Load Balancing As A Service In Openstack-Liberty

Rashmi T V, Dr. Keshava Prasanna, Mr. Girish L

Abstract: Cloud computing is a technology which provides computing resource on demand over the internet as a service. To meet this, many opensource cloud operating system are provided for the tenants, in order to get useful services from the cloud. There are many opensource cloud OS like AWS, Open Shift, HP, OpenStack etc. Out of all these OpenStack comes with free of cost and it has got a huge community. It can be installed and deploy in private institution or company with free of cost. This paper provides a model and techniques for the dynamic load balancing in OpenStack for managing the traffic/loads among the Virtual Machines. The main purpose is to increase the utilization of computing resources and minimize the traffic. Load Balancing as a Service is one of the main service in OpenStack Networking. OpenStack is an opensource platform which provides Infrastructure as a Service. It allows users/tenants to\create their own private clouds and to deploy Virtual Machines, which manages different workloads. In this paper, we provide an architecture of openstack LBaaS, for dynamic load balancing in open stack cloud deployment.

Keywords: IaaS, Neutron, Octavia, Load Balancer.

I INTRODUCTION

Cloud computing model makes use of virtualization to provide computing resources to the users over internet on demand and pay per use pricing model [1]. Cloud services allow users to use software and hardware those are set up and handled by third persons at distributed locations. It leverages the characteristics of on demand self-service and rapid elasticity so that it enables customers to dynamically and efficiently manage their resource usage according to the present workload conditions. These characteristics of the cloud computing model allow the users not to invest high on infrastructure and thereby minimizing the time to market and provide the space for innovation. Cloud computing resources are handed over to the users through three basic service models listed below:

1. Infrastructure as a service (IaaS): IaaS provides access to the computing resources in the form of virtual machines. Virtual machine brings the user an aspect of dedicated physical machine. The user is able to operate the system within a virtual machine and run the required software. Examples: Windows Azure, Google Compute Engine, Amazon EC2.
2. Platform as a Service (PaaS): PaaS provide access to the computing resources in the form of an application program interface. It is used by customer to develop and run their own applications. The user doesn't have the rights to access the system recourses. Allocation of resources to the application is done automatically by platform. Examples: Microsoft Azure, Google App Engine
3. Software as a Service (SaaS): SaaS provides software applications as a service to the users on subscription basis. Users don't have to bother about installation, setup and running the software application. Examples: Google Apps, Microsoft Office 365

- *Rashmi T V, Dr. Keshava Prasanna, Mr. Girish L*
- *PG Student, Dept. of CSE, CIT, Gubbi*
- *Professor, Dept. of CSE, CIT, Gubbi*
- *Asst.Professor, Dept. of CSE, CIT, Gubbi*

In this project, we mainly work on infrastructure as a service platform. Other than computing service models, cloud computing services are also classified according to deployment models. OpenStack Networking (Neutron) [4] manages all the networking aspects for the Virtual Networking Infrastructure (VNI) and access the layer aspects of the Physical Networking Infrastructure (PNI) for the OpenStack. Networking enables tenants to create advanced virtual network topologies which may include services such as a firewall, a load balancer, and a virtual private network (VPN). Networking also provides networks, [4] subnets, and routers as object abstractions. Each abstraction has functionality that mimics its physical counterpart: networks contain subnets, and routers route traffic between different subnets and networks. Load balancing is dividing the amount of work that a computer has to do between two or more computers so that more work gets done in the same amount of time and, in general, all users get served faster. Load balancing can be implemented with hardware, software, or a combination of both. Load balancing plays an essential role in providing quality of service (QoS) guarantees in cloud computing, and it has been generating substantial interest in the research community. Typically, load balancing is the main reason for computer server clustering.

- A load balancer is a device that acts as a reverse proxy and distributes network or application traffic across a number of servers.
- Load balancers are used to increase capacity (concurrent users) and reliability of applications.

Cloud Load Balancing is the process of distributing the workloads across multiple computing environments. Load balancing helps in fair allocation of computing resource to achieve a high User satisfaction and proper Resource utilization. Load balancing is a technique that helped networks and resources by providing a Maximum throughput with minimum response time.

II. RELATED WORK

OpenStack

OpenStack is a cloud operating system that controls large pools of compute, storage, and networking resources throughout a datacenter, all managed through a dashboard that gives administrators control while empowering their users to provision resources through a web interface. OpenStack [3] project is a cloud operating system that provides software

tools for managing and building cloud computing platforms. It controls large pool of services throughout the data centers through dashboard service. OpenStack [3] allows the clients to initiate virtual machines and other instances on fly according to current work load situation with in cloud environment. This is an open source platform which supports all types of cloud environment. The main objective is to provide simple implementation, horizontal scaling and comes with rich set of features. OpenStack offers an infrastructure as a Service platform through various complemented services. Offering IaaS means it is easy for the clients to deploy new instances upon which other cloud components may execute. It provides massive scalability which means jobs executing concurrently can serve users easily on fly by just adding or deleting instances. All services are accessed through application programming interfaces that also provide integration between different services. As this is open source software any one can make access to the source code, do alterations or modifications as they need and share the changes made with other community. All over the world OpenStack developers are trying to build most solid, secure and robust project they can.

OpenStack Components

OpenStack has a standard architecture with many different going parts. Due to its open nature, additional components can be appended to OpenStack by anyone to meet their requirements. OpenStack community has described some components that are shown in figure 2 are core part of OpenStack which are listed and explained as follows.

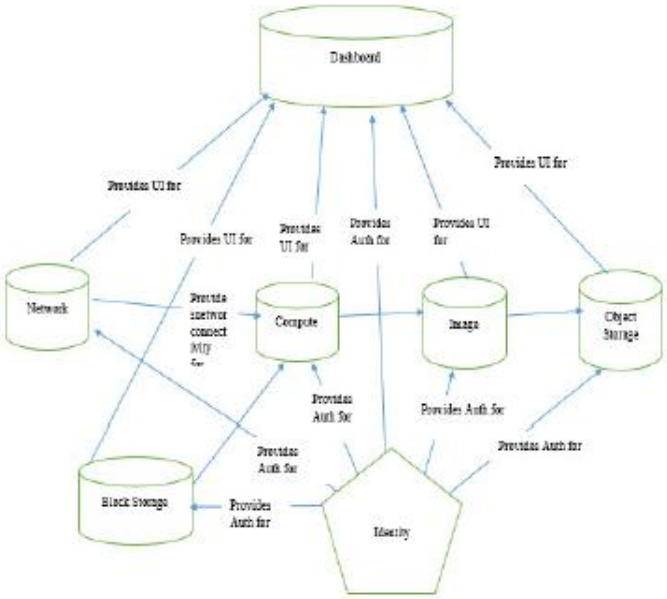
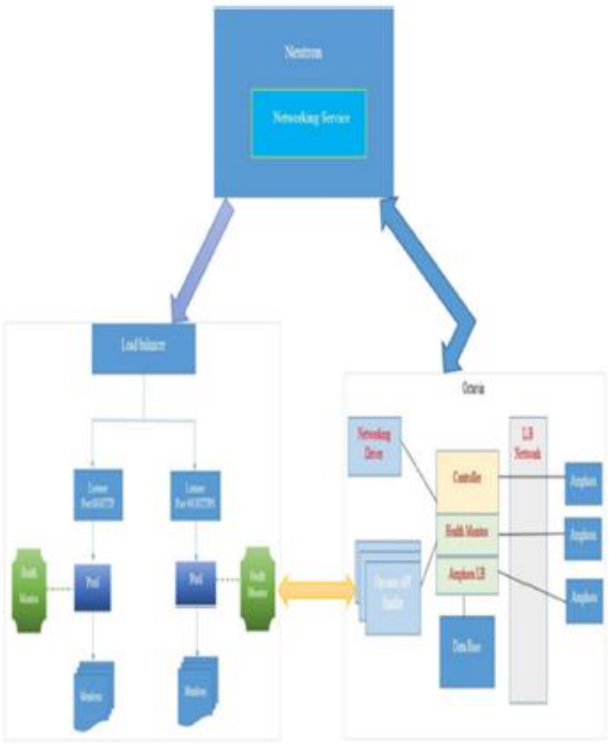


Fig 2: *OpenStack Components. [5]*

1) **Nova:** Nova is a primary OpenStack compute service. It is used for hosting and managing the cloud compute instances in an OpenStack environment. It also manages the life cycle of compute instances. The responsibilities include spawning, decommissioning and scheduling of virtual machines on demand.
2) **Swift:** Swift is storage system provided for objects and files. Instead of referring the files by their position on disk drive, they are referred by unique identifier or a piece of information and then decide where to store the information. Stores and retrieves unstructured data objects via RESTful or HTTP based API. It is more faults tolerant with its data replication and makes the scaling easy. It provides the system to make sure that data is backed up in case of network or machine failure.
3) **Cinder:** Cinder gives block storage as service functionality. It is more correspondents to traditional notion of system that access particular locations on the disk drive. The way of accessing files is very important where data access speed is important consideration. This architecture mainly includes API service, scheduler service and volume service. Its pluggable driver provides the facility to create and manage block storage devices.
4) **Neutron:** This project provides networking capability as a service for other OpenStack services. It facilitates to ensure that all components within the OpenStack deployment, can communicate with each other efficiently. It provides an API for clients to define networks and the attachments. It also has a pluggable architecture that supports networking vendors and technologies.
5) **Horizon:** Horizon is the implementation of dashboard service. It provides a web based user with self-service portal to interact with underlying OpenStack services like Nova, Swift, Keystone etc. It is built in two parts: 1. a set of libraries that implements dashboard; 2. a reference dashboard implementation that uses these libraries. Users can access all OpenStack components through an application programming interface. Dashboard also helps the system administrators to keep track what is going inside cloud and how to manage it.
6) **Keystone:** Keystone provides authentication and authorization services for OpenStack. This is the central list of all the users, represented against the services provided by cloud environment which users have permission to use. Developers can have multiple ways of access so that mapping of user access methods versus keystone is easy.
7) **Glance:** This project provides a service where clients can transfer and describe data assets that are intended to use with other OpenStack services. This includes metadata definitions and image services. It also stores and regains virtual machine disk images. Compute node makes use of this service during virtual machine provisioning.
8) **Ceilometer:** The fundamental goal of this project has the capacity to meter the performance and utilization of cloud environment. It supervises and measures the OpenStack cloud for billing, scalability, benchmarking and statistical purpose [9]. This framework is extensible to accumulate the usage of other needs.

III. System Design

The proposed system architecture is as shown below. The load balancer is configured for one of the VM's in openstack and it will manage and maintain the load balancing, when it gets more number of requests. The fundamental objective of this mission is to give an extensible structure design to equalize the load among virtual machines; to reduce the load traffic and to minimize the time. This proposed system uses neutron as the basic component, which provides the service called load balancing for controlling and managing the traffic among serval VMs, using Octavia [11] as a reference model. Octavia is one of the types of Load Balancer introduced in the Liberty version of OpenStack.



Architecture of Proposed System [12]

The above Figure displays the deployment of Load Balancing frame work with the openstack clouds. This framework provides an infrastructure, needed for monitoring the traffic among the VM's and to manage the loads equally to all virtual machines, to get the best working condition. The environment we are using is IaaS and Load Balancing algorithm to solve the issues. In Octavia Amphora behaves like a load balancer and all other components like driver, controller is connected to each other via amphorae. There are three noteworthy segments to a load balancer in Neutron

1. **Pool member(s):** A pool member is a layer 4 question and is made out of the IP location of an administration and the listening port of the administration. For instance, a pool part may be a web server with an arranged IP address, 10.30.0.2, listening on TCP port 80.
2. **Pool(s):** A pool is a gathering of pool individuals that commonly serve indistinguishable substance.
3. **Virtual IP(s):** A virtual IP, or VIP, is an IP address that dwells on the heap balancer and listens for approaching associations. The heap balancer then adjusts customer associations among the individuals from the related pool. A virtual IP is typically presented to the Internet and is frequently mapped to a space name.
   - **Load balancer [6]:** The Cloud Load Balancers administration incorporates a wellbeing observing operation that keeps load balancer working easily by directing movement just to hubs that are working legitimately.
   - **Listener [7]:** It is a procedure that checks for association demands. It is designed with a convention and a port for front-end (customer to load balancer) associations, and a convention and a port for back-end (load balancer to backend example) associations.
   - **Pool [8]:** A load adjusting pool is a legitimate arrangement of gadgets, for example, web servers, that you aggregate together to get and handle activity.
   - **Member [9]:** A Member is an IP address and port combination tied to a particular Node.
   - **Health monitor [10]:** The health monitor periodically checks the health of each node associated with load balancer, including new nodes that are added. If the health monitor detects a node that is not responding, the node is removed from the load balancer rotation until the health monitor determines that the node is functional.

The components of the Octavia are as described below:

Driver: This is the part of the load adjusting administration that really interfaces between the (cleaned) client and administrator setup and the back-end load adjusting machines or other "administration providing element".

Administrator API Handler: This is precisely similar to the User API Handler in capacity, with the exception of that usage subtle elements are presented to the administrator, and certain administrator level elements are uncovered (ex. posting a given inhabitant's load balancers, and so forth.)

Controller: This is the segment giving all the summon and control for the amphorae. Toward the front, it takes its summons and controls from the LBaaS driver. It ought to be noticed that in later arrivals of Octavia, the controller capacities will be part of a few segments. At this stage, we are less worried about how this inside communication will happen, and are most worried about guaranteeing correspondence with amphorae, the amphora LB driver, and the Network driver are all made as flawless as could be expected under the circumstances.

Amphora Load Balancer (LB) Driver [11]: This is the reflection layer that the controller converses with for speaking with the amphorae an amphora LB driver additionally gives the administrator the capacity to have diverse open-source amphorae with possibly distinctive abilities (got to by means of various flavors) which can be convenient for, for instance, field-testing another amphora picture.

LB Network: This is the subnet that controllers will use to speak with amphorae. This implies controllers must have availability (either layer 2 or steered) to this subnet keeping in mind the end goal to capacity, and vice versa. Since amphorae will impact on it, this implies the system is not part of the "under cloud."

Amphorae: This is a Nova VM which really gives the heap adjusting administrations as designed by the client. Obligations of these elements include: Really fulfilling the heap adjusting administrations for client designed load balancers utilizing HA proxy.

IV Conclusion

In this work, we have presented the design of source implementation of LBaaS for monitoring the incoming loads and it uses the specific algorithms to balance the loads among the virtual machines. Load adjusting as an administration furnishes occupants with the capacity to scale their application automatically through the Neutron API. Clients can adjust

activity to pools, comprising of numerous application servers and can give high accessibility of their application using clever wellbeing screens. The proposed structure is straight forward and flexible for the openstack-Liberty version. The project outcome shows this model is best suited for all types of test cases like functional, data driven and it has the capacity to minimize the load imbalance, and to provide high availability and L7 content Switching.

REFERENCES

[1] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and others, “A view of cloud computing,” Communications of the ACM, vol. 53, pp. 50–58, 2010.

[2] Gulshan Soni, Mala Kalra. “Comparative study of live virtual machine migration techniques in cloud”, vol. 84-No 14. December 2013.

[3] http://www.openstack.org/software/

[4] Networking- this can be found at https://developer.rackspace.com/blog/neutronnetworking-the-building-blocks-of-an-openstackcloud/

[5] openstack components can be found at www.openstack.architecture

[6] The explanation of this can be seen in technet.microsoft.com

[7] The documentation of listener can be seen at docs.aws.amazon.com

[8] The explanation with respect to Pool of LB can be found ataccess.redhat.com/documentation

[9] This document can be found at access.redhat.com/documentation

[10] http://docs.openstack.org/liberty/networkingguide/adv-configlbaas.html

[11] The explanation of Octavia can be found at http://www.octavia.io/review/master/design/version0.5/component-design.html

[12] The proposed architecture is adopted from LBaaS integration and Octavia Components.