A Comparative Study of OpenStack and CloudStack

Diya Burman, Shuchita Singh and Zarif Masud

***Abstract*— Cloud Computing is a continuously growing technology for enterprise software development and delivery. With increasing demands among a large number of organizations for higher level of privacy and control over cloud solutions the need to have private cloud solutions has gone up. In order to attend to this need, there are several open-source efforts at developing frameworks for building public and private clouds. Among them, two of the most recognizable names these days are OpenStack and CloudStack – both of whom are growing at a fast pace. In order to understand which of these two frameworks suits an enterprise's needs, an extensive analysis of these frameworks is required. This paper's main contribution is therefore an in-depth study and comparison of the cloud properties of these two open source frameworks, providing useful information that can aid in the decision making process of which framework to choose.**

I. INTRODUCTION

A. What is Cloud Computing

Simply put, Cloud Computing is the delivery of computer services such as servers, storage, databases, analytic, etc. over the Internet. Companies that offer these services charges users based on the usage, similar to how one might pay electricity bills based on usage. Cloud Computing is revolutionary in the sense that it brings together technology and business to deliver information technology resources on-demand. This fits in well in a scenario where resources have to be provisioned dynamically.

B. Uses of Cloud Computing

People around the world are using cloud computing at any given moment, even if they dont realize it. If you use an online system to send emails, edit documents, watch videos, listen to music, play games, or store pictures, etc., its likely that behind the scenes, it is cloud computing that is making it all possible. Even though cloud computing is still a young field, from small startups to large companies, from government organizations to non-profits, everyone is availing cloud computing services. Some of the commonly available services include:

- Create new apps and services
- Store, back up and recover data
- Host websites and blogs
- Stream audio and video
- Deliver software on demand
- Analyse data for patterns and make predictions

C. Benefits of Cloud Computing

Cloud computing is a prominent shift from the conventional way businesses perceive software delivery. So what are the specific points that work so much in favor of cloud Computing? Here are 6 common reasons why organizations are turning to cloud computing services:

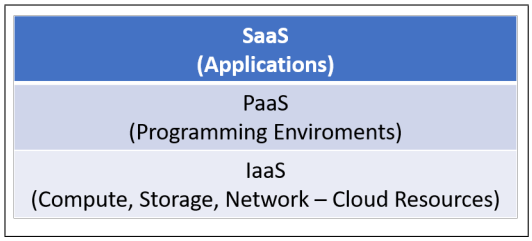


Fig. 1. Types of Cloud Computing

1. Cost

The capital expenses of purchasing hardware and software, setting up and running data centers, establishing servers and not to mention the round-the-clock electricity consumed for running all this and the number of technicians needed to manage the infrastructure - it can all add up pretty quickly. Cloud Computing helps eliminate a lot of these costs depending on the organization.

2. Speed

Most Cloud Computing services are grounded in the fact that they are all self-service and provided on-demand. Therefore, even when large amounts of computing resources are required to be provisioned it can be done in minutes and within a few clicks. This gives businesses a significant amount of flexibility and takes pressure off of capacity planning.

3. Global scale

One of the major benefits of cloud computing is the ability to scale up and down quickly and easily to meet demand (elastic scaling). This ensures the delivery of the correct amount of resources such as computing power, storage, bandwidth etc. for exactly when it is needed and precisely from the requisite geographic location.

4. Productivity

Along with cost, on-site data centers typically require investing a significant amount of effort in setting up hardware, patching software and other time-consuming IT management chores. Cloud computing steps in to remove most of these redundant tasks so that teams can spend time and energy in achieving the more important business goals.

5. Performance

The major cloud computing services are run on a worldwide network of secure data centers which are upgraded regularly to maintain the latest crop of fast and efficient computing hardware. This offers significant benefits over a singular data center since a network of data centers

reduce network latency for applications and allow economic scalability.

6. Reliability

Reliability was one of the major concerns during the initial days of cloud computing but today major providers successfully ensure ease of data recovery and data backup, consequently ensuring business continuity. The fact that data can be mirrored at multiple sites on the cloud provider's network contributes to this reliability.

D. Types of Cloud Services

Cloud computing can be broken down into three cloud computing models.

- **Infrastructure-as-a-Service (IaaS)** is the bottom most layer i.e. it's about renting out fundamental computing infrastructural such as: physical or virtual servers, storage and networking. This is appealing to companies that want to build applications from the very ground up and want control over nearly all the aspects of the development and deployment process themselves. It is generally opted for by companies with the kind of technical expertise on their teams to orchestrate services at the infrastructural level. In that sense, utilizing online infrastructure makes it simpler to innovate, cuts deployment time for new applications and services and significantly cut on-going maintenance costs. However, there is a concern that IaaS isn't secure enough for most critical data.
- **Platform-as-a-Service (PaaS)** is the next layer up – in addition to the underlying storage, networking, and virtual servers this kind of service will also incorporate the tools and software needed to build applications on top: that would include database management, operating systems, middleware and development tools.
- **Software-as-a-Service (SaaS)** is the delivery of applications-as-a-service and the version of cloud computing that is most commonly in use. The underlying complexity of the hardware and operating system is of no consequence to the end user, who need only access the service via a web browser or an app.

E. Types of Cloud Deployments

All cloud ecosystems are not the same. There are 3 different ways to deploy cloud computing resources: public cloud, private cloud and hybrid cloud.

- Public cloud Public clouds are owned and operated by a third-party cloud service provider, which delivers computing resources such as servers and storage over the Internet. Microsoft Azure is an example of a public cloud. With a public cloud, all hardware, software and other supporting infrastructure are owned and managed by the cloud provider. You access these services and manage your account using a web browser.
- Private cloud A private cloud refers to cloud computing resources used exclusively by a single business or organisation. A private cloud can be physically located on the companys on-site data centre. Some companies also pay third-party service providers to host their private cloud. A private cloud is one in which the services and infrastructure are maintained on a private network.
- Hybrid cloud Hybrid clouds combine public and private clouds, bound together by technology that allows data and applications to be shared between them. By allowing data and applications to move between private and public clouds, hybrid cloud gives businesses greater flexibility and more deployment options.

II. LITERATURE REVIEW

In [**6**] Vogel et al. contribute an analysis of Infrastructure as a Service by comparatively analyzing OpenNebula, OpenStack and CloudStack and evaluating their differences regarding support for flexibility and resiliency. This study provides relevant inspiration for our paper as it digs deep into the architectural differences that contribute to the respective behaviours of these IaaS frameworks. The study arrived at the conclusion of OpenStack being most resilient whereas CloudStack offering the most flexibility in deployment. The same team of authors headed by Vogel has previously conducted studies regarding differences in layer management between OpenNebula and OpenStack [**7**] to emphasize that OpenStack offers more robustness owing to its fragmented architecture.

In [**8**], Jayasinghe et al. studied six IaaS public providers utilizing standard benchmark tools. The aim was to analyze variations in performance and scalability during the migration of a multi-tier application from a traditional data-center environment to a cloud infrastructure. Furthermore, they evaluated this performance on 3 commonly used hypervisors: XEN, KVM and CVM. The results showcased significant variations in performance between the hypervisors.

In [**9**] Sefraoui et al. conduct a comparative study of different open source IaaS cloud providers such as OpenStack, Eucalyptus, OpenNebula and CloudStack with the aim to provide information on which open source product is most suited to building a hosting architecture, massively scalable but completely open source and overcoming the constraints and the use of proprietary technologies. The set of criteria used to perform this evaluation included: Management, Virtualization, Storage, Security and Community Vendor Support Criteria etc.

In this paper we try to compare and contrast the two frameworks OpenStack and CloudStack from an architectural point, from installation to running instances. During the course of this study we aim to understand the architectural complexities and intricacies of each platform and try to form an opinion, if possible, on which platform is better in which way and if one is more preferable than the other when it comes to business requirements.

III. OPENSTACK

OpenStack is a cloud operating system that offers the ability to control large pools of compute, storage, and networking resources throughout a data center. Everything is managed through a dashboard that gives administrators control while

empowering their users to provision resources through a web interface [**10**]. Launched in 2010, OpenStack was developed by Rackspace Hosting and NASA aimed at creating an open source alternative to build public and private clouds. The objective of the platform was to enable companies to create and offer their own cloud computing services over standard hardware while eliminating the complexities of software setup for such purposes. OpenStack was built keeping certain principles in mind:

- It is open source: all code is publicly available under the Apache 2.0 license.
- Open and Collaborative Design: every 6 months the development community meets to discuss requirements and specifications for the next release.
- Open Community: the platform is backed by a very active community that ensures a healthy development process and user community through an open and transparent process.

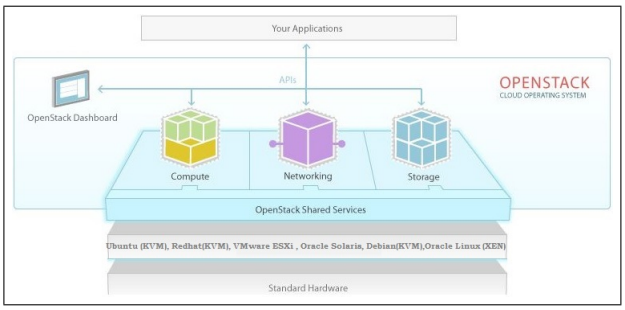


Fig. 2. A General OpenStack Architecture [**12**]

A. Components

As with any cloud service platform the infrastructure underneath OpenStack can comprise of any standard hardware configuration such as network devices, servers and disks. OpenStack develops virtualization layers in order to provide its cloud services. The layers are built up with various components described below. The 3 primary components of the OpenStack architecture are: Compute (Nova), Network (Neutron) and Storage (Object Storage specifically - Swift). On the basis of these founding pillars, OpenStack has developed several other services over the years to further help ease the infrastructure setup process. All these services are integrated through each service's own API (Application Programming Interfaces) which is publicly exposed to the other services.

OpenStack has core services for computing and networking as well as shared services such as identity-service imageing-service, telemetry, etc. Here's a detailed description of each component of the OpenStack Architecture:

1) Compute (Nova): Compute is the most important core component of the Openstack family of services. It is codenamed Nova and is written in Python. It is responsible for Virtual Machine (VM) Management. These VMs in turn host multi-tier applications, dev or test environments, Big Data clusters and high-performance computing applications. Compute manages large networks of VMs – it manages the entire life-cycle of an instance in the cloud i.e. from VM initialization to termination. Compute facilitates this management through an abstraction layer that interfaces with supported hypervisors.
   Compute consists of six subcomponents that we will discuss in detail when we elborate on the OpenStack architecture. They are: Nova-api, Nova-Compute, Nova-Network, ova-Volume, Message Queue and Nova-Scheduler [**10**].
2) Network (Neutron): Networking is key to cloud computing. Simply put, it is the channel through which offered resources and services can be accessed and it provides address-binding between different services in order to support multi-tier applications. Automatic network configuration capability is therefore necessary and important.
   The OpenStack Networking service codenamed Neutron (previously called Quantum) provides different networking services to cloud users such as: DN, DHCP, IP address management, load balancing, and security services like firewall policies etc. Neutron provides an SDN (Software Defined Network) framework that facilitates integration with other networking solutions [**10**]. It has a flexible plug-in plug-out architecture that allows developers to, for instance, use their own load balancing algorithms to achieve better workload control. Neutron allows cloud users the ability to manage their guest network configurations. Security concerns dealt with by Neutron include but are not limited to network traffic isolation, availability, integrity, and confidentiality.
3) Storage (Swift and Cinder): OpenStack supports both Object Storage (codenamed Swift) and Block Storage (codenamed Cinder) along with multiple deployment options for each, depending on the specific scenario.
   Swift is a scalable Object Storage system. It provides a fully distributed system, accessible through an API and that can be integrated directly into applications to aid data backup, recovery and retention. It is important to understand that object storage differs from traditional file system storage. Object storage is best used for static data such as media files (MP3s, images, or videos), virtual machine images, and backup files. With Swift data is written to multiple hardware devices and the Openstack software handles data replication and integrity across clusters. Object Storage clusters are scaled horizontally by adding new nodes. If a node fails then OpenStack can replicate the data from other active nodes. OpenStack uses software logic that requires inexpensive commodity hard drives and servers in comparison to expensive equipments. As a result, Swift provides a cost-effective scaling storage system.
   Block Storage divides data into evenly sized chunks allowing incrementally manipulating data. With Cinder,

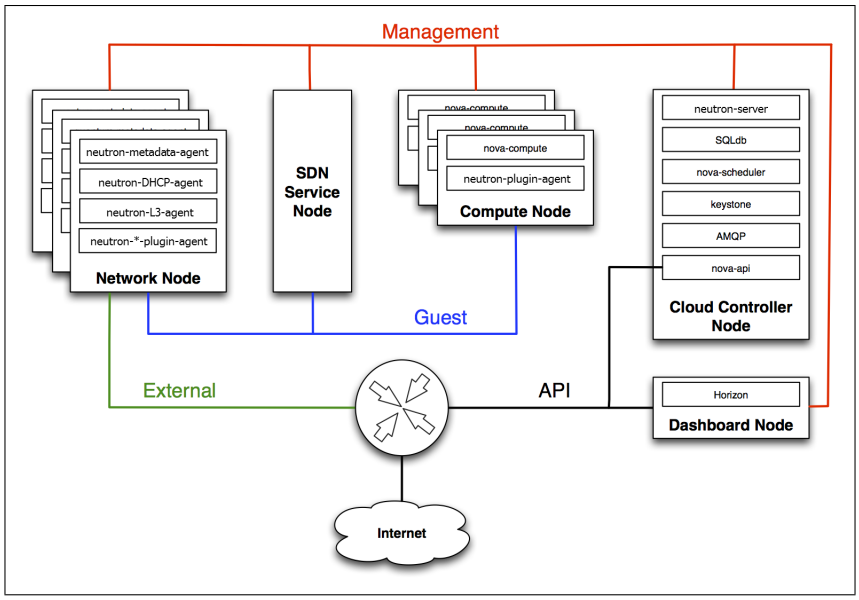


Fig. 3. Network (Neutron) Architecture [**10**]

OpenStack's Block Storage component, block devices are connected to Compute instances for more storage and better performance as well as better integration with other storage platforms such ass NetApp and SolidFire. Cinder is appropriate in performance sensitive scenarios.

4) Dashboard (Horizon): OpenStack provides an user nterface to control and manage the other components such as Compute, Networking etc. The dashboard, codenamed Horizon, s available for users as well as administrators who can set limits and restrictions on resource access by users.
5) Identity Service (Keystone): The OpenStack Identity service (Keystone) is a shared service that provides common authentication and authorization services throughout the entire cloud infrastructure. It support various forms of authentication such as crednetial based, token based and AWS style logins. The Identity service has pluggable support for multiple forms of authentication. Security concerns with identity service include trust in authentication, the management of authorization tokens, and secure communication.
6) Image Service (Glance): The OpenStack Image service (Glance) provides disk-image management services. This includes registration, image discovery and delivery services to the Compute service Nova, as needed. This image repository is used as a database of image templates that can be used to get new servers up and running quickly and easily.
7) Telemetry: Like most cloud platforms OpenStack has its own service that offers billing and analytical abilities. Users can track resource usage and performance data of the services deployed on the OpenStack platform.
8) Orchestration: This 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.

B. Architecture

OpenStack's architecture is governed by three kinds of requirements: Enterprise requirements, Operational requirements and High-Availability requirements. As a result there is no fixed architecture that OpenStack deployments follow. However, in this paper, we discuss a bare-bones, minimalistic architecture that is required to have a functional OpenStack deployment:

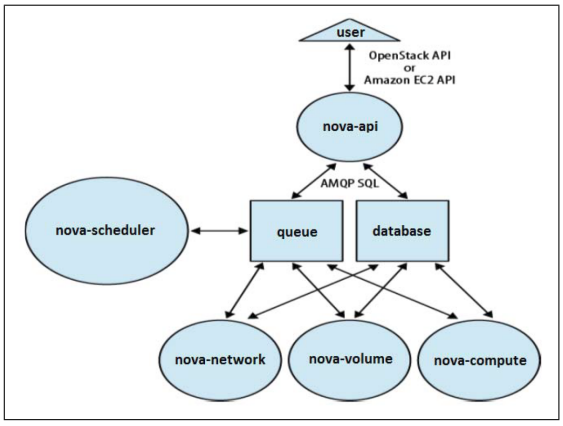


Fig. 4. Compute (Nova) Architecture [**3**]

1) Compute (Nova) Architecture
As mentioned above, Nova is a distributed application that consists of six components: Nova-api, Nova-Compute, Nova-Network, Nova-Volume, Message Queue and Databases, and finally Nova-Scheduler

[**10**]. The components follow the architecture as illustrated in Figure 4. and as explained below:

- Nova-api: This component accepts and responds to end user compute API calls. In addition to providing its own OpenStack Compute API, Nova-api is further compatible with Amazon EC2 API, thus encompassing the capability to integrate with Amazon cloud services. It has a special Admin API which is reserved for users with the privilege to perform administrative actions. Various orchestration activities like running an instance, enforcing policies such as quota checks are also controlled by this component.
- Nova-compute: This component is primarily a worker daemon that is responsible for creating and terminating VM instances via hypervisor APIs. It accepts actions from the queue and executes system commands to fulfill them, and updates the database state accordingly. OpenStack supports several commodity hypervisors while maintaining the openness that allows for interfacing with other hypervisors through its standard library.
- Nova-volume: This component is responsible for managing the creation, attachment and detachment of persistent volumes to compute instances. Nova-Volume supports two types of block devices for an VM instance [**7**]: (1) Ephemeral Storage: is associated to a single unique instance. The life-cycle for this kind of storage follows the instance life-cycle – when the instance is terminated, data on this storage will also be deleted. (2) Volume Storage: This is a persistent form of storage and functions independent from any 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: This is basically a worker daemon which is tasked with handling the network-related tasks. It accepts and performs networking tasks from the queue related to manipulating the network e.g. setting up bridging interfaces or changing iptable rules.
- Nova-schedule: This component takes care of the scheduling of VMs among physical machines. It accepts a virtual machine instance request from the queue and decides the physical host to place the instance on. While the users can define their own scheduling algorithms, Nova-schedule supports three algorithms by default [**6**]: (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. As mentioned earlier, the flexibility the users have in defining their own scheduling algorithms, goes a long way in building a fault tolerant and load-balanced system.
- Queue and Databases: The Message Queue (Rabbit MQ) provides a central hub for message-passing between daemons. While the default queue used is Rabbit MQ, OpenStack supports almost all other AMQP (Advanced Message Queuing Protocol) message queue. Databases store the build-time and run-time state of a cloud infrastructure. For instance, databases provide information regarding instances which are available for use or are currently in use, availability of networks or storage information. While OpenStack Nova claims to be able to support any SQLbased database, the most widely used databases currently are SQLite3, PostgreSQL and MySQL.

2) Network (Neutron) Architecture
The Network architecture consists of four distinct physical data center networks [**2**] [**10**]:

- Data network: this network is responsible for VM data communication within the cloud deployment. The IP addressing requirements of this network are decided by the OpenStack Networking plug-in that is in use.
- Management network: this network is used by OpenStack to communicate internally. The IP addresses on this network should be reachable only within the data center.
- API network: this network exposes all OpenStack APIs, including the OpenStack Networking API, to users. The IP addresses on this network should ideally be accessible to anyone on the Internet.
- External network: this network is used to provide VMs with Internet access at the time of deployment. The IP addresses on this network should also be accessible to anyone on the Internet.

IV. CloudStack

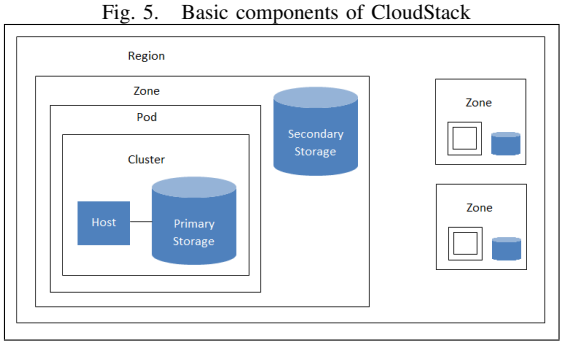
Apache CloudStack, written in Java, [**1**] is an open source software for creating, deploying and managing public and private Infrastructure-as-a-Service (IaaS) clouds. It provides an open and flexible cloud orchestration platform to deliver reliable and scalable clouds. It is being used by a large number of service providers and IT companies to create and manage public, private, and hybrid clouds.
The development of CloudStack started in 2008 by VMOps, which was later renamed to Cloud.com . It was open source under GNU General Public License in 2010. In July 2011, Citrix Systems bought Cloud.com and in April 2012, Citrix donated CloudStack to the Apache Software Foundation (ASF), where it was accepted into the Apache Incubator;Citrix changed the license to the Apache License version 2. [**2**]

A. Components

In order to understand the basic architecture of CloudStack, we describe each component:

- *Region*: A region is the largest available organizational unit within a CloudStack deployment. It is consists of multiple, geographically separated zones where each zone is roughly the size of a data center. Within a region, the distributed zones are close to each other. Fig 5

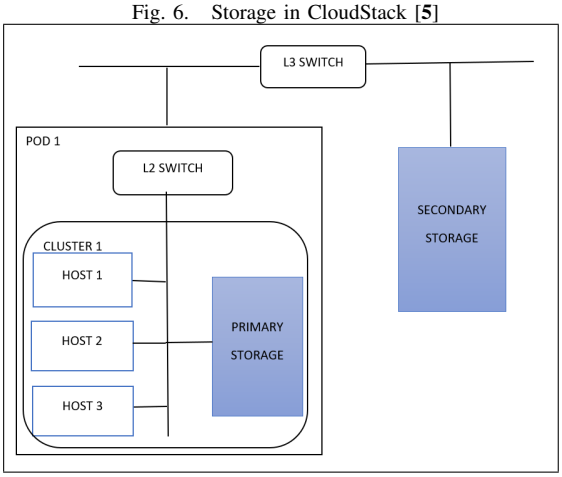
Fig. 5. Basic components of CloudStack



- *Zone*: A zone is the second largest organizational unit within a CloudStack deployment. It is basically data center. Its primary purpose is isolation and redundancy. A zone consists of following:
  1) Single or multiple pods
  2) A secondary storage which is shared amongst all the pods in a zone
  3) Optional primary storage servers which are shared amongst all the pods within the zone

  Unlike regions, zones are visible to the end user.
- *Pods*: Pods are contained within a zone and represent a single rack in the data center. It contains large number of different clusters linked with layer 2 switch. They are not visible to the end user.
- *Cluster*: Clusters are used to group hosts together with the primary storage server and have same IP subnet. They are contained within pods and the size of a cluster depends on the underlying hypervisor. A cluter can support more than one primary storage server. [**c3**]
- *Host*: They are the physical servers containing local storage onto which services are provisioned. They provide the CPU, memory, storage, and networking resources needed to host the virtual machines. CloudStack supports a large number of hypervisors such as, XenServer, VMware, Oracle VM, KVM, HyperV, vSphere and bare metal. Hypervisors need to be same within a cluster but can we can have different hypervisor in a different cluster. [**3**]
- *Management Server*: The management server is responsible for allocating and orchestrating resources in the cloud using APIs or UI. A single management server can manage multiple zones and roughly ten thousand hosts. It controls allocation of virtual machines to hosts and assigns storage and IP addresses to the virtual machine instances. [**1**] A database is required for the management servers to be persistent. CloudStack uses MYSQL as the database. Management server is stateless and can be deployed as physical server on a virtual machine. Multiple management server nodes can be deployed for scaling up the architecture as well as for redundancy.

Fig. 6. Storage in CloudStack [**5**]



- *Storage*: CloudStack has two main types of storage described as follows: Fig6
  1) Primary Storage: Primary storage is associated with a cluster. It is kept close to hosts for better performance as it directly interacts with the application. It is responsible for storing virtual disks for all virtual machines running on that particular cluster. Multiple primary storage servers can be setup at per-zone or per-cluster level but a minimum of one is necessary. Primary storage is needed at per-zone level to avoid extra data copy operations. [**4**]
  2) Secondary Storage: The secondary storage is configured at per-zone or per-region level and the items stored on it are available to all hosts in the scope of the secondary storage. It currently supports two different types - NFS and object storage. It stores the following components [**4**]:
     a) Templates: The base operating system image that user wants to use when creating the virtual machine. It may also contain some configurational information such as installed application.
     b) ISO images: These are the disks images containing the bootable media for operating system for which new VM instance will be created
     c) Disk volume snapshots: These are the saved copy of the already existing VMs. There main purpose is for data recovery and creation of new templates.
- *Networking*: CloudStack provides two networking styles:
  1) Basic: It is used for AWS-style networking. It provides a single network where guest isolation can be provided through layer-3 means such as security groups such as IP address source filtering.
  2) Advanced: This networking type is more flexible compared to the basic architecture as it allows

multiple networks and many additional features. CloudStack provides many networking services such as isolation of networks, load balancing, VPN etc.

B. Architecture

In the previous section we described how various components of CloudStack. In this section we discuss the basic architecture of CloudStack. The architecture of CloudStack is simple and the deployment varies depending on the size and purpose of deployment. For example, one can have a small scale setup of CloudStack in order to just understand and run some trials on CloudStack or a multi-site deployment for large scale infrastructure for IT firms.
The hierarchical structure of CloudStack allows it to easily scaled from a small scale setup to large scale setup as a single management server can handle upto thousands of physical server. The lowest level consist of hosts along with a primary storage. Together many such hosts along with the primary storage server form a cluster. Multiple clusters combined with the layer 2 switch form a pod. As we go the hierarchy, the pods are clubbed together along with layer 3 switch to form zones. Zones can be visualized as data centers. The management server along with Mysql database together controls a single or multiple zones. The biggest advantage of CloudStack is that it is very easy to scale the architecture up or down. One can keep adding new zones and provide the required information about the zone to the management server and scale up the architecture. Also multiple management servers can be added to handle large amount of traffic or for redundancy.

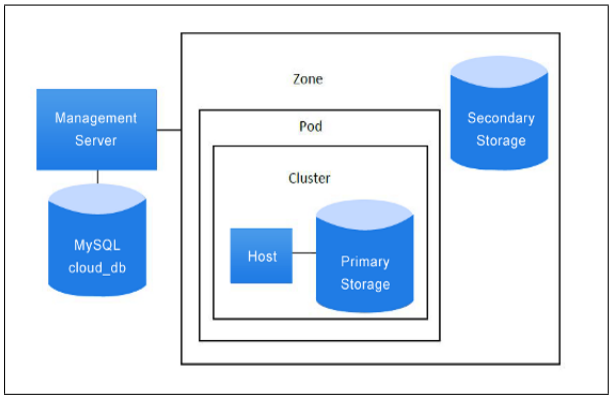


Fig. 7. CloudStack Architecture

V. CONTRAST AND COMPARISON

A. Common Characteristics

There a few properties, common to both OpenStack and CloudStack, grounded in the philosophy that technology lock-ins need to be gotten rid off and users should have the freedom to choose the best slot that matches their requirements. We managed to narrow them down to some of the following:

Fig. 8. Comparison between OpenStack & CloudStack [3]

	OpenStack	CloudStack
Open Source License	Apache 2.0	Apache 2.0
Installation	Medium	Difficult
Supported hypervisors	Xen, KVM, HyperV, VMWare, LXC, vSphere	Xen, KVM, HyperV, VMWare, LXC, vSphere
Live migration	Yes	Yes
Load balancing	VM level, PM level	VM level, Host level
Fault tolerance	VM scheduling, replication	VM scheduling, replication
Security	VPNs, firewall, user authentication, others	VPNs, firewall, user management, others
Compatibility	Amazon EC2, Amazon S3	Amazon EC2, Amazon S3

- *Compatibility*: Both OpenStack and Cloudstack are highly compatible with the Amazon API. As a result client applications written for Amazon Web Services such as EC2 and S3 can be used on these stacks with minimal porting efforts [**1**] [**10**]. In terms of hypervisors, both platforms support the spectrum of most of the commonly used hypervisors such as Xen, KVM, VMWare etc. Even other hypervisors, if they have standard drivers, can be interfaced with these platforms through their inbuilt libraries e.g. *libvirt* in the case of OpenStack.

- *Security*: Both frameworks have a heavy emphasis on data security: both of them offer isolation through protected and segregated user accounts (such as Keystone in OpenStack). Both frameworks offer credential based, token based as well as AWS type logins for authentication purposes. In addition to isolation using different accounts, both platforms offer isolation of traffic using security group strategies and allows setting up firewalls and VPNs as well.

- *Availability*: Both platforms seek to lessen downtime and ensure availability of services. In OpenStack, high availability is achieved through different setup confguration depending on types of services, i.e, stateless or stateful services. Stateless services can answer a request without needing any further information about the other service but stateful services on the other hand require information in order to answer a request. Therefore stateful services are the ones that give trouble in maintaining high availability. CloudStack ensures high availability of the system by using multiple management server nodes which may be deployed with load balancers.

- *Fault tolerance*: Both platforms do an excellent job of showcasing fault tolerance. With flexible architectures, fault tolerance is handled at different levels using a variety of mechanisms: scheduling algorithms prevent failures at VMs level, multi-node configurations ensure against one or more node failures, failures at database level are handled through replication, hosts failing is tackled through image-repositories (taking image from

the secondary storage and running application data in primary storage).

- *Load Balancing*: Both OpenStack and CloudStack support load balancing. In OpenStack, it is supported at different scales and is an optional feature in CloudStack where the traffic can be distributed amongst different management server. CloudStack also provides the users with an option of using external Load Balancer such as Citrix NetScaler. [**11**] Currently, OpenStack has an on-going project called Load Balancing as a Service (LBaaS) that is aimed to provide load balancing service to end users. [**3**]

- *Live Migration*: Both the platforms support live migration. In OpenStack the live migration is supported at VM as well as PM level. It supports two types of live migration:
  1) Shared storage based live migration in which shared storage can be accessed by source and destination hypervisor.
  2) Block live migration where there is no requirement of shared storage.

  CloudStack supports live migration at VM level as host level via Dashboard. Depending on which VMs hypervisor is being used, the migration conditions are different.[**3**]

- *Scalability and Extensibility*: While both platforms offer ways to extend their functionalities, they do so in different ways.
  OpenStack has a very flexible and open architecture as per their policy of not being technology-tied. As a result, it is easy to replace or modify a particular component without having to change much else in the stack. The downside to this approach is that one needs to have a good understanding of the components they wish to modify. This involves having to go through the codebase, getting familiar with the existing functionalities and understanding how they are implemented. Most of OpenStack's components are coded in Python. Hence, Python familiarity is also essential.
  CloudStack on the other hand is a more compact packaged framework. But it is easy to extend functionalities with CloudStack through their rich API. It is simple to automate tasks, build event-based features using the API exposed by CloudStack. The benefit of this API architecture is that one does not need to have a good knowledge of the CloudStack codebase or its components to extend certain functionalities. The API design is intuitive and follows conventional API architecture.

B. Key Differences

To summarize the key differences between OpenStack architecture and that of CloudStack, we have provided the following.

1) *Code Architecture*: OpenStack is built on API stacks and is component based. Each component exposes an endpoint and the OpenStack compute and dashboard module is in charge of connecting them. On the other hand, CloudStack is built upon stacks of services and internal agents. The internal agents are in charge of making sure the stacks of services are working with each other and communicating as needed.
2) *Language*: OpenStack is built with Python for most parts. CloudStack is Java-based.
3) *Setup*: OpenStack is notorious for being very difficult to set up properly. In contrast, CloudStack is cleaner and simpler since its one entire ecosystem as opposed to having many different pluggable components that have to work together.
4) *Extending*: As already mentioned in the previous section, they are both extensible but in different ways. It may be argued that it is easier to extend CloudStack than OpenStack but OpenStack does offer a lot of flexibility in terms of components that the user wishes to use.
5) *Scaling*: While both frameworks are meant to be highly scalable, it may be argued that CloudStack if easier to scale with a defined hierarchical design.
6) *Dashboard*: OpenStack has a simpler dashboard compared to CloudStack. That, along with the API exposed by CloudStack makes its dashboard very easily adaptable to requirements.

C. Choosing a Cloud Framework

After having read through existing literature, numerous blog posts by industry experts, and through our own experiences working with the two frameworks, we have formed our own opinion on the matter. We have identified the following as key factors:

1) *Desirable Cloud Properties*: Desirable properties for a Cloud environment include Compatibility, Security, Availability, Fault Tolerance, Load Balancing, Live Migration, Scalability and Extensibility. These features are well supported by both platforms as already mentioned in the previous section.
2) *Status as Open Source Projects*: As they are both Open Source projects, they have similar licensing policies. A key factor for using open-source projects is that they need to have an active community, a good user-base and regular releases. Both projects have active communities that endorse the projects and work on them. OpenStack is perhaps the more popular with the larger community behind it, but CloudStack is itself a major platform with a large enough community behind it. OpenStack and CloudStack both have two releases a year. OpenStack has large names like HP and Google adopting their platform. CloudStack has companies like Disney.

3) *Ease of Use*: We consider the following when we discuss ease of use-
   - Time to production: CloudStack is arguably easier to setup and has a simpler hierarchical architecture that lets developers scale it without undue effort. On the other hand, OpenStack is known for being hard to setup. [**14**]
   - Modifications and improvements: As mentioned in the previous section, CloudStack offers simple APIs to extend the features. However, both platforms are flexible. CloudStack is perhaps simpler for the average developer to work with.
   - Documentation and Troubleshooting: Since both platforms have an active community, developers can count on them to provide support regarding bugs and issues faced. In terms of documentation, OpenStack and CloudStack both have thorough documentation.

   Thus, from the point of view of companies looking to adopt a private cloud platform, CloudStack would be more beginner friendly

Our conclusion is that there is no clear winner in terms of architecture or community support when it comes to the two platforms. In terms of ease of use, CloudStack is just slightly ahead of OpenStack. Thus, the debate of which platform to use is entirely situational and depends on the company using it, what kind of expertise they have, what use cases for the company are best supported by which platform. In many ways, it comes down to just personal preference as evidenced by the various industry experts opining that they platform they endorse is better.

VI. EXPERIMENT AND RESULTS

. To further our understanding of the two private cloud platforms, we have implemented our own toy example at the University of Waterloo using clusters available to Computer Science students at the university. This section describes experimental setup and presents some key findings from some experiments we conducted.

A. Experimental Setup

For our example, we used four separate machines to setup own cloud environments. The machines are named ugster01, ugster02, ugster03 and ugster04. We setup OpenStack on ugster03 and ugster04 and we setup CloudStack ugster01 and ugster02. The machines have identical configurations with Sun X2100s with 1 CPU, 1GB memory, and an 80GB hard drive. All machines had Ubuntu16.04 running on them.

For OpenStack installation, we set up the Compute node in ugster03 and we setup the controller node on the ugster04 (figure-9). For CloudStack installation, we set up the Management Server and the Database Server in ugster01, and we setup the cluster with a Host, Primary Storage as well as a Secondary Storage in ugster02 (figure-10).

Our installation is a bare-bones installation with the minimal necessary components. Real world implementations of either platform is significantly more complex and spans across multiple centers across geographic locations.

Fig. 9. OpenStack Setup

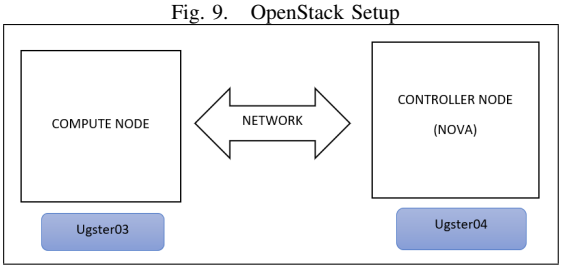
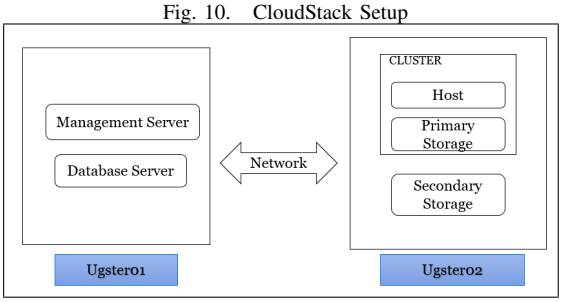


Fig. 10. CloudStack Setup



B. Benchmarking Kits

Initially, we wanted to run some benchmarks to compare the two architectures. On further reading we realized the following.

- There are little to no benchmarking tools that work with both OpenStack and CloudStack.
- PerfKit [**15**] claims to support both platforms but in practice is difficult to make work for CloudStack.
- Microbenchmarks are not good indicators of performance for Cloud Computing. As the overhead for either platform for an operation is more or less the same as the other, microbenchmarks give very similar results [**6**],[**9**]. The minor deviation in results is inconsequential when considering the grand scheme of things such as the architecture, time to production or deployment.

Instead, we focused most of our work on understanding the architecture of the two platforms. We ran some basic network latency tests to show that our assumptions are correct. On the systems, we ran *iperf* [**13**] for varying bandwidth caps to measure the transfer size.

C. Result

We ran iperf on the Linux systems without OpenStack installation or CloudStack installation and measured the baseline performance. We then ran the experiment for both platforms once for TCP (Figure-11) and once for UDP (Figure-12).

As we see in our results, there is no discernible different between CloudStack and OpenStack in terms of performance in this experiment. Both CloudStack and OpenStack have a slightly lower performance than baseline. This can be explained by the fact that there is a small overhead for both platforms. We see performance close to baseline because

Fig. 11. Bandwidth Vs. Transfer (TCP)

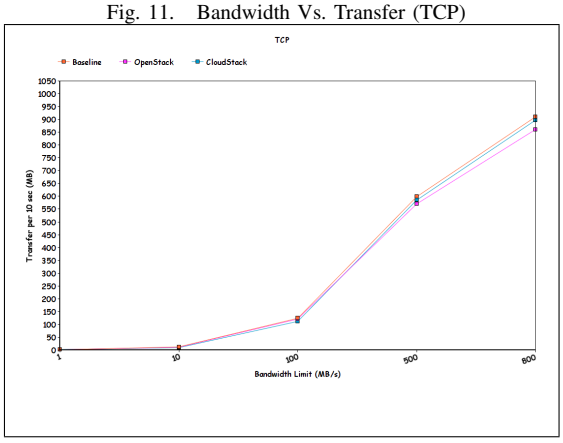
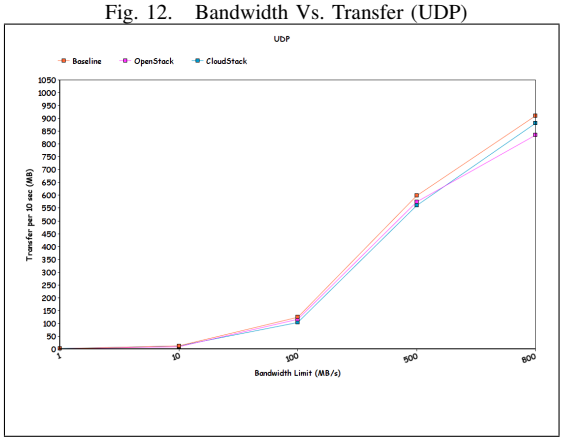


Fig. 12. Bandwidth Vs. Transfer (UDP)



there is very little load on the machines. The difference between the OpenStack and CLoudStack is minimal.

VII. CONCLUSION AND FUTURE WORK

In this section, we leave some concluding remarks summarizing our project as well as highlighting some key lessons we learned doing it. Lastly, we identify some limitations of our work and mention what future directions for work on this topic can be.

A. Conclusion

The need for private cloud has increasingly become apparent to many companies over the years. OpenStack and CloudStack are two of the biggest private cloud computing platforms at present. Both platforms are quite popular with many big companies endorsing them and a large community behind it. As such, it is often a tricky question to choose one over the other.

In this paper, we perform a thorough study on the architecture of the two platforms and compare them. We highlight our understanding from reading existing literature on the domain as well as opinions by industry experts. They support the same features and offer key requirements for cloud platforms. Our understanding is that there is no clear winner between the two platforms and most experts do not seem to think they are competing [**13**]. Both platforms have their place and have something different to offer. They are architecturally very similar in terms of features as well as in terms of performance as shown in our experiments.

B. Lessons Learned

The project has been a learning experience for us. There are a few key takeaways from this that are worth mentioning. Although OpenStack and CloudStack can be difficult to set up, particularly when we consider large networks across large geographic regions, the ideas behind the implementations are somewhat straightforward. Most of the complexities are introduced by the network configurations necessary as well as the operating system environment. That said, a big part of the time taken goes behind troubleshooting errors that arise from said configurations not working correctly. OpenStack in particular is susceptible to this issue as it has all those separate components that need to work in tandem. As such, for any newcomers to the technology, we recommend a certain degree of networking expertise as well as operating system expertise when approaching either platform.

Secondly, we have found that both platforms are more geared towards CentOS/RedHat distributions of Linux and have more documentation for them as opposed to Ubuntu/Debian. Thus, we recommend beginners to use CentOS rather than Ubuntu for setting up these platforms.

Lastly, we only worked with a simple toy example for both frameworks. We understand that real-life implementations of cloud platforms are much more complex. However, in working with the scaled down example, we have taken away key ideas and the necessary knowledge we need to get started with a larger and more complex architecture.

C. Limitations and Future Work

Due to installation issues we faced with OpenStack and CloudStack, we ended up with very little time for experimentation and evaluation. As such, the results from our experiments are not conclusive. More rigorous experiments can be performed to draw more meaningful and conclusive results.

Existing literature on the topic are mostly by university researchers, students and faculty. This is a major drawback as none of them are users of these cloud computing platforms. We feel that there needs to be work to bridge the gap between industry and academia. In particular, opinions from industry experts need to be taken into account for this discussion. For our work, we have taken into consideration opinions from publicly available documents and blogs, but it would be more meaningful if there was an exchange of ideas.

ACKNOWLEDGMENT

We would like to express our sincere gratitude to Professor Martin Karsten for his continuous support, guidance and motivation during the entire project. We would also like to thank our fellow classmates for helping and supporting us in finishing the project.

REFERENCES

[1] Apache Cloudstack. Apache Cloudstack, cloudstack.apache.org/.

[2] Apache CloudStack. Wikipedia, Wikimedia Foundation, 12 Apr. 2018, en.wikipedia.org/wiki/Apache_CloudStack.

[3] Barkat, Amine, et al. Open Stack and Cloud Stack: Open Source Solutions for Building Public and Private Clouds. 2014 16th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing, 2014, doi:10.1109/synasc.2014.64

[4] Concepts and Terminology. Concepts and Terminology - Apache CloudStack 4.11.0 Documentation, docs.cloudstack.apache.org/en/latest/concepts.html#management-server-overview.

[5] CloudStack - Open Source Cloud Computing Project Follow. CloudStack Architecture. LinkedIn SlideShare, 3 May 2012, www.slideshare.net/cloudstack/cloudstack-architecture.

[6] Vogel, Adriano, et al. "Private IaaS clouds: a comparative analysis of OpenNebula, CloudStack and OpenStack." Parallel, Distributed, and Network-Based Processing (PDP), 2016 24th Euromicro International Conference on. IEEE, 2016.

[7] Vogel, Adriano Jose, Dalvan Griebler, and Demetrius Roveda. "Understanding, Discussing and Analyzing the OpenNebula and OpenStack's IaaS Management Layers." Revista Electrnica Argentina-Brasil de Tecnologias de Informao e Comunicao

[8] Jayasinghe, Deepal, et al. "Variations in performance and scalability: An experimental study in iaas clouds using multi-tier workloads." IEEE Transactions on Services Computing 7.2 (2014): 293-306.

[9] Sefraoui, Omar, Mohammed Aissaoui, and Mohsine Eleuldj. "Comparison of multiple iaas cloud platform solutions." Proceedings of the 7th WSEAS International Conference on Computer Engineering and Applications,(Milan-CEA 13). ISBN. 2012.

[10] OpenStack Software Official OpenStack Documentation, www.openstack.org/software/.

[11] P. N. Sabharwal, Integrating netscaler with cloudstack, in Apache CloudStack Cloud Computing. Packt Publishing

[12] "Redhat OpenStack" Redhat's OpenStack Platform, www.redhat.com/en/technologies/linux-platforms/openstack-platform

[13] Install IPerf to Diagnose Network Speed in Linux. Linode Guides & Tutorials, 28 Feb. 2017, linode.com/docs/networking/diagnostics/install-iperf-to-diagnose-network-speed-in-linux/.

[14] CloudStack vs. OpenStack: My Experience. CloudOps, www.cloudops.com/2013/02/cloudstack-vs-openstack-a-personal-experience/.

[15] GoogleCloudPlatform. GoogleCloudPlatform/PerfKitBenchmarker. GitHub, github.com/GoogleCloudPlatform/PerfKitBenchmarker.