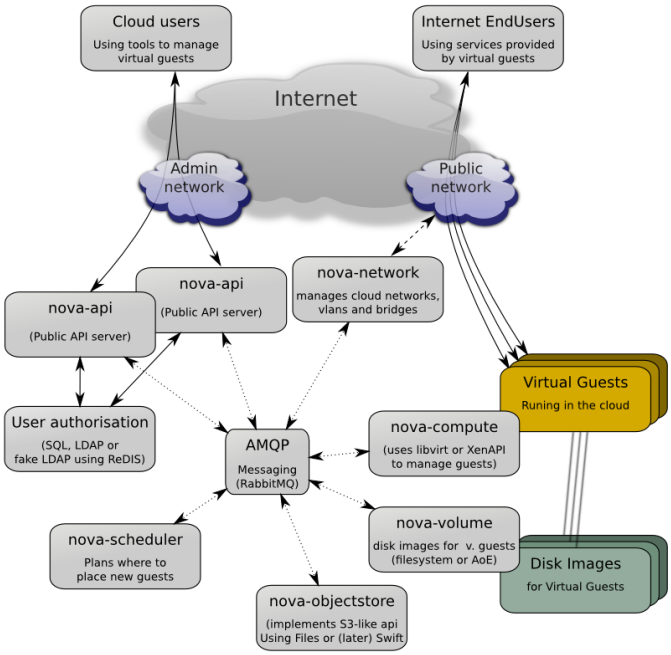

Figure 5: OpenStack (Nova)

6.3 OpenNebula

OpenNebula [19] 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.

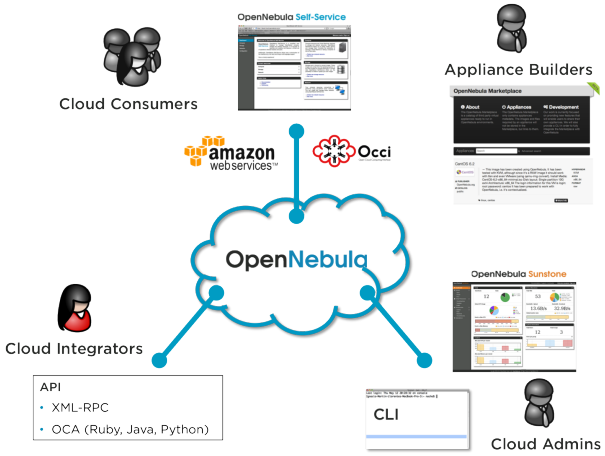

Figure 6: OpenNebula

7. USER-CENTRIC APPROACH

Standardization, 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 listed in the taxonomy. But history has already taught us that the industry might take years or even decades to agree upon such common standards. 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.

7.1 Xen Blanket

Based on nested virtualization [20], the Xen-Blanket [21] 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.

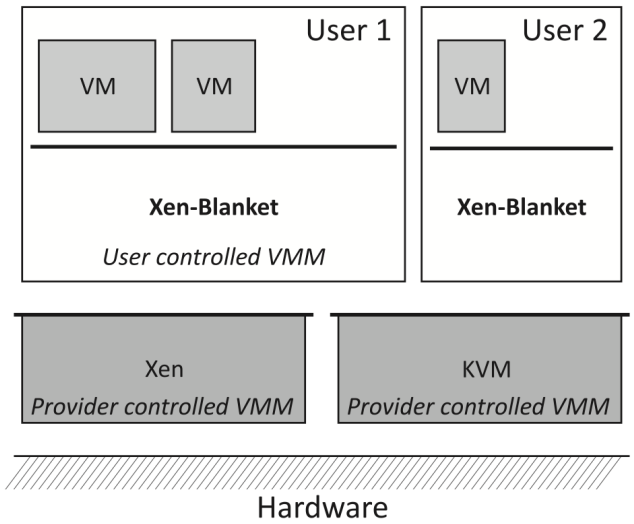

Figure 7: Xen-Blanket

7.2 OpenCirrus

OpenCirrus [22] is an initiative 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.

8. RESEARCH CHALLENGES

8.1 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.

8.2 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 [26] need to constantly incorporate user feedback and customization features into the SLA evaluation framework.

8.3 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.

9. CONCLUSION

James Gosling[23] 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.

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

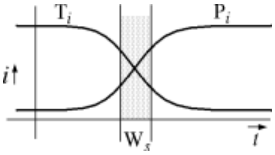

Figure 8: Standardization process

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.

10. REFERENCES

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

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

[3] Peter Mell and Tim Grance, "The NIST Definition of Cloud Computing," http://csrc.nist.gov/groups/SNS/cloud-computing/cloud-def-v15.doc, 2009.

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

[5] 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.

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

[7] 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.

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

[9] 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.

[10] Amazon web services. [Online]. Available: http://aws.amazon.com

[11] Google compute engine. [Online]. Available: http://cloud.google.com/products/compute-engine.html

[12] Rackspace. [Online]. Available: http://www.rackspace.com/

[13] Gogrid. [Online]. Available: http://www.gogrid.com/

[14] Microsoft azure. [Online]. Available: http://www.windowsazure.com/en-us/

[15] Cloud data management interface by snia. [Online]. Available: http://www.snia.org/cdmi

[16] Open cloud computing interface. [Online]. Available: http://occi-wg.org/

[17] Eucalyptus. [Online]. Available: http://www.eucalyptus.com/

[18] Openstack. [Online]. Available: http://www.openstack.org/

[19] Opennebula. [Online]. Available: http://opennebula.org/

[20] 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.

[21] 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.

[22] 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.

[23] Phase relationships in the standardization process. [Online]. Available: http://nighthacks.com/roller/jag/resource/StandardsPhases.htm

[24] 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.

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

[26] 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.

[27] 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.

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

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

[30] 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.

[31] 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.

[32] 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.