# Precentor 0.7:
# An Open Cloud Reference Architecture

**About this paper**:

This paper introduces the Precentor project, which is a path finding effort to understand how to build OpenStack based IaaS cloud on Intel Architecture. This paper describes our preliminary work based on OpenStack Grizzly release, including both the hardware and software reference architecture, as well as the tunings and optimizations that matter to scalability and performance. This paper gives a start point for those who want to build a similar cloud and need a reference point. For further information, please visit https://github.com/intel-cloud/Precentor.

Due to the limit of resource and knowledge, there must be many improvement areas or mistakes in the current work. We appreciate any feedback and expect more companies and individuals to work with us together to improve open source software quality and help accelerate the ecosystem ramp up.

**About the author**:

Cloud Infrastructure Technology (CIT) Lab is part of Intel’s Software and Services Group in Intel Asia-Pacific Research & Development Ltd, Shanghai China. We do evaluation, analysis, development and optimization for open source cloud technologies on Intel platforms. For more interesting stuff we are working on, please refer to https://github.com/intel-cloud.

## Table of Contents

## Introduction

OpenStack is gaining momentum as a prominent technology of private IaaS cloud. However, building an OpenStack based cloud is not a trivial work. At the software front, an OpenStack cloud actually involves many software components (most of them are open-sourced), and it requires considerable expertise to choose/configure/tune those software properly. At the hardware front, building a balanced hardware configuration also requires deep knowledge in the software and workload characteristics. Hence, it would be very useful to have some reference design available as a start point to build real IaaS clouds.

Precentor project is a path finding effort to showcase and improve OpenStack based IaaS cloud reference design on Intel Architecture. It consists of the recipes and optimizations we work out in Intel's laboratory, and is shared to the community for free.

In this paper we share the preliminary result of our work – Precentor 0.7, which is based on OpenStack Grizzly release and 20+ Xeon® based dual-processor servers. Instead of offering a complete solution with production quality, we focus on the scalability and performance of the core components. Due to the limit of resource and knowledge, there must be many improvement areas or mistakes in the current work. We will keep updating the documents when new learning or finding identified. Any comments will be appreciated. For more information please check our website at https://github.com/intel-cloud/Precentor, where you can find the contact information, mailing list, wiki pages and Q&A etc.

## Hardware Architecture

### Design Goal

The basic goal is to build a small scale (one to two racks) that can showcase what a *balanced* hardware configuration looks like. We set the number of compute nodes to 16, and then add the rest supporting nodes in proportional, to make the setup balanced.

### Compute Nodes

The compute nodes in this reference design are dual-processor Xeon® E5-2670 based standalone servers. Every server has 128 GB memory and 6 enterprise grade SATA HDD installed: 1 of them for host O/S and the rest 5 for local instance store in RAID5. Every compute node also needs to have two network ports: a 1Gb port for management network and a 10Gb port for data network.

### Storage Nodes

This reference design does not include Swift. Instead, Cinder volume service is referred as the storage service in this paper. The Cinder volume service in this design is made up of a dedicated storage nodes cluster. Each node in the cluster is a dual-processor Xeon® E5-2670 based standalone server which has 128GB memory, one 1Gb port, one 10Gb port, 20 enterprise grade SATA HDDs for data and 4 Intel DC S3700 SSDs for journal (not including disks for O/S).

## Supporting Nodes

Supporting nodes include one node for all the public-facing API servers except for Glance API server; one node for Image service (including Glance API server) which uses the storage cluster as its backend; and one dedicate network node for external L3 routing.

Table Summary of hardware configurations

| Role | Quantity | Spec | | | | | |
|---|---|---|---|---|---|---|---|
| | | Processor | 1 Gb Port | 10 Gb Port | O/S Disk | Data Disk (HDD*) | Data Disk (SSD**) |
| Compute Node | 16 | Dual processor Xeon® E5-2670 | 1 | 1 | 1 HDD | 5 HDD in RAID5 | 0 |
| Storage Node | 4 | | 1 | 1 | | 20 HDD | 4 SSD |
| API Node | 1 | | 2 | 0 | | 0 | |
| Image Node | 1 | | 1 | 2 | | 0 | 0 |
| Network Node | 1 | | 1 | 2 | | 0 | 0 |

* 7200 RPM 1TB enterprise grade SATA HDD

** Intel DC S3700

## Networking

Three networks are involved in this design. The first one is management network, every node needs to have a link to the management network and all of them are 1Gb links.

The second one is the data network, which is a pure 10Gb network. VMs access to each other and access to the external of the cloud via the data network. Image node distributes VM images to compute nodes via this network too. The volume data network also shares the same network.

The third one is the external network, where the public-facing API node has a 1Gb link connected. Besides, the image node needs to serve for image uploading and the network node needs to route external traffic, so both need a 10Gb link to the external network.

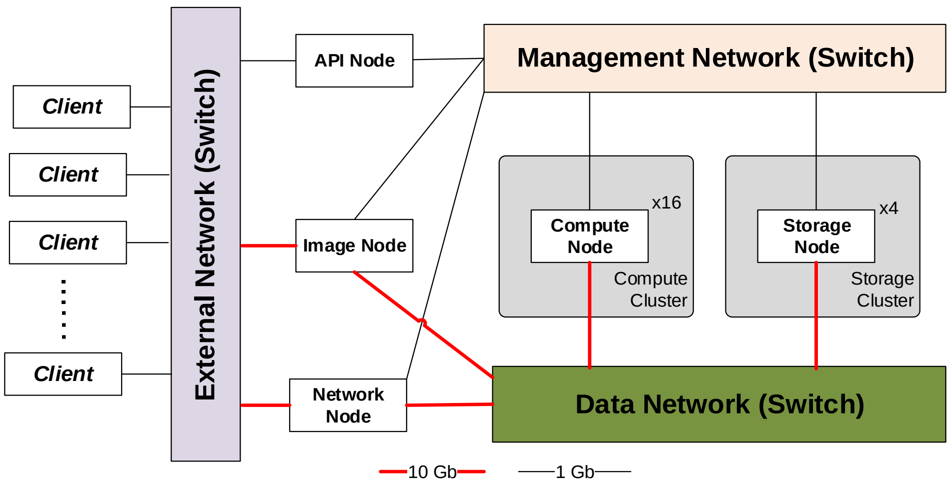

Figure Physical topology of the reference design

### Summary

The Figure  depicts the physical topology of the reference design.

## Software Architecture

### Software Components

The Table  below lists all the key software components used in this reference design. The following sections will go through the software configurations of each type of node.

Table  Software Components List

| | |
|---|---|
| Base O/S | CentOS 6.4 |
| OpenStack | 2013.1.2 (the 2nd Grizzly update release) |
| General: Database | mysql-server-5.1.69-1 |
| General: AMQP Broker | qpid-cpp-server-0.14-22 |
| Compute: Hypervisor | KVM |
| Compute: Host Kernel | 2.6.32-358.111.1.openstack (with ns enabled) |
| Compute: QEMU | qemu-kvm-0.12.1.2-2.355 (with Ceph RBD support enabled) |
| Compute: libvirt | libvirt-0.10.2-18 |
| Storage: Linux Kernel | 3.8.13 (upstream) |
| Storage: Ceph | Ceph 0.61 |
| Storage: Ceph librbd | librbd1-0.61 |
| Storage: Ceph librados | librados2-0.61 |
| Network: Open vSwitch | openvswitch-1.10.0-1 |
| Keystone Token Store | memcached-1.4.4-3 |

### Base Environment

We used CentOS 6.4 for all the systems, as it is a widely-used O/S in IPDC data centers and is also a clone of the Red Hat EL 6.4 which is widely used in enterprise data center. OpenStack 2013.1.2, which is the 2nd bug fix release for Grizzly, is used in this design, although it has been updated to 2013.1.3 recently. On CentOS, Red Hat RDO is available as a simple package of the vanilla OpenStack, so we just use RDO as the installation source.

### Compute Nodes

KVM is used as the hypervisor for compute nodes. The default CentOS 6.4 kernel needs to be upgraded to support namespace. The default QEMU in CentOS does not have Ceph RBD support, so a different version needs to be used.

The following services run on compute node:

- nova-compute
- quantum-openvswitch-agent
- openvswitch
- libvirtd

The Nova instance store on compute node is based on the file system created on the RAID5 of the 5 data disks.

## Storage Nodes

- Ceph is used as the converged storage backend for both volume service (Cinder) and image service (Glance). In this reference design we have a 4-node cluster running the following services:cinder-volume (on the first node only)
- ceph-mon (on the first node only)
- ceph-osd (on every node)

The cluster's 10Gb interfaces are configured in a separate subnets. All the Ceph daemons communicate with each other and with clients through this network, i.e. this is the shared network for Ceph Public Network and Ceph Cluster Network. See Figure  for network topology. In order to access the volumes in the storage cluster, the compute nodes' 10Gb interfaces need to have an IP address configured in the aforementioned subnet too.

Each storage node has 20 Ceph OSD daemons to manage the 20 data disks (HDD) respectively. The data disks are formatted in XFS file system. The 4 SSDs accommodate 20 partitions as the journals for the 20 data disks (5 for each). More tuning information is available in the Ceph Tunings section.

## Network Node

The network node has most of the Quantum services running, except for quantum-api which runs on the dedicated API node.

- quantum-dhcp-agent
- quantum-l3-agent
- quantum-openvswitch-agent
- openvswitch

## Image Node

The image node has both glance-api and glance-registry service running. Ceph RBD is used as the backend of the image store, so this node's 10 Gb interface that connects to the data network needs to have an IP address configured in the storage cluster's subnet.

- glance-api
- glance-registry

## API Node

The API node hosts all the public-facing API services except for the glance-api. It also hosts the Dashboard and some other supporting services including database and message queue services.

It is worth mentioning that the Keystone service use a dedicated memcached service as the token storage backend, for the performance consideration. More details about the tuning are available in the next chapter.

- nova-api
- nova-cert
- nova-scheduler
- nova-conductor
- keystone

- quantum-server
- cinder-api
- cinder-scheduler
- dashboard
- mysqld
- httpd
- qpidd
- memcached

## Tunings and Optimizations

### Basic MySQL Tuning

The default MySQL configuration is not enough to handle large load. While MySQL tuning itself requires a lot of domain expertise, there are low-hanging fruits that can help improve the MySQL performance with some simple changes. We recommend one change here: switch the MySQL engine to innodb (if it is not the default) and increase the buffer pools size to a larger number, say 2GB.

```
[mysqld]
default-storage-engine=innodb
innodb_buffer_pool_size = 2G
```

### Use memcached as Keystone token storage backend

By default Keystone stores everything in database, including tokens. However, as Keystone never removes the expired tokens from database, the token table's size would increase dramatically after some time. What's worse is that there is no index built on the token table so queries will get significantly slower as the table size getting larger (already fixed in Havana). As an alternative, memcached can be used as the backend for Keystone token storage, and any expired tokens would be removed by memcached automatically with its TTL feature. Below is an example of configuring Keystone to use memcached.

```
[token]
driver = keystone.token.backends.memcache.Token

[memcache]
servers = 127.0.0.1:11211
```

### Disable rootwrap scripts

The rootwrap scripts allow fine-grained control on the commands that OpenStack services need to run as root. However, it causes significant overhead due to its Python implementation. Disabling rootwrap can bring more than 5x performance boost in some cases. We recommend disabling rootwrap for quantum service at least. Two changes are required to disable it:

1. Set `root_helper=sudo` in `quantum.conf`
2. Add `quantum` user to `/etc/sudoers`

### Increase SQLAlchemy QueuePool size for quantum-server

The default SQLAlchemy QueuPool's size for quantum-server is 5, which is too small and would cause errors under heavy load. We recommend increasing it to 60 in ovs_quantum_plugin.ini:

```
sqlalchemy_pool_size=60
```

However, the above tuning knob is not available in the Grizzly releases. A patch 24986 has been merged into Havana to fix it, and back-port is required to make it work in Grizzly.

## Increase Quantum agents' state reporting interval and timeout threshold

Quantum agents report their UP state to quantum-server at a specified interval. Quantum-server determines an agent is DOWN if it hasn't received its state report after a specified threshold. Due to some reason the state report messages from agents may get jammed and eventually timed out and then be marked as DOWN. We increase the first interval to reduce the traffic rate between agents and quantum server and increase the second threshold to avoid bogus DOWN stated marked for agents. The changes are made in `/etc/quantum/quantum.conf`:

```
[DEFAULT]
agent_down_time = 300
[AGENT]
report_interval = 40
```

## Increase dhcp-lease-max

By default the dpch-lease-max is set to 150, which limits maximum number of IP addresses it can lease. It is recommended to change it to a large enough value.

In `/etc/quantum/quantum.conf` :

```
dnsmasq_config_file= /etc/dnsmasq.conf
```

In `/etc/dnsmasq.conf`:

```
dhcp-lease-max=1000
```

## Ceph Tunings

The key Ceph configurations are listed as below:

```
[osd]
osd mkfs type = xfs
osd mount options xfs = rw,noatime,inode64,logbsize=256k,delaylog
osd mkfs options xfs = -f -i size=2048
filestore max inline xattr size = 254
filestore max inline xattrs = 6
osd_op_threads=20
filestore_queue_max_ops=500
filestore_queue_committing_max_ops=5000
journal_max_write_entries=1000
journal_queue_max_ops=3000
objecter_inflight_ops=10240
filestore_queue_max_bytes=1048576000
filestore_queue_committing_max_bytes=1048576000
journal_max_write_bytes=1048576000
journal_queue_max_bytes=1048576000
ms_dispatch_throttle_bytes=1048576000
objecter_infilght_op_bytes=1048576000
filestore_max_sync_interval=10
filestore_flusher=false
filestore_flush_min=0
filestore_sync_flush=true
```

We applied the following Linux O/S tuning to the storage nodes that run Ceph OSDs.

```
echo "2048" > /sys/block/${device}/queue/read_ahead_kb
```

On compute nodes, we applied the following tunings to Ceph.conf:

```
objecter_infilght_op_bytes=1048576000
objecter_inflight_ops=10240
```

## Future Work

The current version of Precentor is just a start of our commitment to continuously build, optimize and share OpenStack cloud reference design on Intel architecture. There are numerous issues requiring resolutions in the next phase work, and we will continue to focus on scalability and performance as usual.

At the same time we will keep up with the evolving of OpenStack software and Intel hardware platforms. In the next phase we will rebase our reference design on the upcoming OpenStack Havana release and it is expected to see many issues will be fixed while new issues arise. Next generation Intel products will also be evaluated to reflect the stat of art performance.

Again, Precentor is a path finding project – intend to work with community together to improve open source software quality and help accelerate the ecosystem ramp up. We appreciate any feedback and expect more companies and individuals to work with us together.