2

VIRTUALIZATION IN THE CLOUD

Lisandro Zambenedetti Granville, Rafael Pereira Esteves, and Juliano Araujo Wickboldt

Institute of Informatics, Federal University of Rio Grande do Sul, Porto Alegre, Brazil

2.1 THE NEED FOR VIRTUALIZATION MANAGEMENT IN THE CLOUD

Cloud infrastructures are aggregates of computing, storage, and networking resources deployed along centralized or distributed data centers devoted to support companies' applications or, in the case of services being offered through the Internet, to support cloud customers' applications. Companies such as Google, Amazon, Facebook, and Microsoft rely on cloud infrastructures to support various services such as Web search, e-mail, social networking, and e-commerce. By leasing physical infrastructure to external customers, cloud providers encourage the development of novel services and, at the same time, generate revenue to cover deployment and operation costs of clouds. Cloud resource sharing is then critical for the cloud computing model.

To allow multiple customers, cloud providers rely on virtualization technologies to build *virtual infrastructures* (VIs) comprising logical instances of physical resources (e.g., servers, network, and storage). The provisioning of VIs must consider requirements of both cloud providers *and* customers. While the main objective of cloud providers is

Cloud Services, Networking, and Management, First Edition.
Edited by Nelson L. S. da Fonseca and Raouf Boutaba.

to generate revenue by accommodating a large number of VIs, customers, in their turn, have specific needs, such as storage capacity, high availability, processing power (usually represented by the number of leased virtual machines or VMs), guaranteed bandwidth among VMs, and load balancing. Inefficiencies in the provisioning process can lead to negative consequences for cloud providers, including customer defection, financial penalties when service-level agreements (SLAs) are not satisfied, and low utilization of the physical infrastructure. In summary, management of physical and VIs is vital to enabling proper cloud resource sharing.

Current cloud provisioning systems allow customers to select among different resource configurations (e.g., CPU, memory, and disk) to build a VI. Customers are the main responsible for choosing the resources that will better fit their application's needs. The cloud provider, in turn, either (a) allocates resources for the VI on physical data centers, or (b) rejects the allocation if there are not enough resources to satisfy the customers' requirements. Cloud providers run allocation algorithms to find the best way to map VIs onto the physical substrate according to well-defined objectives, such as minimizing the allocation cost, reducing energy consumption, or maximizing residual capacity of the infrastructure. Mapping virtual to physical resources is commonly referred to as *embedding* and has been extensively studied in the context of network virtualization [1–3].

Embedding is an example of a network virtualization aspect that needs to be properly managed. Choosing the appropriate embedding algorithm, and deciding when it should be triggered (e.g., when new VI requests arrive or when on-the-fly optimizations of the physical substrate are needed) is a management activity that needs to be consciously performed by the cloud management operator or team. Other virtualization management aspects encompass operations such as: monitoring, to detect abusing applications/customers; configuration, to tune VI and physical substrate; and discovery, to identify collaborating VIs that would be better placed closer to one another. In addition to management operations, virtualization management requires understanding of the diversity of target elements because both VI and physical substrate are quite heterogeneous in regard to the resources they use/offer. That impacts the management operations themselves, since, for example, monitoring and configuring a physical server can be quite different than monitoring and configuring network devices and traffic. Operations and target elements are thus two important dimensions of cloud virtualization management.

In this chapter, we cover the management of virtualization in the cloud. Our observations primarily take the perspective of cloud providers who need to manage their substrate and hosted VIs to guarantee that the services offered to customers are operating properly. Virtualization management is a quite new discipline because virtualization itself, at least as it has been employed these days, is also quite recent. Management is achieved by borrowing techniques from other areas, such as network management and parallel processing. We concentrate our discussion on the two management dimensions mentioned before, i.e., management operations and target elements. Although other dimensions do exist, we will focus on the operations and elements because they are the essential dimensions a cloud manager needs to take into account in the first place.

The remaining of this chapter is organized as follows. In Section 2.2, we review some basic concepts of virtualization in cloud computing environments. In Section 2.3, we describe the main elements of a virtualized cloud environment. In Section 2.4, we list the main virtualization-related operations that need to be supported by a cloud platform. In Section 2.5, we review some of the most important efforts towards the definition of open standard interfaces to support virtualization and interoperability in the cloud. In Section 2.6, we list some of the most important efforts currently targeted to build tools and systems for virtualization and cloud resource management. Finally, in Section 2.7, we list key challenges that can guide future developments in the virtualization management and also mention some ongoing research in the field.

2.2 BASIC CONCEPTS

Clouds can be public, private, or hybrid. Public clouds offer resources to any interested tenant (e.g., Amazon EC2 and Windows Azure). Private clouds usually belong to a single organization and only the members of that organization have access to the resources. Hybrid clouds are combinations of different types of cloud. For example, a private cloud that needs to temporarily extend its capacity can borrow resources from a public cloud, thus forming a hybrid cloud.

Cloud services are organized according to three basic business models. In infrastructure as a service (IaaS), cloud providers offer logical instances of physical resources, such as VMs, virtual storage, and virtual links to interested tenants. In platform as a service (PaaS), tenants can request a computing platform including an operating system and a development environment. The software as a service (SaaS) model offers end-applications (e.g., Google Drive and Dropbox) to customers. Other models, such as network as a service (NaaS) are also possible in the cloud, but are not found so frequently in the literature [4].

Virtualization is the key technology to enable cloud computing. Virtualization abstracts the internal details of physical resources and enables resource sharing. Using virtualization, a physical resource (e.g., server, router, link) can be shared among different users or applications. The core of cloud computing environments is based on virtualized data centers, which are facilities consisting of computing servers, storage, network devices, cooling, and power systems.

Virtualization can be accomplished by different technologies according to the target element. Server virtualization, for example, relies on a layer of software called hypervisor, also known as VM monitor. The hypervisor is the component responsible for actually creating, removing, copying, migrating, and running VMs. Virtual links, in turn, can be created by configuring Ethernet VLANs between the physical nodes hosting the virtual ones. Multiprotocol label switching (MPLS) label switched paths (LSPs) and generic routing encapsulation (GRE) tunnels are other candidates to establish virtual links.

Participants in the cloud comprise two main roles: the cloud provider, also known as infrastructure provider, owns the physical resources that can be leased to one or more tenants, also known as service providers, who build VIs composed of virtual instances of computing, storage, and networking resources. A VI can be also referred to as a cloud slice. After the instantiation of a VI, tenants can deploy a variety of applications that will rely on these virtual resources.

A cloud platform is a software that allows tenants to request and instantiate VIs. Tenants can specify the amount of resources to build their VIs and the specific characteristics of each resource, such as CPU and memory for computing, disk size for storage, and bandwidth capacity for links. The cloud platform then interacts with the underlying virtualization software (hypervisor) to create and configure the VI. In order to facilitate resource management and allow interoperability, cloud platforms offer specific interfaces for applications running in VIs. Such interfaces define operations that can be executed in the cloud platform.

2.3 VIRTUALIZED ELEMENTS

As stated in previous sections, virtualization plays a key role in modern cloud computing environments by improving resource utilization and reducing costs. Typical elements that can be virtualized in a cloud computing environment include computing and storage. Recently, virtualization has been extended also to the networking domain and can overcome limitations of current cloud environments such as poor isolation and increased security risks [5]. In this section, we describe the main elements that can be virtualized in a cloud computing environment.

2.3.1 Computing

The virtualization of computing resources (e.g., CPU and memory) is achieved by server virtualization technologies (e.g., VMWare, Xen, and QEMU) that allow multiple virtual machines (VMs) to be consolidated in a single physical one. The benefits of server virtualization for cloud computing include performance isolation, improved application performance, and enhanced security.

Cloud computing providers deploy their infrastructure in data centers comprising several virtualized servers interconnected by a network. In the IaaS model, VMs are instantiated and allocated to customers (i.e., tenants) on-demand. Server virtualization adds flexibility to the cloud because VMs can be dynamically created, terminated, copied, and migrated to different locations without affecting existing tenants. In addition, the capacity of a VM (i.e., CPU, memory, and disk) can be adjusted to reflect changes in tenants' requirements without hardware changes.

Cloud operators have flexibility to decide where to allocate VMs in the physical servers considering diverse criteria such as cost, energy consumption, and performance. In this regard, several VM allocation schemes have been proposed in the literature that leverage VM flexibility to optimize resource utilization [6–8, 10–12, 20].

2.3.2 Storage

Storage virtualization consists of grouping multiple (possibly heterogeneous) storage devices that are seen as a single virtual storage space. There are two main abstractions to represent storage virtualization in clouds: virtual volumes and virtual data objects. The virtualization of storage devices as virtual volumes is important in this context because it simplifies the task of assigning disks to VMs. Furthermore, many implementations also include the notion of virtual volume pools, which represent different sources of available virtualizable storage spaces to allocate virtual volumes from (e.g., separate local physical volumes or a remote Network File System or NFS). On the other hand, cloud storage of virtual data objects enables scalable and redundant creation and retrieval of data objects directly into/from the cloud. This abstraction is also often accompanied by the concept of containers, which in general serve to create a hierarchical structure of data objects similar to files and folders on any operating system.

Storage virtualization for both volumes and data objects is of utmost importance to enable the elasticity property of cloud computing. For example, VMs can have their disk space adjusted dynamically to support changes in cloud application requirements. Such adjustment is too complex and dynamic to be performed manually and, by virtualizing storage, cloud providers offer a uniform view to their users and reduce the need for manual provisioning. Also, with storage virtualization, cloud users do not need to know exactly where their data are stored. The details of which disks and partitions contain which objects or volumes are transparent to users, which also facilitates storage management for cloud providers.

2.3.3 Networking

Cloud infrastructures rely on local and wide area networks to connect the physical resources (i.e., servers, switches, and routers) of their data centers. Such networks are still based on the current IP architecture that has a number of problems. These problems are mainly related to the lack of isolation, which can allow that one VI or application interferes with another, resulting in poor performance or, even worse, in security problems. Another issue is the limited support for innovation, which hinders the development of new architectures that could suit better cloud applications.

To overcome the limitations of current network architectures, virtualization can also be extend to the cloud networks. ISP network virtualization has been a hot topic of investigation in recent years [13, 14] and is now being considered in other contexts, such as cloud networking. Similar to virtualized ISP networks, in virtualized cloud networks, multiple virtual networks (VNs) share a physical network and run isolated protocol stacks. A VN is part of a VI that comprises VN nodes (i.e., switches and routers) and virtual links.

The advantages of virtualization of cloud networks include network performance isolation, improved security, and the possibility to introduce new protocols and addressing schemes without disrupting production services. Figure 2.1 shows how virtualization can be tackled in cloud network infrastructures. In the substrate layer, physical nodes and links from different network administrative domains serve as a substrate for the deployment of VNs. Physical nodes, at the core of the physical networks, represent network devices (e.g., switches and routers) that internally run virtual (or logical) routers instantiated to serve VNs' routing necessities.

In the virtualization layer, virtual nodes and links are created on the top of the substrate and combined to build VNs. A VN can use resources from different sources,

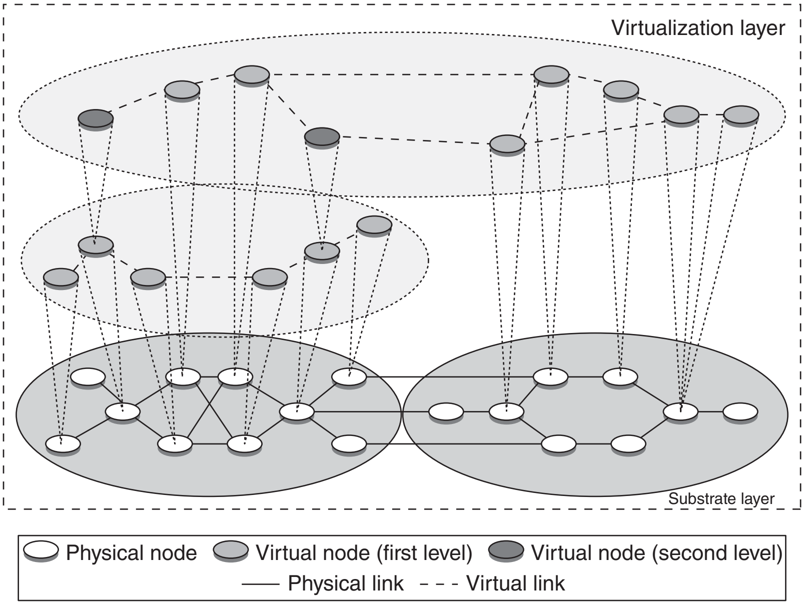

Figure 2.1. Virtualized cloud network infrastructure.

including resources from other VNs, which in this case results in a hierarchy of VNs. VNs can also be entirely placed into a single physical node (e.g., physical end-host). In this case, since virtual links are not running on top of any physical counterpart, isolation, and performance guarantees should be offered, for example, through memory isolation and scheduling mechanisms. In another setup, VNs can spread across different adjacent physical infrastructures (i.e., different administrative domains). In this case, network operators, at the substrate layer, must cooperate to provide a consistent view of the underlying infrastructure used by networks from the virtualization layer.

2.3.4 Management

The management of cloud infrastructures plays a key role to allow cloud providers to efficiently use the resources and increase revenue. At the virtual level, each VI can operate its own management protocols, resource allocation schemes, and monitoring tools. For example, one tenant can use Simple Network Management Protocol (SNMP) to manage his/her VIs, while other can use NETCONF or Web services.

Different resource allocation schemes tailored for specific cloud applications define how virtual resources are mapped in the data center. Adaptive, application-driven resource provisioning allows multiple tenants and a large diversity of applications to efficiently share a cloud infrastructure.

Monitoring is another management aspect that can be virtualized. Once a new VI is created, a set of monitoring tools need to be configured [15] in order to start monitoring the computing resources that form the VI. The set of monitoring tool configurations and the corresponding monitored metrics is referred to as a monitoring slice. Every VI is coupled with a monitoring slice [16]. To monitor the computing resources that form VIs, cloud operators generally use in their monitoring slices tools with native support to cloud platforms.

2.4 VIRTUALIZATION OPERATIONS

Virtualization operations are structured according to the components described in the previous section: *computing*, *storage*, *networking*, and *management*. A non-exhaustive list of the main virtualization operations derived from existing cloud platforms [9, 17–19, 21] is described next.

- Computing (Virtual Machines)
  - *Create/Remove*: defines/undefines the internal representation of a VM with its specified characteristics (*e.g.*, CPU, memory, and guest image).
  - *Deploy/Undeploy*: defines/undefines a VM within the hypervisor of a node of the cloud infrastructure, including the transfer/removal of the image file.
  - *Start/Stop/Suspend/Resume*: basic operations to handle the state of the guest operating system.
  - *Migrate*: undefines a VM in one node and defines it on another. The destination node needs to be specified.
  - *Modify*: modifies the attributes of a VM.
  - *Snapshot*: creates a snapshot of a VM.
  - *Restore*: restores a VM from a snapshot.
  - *List*: lists currently deployed VMs.
- Computing (Images)
  - *Create/Remove*: defines/undefines a guest operating system image at the main repository of the platform, including the transfer/removal of the file.
- Storage (Virtual Volumes)
  - *Create/Remove*: allocates/deletes chunks of storage on nodes.
  - *Attach Volume to VM*: attaches a volume file to a given VM.
- Storage (Virtual Volume Pools)
  - *Create/Remove*: defines/undefines a pool for storing virtual volumes (typically a local or remote/NFS directory).
  - *Add Volume*: adds a virtual volume to a volume pool.
- Storage (Virtual Data Objects)
  - *Create/Remove*: allocates/deletes storage for data objects on nodes.
  - *Upload/Download*: transfers the actual data in and out of the cloud environment.
  - *Stream*: sends the content out to the general public, sometimes even adopting massive scale distribution employing concepts of Content Delivery Networks (CDNs).

- Storage (Virtual Containers)
    - *Create/Remove*: defines/undefines a container for storing data objects.
    - *Add Data Object*: adds a virtual data object to a container.
- Networking (Virtual Links)
    - *Create/Remove*: defines/undefines the internal representation of a virtual link that connects point-to-point virtual interfaces of two virtual devices (i.e., VMs or virtual routers).
    - *Establish/Disable*: establishes/disables the virtual link within the network, enabling/disabling traffic to flow between the connected devices.
    - *Configure*: configures additional parameters of a virtual link (e.g., bandwidth).
- Networking (Virtual Routers)
    - *Create/Remove*: defines/undefines the internal representation of a virtual router that has multiple virtual ports to interconnect multiple virtual interfaces of virtual devices.
    - *Deploy/Undeploy*: deploys/undeploys the virtual router into a node of the infrastructure.
    - *Add/Edit/Remove Routes*: defines/modifies/undefines routes for a virtual router.
- Management (Virtual Devices)
    - *Monitor/Unmonitor*: deploys/undeploys the monitoring infrastructure required to monitor a given virtual device.
    - *Get Monitoring Information*: fetches monitoring information within the monitoring system for a given virtual device.
- Management (Events)
    - *Create/Remove*: defines/undefines the internal representation of an event that belongs to a specific slice or operates in global scope.
    - *Deploy/Undeploy*: deploys/undeploys the event on the monitoring infrastructure to be triggered on demand.
- Management (Physical)
    - *Discover Resources*: this is actually a collection of operations to discover nodes and network topology available on the infrastructure. This collection also retrieves information about resource allocation on these physical elements.
    - *Get Monitoring Information*: fetches monitoring information within the monitoring system for a given physical device (e.g., node or switch).

2.5 INTERFACES FOR VIRTUALIZATION MANAGEMENT

Today, there are many heterogeneous cloud platforms that support the provisioning of virtualized infrastructures under a plethora of different specifications and technologies. Each cloud provider chooses the platform that suits it better or designs its own platforms to provide differentiated services to its tenants. The problem with this heterogeneity is that it hinders interoperability and causes vendor lock-in for tenants. In order to allow the remote management of virtual elements, many platforms already offer specific interfaces (e.g., Amazon EC2/S3, Elastic Hosts, Flexiscale, Rackspace Cloud Servers, and VMware vSphere) to communicate with external applications.

To cope with this variety of technologies and support the development of platform-agnostic cloud applications, some proposals use basically two different approaches: (1) employing proxy-style APIs in order to communicate with multiple providers using a set of technology-specific adapters and (2) creating standardized generic interfaces to be implemented by cloud platforms. The first approach has a drawback of introducing an additional layer of software in cloud systems, which results in overhead and increased latency. Nevertheless, there are libraries and tools that are widely employed, such as Apache Deltacloud and Libcloud, which are further discussed in the next section. The second approach, on the other hand, represents a more elegant solution to the problem by proposing some sort of *lingua franca* to communicate among cloud systems. The problem with standardization is to make participants to agree onto the same standard [22]. Ideally, a standardized interface should be open and extensible to allow widespread adoption by cloud management platforms and application developers. In this section, we review some of the most important efforts towards the definition of open standard interfaces to support virtualization and interoperability in the cloud.

2.5.1 Open Cloud Computing Interface

The Open Cloud Computing Interface (OCCI) [23] introduces a set of open, community-driven specifications to deal with cloud service resource management [24]. OCCI is supported by the Open Grid Forum and was originally conceived to create a remote management API for IaaS platforms allowing interoperability for common tasks, such as deployment, scaling, and monitoring virtual resources. Besides the definition of an open application programming interface (API), this specification also introduces a RESTful Protocol for exchanging management information and actions. The current release of OCCI is not anymore focused only in IaaS and includes other cloud business models, such as PaaS and SaaS.

The current version of the specification[1] is designed to be modular and extensible, thus it is split in three complementary documents. The OCCI Core document (GFD.183) describes the formal definition of the OCCI Core Model. This document also describes how the core model can be interacted with renderings (including associated behaviors) and expanded through extensions. The second document is OCCI Infrastructure (GFD.184), which contains the definition of the OCCI infrastructure extension for the IaaS domain. This document also defines additional resource types, their attributes, and actions that each resource type can perform. The third document, OCCI HTTP Rendering (GFD.185), defines means of interacting with the OCCI Core Model through the RESTful OCCI API. Moreover, this document defines how the OCCI Core Model can be communicated and serialized over HTTP.

[1] As of the ending of 2013, the current version of OCCI is v1.1 (release date April 7, 2011)

The OCCI Infrastructure document describes the modeling of virtual resources in IaaS as three basic element types: (1) compute that are information processing resources, (2) storage that are intended to handle information recording, and (3) network representing L2 networking elements (e.g., virtual switches). Also, there is an abstraction for creation of links between resources. Links can be of two types: i.e., Network Interface or Storage Link, depending on the type of resource they connect. It is also possible to use this specification to define Infrastructure Templates, which are predefined virtual resource specifications (e.g., small, medium, and large VM configurations). Moreover, the OCCI HTTP Rendering document complements these definitions by specifying management operations, such as creating, retrieving, updating, and deleting virtual resources. The document also details general requirements for the transmission of information over HTTP, such as security and authentication.

OCCI is currently implemented in many popular cloud management platforms, such as OpenStack, OpenNebula, and Eucalyptus. There are also base implementations in programming languages, such as rOCCI in Ruby and jclouds in Java, and automated compliance tests with doyouspeakOCCI. One particular effort aims to improve the intercloud networking standardization by proposing an extension to OCCI, called Open Cloud Networking Interface (OCNI) [25]. There is also a reference implementation of OCNI called pyOCNI, written as a Python framework including JSON serialization for resource representation.

2.5.2 Open Virtualization Format

The Open Virtualization Format (OVF) [26], currently in version 2.0.1, was introduced late in 2008 within the Virtualization Management initiative of the Distributed Management Task Force (DMTF), aiming to provide an open and extensible standard for packaging and distribution of software to be run in VMs. Its main virtue is to allow portability of virtual appliances onto multiple platforms through so-called OVF Packages, which may contain one or more virtual systems. The OVF standard is not tied to any particular hypervisor or processor architecture. Nevertheless, it is easily extensible through the specification of vendor-specific metadata included in OVF Packages.

OVF Packages are a core concept of the OVF specification, which consist of several files placed into one directory describing the structure of the packed virtual systems. An OVF Package includes one OVF Descriptor, which is an XML document containing metadata about the package contents, such as product details, virtual hardware requirements, and licensing. The OVF Package may also include certificates, disk image files, or ISO images to be attached to virtual systems.

Within an OVF Package, an Envelope Element describes all metadata for the VMs included in the package. Among this metadata, a detailed Virtual Hardware Description (based on CIM classes) can specify all types of virtual hardware resources required by a virtual system. This specification can be abstract or incomplete, allowing the virtualization platform to decide how to better satisfy the resource requirements, as long as the required virtual devices are deployed. Moreover, OVF Environment information can be added to define how the guest software and the deployment platform interact.

This environment allows the guest software to access information about the deployment platform, such as the values specified for the properties defined in the OVF Descriptor.

This standard is present in many hypervisor implementations and has shown to be very useful for migrating virtual systems information among many hypervisors or platforms, since it allows precise description of VMs and virtual hardware requirements. However, it is not within the objectives of OVF to provide detailed specification for complete VIs (i.e., detailing interconnections, communication requirements, and network elements).

2.5.3 Cloud Infrastructure Management Interface

The Cloud Infrastructure Management Interface (CIMI) [27] standard is another DMTF proposal within the context of the Cloud Management initiative. This standard defines a model and a protocol for managing interactions between cloud IaaS providers and tenants. CIMI's main objective is to provide tenants with access to basic management operations on IaaS resources (VMs, storage, and networking), facilitating portability between different cloud implementations that support this standard. CIMI also specifies a RESTful protocol over HTTP using both JSON or XML formats to represent information and transmit management operations.

The model defined in CIMI includes basic types of virtualized resources, where Machine Resources are used to represent VMs, Volume Resources for storage, and Network Resources for VN devices and ports. Besides, CIMI also defines a Cloud Entry Point type of resource, which represents a catalog of virtual resources that can be queried by a tenant. A System Resource in this standard gathers one or more Network, Volume, or Machine Resources, and can be operated as a single resource. Finally, a Monitoring Resource is also defined to track progress of operations, metering, and monitoring of other virtual resources.

The protocol relies on basic HTTP operations (i.e., PUT, GET, DELETE, HEAD, and POST) and uses either JSON or XML to transmit the message body. To manipulate virtual resources, there are four basic create, read, update, and delete (CRUD) operations. It is also possible to extend the protocol by creating or customizing operations to manipulate the state of each particular resource. Moreover, the CIMI specification can also be integrated with OVF, in which case VMs represented as OVF Packages can be used to create Machine Resources or System Resources.

Today, implementations of the CIMI standard are not so commonly found as OCCI or OVF are. One specific implementation that is worth noting is found within the Apache Deltacloud[2] project, which exposes a CIMI REST API to communicate with external applications supporting manipulation of Machine and Volume Resources abstractions.

2.5.4 Cloud Data Management Interface

The Cloud Data Management Interface (CDMI) [28] is a standard specifically targeted to define an interface to access cloud storage and to manage data objects. CDMI is

[2] http://deltacloud.apache.org/cimi-rest.html

comparable to Amazon's S3 [29], with the fundamental difference that it is conceived by the Storage Networking Industry Association (SNIA) to be an open standard targeted for future ANSI and ISO certification. This standard also includes a RESTful API running over HTTP to allow accessing capabilities of cloud storage systems, allocating and managing storage containers and data objects, handling users and group access rights, among other operations.

The CDMI standard defines a JSON serializable interface to manage data stored in clouds based on several abstractions. Data objects are fundamental storage components analogous to files within a file system, which include metadata and value (contents). Container objects are intended to represent grouping of data, analogous to directories in regular file systems; this abstraction links together zero or more Data objects. Domain objects represent the concept of administrative ownership of data stored within cloud systems. This abstraction is very useful to facilitate billing, to restrict management operations to groups of objects, and to represent hierarchies of ownership. Queue objects provide first-in, first-out access to store or retrieve data from the cloud system. Queuing provides a simple mechanism for controlling concurrency when reading and writing Data objects in a reliable way. To facilitate interoperability, this standard also includes mechanisms for exporting data to other network storage platforms, such as iSCSI, NFS, and WebDAV.

Regarding implementations, CDMI is also not so commonly deployed in most popular cloud management platforms. SNIA's Cloud Storage Technical Working Group (TWG) provides a Reference Implementation for the standard, which is currently a working draft and provides support only for version 1.0 of the specification. Some independent projects, such as CDMI add-on for OpenStack Swift and the CDMI-Serve in Python, have implemented basic support for the CDMI standard but do not present much recent activity.

Besides all the aforementioned efforts to create new standardized interfaces for virtual resource management in cloud environments, other approaches, protocols, and methods have been studied and may be of interest in particular situations [30]. Moreover, many organizations, such as OASIS, ETSI, ITU, NIST, and ISO, are currently engaged with their cloud and virtualization related working groups on developing standards and recommendations. We recommend the interested reader to look at DMTF's maintained wiki page Cloud-Standards.org[3] to keep track of future standardization initiatives.

2.6 TOOLS AND SYSTEMS

In this section, we list some of the most important efforts currently targeted to build tools and systems for virtualization and cloud resource management. Initially, we describe open source cloud management platforms, which are in fact complete solutions to deploy and operate private, public, or hybrid clouds. Afterwards, we discuss some tools and

[3] http://cloud-standards.org/

libraries to perform specific operations for virtual resource management and cloud integration.

2.6.1 Open Source Cloud Management Platforms

2.6.1.1 Eucalyptus. Eucalyptus started as a research project in the Computer Science Department at the University of California, Santa Barbara, in 2007, within a project called Virtual Grid Application Development Software Project (VGrADS) funded by the National Science Foundation. This is one of the first open source initiatives to build cloud management platforms that allow users to deploy their own private clouds [18]. Currently, Eucalyptus is in version 3.4 and comprises full integration with Amazon Web Services (AWS)—including EC2, S3, Elastic Block Store (EBS), Identity and Access Management (IAM), Auto Scaling, Elastic Load Balancing (ELB), and CloudWatch—enabling both private and hybrid cloud deployments.

Eucalyptus architecture is based on four high-level components: (1) Node Controller executes at hosts and is responsible for controlling the execution of VM instances; (2) Cluster Controller works as a front-end at the cluster-level (i.e., Availability Zone) managing VM execution and scheduling on Node Controllers, controls cluster-level SLAs, and also manages VNs; (3) Storage Controller exists both at cluster-level and at cloud-level (Walrus) and implements a put/get SaaS solution based on Amazon's S3 interface, providing a mechanism for storing and accessing VM images and user data; and (4) Cloud Controller is the entry-point into the cloud for users and administrators, it implements an EC2-compatible interface and coordinates other components to perform high-level tasks, such as authentication, accounting, reporting, and quota management.

For networking, Eucalyptus offers four operating modes: (1) Managed, in which the platform manages layers 2 and 3 VM isolation, employing a built-in DHCP service. This mode requires a switch to forward a configurable range of VLAN-tagged packets; (2) Managed (no VLAN), in which only layer 3 VM isolation is possible; (3) Static, where there is no VM isolation, employs a built-in DHCP service for static IP assignment; and (4) System, where there is also no VM isolation and, in this case, no automatic address handling since Eucalyptus will rely on an existing external DHCP service. In version 4.0, released in April 2014, Eucalyptus has introduced new functionality for networking support through a new Edge Networking Mode.

The main technical characteristics of the Eucalyptus platform as the following:

- *Programming Language*: Written mostly in C and Java
- *Compatibility/Interoperability*: Fully integrated with AWS
- *Supported Hypervisors*: vSphere, ESXi, KVM, any AWS-compatible clouds
- *Identity Management*: Role-Based Access Control mechanisms with Microsoft Active Directory or LDAP systems
- *Resource Usage Control*: resource quotas for users and groups
- *Networking*: Basic support with four operating modes
- *Monitoring*: CloudWatch

- *Version/Release*: 3.4.1 (Released on December 16, 2013)
- *License*: GPL v3.0

2.6.1.2 OpenNebula. In its early days OpenNebula was a research project at the Universidad Complutense de Madrid. The first version of the platform was released under an open source license in 2008 within the European Union's Seventh Framework Programme (FP7) project called RESERVOIR—Resources and Services Virtualization without Barriers (2008–2011). Nowadays, OpenNebula (version 4.4 Retina released December 3, 2013) is a feature-rich platform used mostly for the deployment of private clouds, but is also capable of interfacing with other systems to work as hybrid or public cloud environment.

OpenNebula is conceptually organized in a three-layered architecture [31]. At the top, the Tools layer comprises higher level functions, such as cloud-level VM scheduling, providing CLI and GUI access for both users and administrators, managing and supporting multi-tier services, elasticity and admission control, and exposing interfaces to external clouds through AWS and OCCI. At the Core layer, vital functions are performed, such as accounting, authorization, and authentication, as well as resource management for computing, storage, networking, and VM images. Also at this layer, the platform implements resource monitoring by retrieving information available from hypervisors to gather updated status of VMs and manages federations, enabling access to remote cloud infrastructures, which can be either partner infrastructures governed by a similar platform or public cloud providers. At the bottom, the Drivers layer implements infrastructure and cloud drivers to provide an abstraction to communicate with the underlying devices or to enable access to remote cloud providers.

OpenNebula allows administrators to set up multiple zones and create federated VIs considering different federation paradigms (e.g., cloud aggregation, bursting, or brokering), in which case each zone operates their network configurations independently. From the user's viewpoint, setting up a network in the OpenNebula platform is restricted to the creation of a DHCP IP range that will be automatically configured in each VM. The administrator can change the way VMs connect to the physical ports of the host machine using one of many options, that is, VLAN 802.1Q to allow isolation, EBtables and Open vSwitch to permit implementation of traffic filtering, and VMware VLANs, which isolate VMs running over VMware hypervisor. It is also possible to deploy Virtual Routers from OpenNebula's Marketplace to work as an actual router, DHCP, or DNS server.

The main technical characteristics of the OpenNebula platform as the following:

- *Programming Language*: C++ (Integration APIs in Ruby, JAVA, and Python)
- *Compatibility/Interoperability*: AWS, OCCI, and XML-RPC API
- *Supported Hypervisors*: KVM, Xen, and VMWare
- *Identity Management*: Sunstone, EC2, OCCI, SSH, x509 certificates, and LDAP
- *Resource Usage Control*: resource quotas for users and groups
- *Networking*: IP/DHPC ranges customizable by users, many options for administrator require manual configuration

- *Monitoring*: Internal, gathers information from hypervisors
- *Version/Release*: 3.4.1 (Released on December 3, 2013)
- *License*: Apache v2.0

2.6.1.3 OpenStack. OpenStack started as a joint project between Rackspace Hosting and NASA around mid 2010, aiming to provide a cloud-software solution to run over commodity hardware [19]. Right after the first official release (beginning of 2011), OpenStack was quickly adopted and packed within many Linux distributions, such as Ubuntu, Debian, and Red Hat. Today, it is the cloud management platform with the most active community counting on more than 13,000 registered people from over 130 countries. OpenStack is currently developed in nine parallel core projects (plus four incubated) all coordinated by the OpenStack Foundation, which is embodied by 9,500 individuals and 850 different organizations.

The OpenStack architecture consists of a myriad of interconnected components, each one developed under a separate project, to deliver a complete cloud infrastructure management solution. Initially, only two components were present, Compute (Nova) and Object Storage (Swift), which respectively provide functionality for handling VMs and a scalable redundant object storage system. Adopting an incremental approach, incubated/community projects were gradually included in the core architecture, such as Dashboard (Horizon) to provide administration GUI access, Identity Service (Keystone) to support a central directory of users mapped to services, and Image Service (Glance) to allow discovery, registration, and delivery of disk and server images. The current release of OpenStack (Havana) includes advanced network configuration with Neutron, persistent block-level storage with Cinder, a single point of contact for billing systems through Ceilometer, and a service to orchestrate multiple composite cloud applications via Heat.

As for networking, a community project called Quantum started in April 2011 and was targeted to further develop the networking support of OpenStack by employing VN overlays in a Connectivity as a Service perspective. From release Folsom on, Quantum was added as a core project and renamed Neutron. Currently, this component lets administrators to employ from basic networking configuration of IP addresses, allowing both dedicated static address assignment and DHCP, to complex configuration with software-defined networking (SDN) technology like OpenFlow. Moreover, Neutron allows the addition of plug-ins to introduce more complex functionality to the platform, such as quality of service, intrusion detection systems, load balancing, firewalls, and virtual private networks.

- *Programming Language*: Python
- *Compatibility/Interoperability*: Nova and Swift are feature-wise compatible to EC2 and S3 (applications need to be adapted though), OCCI support (under development)
- *Supported Hypervisors*: QEMU/KVM over libvirt (fully supported), VMware and XenAPI (partially supported), many others at nonstable development stages
- *Identity Management*: Local database, EC2/S3, RBAC, token-based, SSL, x509 or PKI certificates, and LDAP

- *Resource Usage Control*: configurable quotas per user (tenant) defined by each project
- *Networking*: several options via Neutron component, extensible with plug-ins
- *Monitoring*: simple customizable dashboard relies on information provided by other components
- *Version/Release*: Havana (Released on October 17, 2013)
- *License*: Apache v2.0

2.6.1.4 CloudStack. CloudStack started as a project from a startup company called VMOps in 2008, later renamed Cloud.com, and was first released as open source in mid 2010. After Cloud.com was acquired by Citrix, CloudStack was relicensed to Apache 2.0 and incubated by the Apache Software Foundation in April 2012. Ever since, the project has developed a powerful cloud platform to orchestrate resources in highly distributed environments for both private and public cloud deployments [21].

CloudStack deployments are organized into two basic building blocks, a Management Server and a Cloud Infrastructure. The Management Server is a central point of configuration for the cloud (these servers might be clustered for reliability reasons). It provides a Web user interface and API access, manages the assignment of guest VMs to hosts, allocates public and private IP addresses to particular accounts, manages images, among other tasks. A Cloud infrastructure comprises distributed Zones (typically, data centers) hierarchically organized into Pods, Clusters, Hosts, Primary and Secondary Storage. A CloudStack Cloud Infrastructure may also optionally include Regions (perhaps geographically distributed), to aggregate multiple Zones, and each Region is controlled by a different set of Management Servers, turning the platform into a highly distributed and reliable system. Moreover, a separate Python tool called CloudMonkey is available to provide CLI and shell environments for interacting with CloudStack-based clouds.

CloudStack offers two types of networking configurations: (1) Basic, which is an AWS-style networking providing a single network where guest isolation can be achieved through layer 3 means, such as security groups and (2) Advanced, where more sophisticated network topologies can be created. CloudStack also offers a variety of NaaS features, such as creation of VPNs, firewalls, and load balancers. Moreover, this tool provides the ability to create a Virtual Private Cloud, which is a private, isolated part of CloudStack that can have its own VN topology. VMs in this VN can have any private addresses since they are completely isolated from others.

- *Programming Language*: Mostly Java
- *Compatibility/Interoperability*: CloudStack REST API (XML or JSON)
- *Supported Hypervisors*: XenServer/XCP, KVM, and/or VMware ESXi with vSphere
- *Identity Management*: Internal or LDAP
- *Resource Usage Control*: Usage server separately installed provides records for billing, resource limits per project
- *Networking*: two operating modes, several networking as a service options in advanced configurations

- *Monitoring*: some performance indicators available through the API are displayed to users and administrators
- *Version/Release*: 4.2.0 (Released on October 1, 2013)
- *License*: Apache v2.0

2.6.2 Specific Tools and Libraries

The following describes some tools and libraries mainly designed to deal with the diversity of technologies involved in cloud virtualization. Unlike cloud platforms, these tools do not intend to offer a complete solution for cloud providers. Nevertheless, they play a key role in integration and allow applications to be written in a more generic manner in terms of virtual resource management.

2.6.2.1 Libcloud. Libcloud is a client Python library for interacting with the most popular cloud management platforms [32]. This library originally started being developed within Cloudkick (extinct cloud monitoring software project, now part of Rackspace) and today is an independent free software project licensed under the Apache License 2.0. The main idea behind Libcloud is to create a programming environment to facilitate developers on the task of building products that can be ported across a wide range of cloud environments. Therefore, much of the library is about providing a long list of drivers to communicate with different cloud platforms. Currently, Libcloud supports more than 26 different providers, including Amazon's AWS, OpenStack, OpenNebula, and Eucalyptus, just to mention a few.

Moreover, this library also provides a unified Python API, offering a set of common operations to be mapped to the appropriate calls to the remote cloud system. These operations are divided into four abstractions: (1) Compute, which enables operations for handling VMs (e.g., list/create/reboot/destroy VMs) and its extension Block Storage to manage volumes attached to VMs (e.g., create/destroy volumes, attach volume to VM); (2) Load Balancer, which includes operations for the management of load balancers as a service (e.g., create/list members, attach/detach member or compute node) and is available in some providers; (3) Object Storage, which offers operations for creating an environment for handling data objects in a cloud (list/create/delete containers or objects, upload/download/stream object) and its extension for CDNs to assist providers that support these operations (e.g., enable CDN container or object, get CDN container or object URL); and (4) Domain Name System (DNS), which allows management operations for DNS as a service (e.g., list zones or records, create/update zone or record) in providers that support it, such as Rackspace Cloud DNS.

2.6.2.2 Deltacloud. Deltacloud follows a very similar philosophy as compared to Libcloud. It is also an Apache Software Foundation project—left incubation in October 2011 and is now a top-level project—and is similarly targeted to provide an intermediary layer to let applications communicate with several different cloud management platforms. Nevertheless, instead of providing a programming environment through a specific programming language, Deltacloud enables management of resources in different clouds by the use of one of three supported RESTful APIs [33]: (1) Deltacloud classic, (2) DMTF CIMI, (3) Amazon's EC2.

Deltacloud implements drivers for more than 20 different providers and offers several operations divided into two main abstractions: (1) Compute Driver, which includes operations for managing VMs, such as create/start/stop/reboot/destroy VM instances, list all/get details about hardware profiles, realms, images, and VM instances; and (2) Storage Driver, providing operations similar to Amazon S3 to manage data objects stored in clouds, such as create/update/delete buckets (analogous to folders), create/update/delete blobs (analogous to data files), and read/write blobs data and attributes.

2.6.2.3 Libvirt. Libvirt is a toolkit for interacting with multiple virtualization providers/hypervisors to manage virtual compute, storage, and networking resources. It is a free collection of software available under GNU LGPL and is not particularly targeted to cloud systems. Nevertheless, Libvirt has shown to be very useful to handle low level virtualization operations and is actually used under the hood by cloud platforms like OpenStack to interface with some hypervisors. Libvirt supports several hypervisors (e.g., KVM/QEMU, Xen, VirtualBox, and VMware), creation of VNs (e.g., bridging or NAT), and storage on IDE, SCSI, and USB disks and LVM, iSCSI, and NFS file systems. It also provides remote management using TLS encryption, x509 certificates, and authentication through Kerberos or SASL.

Libvirt provides a C/C++ API with bindings to several other languages, such as Python, Java, PHP, Ruby, and C#. This API includes operations for managing virtual resources as well as retrieving information and capabilities from physical hosts and hypervisors. Virtual resource management operations are divided into three abstractions: (1) Domains, which are common VM-related operations, such as create, start, stop, and migrate; (2) Storage, for managing block storage volumes or pools; and (3) Network, which includes operations such as, creating bridges, connecting VMs to these bridges, enabling NAT and DHCP. Note that network operations are all performed within the scope of a single physical host, that is, it is not possible to connect two VMs in separate hosts to the same bridged network, for example.

2.7 CHALLENGES

Because virtualization management in the cloud is still in its infant days, important challenges are in place. In this section, we list key challenges that can guide future developments in the virtualization management area. We also mention some ongoing research in the field.

2.7.1 Scalability

Although the benefits of virtualization enables the cloud model, from the management perspective, virtualization impacts the scalability of management solutions. The transition from the traditional management of physical infrastructures to virtual one is not smooth in terms of scale because few physical devices can host a much larger number of virtual device, each one requiring management actions. The number of management elements immediately explodes because such number not only duplicates but is proportional to the number of virtual devices each physical one supports. Traditional management applications have not been conceived to support a so drastic increase in the number of elements, and as a consequence, such solutions do not scale.

Novel management approaches need to be considered, or traditional approaches need to be adapted (if possible) to the cloud context and scale. The problem is also exacerbated because the managed environments (i.e., clouds) are much more dynamic, having new elements created very quickly, while older elements can be destroyed frequently too. Virtual servers can go up and down (even forever) quite fast, which is unusual for traditional management solutions. Adaptation is then required not only because of the new scales of cloud environments but also because they are much more dynamic than traditional IT infrastructures. Very distributed solutions have to be investigated, like the usage of peer-to-peer for management [34]. Autonomic management also becomes an alternative, in order to reduce human intervention as much as possible [35].

2.7.2 Monitoring

Monitoring is a permanent challenging task in the cloud because of the large number of resources in production cloud data centers. Centralized monitoring approaches suffer from low scalability and resilience. Cooperative monitoring [36] and gossiping [37] aim to overcome these limitations by enabling distributed and robust monitoring solutions for large scale environments. The goal is to minimize the negative impact of management traffic on the performance of the cloud. At the same time, finding a scalable solution for aggregating relevant monitoring information without hurting accuracy is a challenge that needs to be tackled by monitoring tools designed specifically for cloud data centers.

Usually, monitoring generates more management data than other activities. With virtualization in the cloud, and the aforementioned scalability issues, the overwhelming amount of monitoring data can hinder proper observation of the cloud environment. As such, monitoring considering big data techniques may be a possible path to follow. Compressing of data structures [38], for example, can be convenient to find a reasonable balance between amount of data and analysis precision.

2.7.3 Management Views

Since cloud computing creates an environment with different actors with particular management roles, such different actors need different management views. Operators of a cloud infrastructure need to have a broader, possibly complete view of the physical infrastructure, but should be prevented of accessing management information that are solely related to a tenant application, because of privacy issues. Cloud tenants also need to have access to management information related to their rented VI, but must also be isolated from accessing management information of both other tenants and physical infrastructure operator.

Different management views are already supported in traditional solutions, but in the case of cloud environments, trust relationships between cloud provider and tenants become more apparent. Since the management software runs in the cloud itself, tenants accessing their management view need to trust the cloud provider assuming that sensitive information is not available to the cloud operator. Tenants can also employ their own management system operating at his/her local IT infrastructure. In this case, management interfaces and protocols that connect the tenant management solution and remote managed virtual elements need to be present.

2.7.4 Energy Efficiency

Efficient energy management aims to reduce the operational cost of cloud infrastructures. A challenge in optimal energy consumption is to design energy-proportional data center architectures, where energy consumption is determined by server and network utilization [39, 40]. ElasticTree [39], for example, attempts to achieve energy proportionality by dynamically powering off switches and links. In this respect, cloud network virtualization can further contribute to reduce power consumption through network consolidation (e.g., through VN migration [41]).

Minimizing energy consumption, however, usually comes with the price of performance degradation. Energy efficiency and performance is often conflicting, representing a tradeoff. Thus, designing energy-proportional data center architectures factoring in cloud virtualization, and finding good balance between energy consumption and performance are interesting research questions.

2.7.5 Fault Management

Detection and handling of failures are requirements of any cloud, especially because in cloud environments failures of a single physical resource can potentially affect multiple customers' virtual resources. Because failures also tend to propagate, the damage caused by a faulty cloud physical device impacts much more severely the cloud business. In additional, the lack of faults in physical devices is not always a synonym that there is not faulty virtual devices. As such, the traditional fault management needs to be expanded to consider faulty virtual devices too.

Most existing architectures rely on reactive failure handing approaches. One drawback is the potentially long response time, which can negatively impact application performance. Ideally, fault management should be implemented in a proactive manner, where the management system predicts the occurrence of failures and acts before they occur. In practice, proactive fault management is often ensured by means of redundancy, for example, provisioning backup paths. As such, offering high reliability without incurring excessive costs or energy consumption is a problem requiring further exploration.

2.7.6 Security

Security issues are challenging in the context of cloud virtualization because of the complex interactions between tenants and cloud providers. Although the virtualization of both servers and networks can improve security (e.g., limiting information leakage, avoiding the existence of side channels, and minimizing performance interference attacks), today's virtualization technologies are still in their infancy in terms of security. In particular, various vulnerabilities in server virtualization technologies, such as VMWare [42], Xen [43], and Microsoft Virtual PC and Virtual Server [44] have been revealed in the literature. Similar vulnerabilities are likely to occur in programmable network components too. Thus, not only network virtualization techniques give no guaranteed protection from existing attacks and threats to physical and VNs, but also lead to new security vulnerabilities. For example, an attack against a VM may lead to an attack against a hypervisor of a physical server hosting the VM, subsequent attacks against other VMs hosted on that server, and eventually, all VNs sharing that server [45]. This raises the issue of designing secure virtualization architectures immune to these security vulnerabilities.

In addition to mitigating security vulnerabilities related to virtualization technologies, there is a need to provide monitoring and auditing infrastructures, in order to detect malicious activities from both tenants and cloud providers. It is known that data center network traffic exhibits different characteristics than the traffic of traditional data networks [46]. Thus, appropriate mechanisms may be required to detect network anomalies. On the other hand, auditability in cloud virtualization should be mutual between tenants and cloud providers to prevent malicious behaviors from either party. However, there is often an overhead associated with such infrastructures, especially in large-scale clouds. In Ref. [47], the authors showed that it is a challenge to audit Web services in cloud environments without deteriorating application performance. Much work remains to be done on designing scalable and efficient mechanisms for monitoring and auditing cloud virtualization.

2.7.7 Cloud Federations

The federation of virtualized infrastructures from multiple cloud providers enables access to larger scale infrastructures. This is already happening with virtualized network testbeds, allowing researchers to conduct realistic network experiments at large scale, which would not have been possible otherwise. ProtoGENI [48] is an example of federation that allows cooperation among multiple organizations. However, guaranteeing predictable performance for participating entities through SLA enforcement has not been properly addressed by current solutions and remains an open issue.

Cloud federations cannot be considered a wide reality in the cloud marketplace. Competition possibly prevents cloud providers to cooperate among one another in federated environments, but the lack of proper technologies devoted to materialize federations of clouds certainly does not improve the current situation either. As in other areas, solutions to federate different resources already exist, but an integrated, global solution to support federating heterogeneous resources between heterogeneous cloud providers also needs further investigation.

2.7.8 Standard Management Protocols and Information Models

The VR-MIB module [49] described a set of SNMP management variables for the management of physical routers with virtualization support. However, it did not progress in the IETF standardization track. More recently, the VMM-MIB module [50] is progressing, but it is limited to manage virtualization of servers (network devices are not explicitly considered); VMM-MIB is also devoted mainly for monitoring, and configuration is weakly supported. In general, the situation of SNMP-based management solutions for cloud environments are still weak.

Other existing management protocols are considered. NETCONF [51], for example, would be more appropriate for configuration aspects, while NetFlow/IPFIX [52] could be expanded for virtual router monitoring. The WS-Management [53] suite, in turn, is more appropriate for server management. A myriad of proprietary solutions is also present in the market, but the large diversity of management interfaces and protocols forces cloud operators to deal with too many different technologies. Although a protocol that fits every need is unlikely to exist or be largely accepted/adopted, there is a clear lack in this area today, which represents an interesting opportunity for research and standardization.

REFERENCES

1. M. Chowdhury, M.R. Rahman, and R. Boutaba. ViNEYard: Virtual Network Embedding Algorithms with Coordinated Node and Link Mapping. *IEEE/ACM Transactions on Networking*, 20(1):206–219, February. 2012.
2. M. Yu, Y. Yi, J. Rexford, and M. Chiang. Rethinking Virtual Network Embedding: Substrate Support for Path Splitting and Migration. *ACM Computer Communication Review*, 38(2):17–29, April 2008.
3. X. Cheng, S. Su, Z. Zhang, K. Shuang, F. Yang, Y. Luo, and J. Wang. Virtual Network Embedding Through Topology Awareness and Optimization. *Computer Networks*, 56(6):1797–1813, 2012.
4. A. Lenk, M. Klems, J. Nimis, S. Tai, and T. Sandholm. What's Inside the Cloud? An Architectural Map of the Cloud Landscape. In *Proceedings of the 2009 ICSE Workshop on Software Engineering Challenges of Cloud Computing (CLOUD '09)*, pages 23–31, Washington, DC, 2009. IEEE Computer Society.
5. M. F. Bari, R. Boutaba, R. Esteves, L. Granville, M. Podlesny, M. Rabbani, Q. Zhang, and F. Zhani. Data Center Network Virtualization: A Survey. *IEEE Communications Surveys and Tutorials*, 15(2):909–928, 2012.
6. Q. Zhu and G. Agrawal. Resource Provisioning with Budget Constraints for Adaptive Applications in Cloud Environments. In *Proceedings HPDC 2010*, Chicago, IL, 2010.
7. M. E. Frincu and C. Craciun. Multi-objective Meta-heuristics for Scheduling Applications with High Availability Requirements and Cost Constraints in Multi-Cloud Environments. In *Proceedings of the Fourth IEEE International Conference on Utility and Cloud Computing (UCC)*, pages 267–274, Victoria, NSW, December 2011.

8. J. Rao, X. Bu, K. Wang, and C.-Z. Xu. Self-adaptive Provisioning of Virtualized Resources in Cloud Computing. In *Proceedings SIGMETRICS 2011*, 2011.
9. OpenNebula. The Open Source Solution for Data Center Virtualization, 2008. http://www.opennebula.org. Accessed on November 2013.
10. J. Rao, Y. Wei, J. Gong, and C.-Z. Xu. DynaQoS: Model-free Self-Tuning Fuzzy Control of Virtualized Resources for QoS Provisioning. In *19th IEEE International Workshop on Quality or Service (IWQoS)*, pages 1–9, San Jose, CA, June 2011.
11. J. Rao, X. Bu, C.-Z. Xu, and K. Wang. A Distributed Self-learning Approach for Elastic Provisioning of Virtualized Cloud Resources. In *Proceedings IEEE MASCOTS 2011*, pages 45–54, Singapore, July 2011.
12. J. Z. W. Li, M. Woodside, J. Chinneck, and M. Litoiu. CloudOpt: Multi-goal Optimization of Application Deployments across a Cloud. In *Proceedings CNSM 2011*, pages 1–9, October 2011.
13. A. Khan, A. Zugenmaier, D. Jurca, and W. Kellerer. Network Virtualization: A Hypervisor for the Internet? *IEEE Communications Magazine*, 50(1):136–143, January 2012.
14. N. Chowdhury and R. Boutaba. Network Virtualization: State of the Art and Research Challenges. *IEEE Communications Magazine*, 47(7):20–26, July 2009.
15. J. Montes, A. Sánchez, B. Memishi, M. S. Pérez, and G. Antoniu. GMonE: A Complete Approach to Cloud Monitoring. *Future Generation Computer Systems*, 29(8):2026–2040, 2013.
16. M. Carvalho, R. Esteves, G. Rodrigues, L. Z. Granville, and L. M. R. Tarouco. A Cloud Monitoring Framework for Self-Configured Monitoring Slices Based on Multiple Tools. In *9th International Conference on Network and Service Management 2013 (CNSM 2013)*, pages 180–184, Zürich, Switzerland, October 2013.
17. Amazon. Amazon elastic compute cloud (Amazon EC2), 2013. http://aws.amazon.com/ec2/. Accessed on May. 2013.
18. Eucalyptus. The Open Source Cloud Platform, 2009. http://open.eucalyptus.com. Accessed on November 2013.
19. Rackspace Cloud Computing. OpenStack Cloud Software, 2010. http://openstack.org. Accessed on December 2013.
20. S. Islam, J. Keung, K. Lee, and A. Liu. Empirical Prediction Models for Adaptive Resource Provisioning in the Cloud. *Future Generation Comp. Syst.*, 28(1):155–162, January 2012.
21. Apache Software Foundation. Apache CloudStack: Open Source Cloud Computing, 2012. http://cloudstack.apache.org. Accessed on December 2013.
22. S. Ortiz. The Problem with Cloud-Computing Standardization. *IEEE Computer*, 44(7):13–16, 2011.
23. Open Grid Forum. Open Cloud Computing Interface, 2012. http://occi-wg.org/. Accessed on September 2012.
24. A. Edmonds, T. Metsch, A. Papaspyrou, and A. Richardson. Toward an Open Cloud Standard. *Internet Computing, IEEE*, 16(4):15–25, 2012.
25. H. Medhioub, B. Msekni, and D. Zeghlache. OCNI—Open Cloud Networking Interface. In *22nd International Conference on Computer Communications and Networks (ICCCN)*, pages 1–8, Nassau, Bahamas, 2013.
26. Distributed Management Task Force (DMTF). Open Virtualization Format (OVF) Specification—Version 2.0.1, Ago 2013. http://dmtf.org//standards/cloud. Accessed on December 2013.

27. Distributed Management Task Force (DMTF). Cloud Infrastructure Management Interface (CIMI)—Version 1.0.0, 2013. http://dmff.org/standards/cloud. Accessed on May 2013.

28. Storage Networking Industry Association (SNIA). Cloud Data Management Interface (CDMI)—version 1.0.2, June 2012. http://www.snia.org/cdmi. Accessed on December 2013.

29. Storage Networking Industry Association (SNIA). S3 and CDMI: A CDMI Guide for S3 Programmers—version 1.0, May 2013. http://www.snia.org/cdmi. Accessed on December 2013.

30. R. P. Esteves, L. Z. Granville, and R. Boutaba. On the Management of Virtual Networks. *IEEE Communications Magazine*, 51(7):80–88, 2013.

31. R. Moreno-Vozmediano, R. S. Montero, and I. M. Llorente. IaaS Cloud Architecture: From Virtualized Datacenters to Federated cloud infrastructures. *IEEE Computer*, 45(12):65–72, 2012.

32. Apache Software Foundation. Apache Libcloud a Unified Interface to the Cloud, 2012. http: //libcloud.apache.org/. Accessed on December 2013.

33. Apache Software Foundation. Apache DeltaCloud an API that Abstracts the Differences between Clouds, 2011. http://deltacloud.apache.org/. Accessed on December 2013.

34. L. Z. Granville, D. M. da Rosa, A. Panisson, C. Melchiors, M. J. B. Almeida, and L. M. Rockenbach Tarouco. Managing Computer Networks Using Peer-to-Peer Technologies. *Communications Magazine, IEEE*, 43(10):62–68, 2005.

35. C. C. Marquezan and L. Z. Granville. On the Investigation of the Joint Use of Self-* Properties and Peer-to-Peer for Network Management. In *2011 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, pages 976–981, 2011.

36. K. Xu and F. Wang. Cooperative Monitoring for Internet Data Centers. In *IEEE International Performence, Computing and Communications Conference (IPCC)*, pages 111–118, Austin, TX, December 2008.

37. F. Wuhib, M. Dam, R. Stadler, and A. Clemm. Robust Monitoring of Network-Wide Aggregates through Gossiping. *IEEE Transactions on Network and Service Management*, 6(2): 95–109, 2009.

38. L. Quan, J. Heidemann, and Y. Pradkin. Trinocular: Understanding Internet Reliability Through Adaptive Probing. In *Proceedings of the ACM SIGCOMM 2013 Conference on SIGCOMM (SIGCOMM '13)*, pages 255–266, Hong Kong, China, 2013. ACM, New York.

39. B. Heller, S. Seetharaman, P. Mahadevan, Y. Yiakoumis, P. Sharma, S. Banerjee, and N. McKeown. ElasticTree: Saving Energy in Data Center Networks. In *Proceedings USENIX NSDI*, April 2010.

40. H. Yuan, C. C. J. Kuo, and I. Ahmad. Energy Efficiency in Data Centers and Cloud-based Multimedia Services: An Overview and Future Directions. In *Proceedings IGCC*, Chicago, IL, August 2010.

41. Y. Wang, E. Keller, B. Biskeborn, J. van der Merwe, and J. Rexford. Virtual Routers on the Move: Live Router Migration as a Network-Management Primitive. *ACM Computer Communication Review*, 38:231–242, August 2008.

42. VMware. VMware Shared Folder Bug Lets Local Users on the Guest OS Gain Elevated Privileges on the Host OS. VMWare vulnerability. http://securitytracker.com/alerts/2008/Feb/1019493.html, 2008.

43. Xen. Xen Multiple Vulnerabilities. Xen vulnerability. http://secunia.com/advisories/26986, 2007.

44. Microsoft. Vulnerability in Virtual PC and Virtual Server Could Allow Elevation of Privilege. Virtual PC Vulnerability. http://technet.microsoft.com/en-us/security/bulletin/MS07-049, 2007.

45. J. Szefer, E. Keller, R. B. Lee, and J. Rexford. Eliminating the Hypervisor Attack Surface for a More Secure Cloud. In *Proceedings of the 18th ACM conference on Computer and Communications Security (CSS)*, pages 401–412, Chicago, IL, October 2011.

46. T. Benson, A. Anand, A. Akella, and M. Zhang. Understanding Data Center Traffic Characteristics. *ACM SIGCOMM Computer Communication Review*, 40(1):92–99, 2010.

47. A. Chukavkin and G. Peterson. Logging in the Age of Web Services. *IEEE Security and Privacy*, 7(3):82–85, June 2009.

48. ProtoGENI. ProtoGENI, Dec 2013. Available at: http://www.protogeni.net/trac/protogeni (Dec. 2013).

49. E. Stelzer, S. Hancock, B. Schliesser, and J. Laria. Virtual Router Management Information Base Using SMIv2. *Internet-Draft draft-ietf-ppvpn-vr-mib-05 (obsolete)*, June, 2013. http://tools.ieff.org//html/draft-ietf-ppvpn-vr.mib-05. Accessed on December 2014.

50. H. Asai, M. MacFaden, J. Schoenwaelder, Y. Sekiya, K. Shima, T. Tsou, C. Zhou, and H. Esaki. Management Information Base for Virtual Machines Controlled by a Hypervisor. *Internet-Draft draft-asai-vmm-mib-05 (work in progress)*, October 13, 2013.

51. R. Enns. RFC 4741: NETCONF Configuration Protocol, December 2006. http://tools.ietf.org//html/rfc4741. Accessed on December 10, 2014.

52. B. Claise, B. Trammell, and P. Aitken. RFC 7011: Specification of the IPFIX Protocol for the Exchange of Flow Information, September 2013. http://tools.ieff.org//html/rfctoll. Accessed on December 2014.

53. Distributed Management Task Force (DMTF). Web Services for Management (WS-Management) Specification. DMTF, Ago 2012.