2014 16th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing

OpenStack and CloudStack: Open Source Solutions for Building Public and Private Clouds

Amine Barkat, Alysson Diniz dos Santos, Thi Thao Nguyen Ho
Politecnico di Milano
Dipartimento di Elettronica Informazione e Bioingegneria
Piazza L. da Vinci, 32 - 20133 Milano, Italy
Email: amine.barkat@polimi.it, alysson@virtual.ufc.br, thithao.ho@polimi.it

***Abstract*—Cloud computing is continuously growing as a prominent technology for enterprises. While several giant public cloud providers, such as Amazon, Microsoft, IBM, Google are competing to extend their market, there is still a large number of organizations asking higher level of privacy and control over cloud solutions. Therefore, the need to have private cloud solutions is obvious. To overcome this need there are several on-going open source software frameworks for building public and private clouds. Among them, OpenStack and CloudStack are growing at fast pace and gaining more attention. An analysis on these software stacks is necessary in order to choose the most suitable solution that matches an enterprise's requirements. This paper main contribution is an in depth study and comparison of the cloud properties of these two open source frameworks, providing useful information on open source cloud solutions that are not available elsewhere.**

***Keywords*—OpenStack, CloudStack, cloud computing, open source, public cloud, private cloud, IaaS**

I. Introduction

Cloud computing is a new computing model that brings together all disciplines, technologies and business models to deliver Information Technology (IT) resources on-demand. This is a new trend that well fits in an environment where resources are provisioned dynamically and exposed as a service on the Internet [1]. In this context, open source cloud technologies such as OpenStack, CloudStack, OpenNebula, Eucalyptus, OpenShift, and Cloud Foundry have gained significant momentum in the last few years. For a researcher and practitioner, they present a unique opportunity to analyze, contribute, and innovate in new services using these technologies [2].

Cloud computing consists basically of three levels of offerings [2]:

1) Infrastructure as a Service (IaaS), where the equipments are provided in the form of virtual machines. The client maintains the applications, runtime, integration SOA (Service Oriented Architecture), databases, server software while the supplier maintains the virtualization layer, server, storage, and network hardware. Among the main actors of IaaS, we find Amazon EC2, Rackspace, GoGrid.
2) Platform as a Service (PaaS), user can develop his own applications using the services provided. The client maintains only his applications, while the supplier maintains all the cloud stack from hardware up to application containers. We have among the key players: Google Apps Engine, Windows Azure.
3) Software as a Service (SaaS), entire applications are available remotely. Among the providers we have GoogleApps, Salesforce, and Facebook.

At the three levels of cloud offerings are shown in Fig. 1, the lowest level is the focus of this paper. The IaaS manages computing resources (computing, storage, network), and the virtualization layer that allows the access to the physical resources (e.g, processor, memory and other devices) providing resource isolation and security.

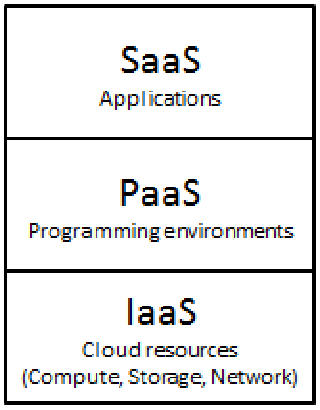

Fig. 1: Offering levels in cloud computing

In the need of having IaaS solutions that can give more privacy and control over the system, open source clouds are born to build private clouds. Eventually, these open source solutions can be used to set up public clouds, private clouds or a mix of them, i.e, hybrid clouds. With the emergence of different open-source cloud solutions, the decision to choose the most suitable one that meets users needs becomes a difficult task, because every platform has its specific characteristics [3]. Moreover, since hybrid clouds are the most widely used nowadays, surveying open source middlewares that simplify cluster management and the creation of private clouds enabled for cloud bursting is an important matter. In this sense, several papers begin to analyze and compare each platform, trying to establish a starting point to look when deciding which open source cloud technology should be adopted. [4]–[6] give essentially an overview of Eucalyptus, Nimbus and OpenNebula solutions, highlighting their different characteristics. [2], [7]–[10] conduct surveys, classify and compare different open source solutions. Concerning specifically OpenStack and

CloudStack, [11] presents briefly both solutions and does some general features comparison, but enter in more details only for OpenStack. Notwithstanding, several updates implemented recently to evolve and improve cloud softwares make this kind of study quickly outdated.

With this in mind, in this paper we present the general features of the newest versions of OpenStack and CloudStack and compare their general features and important properties, trying to provide useful information for users that need to choose an open source cloud software.

The paper is organized as follows: Section II describes the architecture of OpenStack and its important properties, Section III presents CloudStack, Section IV performs comparisons between the two platforms. Conclusions are finally drawn in Section V.

II. OpenStack

OpenStack is a cloud software that offers capability to control large pools of compute, storage and networking resources. It also empowers users providing on-demand resources [12]. Starting from 2010, OpenStack was developed by Rackspace Hosting and NASA [13] aimed to provide open source cloud solution to build public or private clouds. The mission of OpenStack is to enable any organization to create and offer cloud computing services running on standard hardwares. Provisioned as open source solution, OpenStack is built keeping these core principles in mind: (1) Open source: all code will be released under the Apache 2.0 license allowing the community to use it freely; (2) Open design: every 6 months the development community will hold a design summit to gather requirements and write specifications for the upcoming releases; (3) Open development: maintains a publicly available source code repository through the entire development process; (4) Open community: produces a healthy, vibrant development and user community through an open and transparent process.

A. General Architecture

As in any cloud platform, the infrastructure underneath OpenStack is standard hardware, which can contain any pieces of physical devices such as servers, disks or network devices. In order to provide cloud services, OpenStack develops virtualization layers giving the abstract view of physical infrastructure to end users. These virtualization layers are built up by various components as described in Fig. 2. The OpenStack architecture consists of three main components: Compute (Nova), Network (Quantum) and Storage (Swift). Beside these three pillars, OpenStack has been developing many other services, each of those designed to work together to provide a complete IaaS solution. The integration of these services is facilitated through public application programming interfaces (APIs) offered by each service [13].

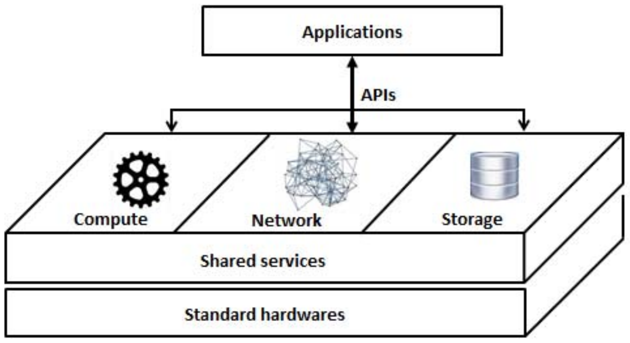

Fig. 2: OpenStack general architecture

In the following, the detailed description of each component is provided.

1) Compute (Nova): Compute is the heart of OpenStack (codename is Nova and it is written in Python), which is the computing fabric controller responsible for managing large networks of virtual machines (VMs), and eventually to properly schedule VMs among available physical machines (PMs) [13]. Compute is a distributed application that consists of six components: Nova-api, Message Queue, Nova-Compute, Nova-Network, Nova-Volume and Nova-Scheduler as shown in Fig. 3. Nova supports the complete life-cycles of an instance in the cloud, starting from the request to initialize a VM until its termination. It follows this architecture:

- Nova-api: accepts and responds to end user compute API calls. Beside providing its own OpenStack Compute API, Nova-api is compatible with Amazon EC2 API, offering the potential to integrate with Amazon cloud services. It has another special Admin API reserved for privileged users to perform administrative actions. The orchestration activities such as running an instance, or enforcing the policies such as quota checks are initiated by this component.
- Nova-compute: is primarily a worker daemon that creates and terminates VM instances via hypervisor APIs. In order to do so, it accepts actions from the queue and performs system commands to fulfill them, while updating the database state accordingly. OpenStack supports several standard hypervisors (listed in Section IV) while keeping the openness that allows to interface other hypervisors through its standard library.
- Nova-volume: manages the creation, attaching and detaching of persistent volumes to compute instances. There are two types of block devices supported for an VM instance: (1) Ephemeral Storage: is associated to a single unique instance. Its life-cycle exists together with the instance life-cycle, which means when the instance is terminated, data on this storage will also be deleted; (2) Volume Storage: is persistent and independent from any particular instance. This storage can be used as external disk device where the data stored on it still remain even when the instance is terminated.
- Nova-network: is a worker daemon that handles network-related tasks. It accepts and performs networking tasks from the queue to manipulate the network such as setting up bridging interfaces or changing iptable rules.
- Nova-schedule: handles the scheduling of VMs among PMs. It takes a virtual machine instance request from

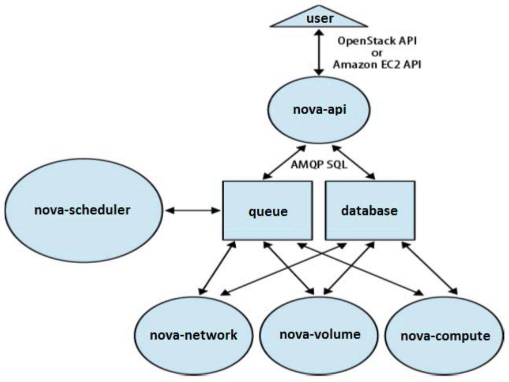

Fig. 3: Compute component architecture [13]

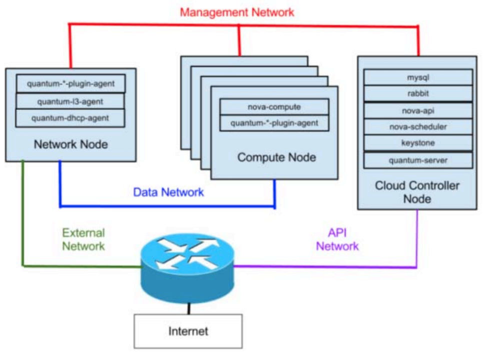

Fig. 4: Networking component architecture [13]

the queue and determines the physical host it should place the instance on. While the scheduling algorithms can be defined by users, Nova-schedule supports by default three algorithms: (1) Simple: attempts to find least loaded host, (2) Chance: chooses random available host from service table, (3) Zone: picks random host from within an available zone. By allowing users to define their own scheduling algorithms, this component is important for building fault tolerant and load-balanced system.

- Queue: provides a central hub for passing messages between daemons. This is usually implemented withRabbitMQ today, but can support any AMPQ message queue.
- Database: stores most of the build-time and run-time state of a cloud infrastructure. For example, it provides information of the instances that are available for use or in use, networks availability or storage information. Theoretically, OpenStack Nova can support any SQL-based database but the most widely used databases currently are sqlite3, MySQL and PostgreSQL.

Given this architecture, all its components follow a shared-nothing and messaging-based policy. Shared-nothing means that each component or each group of components can be installed on any server, in a distributed manner; while the messaging-based policy ensures the communication among all components such as volume, network and scheduler is performed via Queue Server.

2) Network (Quantum): Network is the key of cloud computing for several reasons: (1) Offered resources and services must be accessible; (2) Address binding between different services is essential to support multi-tier applications; (3) Automatic network configuration capability is important, especially in scenarios where auto-scaling installations evolves. The OpenStack Networking component gives operators the ability to leverage different network technologies to power their cloud networking through a rich set of APIs, multiple networking models (e.g, flat or private network) and flexible plug-in architecture. Especially, the plug-in architecture - with the plug-in agent - enables, not only capability of using various network technologies, but also the ability to handle user workloads. It means, at network level, that developers can implement their own load balancing algorithms and plug it in the platform to achieve better workload control. The architecture of Network component is shown in Fig. 4

The Network architecture consists of four distinct physical data center networks:

- Management network: used for internal communication between OpenStack components. The IP addresses on this network should be reachable only within the data center.
- Data network: used for VM data communication within the cloud deployment. The IP addressing requirements of this network depend on the OpenStack Networking plug-in in use.
- External network: used to provide VMs with Internet access in the deployment scenarios. The IP addresses on this network should be reachable by anyone on the Internet.
- API network: exposes all OpenStack APIs, including the OpenStack Networking API, to tenants. The IP addresses on this network should be reachable by anyone on the Internet.

3) Storage: The Storage component, one of three main pillars of OpenStack architecture, is used to manage storage resources. OpenStack has support for both Object Storage and Block Storage, with many deployment options for each, depending on the use case.

Object Storage (codename Swift) is a scalable object storage system. It provides a fully distributed, API-accessible storage platform that can be integrated directly into applications or used for backup, archiving, and data retention [13]. In Object Storage, data are written to multiple hardware devices, with the OpenStack software responsible for ensuring data replication and integrity across clusters. Object storage clusters are scaled horizontally while adding new nodes. If a node fails, OpenStack replicates its content from other active nodes. Because OpenStack uses software logic to ensure data replication and distribution, inexpensive commodity hard drives and servers can be used instead of expensive equipments. Therefore, Object Storage is ideal for cost effective, scale-out storage [13].

Block Storage (codename Cinder), is the storage system that allows block devices to be exposed and connected to compute instances for expanded storage, better performance and integration with enterprise storage platforms, such as NetApp, Nexenta and SolidFire [13]. By managing the storage resources in blocks, Block Storage is appropriate for performance sensitive scenarios such as database storage, expandable file systems, or providing a server with access to raw block level storage.

4) User interface - Dashboard: The OpenStack dashboard provides to administrators and users a graphical interface to control their compute, storage and networking resources. Through the dashboard, administrators can also manage users and set limits on resources access for each user.

5) Shared Services: OpenStack Shared services are a set of several services that span across three pillars of compute, storage and networking, making it easy to perform cloud management operations. These services include [13]:

- Identity Service (code-named Keystone): is the security service to protect resources access and usage. This service provides a central directory management, mapping users to OpenStack accessible services. It acts as a common authentication system across the cloud operating system. It supports multiple forms of authentication including standard username and password credentials, token-based systems and AWS-style logins.
- Image Service (code-named Glance): is the repository for virtual disk and server images used by the VMs. In OpenStack, user can copy or snapshot a server image and immediately store it away. Stored images can be used as a template to get new servers up and running quickly and consistently.
- Telemetry Service: aggregates resources usage and performance data of the services deployed in OpenStack cloud. This powerful capability provides visibility into the usage of the cloud infrastructure and allows cloud operators to view metrics globally or individually.
- Orchestration Service: is a template-driven engine that allows application developers to describe and automate the deployment of the cloud infrastructure as well as detailed post-deployment activities of infrastructure, services and applications.
- Database Service: allows users to quickly and easily utilize the features of a relational database. Cloud users and database administrators can provision and manage multiple database instances as needed.

B. Properties

Provisioned as IaaS, OpenStack is built following an open philosophy: avoid technology lock-ins by not requiring specific technologies and providing user freedom to choose the best slot that matches its needs [13]. In this section, we will analyze some important properties of OpenStack.

- Live migration: is the process of moving a running VM from one PM to another, while the VM is still powered-on. It is important to remember that memory, network connectivity and storage of the migrated VM are also transferred to the destination PM. This capability provides efficient online system maintenance, reconfiguration, load balancing and fault tolerance. OpenStack supports two types of live migration: (i) Shared storage based live migration, and (ii) Block live migration. The former supports live migration scenarios where the source and destination hypervisors have access to the shared storage, while the latter does not require shared storage.
- Load balancing: is the capability that allows to dynamically control the workloads among VMs or physical servers in order to achieve better performance. OpenStack supports load balancing at different scales. First of all, the supporting feature of live migration has enabled system administrators to distribute application workloads among physical servers by means of adjusting VM placement. Moreover, it is possible to control application workloads at VM level, service provided by OpenStack Network layer, controlled by Network component. This component, with a flexible plug-in architecture allows the development of runtime custom algorithms to distribute workloads among VMs. Indeed, OpenStack has an on-going project called Load Balancing as a Service (LBaaS) that is aimed to provide load balancing service to end users. This service has monitoring feature to determine whether the VMs are available to handle user requests and take routing decisions accordingly. Several routing policies are supported such as round robin (i.e, rotates requests evenly between multiple instances), source IP (i.e, requests from a unique source IP address are consistently directed to the same instance) and least connections (i.e, allocate requests to the instance with the least number of active connections).
- Fault tolerance: Within the flexible architecture of OpenStack, fault tolerance can be handled at different levels. These levels depend on the way the IaaS system is configured and deployed. At the VMs level, in order to prevent failures, users can develop scheduling algorithms (besides the three already supported algorithms by OpenStack) for placing the VMs that best fits to his use cases. Some scheduling algorithms have been designed at the present time, such as: group scheduling (i.e, VMs that provide the same functionalities are grouped and placed to separate PMs) and rescheduling (i.e, rescheduling of VMs from failed host to surviving hosts using live-migration). At storage or database level, fault tolerance is achieved by using replication and synchronization to ensure that a failure occurred at one device will not break the whole system.
- Availability: this property seeks to minimize system down time and data loss. In OpenStack, high availability can be achieved through different setups depending on types of services, i.e, stateless or stateful services. Stateless services can provide answer to a request without requiring further information of other

services or historical data. OpenStack stateless services include nova-api, nova-scheduler, etc. For these services, high availability is achieved by providing redundant instances and load balance them. In the opposite, stateful services are ones that requires other information to answer a request, which makes them difficult to obtain high availability. These services, e.g, database or storage, can be highly available by using replication but at the same time the system has to maintain the synchronization between the main version and replicated versions in order to keep the system consistent [14].

- Security: OpenStack has a separated service (Identity service) which provides a central authentication management across the cloud operating system and users. The possibility to set up VPNs and firewalls is also available.
- Compatibility: OpenStack is highly compatible with Amazon EC2 and Amazon S3 and thus client applications written for Amazon Web Services can be used with OpenStack with minimal porting effort [13]. In terms of hypervisors, OpenStack supports multiple hypervisors, e.g, Xen, KVM, HyperV, VMWare, etc. Other hypervisors with existing standard drivers can also be interfaced with OpenStack through standard library, e.g, libvirt library.

III. CLOUDSTACK

CloudStack [15] is an open source software platform, written in Java, designed for development and management of cloud Infrastructure as a Service. It aggregates computing resources for building private, public or hybrid clouds. CloudStack is a turnkey technology that brings together the ”Stack” of features requested by companies and users, like data centers orchestration, management and administration of users and NaaS (Network as a Service).

The start of CloudStack was with Cloud.com in 2008. In May 2010, it was open source under GNU General Public License. Citrix bought CloudStack in July 2011, then in April 2012, Citrix donated CloudStack to Apache Software Foundation (ASF) where it was relicensed under Apache 2.0 and accepted as an incubation project. Since March 2013, CloudStack became a Top Level Project of Apache. Many companies are basing on CloudStack for building and managing their cloud infrastructures. Among these companies, there are: Nokia, Orange, Apple, Disney and many others.

A. *CloudStack Architecture*

In this section, we break down the logical architecture of CloudStack, shown in Fig. 5. In CloudStack, physical resources are organized and managed in a hierarchical structure. The lowest level contains computational devices such as host and primary storage. Hosts are attached together and access shared storage to form a Cluster. The next level consists of clusters which are combined by a layer 2 switch to form a Pod. Go up to higher level, Pods are grouped together with Secondary storage by layer 3 switch to form a Zone. At the highest level, zones are grouped to create a Region. All these resources are managed by a Management Server. In the following, we describe in details each component that forms the whole architecture.

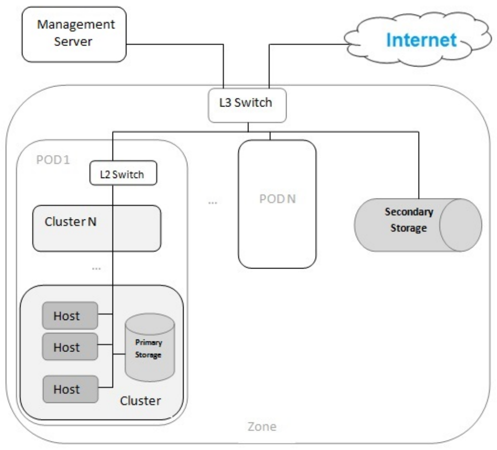

Fig. 5: CloudStack architecture

1) Host: A host represents a physical computational machine that contains local storage. The physical hosts are virtualized by hypervisors. CloudStack supports many hypervisors for VMs management such as Xen, KVM, vSphere, Hyper-V, VMWare, etc. as well as bare metal provisioning. All hosts within a cluster must be homogeneous in terms of the hypervisor, with the possibility of having heterogeneous hypervisors in different clusters.

2) Cluster: Within a cluster, hosts are tied together into the same computational pool with the primary storage and have the same IP subnet. The primary storage can be any kind of storage supported by the hypervisor. One cluster can have more than one primary storage device.

3) Pod: A Pod is a collection of different clusters linked with a layer 2 switch. Hosts in the same Pod are in the same subnet. Pod is not visible to the end user.

4) Zone: The benefit of using Zone is for isolation and redundancy. Often, it corresponds to a data center; although if a data center is large enough, it can have multiple zones. A zone contains Pods that are attached to the secondary storage using a layer 3 switch. Zones are visible to the end user and they can be private or public. Public zones are visible to all users in the cloud while private zones are visible only to users from a particular domain.

5) Region: A region is the largest organizational unit in CloudStack. A region contains multiple zones distributed in geographic locations close to each other.

6) Management Server: A Management Server is used to manage all resources in cloud infrastructure through APIs or UI. One management server can support around 10K hosts and can be deployed on a physical server or a VM. In case we have more than one management server, user interaction to either of them will return the same result. This ensures high availability of CloudStack. A database is required for management servers to be persistent. In order to prevent single point of failure, we can have one primary database and several database replica which always stay synchronized with the primary copy.

7) Storage: In addition to the host local storage, CloudStack manages two main types of storage: primary storage and secondary storage.

- Primary storage: is a storage associated with a cluster or a zone. In the same cluster, we can deploy multiple primary storages. This kind of storage is basically used to run VMs and stores application data. Since this storage interacts directly with applications deployed in VMs, it can be expensive in terms of I/O operations, this is the reason why it is placed physically near to the hosts.
- Secondary storage: is used to store ISO images, templates, snapshots, etc. It supports two different types, NFS and Object Storage.
    - ISO image: is used when user wants to create a VM.
    - Template: is the base operating system image that the user can choose when creating new instance. It may also include additional configuration information such as installed applications.
    - Snapshot: is used as backup for data recovery service. CloudStack supports two types of snapshot: individual snapshot and recurring snapshot. The former is one-time full snapshot, while the latter is either one-time full snapshot or incremental snapshot.

8) Networking: CloudStack supports the use of different physical networking devices (e.g. NetScaler, F5 BIG-IP, Juniper SRX, etc). In CloudStack, users have the ability to choose between two types of networks scenarios: basic and advanced. The basic scenario is for an AWS-style networking. It provides a single network where guest isolation is done through the layer 3 switch. The advanced scenario is more flexible in defining guest networks [15]. For example, the administrator can create multiple networks for use by the guests. CloudStack provides many networking services. Among them we cite:

- Isolation: CloudStack assures the isolation of networks, by allowing the access to the isolated network only by virtual machines of a single account.
- Load Balancing: to balance the traffic in the cloud, the user can create a rule to control and distribute the traffic and apply it to a group of VMs. Within the defined rule, user can choose a load balancing algorithm among the supported ones.
- VPN: for accessing to the VM using CloudStack account, users can create and configure VPNs. Each network has its own virtual router, so VPNs are not shared across different networks. Using VPN tunnels, hosts in different zones are allowed to access each other.
- Firewall: hosts in the same zone can access to each other without passing through the firewall. Users can use external firewalls.

B. Properties

Cloudstack has many properties that motivate companies to use it to manage their infrastructure. The main properties of CloudStack are [15]:

- Live migration: A live migration of running VMs between hosts is allowed in CloudStack through the Dashboard. Depending on the VMs hypervisor, migration conditions can be different. For example, live migration using KVM hypervisor will not support the use of local disk storage, and source and destination hosts have to be in the same cluster; while Xen and VMWare support local disk storage and allow to migrate between different clusters [16].
- Load balancing: a Load balancer is an optional component of CloudStack that allows to distribute the traffic among different management servers [17]. In addition to creating rules and using load balancing algorithms, CloudStack offers the possibility to integrate with external load balancers such as Citrix NetScaler [18].
- Fault tolerance: in CloudStack, fault tolerance is achieved at different scales. In order to prevent failures of management server, the server can be deployed in multi-node configuration. Should one management node fail, other nodes can be used without affecting cloud functioning. Failures at database level are handled by using one or more replication of the database linked to the management server. For host's fail-over, CloudStack recovers the VM instances by taking the images from secondary storage and using application data in primary storage.
- Availability: CloudStack ensures high availability of the system by using multiple management server nodes which may be deployed with load balancers.
- Security: in addition to isolation using different accounts, VPNs and firewalls, CloudStack offers the isolation of traffic using the strategy of security groups which are sets of VMs that filter the traffic on the basis of configuration rules. CloudStack provides a default security group with predefined rules, however they can be modified if necessary.
- Compatibility: CloudStack is built based on a pluggable architecture, one cloud can support different hypervisor implementations including: Hyper-V, KVM, LXC, vSphere, Xenserver , Xen Project and also bare metal provisioning. Moreover, CloudStack is compatible with Amazon API and enables the integration of these two platforms.
- Scalability: CloudStack has the ability to manage thousands of servers distributed in different data centers and different locations thanks to the management server capability (one management server node can manage a big pool of physical resources), and the possibility of using multiple management servers for reducing VMs downtime.
- API extensibility: The CloudStack APIs are very powerful and allows developers to create new command line tools and UIs, and to plug them into CloudStack

architecture. If the developer wants to use new hypervisor, new storage system or new networking service, he just needs to write a new plug-in in Java and integrate it.

IV. Cloud Frameworks Comparison

In this section, we provide a comparison between OpenStack and CloudStack, looking at three different levels:

1) General comparison aimed at providing high level comparison in terms of model, policy and architecture
2) Functional comparison whose goal is to compare supported functionalities, and
3) Property comparison which finally considers cloud properties implemented in these platforms.

It is important to highlight that, since the two platforms are under continuous development, there are not complete documentation and technical reviews provided to fully understanding their technical aspects. Hence, it is difficult to perform a reasonable comparison between the two. However, we attempt to provide a comparison from user perspective, considering the properties that a user needs to look at when choosing a IaaS cloud solution.

A. General Comparison

The general comparison is provided in Table I, considering general aspects: licensing, cloud model compatibility, business model, architecture, etc.

	OpenStack	CloudStack
Open Source License	Apache 2.0	Apache 2.0
Commercial model	Free	Free
Compatibility with	Private, public and hybrid clouds	Private, public and hybrid clouds
Easy installation	Difficult (many choices, not enough automation)	Medium (Few parts to install)
Architecture	Fragmented into many pieces	Monolithic controller
Large organizations adopters	Yahoo, IBM, VMWare, Rackspace, Redhat, Intel, HP, etc.	Nokia, Orange, Apple, Citrix, Huawei, TomTom, Tata, etc.

TABLE I: General comparison between OpenStack and CloudStack

As it can be seen, the two frameworks are generally equal with respect to business model, licensing policy and cloud models. Each of them has been adopted by large organizations. Nevertheless, a big difference is spotted from the architecture viewpoint. While OpenStack is fragmented into modules, CloudStack has a monolithic central controller. This difference is explained by the open philosophy of Open Stack which tries to avoid technology lock-ins and provides high degree of flexibility and extension. As a consequence, OpenStack is characterized by increasing complexity of installation and configuration.

B. Functional Comparison

The functional comparison looks at the offered functionalities or technical aspects of the two solutions, as described in Table II. Supported hypervisors lists all hypervisors that are currently available in these two platforms, even though there are ways to interface with non-supported hypervisors. Administration feature is the interface available to interact with these platforms. Both solutions present Web interfaces (Web UI) and command line interfaces (CLI). Also user management is provided in both platforms. Concerning the monitoring feature, both OpenStack and CloudStack can capture system states and resources usage in order to identify and adapt to occurring problems or perform optimization.

Functionality	OpenStack	CloudStack
Supported hypervisors	Xen, KVM, HyperV, VMWare, LXC, vSphere	Xen, KVM, HyperV, VMWare, LXC, vSphere
Administration	Web UI, CLI	Web UI, CLI
User management	yes	yes
Monitoring	Health monitoring, Resources usage trend monitoring	Health monitoring and usage data monitoring

TABLE II: Functional comparison between OpenStack and CloudStack

Other important functional aspects of open source cloud solutions are related to the activeness of its community. A solution that is always seeing new releases is constantly evolving. An active community, with well documented wiki, good bug reporting and fixing system and active users support are fundamental features for the success of an open source solution.

About the releases, after its insertion on Apache Incubator, CloudStack made its first major release (4.0.0-incubating) on November of 2012. Since then, there were three major releases (4.1 and 4.2 and 4.3) and four minor ones. Meanwhile, OpenStack does, since 2012, two big releases per year. In 2012, there were Essex and Folsom, in 2013 Grizzly and Havana and in 2014 Icehouse and the now under development Juno. In OpenStack, the number of minor releases varies from version to version. From this we can conclude that, even though OpenStack is showing more updates per year, both solutions are regularly updated.

About the activeness of the community, OpenStack remains the largest and most active open source cloud computing project [19]. Nevertheless, CloudStack is growing and has an important participation in the market. Both platforms have well updated wikis and rely on summits to get together their developers. An active forum community for bug reporting and user support can be also found for both solutions. In this point, OpenStack has a bigger and more active community but CloudStack has gained momentum in the last years.

C. Properties Comparison

Properties comparison gives deeper insights into the two platforms, considering some important properties that an IaaS has to provide. The comparison is given in Table III.

Property	OpenStack	CloudStack
Live migration	Yes	Yes
Load balancing	VM level, PM level	VM level, host level
Fault tolerance	VM scheduling, replication	VM scheduling, replication
Availability	Redundancy, load balancer	Redundancy, load balancer
Security	VPNs, firewall, user authentication, others	VPNs, firewall, user management, others
Compatibility	Amazon EC2, Amazon S3	Amazon EC2, Amazon S3

TABLE III: Properties comparison between OpenStack and CloudStack

Live migration is an important property in IaaS solution. Both platforms provide different ways to support live migration. OpenStack offers two types of live migration, while CloudStack has migration conditions depending on the hypervisors.

The two platforms support flexible load balancing mechanisms even though they are made through different strategies. Load balancing at host level is implemented in OpenStack through live migration, which is the same in CloudStack; while load balancing at VM level is achieved in OpenStack through the flexible plug-in architecture of Network component. This capability can be also achieved at network layer in CloudStack with ability of using external load balancers.

Fault tolerance mechanisms are provided in both platforms under the policies to schedule VM placement or services replication.

High availability can be achieved in both platforms by means of using redundant service instances and load balancing to distribute workloads among those instances.

Security is provided by the two platforms through setting up VPNs, firewall and other services. A central user authentication system is available in OpenStack, while CloudStack has user account management. OpenStack has an extensible architecture which allows other systems such as intrusion detection system to be plugged in and deployed; instead, CloudStack offers security group to isolate VMs.

OpenStack and CloudStack are also compatible with Amazon APIs that enable the potential integration with this dominant commercial public cloud solution.

V. Conclusions

OpenStack and CloudStack are two prominent open source platforms that offer tools and services for building private, public, and hybrid clouds. In this paper, we have presented the up-to-date architecture of the two platforms, looking into the details of the provided functionalities and their properties. We conclude the work with the comparison between the two. From the comparison, we realize that it is not easy to say which is the best between them, since they are both stable and used by many large companies, and they continue to evolve. These two solutions support different hypervisors, storages and use different strategies to build a IaaS system. The final choice is for the user who decides to move his business to the cloud, which hypervisors he wants to use, which storage style is more adaptable for his applications, how he wants to deploy his infrastructure, etc. Since the two platforms are on-going developing software, it is necessary to continue updating their development path before selecting an IaaS solution to adopt.

References

[1] L. Schubert, K. G. Jeffery, and B. Neidecker-Lutz, *The Future of Cloud Computing: Opportunities for European Cloud Computing Beyond 2010:–expert Group Report*. European Commission, Information Society and Media, 2010.

[2] O. Sefraoui, M. Aissaoui, and M. Eleuldj, “Comparison of multiple iaas cloud platform solutions,” in *Proceedings of the 7th WSEAS International Conference on Computer Engineering and Applications,(Milan-CEA13), ISBN*, 2012, pp. 978–1.

[3] X. Wen, G. Gu, Q. Li, Y. Gao, and X. Zhang, “Comparison of open-source cloud management platforms: Openstack and opennebula,” in *Fuzzy Systems and Knowledge Discovery (FSKD), 2012 9th International Conference on*. IEEE, 2012, pp. 2457–2461.

[4] B. Sotomayor, R. S. Montero, I. M. Llorente, and I. Foster, “Virtual infrastructure management in private and hybrid clouds,” *Internet Computing, IEEE*, vol. 13, no. 5, pp. 14–22, 2009.

[5] D. Nurmi, R. Wolski, C. Grzegorczyk, G. Obertelli, S. Soman, L. Youseff, and D. Zagorodnov, “The eucalyptus open-source cloud-computing system,” in *Cluster Computing and the Grid, 2009. CCGRID’09. 9th IEEE/ACM International Symposium on*. IEEE, 2009, pp. 124–131.

[6] J. Peng, X. Zhang, Z. Lei, B. Zhang, W. Zhang, and Q. Li, “Comparison of several cloud computing platforms,” in *Information Science and Engineering (ISISE), 2009 Second International Symposium on*. IEEE, 2009, pp. 23–27.

[7] P. T. Endo, G. E. Gonçalves, J. Kelner, and D. Sadok, “A survey on open-source cloud computing solutions,” in *Brazilian Symposium on Computer Networks and Distributed Systems*, 2010.

[8] P. Sempolinski and D. Thain, “A comparison and critique of eucalyptus, opennebula and nimbus,” in *Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference on*. Ieee, 2010, pp. 417–426.

[9] M. Mahjoub, A. Mdhaffar, R. B. Halima, and M. Jmaiel, “A comparative study of the current cloud computing technologies and offers,” in *Network Cloud Computing and Applications (NCCA), 2011 First International Symposium on*. IEEE, 2011, pp. 131–134.

[10] I. Voras, B. Mihaljevic, M. Orlic, M. Pletikosa, M. Zagar, T. Pavic, K. Zimmer, I. Cavrak, V. Paunovic, I. Bosnic *et al.*, “Evaluating open-source cloud computing solutions,” in *MIPRO, 2011 Proceedings of the 34th International Convention*. IEEE, 2011, pp. 209–214.

[11] S. A. Baset, “Open source cloud technologies,” in *Proceedings of the Third ACM Symposium on Cloud Computing*. ACM, 2012, p. 28.

[12] O. C. Software, “Chapter 1. introduction to openstack,” in *http://docs.openstack.org/training-guides/content/module001-ch001-intro-text.html*. OpenStack.

[13] (2014) Openstack history. [Online]. Available: https://www.openstack.org/

[14] OpenStack, “Introduction to openstack high availability,” in *http://docs.openstack.org/high-availability-guide/content/stateless-vs-stateful.html*. OpenStack.

[15] (2014) Cloudstack website. [Online]. Available: https://cloudstack.apache.org/docs/

[16] (2014) Cloudstack live migration. [Online]. Available: http://cloudstack.apache.org/docs/en-US/Apache_CloudStack/4.1.0/html/Admin_Guide/manual-live-migration.html

[17] (2014) Management server load balancing. [Online]. Available: https://cloudstack.apache.org/docs/en-US/Apache_CloudStack/4.0.2/html/Installation_Guide/management-server-lb.html

[18] P. N. Sabharwal, “Integrating netscaler with cloudstack,” in *Apache CloudStack Cloud Computing*. Packt Publishing.

[19] networkworld. (2014, July). [Online]. Available: http://www.networkworld.com/article/2166407/cloud-computing/stack-wars–openstack-v–cloudstack-v–eucalyptus.html