ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

IJARCCE

International Journal of Advanced Research in Computer and Communication Engineering

ICRITCSA

M S Ramaiah Institute of Technology, Bangalore

Vol. 5, Special Issue 2, October 2016

Openstack Architecture Design and Scalability Principles: An Overview

Pallavi V Patil[1], Dr Jagannatha S[2], Balaji B S[3]

Assistant Professor, Department of Computer Science and IT, Jain University, Bangalore, India[1]

Associate Professor, Department of Computer Applications, MSRIT, Bangalore, India[2]

Student, Department of Computer Science and IT, Jain University, Bangalore, India [3]

Abstract: Openstack is an open source software platform for infrastructure provisioning in cloud and hence commonly deployed as Infrastructure as a Service (IaaS). Openstack Architecture has multiple flavours based on usage scenarios. Openstack has specific and coordinated components to manage hardware and storage pool which can be accessed through well define REST API endpoints, command line tool or Web UI. The main challenge in cloud offerings is to handle the growing demands for the infrastructure catering diverse needs. This results in dynamic infrastructure provisioning with efficient and robust scalability design. The main objective of Openstack cloud operator is to hide from user, the failure caused due to the resource limitation and to provision the required infrastructure adhering to Service Level Agreement (SLA). This paper gives in-depth understanding of Openstack design principles like Infrastructure segregation. Host Aggregates and Availability zones to achieve massive scalability in Openstack components and also discuss the architecture design of major Openstack components based on different usage scenarios.

Keywords: IaaS, Openstack nova, Openstack neutron, Host Aggregates, Openstack Heat.

I. INTRODUCTION

Cloud is a remote resource which is offered by a specific provider and consumed by the user on rental basis with adherence to SLA and consumer is charged based on the utilization of resource and hence Cloud computing is also called as Utility Computing. The cloud is classified as public, private, hybrid and community based on the location of resources. Public cloud is available to all users, similarly Private cloud is specific to organization hosting cloud, Hybrid cloud is a combination of both private as well as public for example to achieve load balancing one could transfer intensive peak loads to public while processing less intensive work on private cloud. The cloud offering can also be classified based on the kind of services provided like Software as a Service(SaaS), Infrastructure as a Service(IaaS) and Platform as a Service(PaaS). IaaS is applicable in scenarios where the cloud is used to deploy applications without adhering to any specific platform runtime like java runtime, .Net etc.. PaaS is an extension of IaaS with platform specific runtimes to host applications, for example IBM Bluemix Application with Node.js runtime or with Websphere runtime, Tomcat runtime associated with java build back etc., other providers include Amazon Web Services, Google, Microsoft Azure etc.. Hence these services are coarse grained services. The cloud services can be fine grained aiming on specific type of resource like Storage as a Service, Database as a Service etc. The coarse grained cloud service provider has to face real challenging situations when demands grows from users and it is necessary to have well define scalability principles to be used while designing the cloud architecture.

Openstack architecture design has distinct avatars based on the usage scenarios. The flavours include nova (compute focused), neutron (network focused), Openstack swift/cinder(storage focused), multi-site, hybrid and massively scalable. The user and operational requirements really drives such flavours of Openstack architecture. For general purpose cloud, the requirements are generic like database as a Service, A web application runtime or common application development platform etc, and hence Openstack general purpose cloud is ideally designed to cater 80% of use cases. The important requirements includes Cost which is minimized with maximized utilization, Time to market is obviously less when cloud infrastructure is used than building customized data center from scratch, Performance is a baseline requirement to satisfy user common considerations, Ad-hoc and self - service applications, the cloud should have ability to self-provision the resources required for user based on increasing demand in flexible way. The resources could be storage, network or can be a simple software, Security is the major concern for cloud consumers, but cloud computing works on trust basis but at least primary security considerations like authentication, authorization, confidentiality and Vulnerable free should be ensured. It is not good choice to go for general purpose cloud if security

ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

IJARCCE

International Journal of Advanced Research in Computer and Communication Engineering

ICRITCSA

M S Ramaiah Institute of Technology, Bangalore

Vol. 5, Special Issue 2, October 2016

has highest priority over other requirements. Precise Capacity planning plays important role to achieve good scalability.

Scalability is considered as one of the primary requirements for Coarse grained services like IaaS. Capacity Planning predicts the amount of resources required achieving efficient scalability and hence it also plays important role in finding Cap-Ex (Capital expenditure) and Op-Ex (Operational Expenditure). Openstack has well defined scalability ethics, where it provisions the infrastructure for users without disclosing the failure environments created due to the shortage of cloud resources. Massively scalable architecture results in heavy and large cloud deployments which incur cost for provider and can be affordable by commercial providers.

The term massive here could be used in the case like when there is request for 400 instances and currently, in real time such massive requests are not common. Like General purpose cloud massively scalable clouds are not driven by specific use cases and hence its challenging and serves as platform for heavy workload management. Since private organizations have seldom requirements for such massive scalability and therefore massively scalable Openstack cloud is built as commercial, public cloud offerings. It is required to automate as many as processes to achieve efficient scalability. Automation includes dynamic configuration of resource provisioning, monitoring and alerting system. Especially monitoring and alerting system gives signal to provider to increase the cloud space based on the new requirements. This mitigates the maintenance costs by reducing the human staff and which in turn minimizes capital expenditure as such massive labour intensive work is automated without manual intervention.

In massively scalable architecture, the replacement of the failure system is preferred over the diagnosis of error as it incur more human resource cost, because redundant and common tasks are automated in such environments. Because of automation, the human resources can be used instead for other productive tasks like capacity planning. In other perspective it is not preferred to replace failure system, for example if production environment goes down then it is not practical enough to create new system and build complete environment rather one solution would be to apply patches for failure components to resolve issues. This paper aims at understanding Openstack as massively scalable architecture and usage of such scalability principles across other flavours.

The rest of the paper is organized as follows, Section II briefly describes the scalability requirements, Section III describes the Openstack scalability principles. Section IV describes other flavours of Openstack architecture design and how scalability principles are incorporated in other flavours.

II. OPENSTACK SCALABILITY REQUIREMENTS

There are two main perspectives for requirements of massively scalable Openstack. Two key actors of cloud universe are Cloud Consumers and Cloud Providers where each one of them visualize cloud different from different angles and hence results in diverse class of requirements. Massively scalable cloud results in dynamic components allocation which results in additional stress and overhead for supporting infrastructure such as databases and messages brokers. Therefore architecture should be carefully designed to not negatively impact users' experience.

A. Cloud Consumer Requirements

Deterministic Process – Cloud Management Software, Consumer always expect that, there should exist deterministic, dependable process for managing and deploying cloud resources. Such a solution can be provided either through Web Based or well defined REST API endpoints for managing cloud systems.

Consumption as a Service defining on demand consumption model when massively scalable cloud reaches threshold. Such a consumption model helps in automatic discovery of cloud services used by consumer and provides complete usage statistics of consumed cloud resources, which further helps user to manage their clients' needs and requirements.

Compromise on security, performance or availability, it is common observation that consumers of massively scalable cloud architecture has less expectation on security, performance and availability rather they require robust API endpoint which always up and running with basics SLA offered. Such comprising factor does not exist in scenarios where massively cloud architecture is used for some private organization or government departments where security is major concerns.

Properties Abstraction of Underlying Infrastructure, users of scalable cloud architecture requires the automated and deterministic process for managing growing resources with no botheration on capacity, scalability or other characteristics of underlying infrastructure.

B. Cloud Provider Requirements

While users are non-transparent on underlying infrastructure but in contrast, the provider has primary responsibility for providing such underlying infrastructure which results in new requirements.

Full Fledged Automation, it is less overhead for provider if every components of cloud architecture is deployable automatically. Components include compute-hardware, storage hardware and network hardware. Automation here includes installation of components and configuration of hardware management software. Manual processes are not practical in massively scalable architecture.

ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

IJARCCE

International Journal of Advanced Research in Computer and Communication Engineering

ICRITCSA

M S Ramaiah Institute of Technology, Bangalore

Vol. 5, Special Issue 2, October 2016

Minimize CapEx, Capital expenditure includes the cost for third party supporting software require to support openstack cloud architecture. It is advisable to use open source software and customized as per individual requirements in every layer of stack, this reduces the deployment costs and operational expenses. Open Compute provides the information about finance management on cloud and ideas to build cost effective massively scalable architecture. Some ideas include removal of redundant power supplies, network connections and rack switches.

Minimize OpEx, Operational Expenditure costs of hardware to build data centers. It is recommended to use cloud optimized hardware. Some metrics include power, cooling and physical design of blade chassis. Comprising on such hardware cost is not advisable in all scenarios as hardware failure really affects cloud consumers and finally business. It is advisable to buy cheap and best hardware to avoid operational cost overhead.

Extensible Monitoring Capability, Massively scalable cloud architecture requires well designed resource monitoring, metering and alerting system which automatically creates warnings based on analysis result of monitoring system. This helps the provider to replace/diagnose failure components.

Legal and jurisdictional requirements, This kind of requirements evolve in multi-site architecture where cloud architecture is geographically spread into different regions. Also 3A (Authentication, Authorization and Auditing) requirements of multi-site has impact on scalability.

Consider the Physical Space constraint, Last but not least requirement consideration for provider in horizontal scaling is to be aware of physical space, floor weight, rack height and type and other environmental conditions like power usage and physical security.

III.OPENSTACK SCALABILITY PRINCIPLES

Openstack uses a principle of incremental and radical transformation of existing infrastructure for horizontal scaling. It is hard to deploy the massively scalable cloud installation for very first time from scratch, ensure that initial deployment is forward compatible w.r.t principles and choices that is used for further scaling. For example in multi-site scenario first build the initial site and while spreading site on different geographic locations make sure that the same scaling principles are used. In hyperscale cloud scenario the infrastructure seems to be redundant, the applications are carefully modified to avoid such redundancy and to energize reliability.

A. Infrastructure Segregation

Openstack supports massive horizontal scaling for certain specific infrastructure in particularly database management systems, message queues that openstack services use for data storage and Remote Procedure Call (RPC). Massively scalable cloud is extension of traditional clustering process where additional effort and care is required to mitigate the performance pressure on the components so that it does not impact the overall performance of cloud as a whole and eliminates the single component failure. Horizontal scaling is achieved by segregating independent and self-controlled installation which is accessed with Openstack Identity and dashboard (optional) installations. Such individual physical group is called Region. Every region has its own database management systems and message queue installation. In massively scalable domain it is important to hide the region level failures and yet to provide the required infrastructure to users. This mandates further division of regions into physical groups called Cells as shown in Fig 1[6]. Every cell is associated with its own message queues, database, scheduler, conductor and multiple compute hosts. Even though there exists only single Cell API endpoint for region which handles user requests and diverge them to cell scheduler. The cell scheduler looks for available cell and submits the requests, Filter scheduler then finds the appropriate compute hosts for handling user requests with in cell.. Because of solely independents cells the load is further balanced and shared among the cells of region. There are some drawbacks exists with this principle, Such Regions and cell concepts works well only with compute focused architecture and other flavors. Moreover regions do not support some of the standard and baseline functionality such as security groups and host aggregates. The cell concept is relatively new and has less impact in openstack world. They have been used in CERN and Rackspace[11-12] clouds.

B. Availability Zones

The Availability zone is part of cell or regions which allow further division of regions and cell into physical group installations. Usually Availability zones are formed based on the shared physical characteristics of the nodes like shared power source and physical network connections. Close proximal groups are created because of such shared physical properties.

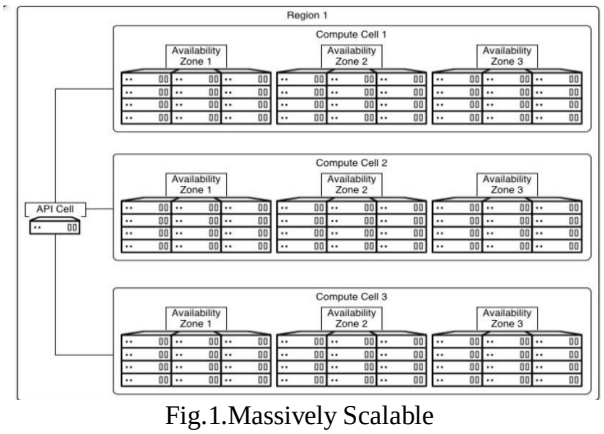

Fig.1.Massively Scalable

ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

IJARCCE

International Journal of Advanced Research in Computer and Communication Engineering

ICRITCSA

M S Ramaiah Institute of Technology, Bangalore

Vol. 5, Special Issue 2, October 2016

May be blade servers with in single rack enclosure can form one availability zone or multiple racks which are close to each other can form one availability zone and hence it is totally in the control of operator. Even if one availability zone fails then the requests are transferred and processed by other zones and such a decision is based on the scheduler. The scheduler itself can be hosted on one of the zone called default Availability Zone. Availability zones do not possess their own database or message queues therefore it represents the arbitrary grouping of compute nodes.An availability zone on each compute node is created by adding a line node_availability_zone=availability_zone_name in /etc/nova/nova.conf then restart corresponding compute services service openstack-nova-api restart, service openstack-nova-compute restart. Availability zones can also be deleted using reverse process. Make sure no Virtual Servers are in running state and uncomment line from nova.conf and restart the same services. Existing Availability zones are listed using nova availability-zone-list.

C. Host Aggregates

As the name itself indicates that its groups the compute hosts. Here the group is the logical group of hosts to achieve load balancing and instance distribution. The host aggregate can be used to further partition the availability zones. The grouping criteria is based on the similarity in shared resources accessed by the compute hosts. The resources could be storage, network, API services, databases so on and so forth. To create such group one can define the group metadata and configure the hosts based on this metadata. The metadata is stored as key/value pair typically JSON. The advantage of such metadata is to handle cloud instance capability management. The host aggregate can be used to define capability requirements and serialize as metadata and tags the host group to this metadata. When there is a request to be served, then scheduler can choose the group of hosts of availability zone having capability to process the requests. When there is a request to shutdown server of such type then only hosts in this group is considered as candidate. Such metadata can be set using nova aggregate-set-metadata 1 <key=value>[ref]. Similarly aggregate-create <name> <zone_name> is used to create host aggregate <name> with zone <zone_name>, nova aggregate-add-host <id> <host> is used to add <host> of zone to host aggregate with host identifier <id>. The API returns the availability zone separately from the general list of metadata, though, as it is a special piece of metadata. Some standard defined keys such as cpu_allocation_ration, ram_allocation_ratio can be used in to aggregate host with equal CPU and RAM capacity. One should be careful and cautious while setting such properties especially when hosts belongs to multiple groups. The flavour types from different hosts can be created using extra_specs for example if extra_specs cpu_allocation ratio 2 then aggregation is performed with hosts having cpu_allocation_ration set to 2. All these services runs in single availability zone and hence host aggregates are possible within zone limit. Hardware procurement and capacity planning plays important role in scalability as deployment is planned out ahead of time. Based on the overcommit ratio one can determine the requirements of Virtual servers(VS) and physical servers hosting VS in well advance.

IV. OTHER FLAVORS OF OPENSTACK ARCHITECTURE

As mentioned before based on use case there are diverse flavours of openstack architecture exists. The flavours are General purpose, Compute focused, Network focused, Storage Focused, Multi Site and Hybrid. The usage of architecture components are flexible and end up in different flavours. The scalability principles are generic principles applied across all the flavours of openstack architecture especially in compute focused architecture.

A. Compute Focused(Openstack Nova)

Compute focused architecture supports computation intensive workloads. The main resources such as RAM, CPU are intensive with minimal requirements on storage and network. The Use cases includes Big data analytics, Platform as a Service, High Performance Computing etc.. Capacity planning has two aspects to consider which plays important role for efficient scalability. The planning for initial deployment footprint and expansion of the environment to forecast the user growing requirements.

The Overcommit ratio is the ratio of number of VCpus to host CPU. The number of expected instances is (Overcommit ratio* cores)/virtual core pere instance. Number of instances times the disk size gives the estimation of storage capacity and hence these ratios plays important role for scalability. Openstack Compute uses 16:1 [6] CPU allocation ratio and 1.5:1 RAM allocation ratio. The components of Openstack compute focused includes Openstack nova, Openstack Image Service(glance), Openstack Identity(keystone), Orchestration(heat) and Telemetry. Openstack heat and telemetry plays important role in autoscaling of infrastructure based on monitoring, metering and alerting systems. Openstack cinder, a block storage service and Openstack neutron is used for network management.

B. Storage Focused

Openstack provides storage in blocks called cinder or basic blob storage called swift. The storage is spawned across SAN Storage Area network, when user requests for storage LUN is created from array of physical disks and the HBA ports of server and Storage controller is mapped to form a zone and after presenting this LUN the storage is formatted for use. Every logical disk i.e LUN can be further used for swift or block storage. Scaling storage

ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

IJARCCE

International Journal of Advanced Research in Computer and Communication Engineering

ICRITCSA

M S Ramaiah Institute of Technology, Bangalore

Vol. 5, Special Issue 2, October 2016

services provides effective way to increase the storage capacity as compared with Block and Object storage services. Adding block storage is fairly simple but to increase the capacity of storage as a hole and bandwidth is challenging.

Scaling Block Storage

The group of storage nodes forms the storage pool. New block storage node can be added to pool without disturbing the existing storage services. The node is installed with required hardware/software configured and this node advertises its presence to scheuler. After registering to scheduler the user requests are transferred to storage node for storage space instantiation. This is the story of capacity planning but it is not just sufficient if storage is created it is also required to scale the network bandwidth to support such capacity growth. This requires dynamic routing protocols and both frontend and back-end network design should encompass the ability to add capacity and bandwidth, quickly and easily.

Scaling Object Storage

Scaling of object storage depends on the partition power of storage service. The partition power determines the number of partitions which can exists. The partition cannot span more than single disk but vice versa is true. The disk with partition power of 4 can have 2^4=16 partitions. When new disk is added the partitions spread and hence each disk can have 8 partitions hence scalability is directly proportional to partition power. As more users add storage capacity to back end system the network bandwidth should also be increased to match the access speed as before or even more. The storage is exposed always as proxy object and hence scalability also required at this level.

C. Network Focused(OpenStack Neutron)

The use cases for network focused architecture includes High Speed Content Delivery, means streaming video, photographs images or accessing distributed cloud storage. Important factors of Network as a Service are bandwidth, latency ,congestion and jitter. The user expects high speed network with low latency. Support of Network management functions, like DNS, NTP or SNMP etc.. Web Application, Web servers hosting web application requires descent bandwidth to serve web pages. Other requirements include Processing of Big data , VOIP and Video/Web Conferencing. Cloud Networks requires proper management on capacity and growth overtime. Capacity planning implies prediction of required network circuits and hardware. Such estimation helps in dynamic scaling of network capacity. There are two variations in openstack neutron Layer 2(Data Link Layer) and Layer 3(Network layer and above) networking. In Layer 3 scalability depends on number of compute nodes i.e if compute nodes increases the corresponding Virtual routers increases. It's hard to achieve such scalability from neuton but with the help of OpenContrail, Layer 3 overlay is created using a vRouter in the kernel corresponding to each of the compute nodes. The policies specified are centrally executed at the Controller and enforced in a distributed fashion within the vRouter .

V. MULTI-SITE

Openstack efficiently manages group of sites which are geographically apart as a single cloud. Use cases include organization with different geographical footprints and location specific sensitive data which also adheres to data locality. The information is fetched from the nearest site. The multi-site deployment is basically improves the availability of resources. It's important to manage the replication of data to avoid loss. The template with the feature of autoscaling heat template is used to deploy the application in three different regions. Web Servers, which uses apache for executing relevant user data to populate the central DNS servers, uses Instance launch and Telemetry alarms that maintains status of the application and handles Instance failure. Orchestration is used because of the in-built support of auto scaling and auto healing in the event of increased load. The capacity limit of multi-site architecture is controlled using Quotas where Quota sets the operational limit which prevents system capacities from being exhausted without prior notification and these Quotas corresponding to a per region basis.

VI. CONCLUSIONS

Infrastructure as a Service cloud offerings relieves the user from infrastructure management rather they can focus on computation tasks. Openstack provides the software platform and supports IaaS. The scalability concepts like Infrastructure segregation, Host aggregates and Availability zones is proof of massively scalable capability of openstack. Its is evident from studies that openstack nova supports most of scalability principles whereas neutron has limited support for massive scalability.

The openstack storage provides the scalability at both object and block storage levels. In holistic view the openstack has good support for scalability.

REFERENCES

[1]. Cruzes, D., Jaatun, M.G.: Cloud Provider Transparency - A View from Cloud Customers. Proceedings of the 5th International Conference on Cloud Computing and Services Science.

[2]. Xiao, L.Y., Wang, Z., Wang, R., Wang, H.N.: Architecture and Key Technologies of Cloud Computing. AMR Advanced Materials Research. 1953–1956.

[3]. Genez, T.A.L., Bittencourt, L.F., Madeira, E.R.M.: Workflow scheduling for SaaS / PaaS cloud providers considering two SLA levels. 2012 IEEE Network Operations and Management Symposium.

[4]. Breiter, G., Behrendt, M., Gupta, M., Moser, S.D., Schulze, R., Sippli, I., Spatzier, T.: Software defined environments based on TOSCA in IBM cloud implementations. IBM Journal of Research and Development IBM J. Res. & Dev.

ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

IJARCCE

International Journal of Advanced Research in Computer and Communication Engineering

ICRITCSA

M S Ramaiah Institute of Technology, Bangalore

Vol. 5, Special Issue 2, October 2016

[5]. IBM Bluemix - Create, Deploy, Manage Your Applications in the Cloud, http://www.ibm.com/cloud-computing/bluemix/.
[6]. OpenStack Architecture Design Guide, http://docs.openstack.org/arch- design/.
[7]. Openstack Operations Guide, http://docs.openstack.org/openstack- ops/content/index.html.
[8]. Sefraoui, O., Aissaoui, M., Eleuldj, M.: OpenStack: Toward an Open-source Solution for Cloud Computing. International Journal of Computer Applications IJCA. 38–42 (2012).
[9]. Li, L., Chou, W.: Design and Describe REST API without Violating REST: A Petri Net Based Approach. 2011 IEEE International Conference on Web Services.
[10]. Frachtenberg, E.: Holistic Data Center Design in the Open Compute Project. Computer. 83–85.
[11]. Andrade, P., Bell, T., Eldik, J.V., Mccance, G., Panzer-Steindel, B., Santos, M.C.D. and S.T., Schwickerath, U.: Review of CERN Data Centre Infrastructure. J. Phys.: Conf. Ser. Journal of Physics: Conference Series. 042002–042002 (2012).
[12]. Rackspace and CERN openlab Collaborate to Deliver, http://blog.rackspace.com/newsarticles/rackspace-and-cern-openlab-collaborate-to-deliver-big-bang-with-hybrid-cloud/.
[13]. Availability Zones and Host Aggregates in OpenStack Compute (Nova), http://blog.russellbryant.net/2013/05/21/availability-zones-and-host-aggregates-in-openstack-compute-nova/.
[14]. Woods, L., Eguro, K.: Groundhog - A Serial ATA Host Bus Adapter (HBA) for FPGAs. 2012 IEEE 20th International Symposium on Field-Programmable Custom Computing Machines.
[15]. Lee, N.-K., Han, T.-D., Kim, S.-D., Yang, S.-B.: High performance RAID system by using dual head disk structure. Proceedings High Performance Computing on the Information Superhighway. HPC Asia '97.
[16]. Block level storage vs. file level storage: A comparison - TechRepublic, http://www.techrepublic.com/blog/the-enterprise-cloud/block-level-storage-vs-file-level-storage-a-comparison/.
[17]. Factor, M., Meth, K., Naor, D., Rodeh, O., Satran, J.: Object Storage: The Future Building Block for Storage Systems A Position Paper. 2005 IEEE International Symposium on Mass Storage Systems and Technology.
[18]. Egevang, K., Francis, P.: The IP Network Address Translator (NAT).
[19]. OpenStack Neutron at scale, http://www.opencontrail. org/openstack-neutron-at-scale/.