A Survey on Cloud Interoperability: Taxonomies, Standards, and Practice

Zhizhong Zhang, Chuan Wu, David W.L. Cheung
Department of Computer Science, The University of Hong Kong, Hong Kong
email:{zzzhang,cwu,dcheung}@cs.hku.hk

ABSTRACT

Cloud computing is a new computing paradigm that allows users with different computing demands to access a shared pool of configurable computing resources (e.g., servers, network, storage, database, applications and services). Many commercial cloud providers have emerged in the past 6-7 years, and each typically provides its own cloud infrastructure, APIs and application description formats to access the cloud resources, as well as support for service level agreements (SLAs). Such vendor lock-in has seriously limited the flexibility that cloud end users would like to process, when it comes to deploy applications over different infrastructures in different geographic locations, or to migrate a service from one provider's cloud to another. To enable seamless sharing of resources from a pool of cloud providers, efforts have emerged recently to facilitate cloud interoperability, *i.e.*, the ability for multiple cloud providers to work together, from both the industry and academia. In this article, we conduct a comprehensive survey on the state-of-the-art efforts, with a focus on interoperability among different IaaS (infrastructure as a service) cloud platforms. We investigate the existing studies on taxonomies and standardization of cloud interoperability, as well as practical cloud technologies from both the cloud provider's and user's perspectives to enable interoperation. We pose issues and challenges to advance the topic area, and hope to pave a way for the forthcoming research.

> *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 has recently emerged as a new computing paradigm for organizing a shared pool of servers in data centers into a cloud infrastructure, which can provide on-demand computing resources (e.g., CPU, storage, network, database, applications and services) to users anywhere anytime, in a pay-as-you-go manner. Cloud computing facilitates scalable and cost-effective computing solutions, by rapidly provisioning and releasing resources with minimal user management effort and cloud provider interaction. According to the NIST definition of cloud computing [2], the cloud computing model is composed of five essential characteristics, *on-demand self-service*, *broad network access*, *resource pooling*, *rapid elasticity* and *measured service*, and three service models, *SaaS* (Software as a Service), *PaaS* (Platform as a Service) and *IaaS* (Software as a Service). A cloud has been very useful in supporting various applications, including computation-intensive applications [3], storage-intensive applications [4], and bandwidth-demanding Web applications [5].

Vendors in the cloud market has rapidly proliferated in the past 6-7 years, including Amazon [6], Google [7], Microsoft [8] and SalesForce [9]. Each of them promotes its own cloud infrastructure, and incompatible standards and formats to access the cloud, preventing them from agreeing upon a widely accepted, standardized way to support cloud applications. Nonetheless, the need for multiple clouds to be able to work seamlessly together, *i.e., cloud interoperability*, is rising.

From the users' point of view, cloud customers are typically reluctant to be bound to the cloud offered by a single provider, since it might not be able to satisfy all the needs of the user, such as on geographic distribution of the cloud data centers, SLAs (Service Level Agreements), etc., and might bring the potential risk of ridiculously high charges later on, due to the development lock-in. The cloud users are most likely looking for interoperable clouds, where they can have full control on where to deploy their applications and migrate their services easily when need arises, without extra development investments.

From the providers' perspective, this incompatibility among cloud providers can protect the interest of each provider temporarily, but not in the long run when the cloud market becomes more mature. Efforts have emerged to establish standards for federating clouds owned by different providers, especially advocated by relatively small cloud providers (*e.g.*, Rackspace [10], GoGrid [11] and late-comers [12] (*e.g.*, Red Hat[13], Dell [14], Oracle[15], etc.). Interoperable clouds promisingly represent the trend of cloud technologies, since they better fulfill the ultimate goal of the cloud computing paradigm, in providing global-scale, "unlimited" computing utilities with unified access interfaces[16].

The importance of cloud interoperability has been highlighted by both the industry and the academia. The industry tries to address the cloud interoperability issues via *standardization* and many cloud interoperability standards have been proposed and even put into use in recent years [17, 18, 19]. The European Commission has also aimed to "identify by 2013 a detailed map of the necessary standards (inter alia for security, interoperability, data portability and

reversibility)" [20]. However, it may take years for the standards to be fully agreed upon and adopted, if ever. As practical, short-term solutions, researchers in both the industry and the academia have been developing technologies to enable interoperation among clouds, from both the cloud provider's and user's perspectives.

In this article, we conduct a comprehensive survey on the state-of-the-art efforts to enable cloud interoperability, with a focus on interoperability among different IaaS (infrastructure as a service) cloud platforms. We investigate the existing taxonomies, standards and implementation of cloud interoperability, as well as practical cloud technologies (*e.g.*, nested virtualization [21]) to enable interoperation. We pose issues and challenges to advance the topic area, and hope to pave a way for the forthcoming research.

The rest of the article is organized as follows. Sec. 2 introduces the definitions and different models of cloud interoperability. Sec. 3 presents a taxonomy of interoperability over IaaS clouds. Sec. 4 discusses three mainstream cloud interoperability standards and their comparisons. Representative practical systems which implement those standards are presented in Sec. 5. User-centric approaches are discussed in Sec. 6. Finally, we summarize open issues and research challenges, and conclude the survey in Sec. 7.

2. CLOUD INTEROPERABILITY: AN OVERVIEW

2.1 Definition

Interoperability generally means the ability for different heterogeneous systems to function and interact together. According to the European Commission (EC) [22], 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. Among clouds, interoperability can be defined as the ability to understand each others' application formats, SLA templates, formats of authentication and authorization token, attribute data and others [23], such that different clouds can *cooperate* or *interoperate* [24].

Cloud computing *interoperability*, *compatibility* and *portability* are closely related terms and may often be confused. Cohen clarifies the similarities and the differences among these terms in an attempt to exemplify and differentiate them [25]. Cloud *interoperability* is the ability for multiple cloud providers to work together. Cloud *compatibility* and *portability* answer to the question "how?": cloud compatibility means that the application and data can work in 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 choice of cloud provider, operating system, storage format or APIs.

2.2 Cloud Interoperability on Different Service Models

A clear understanding and classification of the requirements that ensure interoperability are the first step towards the standardization of cloud computing platforms, APIs and services. Based on the three cloud service models widely adopted by the cloud computing community as follows [26, 2], a semantic approach in delimitating cloud computing interoperability has been presented by Sheth and Ranabahu [27], where the interoperability is closely associated with the type of heterogeneity that arises during the interoperation of different types of clouds.

- *Infrastructure as a Service (IaaS)*. The capability provided to the cloud consumer is to provision processing, storage, networking, and other fundamental computing resources, where the consumer is able to deploy and run arbitrary software include operating systems and applications. Examples of IaaS clouds are Amazon EC2 [6] and Google Compute Engine [7].

- *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 cloud provider. An example PaaS cloud is Google App Engine [28].

- *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 Salesforce CRM (Customer Relationship Management) [9] and GMail [29].

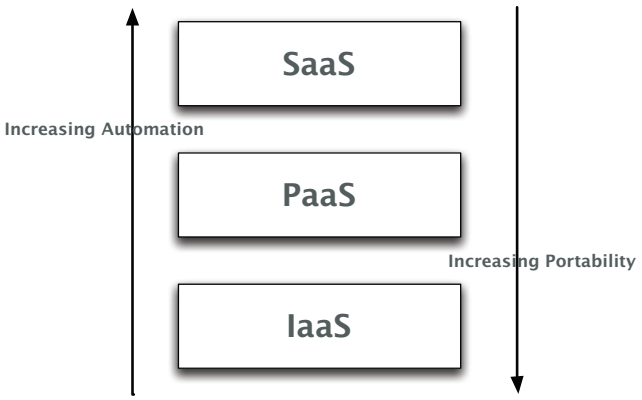

Figure 1: The cloud landscape illustrating the differences in user's responsibility for interoperability on each of the three types of clouds.

In particular, Fig. 1 [27] shows that cloud users have increasing responsibilities for deploying over an SaaS cloud, to a PaaS cloud, and to an IaaS cloud, as well as more flexibility (portability); on the other hand, from IaaS to PaaS to SaaS, an increasing level of automation is achieved, since the users need to deal with less software deployment and management themselves. Cloud users would choose carefully among different cloud types depending on their own needs of portability and automation. For example, PaaS clouds offer faster setup for applications than IaaS clouds, and users can often exploit the free hosting opportunities provided by some PaaS providers (e.g. Google App Engine). When a user's 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).

Different IaaS (or PaaS, or SaaS) providers offer different interfaces for users to access their infrastructures. For a cloud user to deploy its application over different IaaS (or PaaS, or SaaS) clouds, it may often need to employ middleware or common standards to homogenize the APIs. For example, the Open Virtualization Format (OVF) is used to allow migration of virtual appliances (definition to be given in Sec. 3) across IaaS clouds, standard email protocols like IMAP and POP enable the smooth migration of enterprise mail services among different SaaS providers.

PaaS and SaaS are typically provided on top of IaaS, which makes the interoperability of IaaS clouds much more important than the other two. IaaS interoperability is more complicated as well: PaaS mostly depends development frameworks (*e.g.*, Django, Ruby on Rail, PHP, etc.), which can be better supported by the respective development community, when interoperability among different PaaS clouds is demanded; SaaS intends to provide a complete service framework, whose interoperability issue is mostly about the data. Therefore, in this article, we will mainly focus on interoperability among IaaS clouds, and discuss the existing efforts on enabling seamless interoperation and migration of applications over different IaaS platforms.

3. INTEROPERABILITY IN IAAS CLOUDS: A TAXONOMY

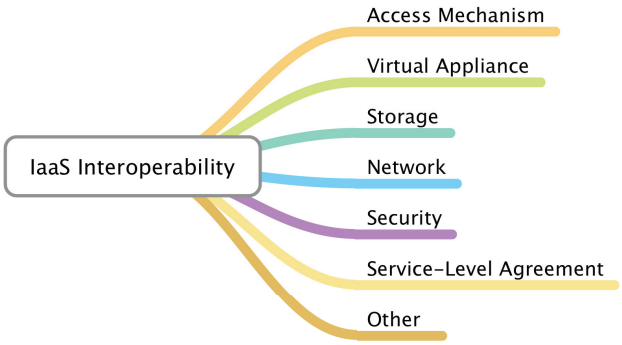

Figure 2: Interoperability of IaaS Taxonomy.

Teckelmann and Reich [30, 31] have presented a taxonomy of interoperability for IaaS clouds (Fig. 2). The taxonomy discusses all important issues, to demonstrate the needs and trends in the state-of-the-art development aiming for IaaS interoperability [32].

(1) *Access Mechanisms.* Access mechanisms define how a cloud service can be accessed by developers or end users. The industry has considered three common standard types of access mechanisms: *Application Programming Interface (API)*, *Graphical User Interface (GUI)*, and *Command-Line Interface (CLI)* [33]. GUIs and CLIs are mainly implemented on top of the existing APIs, which may use proprietary data formats based on XML, JSON or plain HTTP, and lack interoperability.

(2) *Virtual Appliance (VA).* The idea of a virtual appliance is to deliver a service as a complete software stack installed on the VMs, as proposed by DMTF [17]. Three issues of virtual appliances must be addressed, towards its goal of enabling cloud interoperability:

- *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 is needed in all five stages of the life cycle, which will pose a significant problem to cloud providers. Starting from pre-configured VM images, updates should be applied across the whole life cycle of a VA carefully so as not to incur SLA 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 [34]. Furthermore, when hardware-assisted virtualization is taken into consideration, cloud providers may also need to choose between Intel's VT and AMD's AMD-V [34]. Solutions are needed to enable interoperability among clouds employing different virtualization technologies. In addition, *host architecture*, *host operating system* and *license* are also important to enable interoperability of the upper-level layer (*i.e.*, Hypervisor), which, to some extent, depends on the host architecture and operating system.

- *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, *i.e.*, migration is an essential task for the virtualization managers.

(3) *Storage.* Both the management and organization of storage are important issues to address for compatibility among different clouds. There are three topics on storage management in a cloud: *backup*, *replication*, and *snapshots*. The backup functionality is important to guarantee long-term preservation of the data on a non-volatile storage media. Replication means that data are distributed to several sites. Snapshots are full copies of data objects. On the organization of storage, the schemes of organization and the image format are the major concerns. To facilitate compatibility, the introduction of an intelligent object storage layer can be a solution, where VM images and related data are decoupled and stored on the most appropriate storage, and logical connections can be kept using metadata information [35].

(4) *Network.* There are two important topics in terms of network interoperability: *addressing* and *application-level communication.*

- *Addressing.* A major issue is to have a reliable remote access to applications and their underlying VMs, even when they are being moved between subnets within a cloud or across the wide area networks.

- *Application-level communication.* The APIs should be RESTful, in order to minimize the coupling between client and server components in a distributed application. In the context of interoperable IaaS clouds, RESTful APIs are the appropriate choice due to the requirement that the client should be extremely resilient

to changes on the server. There are two major communication protocols in concern, which implement RESTful APIs: *HTTP* (application-level transport protocol) and *XMPP* (XML over TCP).

In terms of IaaS interoperability, IP mobility is essential during live migration of virtual machines; otherwise it would directly cause service downtime and therefore violate the SLA. Many solutions exist to extend the capability of IPv4, including IPv6 and software-defined networking, which can better facilitate IP mobility [36].

(5) *Security*. Important security topics for cloud interoperability include *authentication*, *authorization*, *accounting* and *encryption*.

- *Authentication*. The confirmation of a stated identity is an essential security mechanism in clouds. To achieve cloud interoperability, authentication mechanisms have to be agreed upon among clouds to facilitate accessing resources from each cloud. Two kinds of authentication mechanisms are commonly presented: *HTTP Authentication* and *Public Key Infrastructure*.

- *Authorization*. Cloud providers need to allocate the resources or rights according to a user's credentials after the user has proven to be what he/she stated. In a federation of clouds, users with different backgrounds and requirements should be granted accesses to different resources of each cloud. Therefore, a proper authorization mechanism is necessary for the cloud federation to cooperate together and provide the best user experience possible for the cloud users.

- *Accounting*. In conjunction to security, accounting is necessary for the record of events and operations, and the saving of log information about them, for system and fault analysis. Interoperability among clouds can be seriously affected if no such information is available. The current issue is that a well-defined best-practice guideline (not necessarily standard) on accounting 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 should be agreed upon in order for cloud users from different endpoints to access the resources of a cloud federation. Encryption mechanisms (SSL, TSL, VPN) are some of the widely adopted protocols [37].

(6) *Service Level Agreement*. SLAs are ubiquitous in modern IT systems. In terms of cloud interoperability, four important topics are involved on cloud SLA: *architecture*, *template format*, *monitoring* and *SLA objectives*.

- *Architecture*. Different cloud providers within a federation of clouds might have different SLAs. Thus, it becomes essentially important to have a well-adopted SLA architecture to measure and manage SLA requirements for different clouds. Otherwise, cloud users might get confused of the different SLA specifications. For example, Web Service Agreement Specification (WS-A) is the standard for the SLA management architecture in Web service environments.

- *Template Format*. Template formats are electronic representations of SLAs for the purpose of automated management. In order to achieve interoperability, it is necessary to agree upon a single template format. While most of the previous generation of template formats are *not* machine-readable, which brings trouble for automated management, the machine-readable 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 services as mentioned in the contract.

- *SLA Objectives*. Service level objectives (SLOs) are the core components of a SLA, which are quality-of-service measurements for the performance of the service provider. In a scenario where a customer wants to change its cloud provider, the SLOs should be made consistent in order to compare the old SLA with a new one.

The aforementioned aspects of SLA interoperability share a common characteristic: they tend to provide the necessary functionality in the form of APIs to the cloud providers and users. APIs are crucial to interoperability for exporting SLA management functionalities. By separating SLA managing entities into modular components, their interoperability can be strengthened through exchangeability and extensibility.

4. STANDARDIZATION

Many organizations are involved in various standardization efforts on the common theme of clouds. Notable among them are the working groups operating under the Open Grid Forum (OGF) umbrella [38]. Other prominent industry consortiums active in cloud standardization are Distributed Management Task Force Inc. (DMTF) [39] and the Storage Networking Industry Association (SNIA) [18]. In this section, we discuss three major open standards on cloud interoperability, which are widely adopted.

These open standards help build bridges towards the goal of achieving user application and cloud provider interoperability. A significant progress has been made for pivotal elements such as storage, infrastructure management, and application description formats in the interoperability taxonomy, but much work remains to reach the final destination [23]. Below is a table summarizing the mapping between the coverage of these standards and the cloud interoperability taxonomy.

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

4.1 OVF

The Open Virtualization Format (OVF) [17] is a packaging standard initiated by DMTF [39] for deployment of

virtual appliances. By abstracting a virtual appliance as a single atomic unit, 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 [40].

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 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 vendor-specific information.

The OVF describes an open, portable, efficient and flexible format for packaging and distributing virtual appliances, with the following key features: (1) Enabling optimized distribution of virtual appliances; (2) Providing a simple, automated user experience, as it offers a robust and user-friendly approach to streamline the VA installation process. (3) Supporting both single and multi virtual machine configurations. (4) Enabling portable VM packaging, as OVF does not rely on any specific host platform, virtualization platform, or guest operating system.

OVF addresses the following issues of *Virtual Appliance* in the cloud interoperability taxonomy.

- *Life Cycle*: With a standard format of VA, the five stages of VA life cycle can now be handled freely by the developers, due to the flexibility of the XML descriptor employed in OVF, which contains all the metadata of the VA. Cloud providers are also capable to build pre-configured VM images based on OVF, therefore allowing cloud users to better automate their service deployment.

- *Virtualization Platform*: The extensibility of OVF allows different virtualization platforms to be defined within a VA, whether it be OS-level virtualization, paravirtualization, or hardware-assisted virtualization, thus allowing cloud vendors with different virtualization platforms (*e.g.*, AWS, Microsoft Azure) to interoperate together.

- *Virtualization Manager*: OVF also provides a simple and unified interface for the virtualization managers to perform migration tasks within or across data centers of different cloud providers (*e.g.*, *libvirt* [41]).

4.2 CDMI

The Cloud Data Management Interface (CDMI) [18], proposed by the Storage Networking Industry Association (SNIA), defines the functional interface that applications can use to create, retrieve, update and delete data elements from a cloud. It is the storage backbone in cloud interoperability. CDMI features functions that (1) allow clients to discover the capabilities available in the cloud storage offering, (2) manage containers and the data that are placed in them, and (3) 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 on *Storage* of the cloud interoperability taxonomy.

- *Backup*: Backup is the most common operation for cloud storage, which is enabled by CDMI as a simple interface. Cloud users can schedule the backup or perform backup operations manually via the interface.

- *Replication*: CDMI containers and objects are mapped to a mounted file system's directories and files. This mapping allows their flexible replication in the low-level data storage cloud, even across different data centers.

- *Snapshots*: CDMI defines a snapshot as a point-in-time copy (image) of a container and all of its content. A snapshot operation is requested using the container update operation. 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 a source container to be restored to their values at a previous point in time, even across different cloud providers, using a CDMI copy operation.

4.3 OCCI

Open Cloud Computing Interface (OCCI) [19] is a boundary API that acts as a service front-end to an IaaS provider's internal infrastructure management framework. The OCCI community is an open community under the umbrella of the Open Grid Forum (OGF), with contributing members from both the industry and the academia.

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. It is compatible with the existing standards such as the OVF and the CDMI, and serves as an integration point for standardization efforts among DMTF, IETF[42] and SNIA.

The current OCCI specifications define a set of common standard interfaces of management for IaaS cloud interoperability. The documents are divided into three categories consisting of the OCCI Core, the OCCI Renderings and the OCCI Extensions.

- The *OCCI Core* defines a representation of instance types which can be manipulated through an OCCI rendering implementation. It is an abstraction of real-world resources, including the means to identify, classify, associate and extend those resources. It interacts with the renderings (including their associated behaviors) and can be expanded through extensions.

- The *OCCI Renderings* each describe a particular rendering of the OCCI Core model, *i.e.*, how each OCCI Core model is rendered over the HTTP protocol that leads to a RESTful API implementation. 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 the OCCI Core.

- The *OCCI Extension* each describe a particular addition to the OCCI Core model.

OCCI addresses the following issues on *Access Mechanism*, *Network*, *Security* and *SLA* of the cloud interoperability taxonomy.

- *Access Mechanism*: Access mechanisms are exclusively described in the OCCI Rendering specifications, where RESTful HTTP APIs are defined. Each resource instance within an OCCI system, *e.g.*, Compute, Network, Storage, etc., has a unique identifier, 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 layer-2 networking entity (e.g. a virtual switch), in which the network addressing issue among clouds could be addressed. It can be extended using the mixin mechanism to support layer-3 and layer-4 capabilities such as TCP/IP etc. In regard to application-level communication, OCCI facilitates a RESTful API over HTTP.

- *Security*: Elements described by the OCCI Core 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 an OCCI rendering specification, which in any case must address transport-level security and authentication on the protocol level.

- *SLA*: Standardization on SLAs is addressed in the *Billing and Negotiation/Agreement* part of the OCCI specification, which is currently still in progress.

5. CLOUD MANAGEMENT SYSTEMS SUPPORTING INTEROPERABILITY

Cloud standards could not evolve by themselves without the cooperation of their open source implementation. The representative, state-of-the-art implementation of IaaS cloud management systems includes *OpenStack*, *OpenNebula* and *Eucalyptus*. Based on the current IaaS practice, almost all representative implementation of such cloud management platforms attend to interoperability with the dominating IaaS cloud platforms such as Amazon EC2.

5.1 OpenStack

As an open-source project managed by the OpenStack Foundation [43], OpenStack is an IaaS cloud management platform consisting of a trio of "core" services (Fig. 3):

- The compute controller is named as *Nova*, which provides virtual servers upon demand. Nova's architecture is designed to scale horizontally on standard hardware with no proprietary hardware or software requirements, and with the ability to integrate with legacy systems as well as other IaaS systems. In terms of supported hypervisors, Nova supports Xen [44], KVM [45], LXC [46] and even Microsoft Hyper-V [47]. It also supports different CPU architectures (*e.g.*, x86, amd64 and ARM). These make Nova interoperable with the *Compute* service of other IaaS Clouds, such as Amazon EC2.

- The storage systems, *Swift* and *Cinder*, provide object storage and block storage, respectively. They are designed to be interoperable with Amazon's object storage (S3) and block storage (EBS). In particular, Swift implements a subset of S3 APIs with the WSGI middleware [48].

- The image service, *Glance*, provides a catalog for virtual disk images, which includes many of the same features as Amazon's AMI (Amazon Machine Image) catalog [49]. These disk images are commonly used in the OpenStack Compute module. Although the AMIs from Amazon can not be used directly in Glance (yet), their common APIs enable third-party IaaS management tools such as RightScale to automate management of images in OpenStack (using its own image format, compatible with AMI).

The OpenStack networking system, *Quantum*, manages networks and IP addresses in the cloud, and intends to provide "networking as a service" between interface devices, *e.g.*, virtual network interface cards (vNICs), managed by other OpenStack services (*e.g.*, Nova). Quantum facilitates software-defined network [50] capability in OpenStack, which is quite an effective solution for the network addressing issue over WAN. It also makes possible native live migration of VMs over WAN, given that the underlying hypervisor supports it. Equipped with the software-defined network functionality, OpenStack has overcome many of the obstacles (network-wise) towards interoperable clouds.

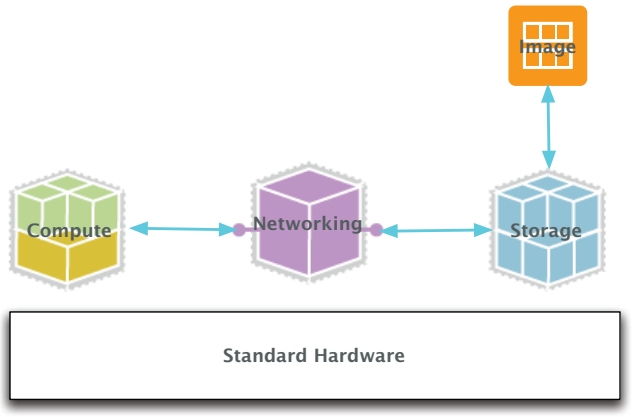

Figure 3: OpenStack.

5.2 Eucalyptus

Eucalyptus [51], created by Eucalyptus Systems, Inc., is a software architecture that creates scalable private and hybrid clouds within one's existing IT infrastructure. It features high-fidelity Amazon AWS (Amazon Web Services) API implementation, in order to be easily interoperable with AWS clouds. Eucalyptus consists of the following key components (Fig. 4):

- Cloud Controller, which provides the Amazon EC2-compatible functionality;

- Walrus Storage, which provides the Amazon S3-compatible functionality;

- Cluster Controller, which manages a cluster in the cloud;

- Storage Controller, which provides the Amazon EBS-compatible functionality;
- Node Controller, which controls the virtual machine instances.

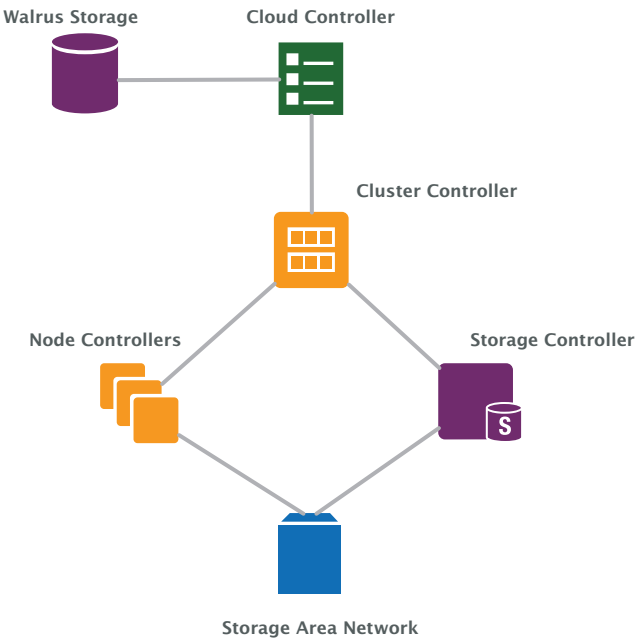

Figure 4: Eucalyptus.

Eucalyptus has announced an agreement [52] that enables customers to more efficiently migrate workloads between their existing data centers and Amazon clouds while using the same management tools and skills across both environments.

- *Flexibility in VM Formats*: Cloud users can run multiple versions of Windows and Linux virtual machine images on Eucalyptus clouds, and can build a library of Eucalyptus or Amazon Machine Images (AMIs) with application metadata that are decoupled from infrastructure details to allow seamless operation on Eucalyptus or AWS clouds. In addition, VMware images and vApps (multi-VM OVF packages) can be easily converted to run on Eucalyptus or AWS clouds.
- *Heterogeneous Hypervisor Management*: Cloud users can build and manage mixed hypervisor cluster environments in Eucalyptus clouds and manage their existing vSphere [53], ESX [54], KVM and Xen virtual environments as AWS-compatible Eucalyptus hybrid clouds. A self-service management portal can manage Xen, KVM, and VMware virtual environments from a single panel, leading to much interoperable and efficient clouds.
- *Interoperable Identity Management*: As for *Access Mechanisms*, Eucalyptus user identity management can be integrated with the existing Microsoft Active Directory or LDAP systems. In addition, Eucalyptus identity management interfaces are compatible with the AWS Identity and Access Management (IAM) APIs, which enables integrated management of hybrid clouds of Eucalyptus and AWS.

In summary, Eucalyptus provides a robust, on-premise cloud platform for cloud users to take advantages of AWS by implementing AWS EC2, EBS, S3 and IAM APIs natively, leading to interoperable hybrid clouds of Eucalyptus and AWS.

5.3 OpenNebula

OpenNebula [55], promoted by the OpenNebula Community, is an open-source data center virtualization technology, offering feature-rich, flexible solutions 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. 5):

- Cloud interfaces for *Cloud Consumers*, including the OCCI, EC2 Query and EBS interfaces, and a simple self-service portal;
- Administration interfaces for *Cloud Administrators*, 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 [56];
- A marketplace for *Appliance Builders* with a catalog of virtual appliances ready to run in OpenNebula environments.

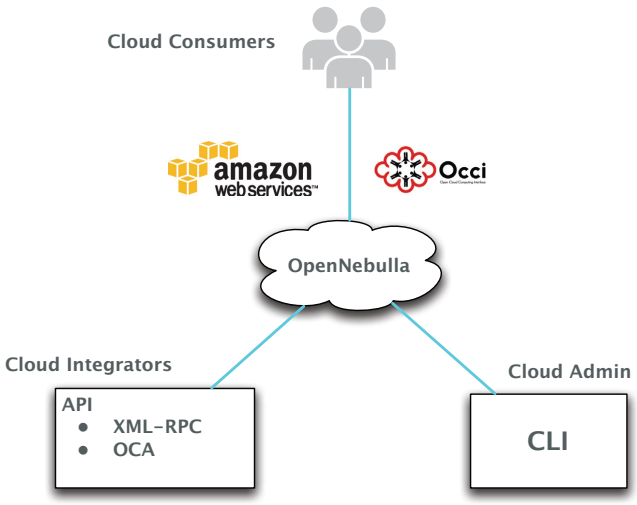

Figure 5: OpenNebula.

In terms of cloud interoperability, OpenNebula implements a full EC2 Query interface, which enables seamless integration with Amazon EC2 clouds. Along with EC2 Query interface, OpenNebula natively includes a second remote interface: the *OCCI* interface which was discussed in details

in Sec. 4.3. By enabling the EC2 Query interface and OCCI interface, large-scale cloud infrastructures can be potentially built, opened to a wider audience. Manageability of a large infrastructure poses a significant challenge on interoperability among resources in the infrastructure. For this reason, OpenNebula implements *oZones* and *VDCs (Virtual Data Centers)*:

- oZones allow the centralized management of multiple running OpenNebula instances that are called zones (whether it be OpenNebula or AWS instances). This allows the oZone administrators to monitor each zone, and to grant access to different zones to particular users, from a centralized CLI and web GUI.

- Inside each zone, it is possible to define multiple VDCs that contain a set of virtual resources (images, VM templates, virtual networks and virtual machines) and users that use and control those virtual resources.

6. USER-CENTRIC APPROACH

Standardization (together with its implementations), as mentioned in the previous sections, aims to address the interoperability issues by building up common standards (OVF, CDMI, OCCI, etc.) in nearly every dimension of the issues and challenges in the taxonomy. But history has taught us lessons that the industry might take years 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 proposed *user-centric* approaches, as practical solutions to achieve interoperability of cloud users' applications and services over the multiple, existing cloud infrastructures. In this section, we overview a few representative developments in this area.

6.1 Xen Blanket

Fundamentally, today's clouds lack the homogeneity necessary for cost-effective multi-cloud deployment of VM images. A single VM image cannot be deployed on more than one IaaS cloud without modification. There is no consistent set of hypervisor-level services across different providers. While progress towards multi-cloud homogeneity is expected through standardization efforts such as the OVF, these approaches will likely be limited to simple cloud attributes like image formats.

As a sub-project of *xCloud* [57], which was proposed by researchers in Cornell University, *Xen-Blanket* [58] is a technology that transforms the existing heterogeneous clouds into a uniform user-centric homogeneous offering. Xen-Blanket is based on nested virtualization [21], a technique that allows multiple second-layer hypervisors to run on top of an underlying hypervisor, and 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 the respective underlying hypervisor of the existing cloud. An illustration of the idea is given in Fig. 6. Particularly, a Blanket layer embodies three important concepts: (1) 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. (2) 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. (3) 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.

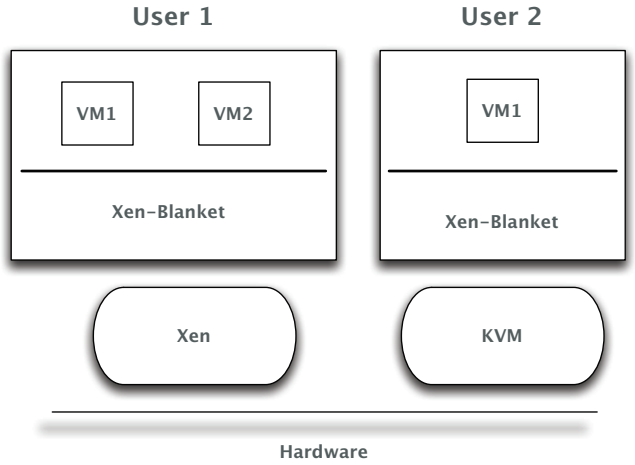

Figure 6: Xen-Blanket.

The Xen-Blanket has been deployed on both Xen-based and KVM-based hypervisors, on public and private infrastructures such as Amazon EC2, an enterprise cloud, and a private cloud in Cornell University. It has been used to facilitate VM migration to and from Amazon EC2 without any modifications to the VMs. As a middleware between the VMs and the underlying hypervisor, the Xen-Blanket is another application of David Wheeler's theorem: *All problems in computer science can be solved by another level of indirection* [59].

6.2 Libraries to Enable Interoperability

Apart from full implementations of an IaaS cloud management system, there have been other efforts aiming to leverage the existing cloud platforms by offering APIs for developing interoperable applications on those platforms. These include *LibCloud* [60] and *δ-Cloud* [61]. Instead of building their own clouds using OpenStack or Eucalyptus, enterprises or developers can also adopt these cloud libraries to achieve interoperable clouds, using the existing cloud infrastructures such as Amazon EC2, Rackspace, Google Compute Engine, etc. In particular, LibCloud and δ-Cloud provide abstractions that help developers write applications without being tied to a specific cloud vendor (or an intermediate layer such as OpenStack).

While lack of lock-in is a concrete benefit, it comes at a price, that the APIs cannot expose features that differentiate the different platforms. This may lead to a significant loss of functionality or performance. Therefore, if the developer needs to create applications against multiple clouds for portability, then *LibCloud* is the way to go. If an enterprise needs rich features and a full ecosystem, then a combination of OpenStack (or Eucalyptus, or OpenNebula) and public clouds (e.g., Amazon EC2) is a better choice.

7. CONCLUDING DISCUSSIONS

James Gosling [62] made a nice note on the phases that a standardization process goes through, as shown in Fig. 7. The i axis describes the 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 better understood, while the political interest of the technology increases, generally fueled by economic pressures. Applying this model to cloud interoperability standardization, we are currently in the phase where technologists are showing strong interest and trying to understand and develop the technology for cloud interoperability. There is potentially still a long way to go for the standard to be usefully formed and adopted.

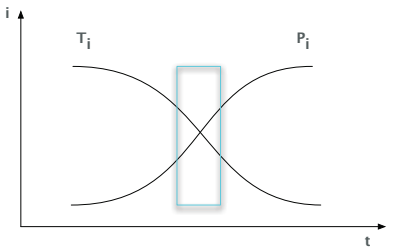

Figure 7: Standardization process. [62]

On the other hand, we have systematically surveyed a number of practical cloud technologies, emerged or still actively being developed in the very recent years, to enable interoperation among IaaS clouds, including full-fledged cloud management platforms, cloud API libraries, and novel virtualization technologies. Corresponding to the taxonomy of cloud interoperability, many issues are still open and a lot of tasks remain. We list a few important ones hereafter.

Virtualizing network. Compared with other elements in cloud computing (*i.e.*, Compute, Storage, etc.), network is more important in terms of achieving cloud interoperability by realizing network virtualization. Some cloud management platforms, such as OpenStack, has incorporated software-defined networking capability, with can improve visualization of servers/network resources for better interoperability. Nevertheless, software-defined networking is still at its infant state. Innovations are expected in exploiting software-defined networking capabilities to better facilitate inter-cloud cooperability.

SLA. 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 they have migrated their core applications onto their entrusted clouds. In other words, it is vital for consumers to obtain guarantees from providers on service delivery, through the SLAs. Federation of clouds raises a number of significant problems for SLAs. For example, it is almost inevitable to avoid downtime during live migration between interoperable clouds, and the cloud providers (both the incoming and outgoing ones) need to possess precise and updated information on the resource usage at any particular time during the migration. Furthermore, advanced SLA mechanisms [63] may be needed to constantly incorporate user feedback and customized features into the SLA evaluation framework, which becomes more challenging in the context of multiple interoperable clouds.

Resource pricing. From a cloud provider's perspective, the elastic resource pool (through either virtualization or multi-tenancy) in a federated cloud has made the cost analysis far more complicated than in regular data centers.For example, 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 they can easily be migrated among interoperable clouds. If we consider the case of nested virtualization, the picture will be even more complicated, because a VM might also be a hypervisor. A VM 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. How to implement cost analysis and pricing of various resources in scenarios of interoperable clouds remain open, significant challenges.

8. REFERENCES

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

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

[3] I. Foster, Y. Zhao, I. Raicu, and S. Lu, "Cloud computing and grid computing 360-degree compared," *Grid Computing Environments ...*, 2008.

[4] *Resource-aware adaptive scheduling for MapReduce clusters*. International Federation for Information Processing, Dec. 2011.

[5] W. Iqbal, M. Dailey, and D. Carrera, "SLA-driven adaptive resource management for web applications on a heterogeneous compute cloud," *Cloud Computing*, 2009.

[6] *Amazon Web Services*, http://aws.amazon.com.

[7] *Google Compute Engine*, http://cloud.google.com/products/compute-engine.html.

[8] *Microsoft Azure*, http://www.windowsazure.com/en-us/.

[9] *SalesForce*, http://www.salesforce.com.

[10] *RackSpace*, http://www.rackspace.com/.

[11] *GoGrid*, http://www.gogrid.com/.

[12] *Open Data Center Alliance*, http://www.opendatacenteralliance.org/.

[13] *Red Hat OpenShift*, https://openshift.redhat.com/app/.

[14] *Dell Cloud Computing*, http://content.dell.com/us/en/gen/dell-cloud-computing.

[15] *Oracle Cloud Computing*, http://www.oracle.com/us/solutions/cloud/.

[16] B. Rochwerger, D. Breitgand, and E. Levy, "The reservoir model and architecture for open federated cloud computing," *IBM Journal of ...*, 2009.

[17] *Open Virtualization Format*, http://www.dmtf.org/standards/ovf.

[18] *Cloud Data Management Interface by SNIA*, http://snia.org/cdmi.

[19] *Open Cloud Computing Interface*, http://occi-wg.org/.

[20] *Unleashing the Potential of Cloud Computing in Europe*.

[21] 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.

[22] *European Interoperability Framework v2*, http://www.informatizacia.sk/ext_dok-european_interoperability_framework/9319c.

[23] 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.

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

[25] R. Cohen, "Examining cloud compatibility, portability and interoperability," *...*, 2009.

[26] 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.

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

[28] *Google App Engine*, http://appengine.google.com/.

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

[30] 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.

[31] 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.

[32] 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.

[33] *Amazon EC2 API Tools Reference*, http://docs.amazonwebservices.com/AWS-EC2/latest/CommandLineReference.

[34] O. Agesen, A. Garthwaite, J. Sheldon, and P. Subrahmanyam, "The evolution of an x86 virtual machine monitor," *SIGOPS Operating Systems Review*, vol. 44, no. 4, 2010.

[35] G. DeCandia, D. Hastorun, and M. Jampani, "Dynamo: amazon's highly available key-value store," *ACM SIGOPS ...*, 2007.

[36] R. Bradford, E. Kotsovinos, A. Feldmann, and H. Schiöberg, "Live wide-area migration of virtual machines including local persistent state," in *the 3rd international conference*. New York, New York, USA: ACM Press, 2007, p. 169.

[37] 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.

[38] *Open Grid Forum*, http://www.gridforum.org/.

[39] *DMTF (Distributed Management Task Force*, http://www.dmtf.org/.

[40] *OVF Members List*, http://www.dmtf.org/about/list.

[41] *Libvirt*, http://libvirt.org/.

[42] *The Internet Engineering Task Force*, http://www.ietf.org.

[43] *OpenStack*, http://www.openstack.org/.

[44] *The Xen Hypervisor*, http://www.xen.org/.

[45] *KVM - Kernel Based Virtual Machine*, http://www.linux-kvm.org/.

[46] *lxc Linux Containers*, http://lxc.sourceforge.net/.

[47] *Microsoft Hyper-V*, http://www.microsoft.com/en-us/server-cloud/hyper-v-server/.

[48] *WSGI Middleware Libraries*, http://wsgi.readthedocs.org/.

[49] *Amazon Machine Images*, https://aws.amazon.com/amis.

[50] *Open Networking Foundation*, https://www.opennetworking.org/.

[51] *Eucalyptus*, http://www.eucalyptus.com/.

[52] *AWS and Eucalyptus Agreement*, http://www.eucalyptus.com/aws-compatibility.

[53] *VMware vSphere Hypervisor*, http://www.vmware.com/products/vsphere-hypervisor/.

[54] *VMware vSphere ESX*, http://www.vmware.com/products/vsphere/esxi-and-esx/.

[55] *OpenNebula*, http://opennebula.org/.

[56] *XML-RPC*, http://en.wikipedia.org/wiki/XML-RPC.

[57] *xCloud*, http://xcloud.cs.cornell.edu/.

[58] 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.

[59] D. Wheeler, *Another level of indirection*.

[60] *LibCloud*, http://libcloud.apache.org/.

[61] *Delta Cloud*, http://deltacloud.apache.org/.

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

[63] 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.