TECHNICAL REPORT

Design Considerations for Using Nimble Storage with OpenStack
## Contents

OpenStack Introduction ............................................................................................................. 3  
Cloud Definition ...................................................................................................................... 3  
OpenStack Services .................................................................................................................. 4  
Nimble Storage Value in OpenStack .......................................................................................... 5  
OpenStack Block Storage Service (Cinder) Integration .............................................................. 5  
  Nimble Storage Cinder Driver Features .................................................................................. 6  
Design Considerations ............................................................................................................... 7  
  Foundation Design ................................................................................................................. 7  
  Storage Service Level Design Consideration .......................................................................... 8  
  Advanced Storage Service Design ........................................................................................11  
Appendix ...................................................................................................................................16
OpenStack Introduction

Cloud Definition
Based on National Institute of Standards and Technology (NIST) guidelines, a cloud computing platform is defined as a service offering with the following attributes:

- **On-demand self-service.** End users should be able to consume infrastructure resources through a self-service portal.

- **Broad network access.** The Platform should be able to orchestrate and consume network resources backed by traditional network virtualization or software defined networking.

- **Resource pooling.** The platform should be able to aggregate compute and storage into a logical resource pool.

- **Rapid elasticity.** The platform must be able to scale resource pools dynamically to meet end-user demands. Conversely, it should be able to free unused resources.

- **Measured Services.** The platform should have the ability to measure resource usage and provide ability to provide showback or chargeback based on consumption

All of these requirements can be addressed by the OpenStack cloud computing platform. OpenStack represents a global collaboration of developers and cloud computing technologists on a ubiquitous open source platform for public and private clouds. The project’s goal has been to deliver simple to implement, massively scalable, and feature rich solutions for all types of clouds. It includes a series of interrelated projects that have produced a variety of components for cloud infrastructure.
OpenStack Services

Openstack dashboard (Horizon): Web-based management portal for cloud administrators and users. It has built in role-based access control (RBAC), and multi-tenancy capability.

OpenStack Identity Service (Keystone): Identity service with two primary functions – user access management and permissions; and catalog of services with their respective API points.

OpenStack Image Service (Glance): Image repository for public or private cloud instance images or operating system installation ISO. Supports various formats including VMware Virtual Machine Disk (VMDK), Microsoft Virtual Hard Disk (VHD), Amazon Web Services (AWS) AMI, ISO, QCOW2

Openstack Compute (Nova): Compute resource abstraction supporting both physical hosts and various hypervisor technologies including KVM, Xen and VMware vSphere.

OpenStack Networking (Neutron): Network abstraction layer consuming physical, virtualization enabled and software defined networking technologies.

OpenStack Object Storage (Swift): Object-based storage backed by local direct attached storage (DAS) serving for eventually persistent storage and/or repository for glance ISO, instance images.

OpenStack Block Storage (Cinder): Block-based storage based on iSCSI protocol, providing persistent, high performance storage for cloud instances.

Orchestration module (Heat): “Infrastructure as code” template based orchestration tool for rapid provisioning of instances or application stacks.

Telemetry module (Ceilometer): Resource consumption metering and chargeback service.

More information about OpenStack services can be found in the OpenStack Administration Guide.
Nimble Storage Value in OpenStack

1. Nimble volume can back the Image Service repository to accelerate read and write performance for ephemeral instances. Nimble space efficient snapshot and replication can be used to backup important cloud images.

2. OpenStack Block Storage (Cinder) integration enables Nimble to serve as the backend for instance persistent block storage. Snapshot and clone operations offload

3. OpenStack controller and database nodes could be backed by Nimble volumes, thereby providing flexibility in retaining working state, for simple recovery due to misconfiguration or OpenStack upgrade failure.

OpenStack Block Storage Service (Cinder) Integration
Nimble fully integrates with OpenStack Block Storage Service, through the Cinder driver, providing persistent block storage for cloud instances.
Nimble Storage Cinder Driver Features

**Volume Operations**
- Volume Create/Delete
- Volume Attach/Detach
- Snapshot Create/Delete
- Create Volume from Snapshot
- Get Volume Stats
- Copy Image to Volume
- Copy Volume to Image
- Clone Volume
- Extend Volume

**Volume Stats/Attributes**
- Driver version
- Free Capacity
- Reservation (%)
- Total Capacity
- Volume Backend
Multi-Tenancy

- Storage data traffic separation on a per tenant basis with 802.1Q VLAN encapsulation
- Storage pool isolation through Cinder Type

Design Considerations

Foundation Design
To build a highly available, highly scalable OpenStack cloud platform with high performance, it is imperative to build a solid foundation for the infrastructure components. OpenStack Controller is the heart of the cloud platform, responsible for all user and application resource requests, and orchestrating communications to all shared services. It works with a backend database for storing all resource requests, resource states and all platform level metadata information.

OpenStack Controller
A minimum of two OpenStack Controller nodes are recommended so that a network load balancer can redirect requests to available nodes in the event of node failure. Nimble space efficient snapshots can be leveraged to provide additional layer of protection, and rapid recovery for controller nodes.

- Controller node backed by a physical host can have its OS and controller services run on a Nimble volume, through a Boot-from-SAN configuration.
- Controller node backed by a virtual machine can have its OS and controller services on the hypervisor clustered file system backed by a Nimble volume (for example, a VMDK on VMFS data store backed by Nimble volume).

MySQL Database
To avoid a single point of failure for MySQL backend database, several approaches can be used:

- Active/standby configuration with synchronous replication
- Active/active configuration with synchronous replication

Active/standby approach is recommended to prevent a database deadlock when multiple nodes are trying to perform writes at the same time. Follow the Nimble and Oracle’s joint Best Practices Guide when designing MySQL database.
Nimble Volume Collection Recommendation

Controller and database volumes should be placed on the same Nimble Volume Collection, so the built-in snapshot schedule or manual snapshots can be taken for all volumes simultaneously. The advantages of this approach are:

- The ability to capture working state of the entire OpenStack infrastructure, enabling the entire platform can be rolled back to a known working state in the event of upgrade failures or service disruption due to misconfiguration.
- The volume backing either the controller or database can be provisioned to new or replacement server node for immediate service recovery in the event of node failure due to hardware issues, eliminating the need for reinstallation of services or databases.
- For upgrade preparation, a Nimble Zero Copy Clone can be instantiated to enable upgrade test. For example, controller node 2 can have its boot volume cloned from the most recent snapshot. The node can be configured to boot from the clone with new OpenStack release upgrades to test for compatibility and potential upgrade issues. Upgrade rollback can be done by simply booting the node with the original volume.
- Built-in Nimble WAN replication can provide disaster recovery protection for OpenStack cloud platform. In the event of datacenter planned or unplanned outage, replicated snapshot copies of the OpenStack environment can be provisioned instantaneously for business continuity.

Glance Image Repository Acceleration

By default, OpenStack Image Service (Glance) is configured to store instance images or ISO files on the local Operating System directory ("/var/lib/glance/images"). This simple design allows ephemeral instances to be booted straight from the image repository. Performance of
these types of instances depend largely on the storage backing the repository; additionally, the repository requires regular backup to ensure cloud instance base OS images, as well as custom images can be protected. Therefore, a Nimble volume should back up the Image Service repository. With a Nimble volume serving as the backend storage for Image Service, a large number of ephemeral instances can be booted rapidly, with acceleration for scratch space. Additionally, efficient Nimble snapshots allow for the frequent backup of all images.

Storage Service Level Design Consideration

After the infrastructure foundation has been designed, the next step is to configure storage service level offerings. Not all application and usage needs have the same requirements for storage performance, capacity and data protection. OpenStack Cinder service provides great flexibility for storage service level design. The following is a reference model for designing a differentiated storage service offering, leveraging the multi-backend Cinder capability (a capability introduced since the Grizzly release of OpenStack).

Basic Tier:

Server nodes are typically equipped with local DAS storage. Such storage can serve as the basic tier of offering, suitable for usages that do not have large capacity or high performance requirements. The local DAS could be aggregated to create a LVM volume group, which in turn, backs Cinder through native LVM Cinder driver.

Premium (Shared):

For application requirements that have a high capacity and performance needs, the premium tier can be backed by the native Cinder volume driver, along with a LVM volume group backed up by a Nimble volume(s). It allows the ability to start small with one Nimble volume in the VG, and dynamically expand to span more multiple volumes as capacity needs grow. CASL leverages Nimble’s Adaptive Flash platform for read and write optimization using flash and high capacity disk spindles. With Nimble volume(s) backing the LVM volume group, high performance and capacity is built-in, with simple and non-disruptive expansion & scalability.
**High Performance (Dedicated):**

For the most performance demanding applications, a dedicated Cinder volume can be allocated through Nimble’s Cinder driver. In this tier, an instance can boot from a dedicated Cinder Nimble volume (for OS), and with one or more Cinder Nimble volumes attached for application data. Additionally, native Nimble CASL snapshots, clones and replication technologies plus InfoSight™ analytics reporting are applicable on individual instance level granularity. Given all the capabilities enabled through Nimble Cinder integration, this tier of offering can be categorized as the highest tier of storage service.

Here is a summary table capturing key use cases enabled by each storage service level, along with feature sets enabled:

<table>
<thead>
<tr>
<th>Service Tier</th>
<th>Cinder Storage Backing</th>
<th>Features</th>
</tr>
</thead>
<tbody>
<tr>
<td>Basic</td>
<td>VG with Local server DAS (LVM Cinder Driver)</td>
<td>• Shared storage pool</td>
</tr>
<tr>
<td>Premier (Shared)</td>
<td>VG with Nimble Storage Volume (LVM Cinder Driver)</td>
<td>• Data reduction (Nimble inline compression)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• LVM level snapshot</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• LVM level volume full copy</td>
</tr>
<tr>
<td>High Performance (Dedicated)</td>
<td>Nimble Storage Cinder Volume (Nimble Cinder Driver)</td>
<td>• Data reduction (Nimble inline compression)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Self-service instance snap, restore and clone</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Per instance volume level space efficient snapshot</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Rapid instance cloning with Nimble CASL Zero Copy Clone</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• InfoSight per instance level capacity and performance analytics and head room report</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Additional advanced capabilities described in “Advanced Storage Service Design” section</td>
</tr>
</tbody>
</table>
Advanced Storage Service Design

Nimble’s Cinder integration enables additional set of capabilities to further differentiate the highest tier of storage offering in providing value to end user applications.

- Storage Multi-Tenancy
- Multipathing
- Rapid Instance Cloning Offload

Storage Multi-Tenancy

Nimble’s iSCSI data interface supports 802.1Q VLAN tagging. Each data interface can be configured with a dedicated or shared subnet tied to a specific VLAN. This capability can be leveraged with the Nimble Cinder driver “subnet” option. The following are high level steps to create a dedicated data interface to serve traffic for a tenant:

Multipathing

Multipathing further improves performance for instance volumes, by enabling I/O operations across all available network interfaces from both the OpenStack Nova nodes and Nimble array. Multipathing can be configured for both LVM VG backed by a Nimble volume, or an individual Cinder Nimble volumes attached to instances. The following high level steps highlight the tasks required to enable multipathing in OpenStack environment:
Install device-mapper-multipath / multipath-tools

- /etc/multipath.conf
- /etc/nova/nova.conf (libvirt_iscsi_multipath=true)

Attach volume to instance

- Add additional iSCSI Connection
- Set target node.session.nr_sessions

*NOTE* For LVM VG backed by Nimble volume, step 3 can be skipped as multipathing operates on the VG itself; individual instances transparently inherit the benefits

**Rapid Instance Cloning Offload**

Instance level cloning can be offloaded to the Nimble array, leveraging Zero Copy Clone technology. Without using additional disk space, each clone provides the same performance, capacity and data protection capabilities. To enable this, the instance needs to boot from a Cinder Nimble volume, followed by invoking an instance volume snapshot to serve as the source for cloning.
Step 1: Create volume using cloud image as the source

Create Volume

Volume Name *
vol1

Description
Additional information here...

Type
Nimble-Cinder

Size (GB) *
2

Volume Source
Image

Use image as a source
centos65 (328.5 MB)

Step 2: Create volume snapshot

Volumes

Volumes

Name | Description | Size | Status | Type | Attached To | Actions
--- | --- | --- | --- | --- | --- | ---
vol1 |  | 2GB | Available | - | | Create Snapshot

Delete Volume
Step 3: Launch instance and specify number of clones to create

Step 4: Enjoy instantaneous array level cloning without duplicating data blocks from instances
Appendix

A.1 Nimble Cinder Driver Installation

Driver Placement

Place "nimble.py" in the following directory of the OpenStack controller node:

/usr/lib/python2.6/site-packages/cinder/volume/drivers/

Single Backend Cinder Configuration

Add the following lines to the /etc/cinder/cinder.conf file, under [DEFAULT] section:

#Nimble Cinder configurations
san_ip=<IP address of Nimble array/group>
san_login=admin <Nimble account login with minimum “power user” privilege if RBAC is used>
san_password=<password of admin account for nimble array>
volume_driver=cinder.volume.drivers.nimble.NimbleISCSIDriver
volume_backend_name=Nimble-Cinder

Note: “volume_backend_name” is not required in a single backend cinder configuration. However, we recommend specifying it to avoid issues in case of moving towards a multi-backend cinder configuration, in the future.

In the CLI of the OpenStack controller node, create Nimble backend as a volume type, and associate the type label with the cinder backend, follow by service restarts:

# cinder type-create Nimble-Cinder
# cinder type-key Nimble-Cinder set volume_backend_name=Nimble-Cinder

NOTE: “openstack-cinder-*” is specific to Red Hat OpenStack distribution; typically, it is only “cinder-volume” for open source distribution

#service openstack-cinder-scheduler restart
#service openstack-cinder-api restart
#service openstack-cinder-volume restart
A.2 Multi-Backend Cinder Implementation

On OpenStack Horizon, go to “Admin” tab, select “Volume”, and define volume type based on service level offering defined:

![OpenStack Horizon Volume Management](image)

Edit the `/etc/cinder/cinder.conf` configuration file to associate each volume type with the corresponding driver and driver options:

```ini
enabled_backends=nimble-cinder,nimble-lvm

[Nimble-Cinder]

san_ip=<management IP of Nimble array or group>
san_login=<admin user for the Nimble array>
san_password=<admin password for the Nimble array>
volume_driver=cinder.volume.drivers.nimble.NimbleISCSIDriver
volume_backend_name=Nimble-Cinder
```
[DAS-LVM]

volume_group=<name of LVM volume group backed by local DAS>
volume_driver=cinder.volume.drivers.lvm.LVMISCSIDriver
volume_backend_name=DAS-LVM

[Nimble-LVM]

volume_group=<name of LVM volume group backed by Nimble volume>
volume_driver=cinder.volume.drivers.lvm.LVMISCSIDriver
volume_backend_name=Nimble-LVM

A.3 Multipathing

1. Use Cinder to create a new volume
2. Attach the volume to a VM
3. Use “iscsiadm -m node” to identify the new volume’s iqn
4. iscsiadm -m node -T <target_iqn> -p 172.17.2.75 --op update node.session.nr_sessions -v 4
   a. for example: iscsiadm -m node -T <target_iqn>iqn.2007-11.com.nimblestorage:volume-81c6d336-e770-4d25-99b9-1aabc9b87a9d-v4161c23a56ce8f8c7.00000002.17af3af -p 172.17.2.75 --op update node.session.nr_sessions -v 4
   b. this will create 4 iSCSI connections/paths to the device
5. /etc/init.d/iscsi restart
6. multipath –ll

# iscsiadm -m session


# multipath -ll

mpathc (2c9d9a9f9ae79ccf36c9ce900fd3aaf17) dm-2 Nimble,Server

size=3.0G features='1 queue_if_no_path' hwhandler='0' wp=rw

`-+- policy='round-robin 0' prio=1 status=active

|- 7:0:0:0 sdd 8:48 active ready running
|- 4:0:0:0 sdf 8:80 active ready running
|- 5:0:0:0 sde 8:64 active ready running
`- 6:0:0:0 sdc 8:32 active ready running

A.4 OpenStack Image Service Offload

1. Identify your storage system's iSCSI Discovery IP address

Use iscsiadm to discover the volumes:

[root@TS-Training-OS-01 ~]# iscsiadm -m discovery -t sendtargets -p discovery_IP

2. Establish a connection to the appropriate volume:


3. Once the volume has been connected, use fdisk -l to identify the new disk, in the example below it is /dev/sdc. Use mkfs to format the disk in ext4:

[root@TS-Training-OS-01 ~]# mkfs.ext4 -b 4096 /dev/sdc

4. After the device has been formatted, create a mount-point and change the permissions on it:

[root@TS-Training-OS-01 ~]# mkdir /mnt/glance

[root@TS-Training-OS-01 ~]# chmod 777 /mnt/glance

5. Configure fstab to automatically mount /dev/sdc to /mnt/glance after a reboot. Add the following line to /etc/fstab:

/dev/sdc /mnt/glance ext4 defaults 0 0

6. Mount /dev/sdc to /mnt/glance by running the following command:

[root@TS-Training-OS-01 ~]# mount /dev/sdc /mnt/glance
7. Update the “/etc/glance/glance-api.conf” file to redirect repository to the newly mounted filesystem backed by Nimble volume

# =========== Filesystem Store Options ========================

# Directory that the Filesystem backend store

# writes image data to

#filesystem_store_datadir=/var/lib/glance/images/

filesystem_store_datadir=/mnt/glance/

8. Restart the glance-api service and any newly uploaded image through glance will be located under /mnt/glance on your controller.

[root@TS-Training-OS-01 ~]# service openstack-glance-api restart