Building Multi-Site & Ultra-Large Scale Cloud with Openstack Cascading

Requirement and driving forces multi-site cloud

Along with the increasing popularity and wide adoption of Openstack as the de facto standard for public and private cloud construction, starting from the very first release in 2011 through to latest Juno version, Openstack software has been continuously improved by the open source community in terms of both functionality enrichment and architecture optimization.

But there are still lots of challenges when building multi-site OpenStack based cloud.

Challenge 1: Unified Cloud service and resource scheduling across multi-site.

For certain customers such as large Enterprises and Telecom carriers, due to the existence of lots of branch offices and Service PoP with geographical proximity to end user access, and due to the limited bandwidth availability, each branch office and Service PoP need to have its corresponding DC sites, and unified Cloud service and resource scheduling across these multiple DC sites will be required. The multi-site scheduling will not only includes computing resources, but also for storage and networking resources. And the tenant also has the requirement for virtual resources to be distributed in multi-site but inter-connected by isolated L2/L3 networking. For the eco-system friendly, the unified cloud service must provide OpenStack API.

Challenge 2: Always workable and manageable cloud services.

As the distributed cloud grow larger and larger, to make the cloud itself always workable and manageable is a big challenge. It's unacceptable for any part of the cloud failure lead to the whole cloud failure, that means the fault must be isolated locally, no propagation. Even for daily maintenance activity and upgrade, if any configuration change, trouble shooting, bug fix, patch update, upgrade has to be done on the whole cloud level, it'll become big challenge even for most skilled engineer. These operation and maintenance activity must be able to be done on small part of the cloud, and then applied to other part of the cloud gradually, even last several month.

Challenge 3: Scalability in multi-site

- It's a big challenge for a single OpenStack to manage ultra large scale cloud for example 1 million VMs. Compute nodes resource track, scheduling, L2 population, DVR(distributed virtual router) route update, too many nodes involved in RPC message interaction, DB CRUD performance, etc. Even if one OpenStack instance can be scaled to million level VMs' cloud, one OpenStack instance itself across data

centers leads to lot of challenges. For example, how to solve the issues for cross data center RPC message communication. If message bus failed, API servers may be alive, but all compute nodes / L2 agents / L3 agents become out of manageable status. Great challenge to do the cross data center integration for multi-vendor hardware/software via RPC.

Challenge 4: co-existence of multi-Vendor OpenStack distribution, multi-OpenStack instance, multi-OpenStack version inside a distributed multi-site cloud.

For certain customers such as large Enterprises and Telecom carriers, branch offices and service PoP also have long standing relationships to their own choice of vendors. Each site can use one or multiple vendors which lead to better use of local resources and capabilities. Each site with its own pace and requirements on building, upgrading and maintenance based on standard OpenStack APIs. Therefore fast and efficient integration of multi-Vendor OpenStack distribution, multi-OpenStack instance, and multi-OpenStack version is a basic requirement.

-

Limited space for innovation, because OpenStack is already there

The challenges are there, and OpenStack also is there. Not only OpenStack community would not accept a revolution on the architecture, but also the eco-system from backend vendor to cloud operator would not allow this to happen. Obviously it's impossible to reinvent a new architecture for OpenStack to solve these challenges; the only way to solve the challenge is how it could be done under current architecture:

- Never break current OpenStack architecture
- Single standard OpenStack API endpoint, still work like one OpenStack, for ecosystem friendly purpose.
- Each site must be able to work standalone with restful API exposed or CLI provided, so that it's always workable and manageable standalone. And restful API is much better than RPC message to work in geographically distribution circumstance, not mention to sometimes the communication will be across public internet.
- At least each site could be built by different vendor, to meet multi-vendor business policy but also reduce the cross vendor integration effort at the same time. Interoperation and integration among multi-vendors' infrastructure always means time consumption and endless torture, configuration change, update, reboot… The smaller granularity of the integration unit is, the worse the situation become.
- Each distributed site could be integrated and hidden by the unified cloud service. And for the fast integration purpose, the restful API exposed by each site must be stable and standard API.

When we looked at these challenges and solution options, there is not too much space for innovation.

Basic design idea of OpenStack Cascading

Since OpenStack itself is designed to manage and schedule the computer cluster, why not treat OpenStack itself as one huge computer?

Currently the community support remote clustered hyper-visors, like vCenter, Ironic running under Nova. But it only works under Nova. Can this mechanism also work for Cinder, Neutron, Ceilometer, even Glance?

The question is easy to be answered. Just look at each OpenStack service's architecture, it is typically consisted of API-server, message bus, DB, distributed nodes with different backend supported.

Now, we have a clearer picture.

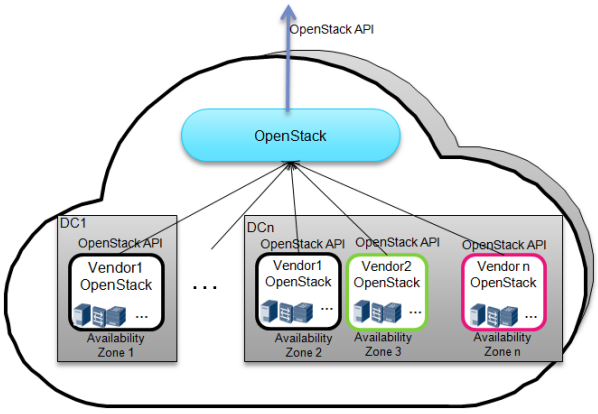
Usually, OpenStack is to manage underlying nova-compute (KVM/Xen/…), cinder-volume (Ceph/LVM/…), L2/L3 agent (OVS/LinuxBridge/Router/…), image store (Ceph/S3/…)

We can leverage the current OpenStack driver/agent mechanism to replace nova-compute's hypervisor to Nova, cinder-volume's storage to Cinder, L2/L3 agent's network facility to Neutron, Glance's image location to Glance, Ceilometer's store to Ceilometer.

That means we can use current OpenStack architecture and mechanism to provide a way for OpenStack to manage and schedule many OpenStacks. And we call this pattern as "OpenStack cascading".

Thus, we can have a summary for OpenStack cascading:

- The parent OpenStack (the cascading OpenStack) exposes standard OpenStack API
- The parent OpenStack use OpenStack standard API to orchestrate many child OpenStacks;
- Each child OpenStack (the cascaded OpenStack) functions as Amazon like available zone, and is hidden by the parent OpenStack, and can be distributed in multi-sites.



The cascading OpenStack: providing API and scheduling, orchestration and networking of the cascaded OpenStacks.

The cascaded OpenStack: provisioning the VM, Volume and virtual networking resources

OpenStack cascading meets the demand of multi-site cloud.

1. The multi-site cloud is unified with OpenStack API exposed
The cascading OpenStack aggregate many cascaded OpenStacks cloud via standard OpenStack API, and expose one OpenStack API endpoint by the cascading OpenStack. All eco-system based on the OpenStack API could be retained. If a propriety cloud management system is used to integrate the multi-site clouds, then no OpenStack API will be exposed, and cannot leverage the power of the OpenStack API eco system to boost business demands.

2. Fault Isolation with availability zone granularity
Each cascaded OpenStack will work as Amazon like availability zone, and provide the CLI and restful API for management purpose. No matter which cascaded OpenStack failed, the fault will not propagated to other cascaded OpenStacks and the cascading OpenStack. Even if the cascading OpenStack failed, all resources in the cascaded OpenStack are still running and being manageable thanks to the OpenStack API it has. This helps to build an always workable and manageable cloud.

3. Upgrade and daily operation and maintenance with availability zone granularity
The community has solid work to make the OpenStack API with backward compatibility and multi-version running in parallel before the version is deprecated. And the cascading OpenStack leverage the built in python client to interact with the cascaded OpenStack, it makes multiple version release of cascaded OpenStacks can run in one unified cloud and be integrated by the cascading OpenStack. Every cascaded OpenStack work standalone, and has no knowledge whether it has sibling cascaded OpenStack, whether it is running under cascading scenario. This mode makes the upgrade, and so far daily operation and maintenance, can be done with availability zone granularity, but not to touch the whole cloud at the same time.

4. Plug & play fast integration without vendor-lock in risk
As mentioned above, the backward compatibility and multiple-version supported by OpenStack API brings a lot of benefit for the distributed cloud integration through OpenStack API. Each vendor’s physical resources could delivered to the customer with built-in OpenStack, that means the boring integration like parameter tuning, configuration, patches and so on could be done before the delivery. And there is also clear responsibility for the part of the cloud to be integrated must be workable in premise. Then the integration will be greatly simplified to calling stable and standard OpenStack API with the access endpoint configuration, no matter which vendor provides the cloud infrastructure. Thus, plug & play fast integration mode without any vendor-lock in risk could be achieved by OpenStack cascading.

5. Scale out architecture leads to horizontally scalability even cross multi-sites

The cloud is unified through the OpenStack restful API by the cascading OpenStack. The restful API makes it feasible to integrated cascaded OpenStack within LAN or across WAN, especially suitable for geographically distributed multi-site integration.

Typical deployment scenario of Openstack Cascading

One data center will be deployed with one or several cascaded OpenStacks, eg,one or several availability zones in one data centers.
One cascading OpenStack can manage this one or several data centers.

Ultra large scale cloud is feasible

The cascading OpenStack only needs to manage max dozens of proxy nodes; the real provisioning of all running resources like VM, Volume, and Network is happened in the cascaded OpenStacks. Suppose that the cascading OpenStack manage 100 cascaded OpenStacks, and each cascaded OpenStack manage 10k VMs, then it's easy to implemented a million level cloud distributed in many data centers. The challenge is much smaller than a single OpenStack instance cloud including 1 million VMs, where thousands of compute nodes to be managed.

Typical application scenario of OpenStack Cascading

All app using OpenStack API can be seamlessly used for the cloud with OpenStack Cascading. And cross data center capability could be introduced because availability zone spread in geographically distributed data centers in nature.

Scenario 1: Isolated virtual data center across different physical data centers.
Big cloud operator wants to create a virtual data center over distributed physical data centers, the virtual data center includes secure and isolated tenant resources: VM, Volume, and cross data center L2/L3 networking with advanced service like FW,LB,VPN. OpenStack cascading can provide virtual isolated L2/L3 network plus VMs,Volumes across different physical data centers to the end user through the standard OpenStack api

Scenario 2: Cross DC auto-scaling / load balance / disaster recovery
The OpenStack cascading can provide global L2 VxLAN networking (or locality VLAN networking) and L3 DVR networking across multiple cascaded OpenStacks located in different data centers, relied on this networking capability, cross DC auto-scaling, load balance and disaster recovery can be achieved.

Scenario3: VM/Volume mobility across physical datacenters
The VM/Volume can be migrated from one data center to another data center: create a VM/Volume snapshot from availability zone located in one data center, and then create the VM/Volume in another availability zone located in another data center according to the snapshot in the Glance. If VM is attached to a global L2 VxLAN network, the IP/mac can

be kept unchanged. If there is storage assisted cross data center replication, the migration time could be reduced to seconds or minutes.

Want to learn more about OpenStack cascading, please refer to:

[1] wiki: https://wiki.openstack.org/wiki/OpenStack_cascading_solution