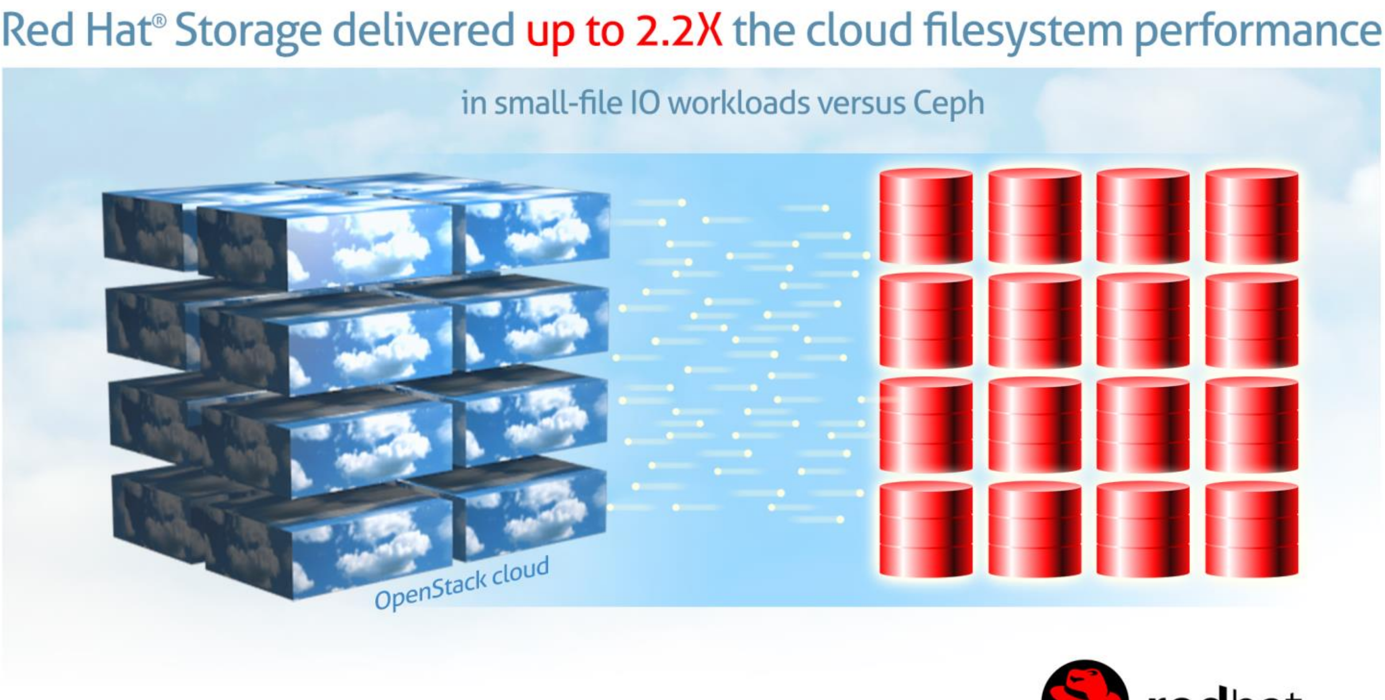
# DISTRIBUTED STORAGE PERFORMANCE FOR OPENSTACK CLOUDS USING SMALL-FILE IO WORKLOADS: RED HAT STORAGE SERVER VS. CEPH STORAGE




OpenStack clouds require fast-acting storage solutions to deliver optimal performance to end users. Software-based distributed storage systems are a popular choice for such environments because they allow for pooled resources with flexible management and scaling capabilities. In cloud environments, IO workloads often use smaller datasets, requiring distributed storage systems to handle small-file workloads and all the associated filesystem actions that occur with this type of IO.

In the Principled Technologies labs, we investigated how two distributed storage solutions, Red Hat Storage Server and Ceph Storage, performed handling small-file IO workloads using the ***smallfile*** benchmark tool. We tested how both storage solutions performed a number of common storage operations across various configurations of nodes, virtual machines (VMs), and threads. In our tests, Red Hat Storage Server delivered greater throughput (faster storage performance) in almost every instance, including 124.0 percent more throughput than Ceph when completing the create operation during the workload.

JANUARY 2014

A PRINCIPLED TECHNOLOGIES TEST REPORT

Commissioned by Red Hat, Inc.

# DISTRIBUTED STORAGE TESTING

## OpenStack and distributed storage

An OpenStack cloud manages compute, storage, and networking resources. For the backing storage in an OpenStack cloud environment, organizations face the challenge of selecting cost-effective, flexible, and high-performing storage. By using distributed scale-out storage with open-source software, companies can achieve these goals. Software such as Red Hat Storage removes the high-cost and specialized skillset of traditional storage arrays and instead relies on servers as storage nodes, which means that datacenter staff can simply add servers to scale capacity and performance. Conversely, if storage needs decrease, administrators can repurpose those server storage nodes if necessary.

The storage components in an OpenStack environment are Cinder, a component that handles persistent block storage for guests; Glance, a component that stores and manages guest images; and Swift, an object storage component. This study focuses on performance of guest (virtual machine) local file systems constructed on Cinder block devices stored in either Red Hat Storage or Ceph.

The clear winner in our performance and scalability tests was Red Hat Storage. With small-file IO workloads using the ***smallfile*** tool, it dramatically outperformed Ceph on nearly every operation at nearly every compute node/VM configuration we tested. In tests where it did not win, Red Hat Storage performed comparably to Ceph. Handling more files per second translates into how an end-user of an OpenStack cloud application could experience the speed of accessing or altering files in the cloud.

Figure 1 shows the performance the solutions achieved with four server nodes, 16 VMs, and 64 threads across all operations. Here, we normalize performance to the Ceph Storage scores, showing Red Hat Storage performance as a factor of what Ceph achieved. For detailed performance results for all configurations and operations, see the Test results section.

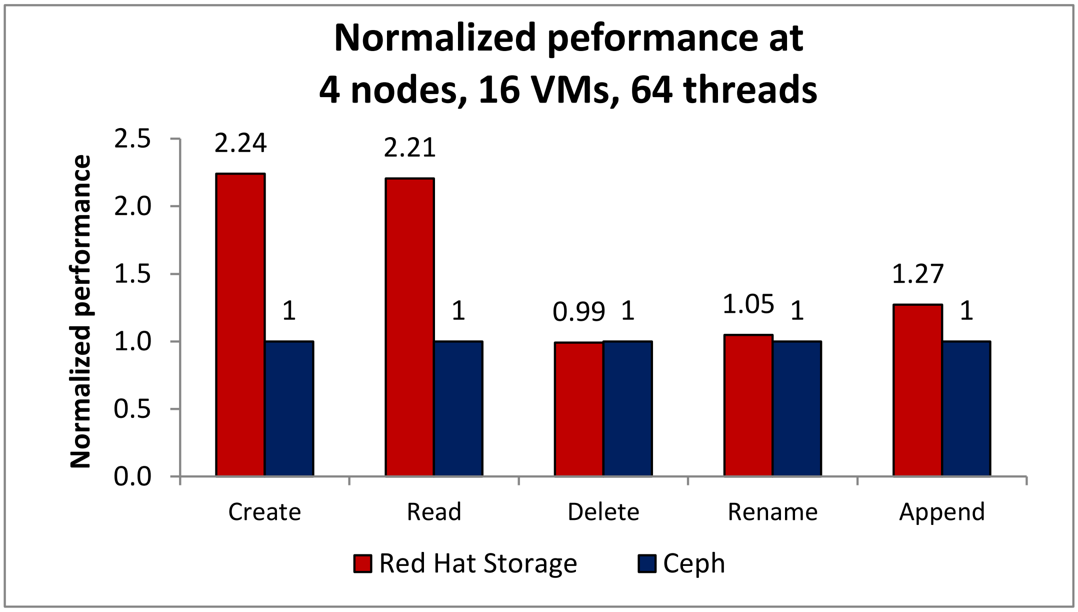



**Figure 1: Performance comparison for 4 nodes, 16 VMs, and 64 threads, normalized to Ceph scores.**

## Software overview

In our tests of two leading open-source distributed storage solutions, we compared small-file performance of Red Hat Storage Server and Ceph Storage, along with the scalability of both solutions using one to four nodes, one to 16 VMs, and four to 64 threads. We used RDO OpenStack for our OpenStack distribution, and we used the ***smallfile*** benchmark running within virtual machine instances on our OpenStack compute nodes to measure filesystem read and write throughput for multiple configurations using each storage solution.

For testing, we used the same hardware for both solutions – four compute nodes running RDO OpenStack[1] and four storage nodes running either Red Hat Storage or Ceph Storage. For detailed system configuration information, see Appendix A. Red Hat commissioned these tests and this report.

## Testing with *smallfile*

To test the relative performance of Red Hat Storage Server and Ceph Storage in such a highly virtualized, multi-tenant scenario, we used ***smallfile***, a python-based, open-source workload tool designed to assess filesystem performance across distributed storage with metadata-intensive file operations. ***Smallfile*** is available via Github at https://github.com/bengland2/smallfile.git.

***Smallfile*** differs from many synthetic storage benchmarks in that it accounts for metadata operations. Other storage benchmarks often send IO directly to drives or use a small number of very large files during the test iterations, completely bypassing

[1] http://openstack.redhat.com

filesystem metadata operations. While these benchmarks can provide useful data, such workloads can be less than ideal for emulating a cloud environment where there is an assumption of high multi-tenancy, smaller VMs, few cores, smaller amounts of vRAM, and fewer available IOPS. In these environments, where small-file IO is common, the application must also use resources working with metadata operations around the IO events, such as opening, closing, deleting, calculating file distribution or sizes, and so on. These conditions lend themselves to use cases involving high numbers of files that are smaller in size, which ***smallfile*** can emulate.

We ran ***smallfile*** within RHEL VMs residing on four identical compute nodes, with virtual disks attached to each VM, which were physically located on four storage nodes. We ran ***smallfile*** from within the guests first using Ceph and then using Red Hat Storage Server as the backing storage on the four storage nodes. We used a random exponential distribution of file sizes (as supported by the ***smallfile*** tool) to provide a distribution of file sizes similar to what many real-world virtualized, multi-tenant environments would use. Figure 2 shows our test setup for both solutions. The virtio block devices were created in the OpenStack framework as Cinder volumes.

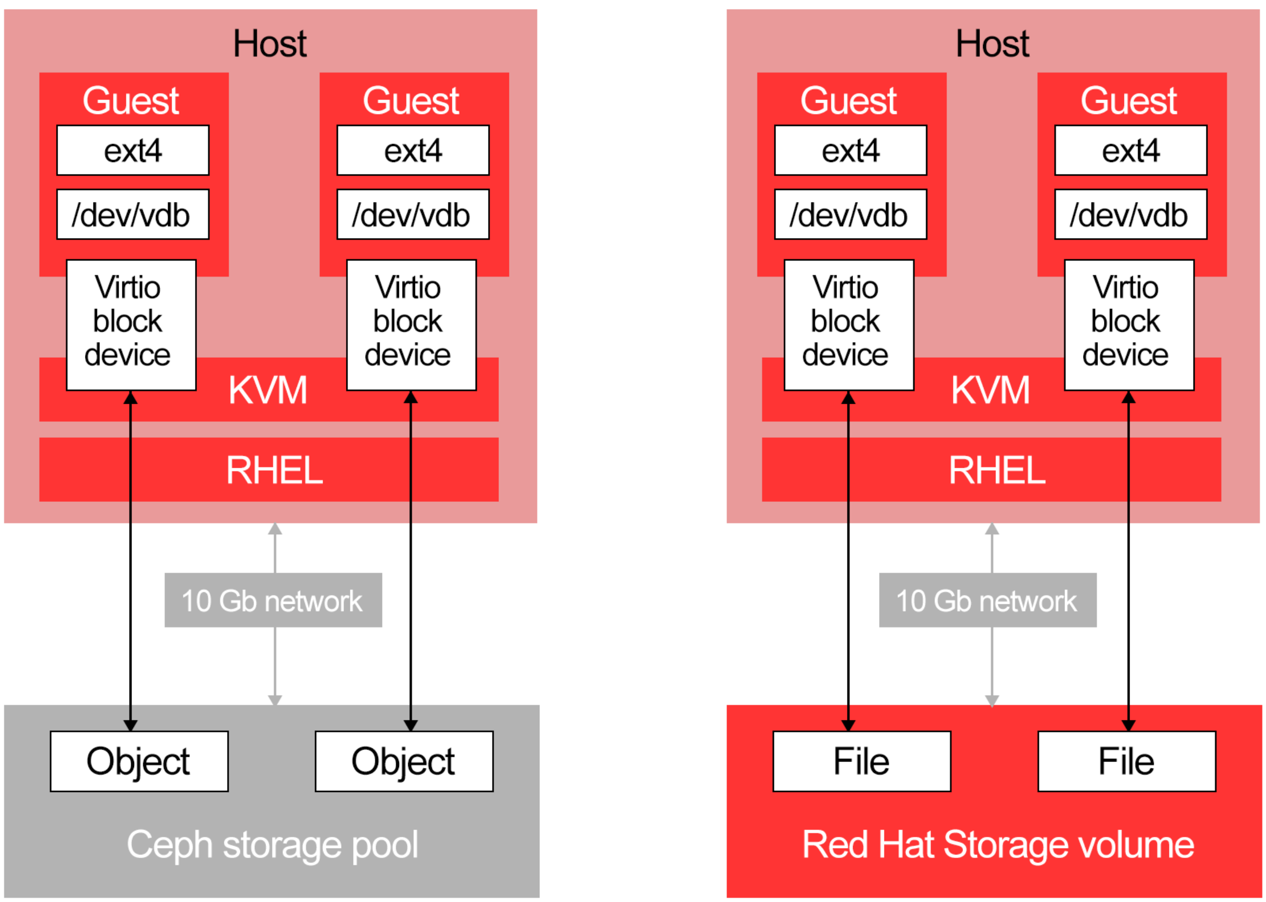



**Figure 2: Our test setup for Ceph Storage and Red Hat Storage Server. The virtio block devices were created as Cinder volumes in OpenStack.**

In both configurations, we started with a single node, single VM, and four ***smallfile*** workload threads and then increased threads, VMs, and nodes in a predictable manner – up to a maximum of 4 nodes, 16 guests, and 64 total threads (staying at a consistent four threads per VM). Each thread in the ***smallfile*** workload operated on a total of 32,768 files for approximately two-million unique files in our maximum configuration (4 node/4VM/64 threads). The file-size distribution averaged 64 KB, however during the append tests the average file size grew to 128 KB. Note: Within the ***smallfile*** test cycle, each operation is executed on every file (one operation at a time). On the first run, all files are created, on the next run all files are appended to, and so on until all operations are complete. The operations are:

- Create
- Read
- Delete
- Rename
- Append

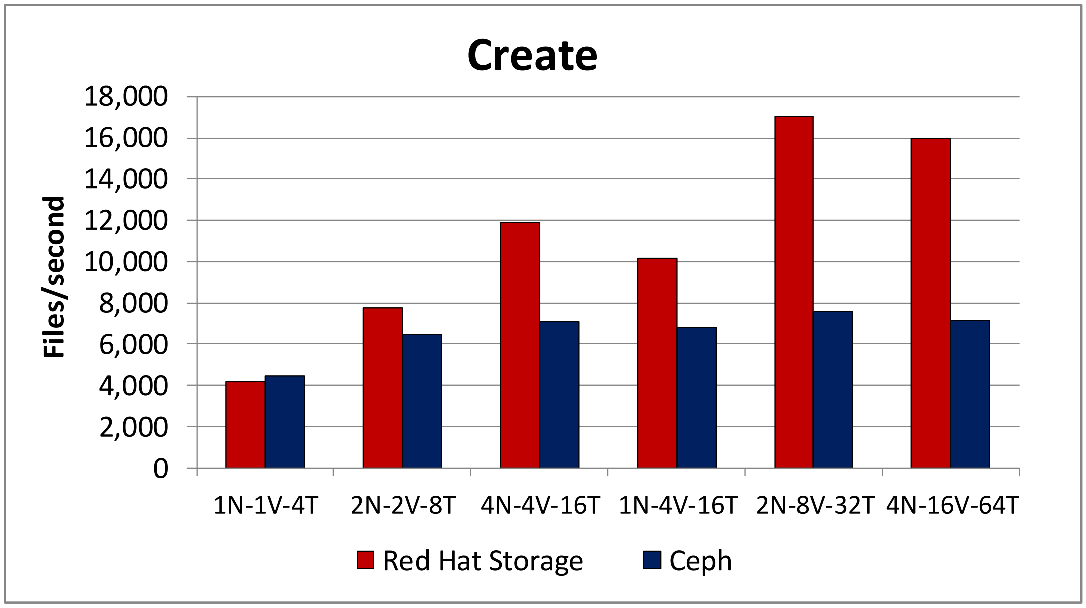
The metadata functions occur throughout each of the test operations including the read, write, and append tests. Every time a file is accessed, an OPEN and CLOSE operation is run.

# TEST RESULTS

## Create test results

Completing the create operation, Red Hat Storage handled up to 124.0 percent more files per second than Ceph Storage. As we increased nodes, VMs, and thread counts (noted as #N-#V-#T in the charts below), Red Hat Storage continued to deliver increased performance for the workload (see Figure 3). For specific throughput data for this operation, see Figure 4.

Figure 3: Throughput comparison of the storage solutions completing the create operation at various node, VM, and thread counts.



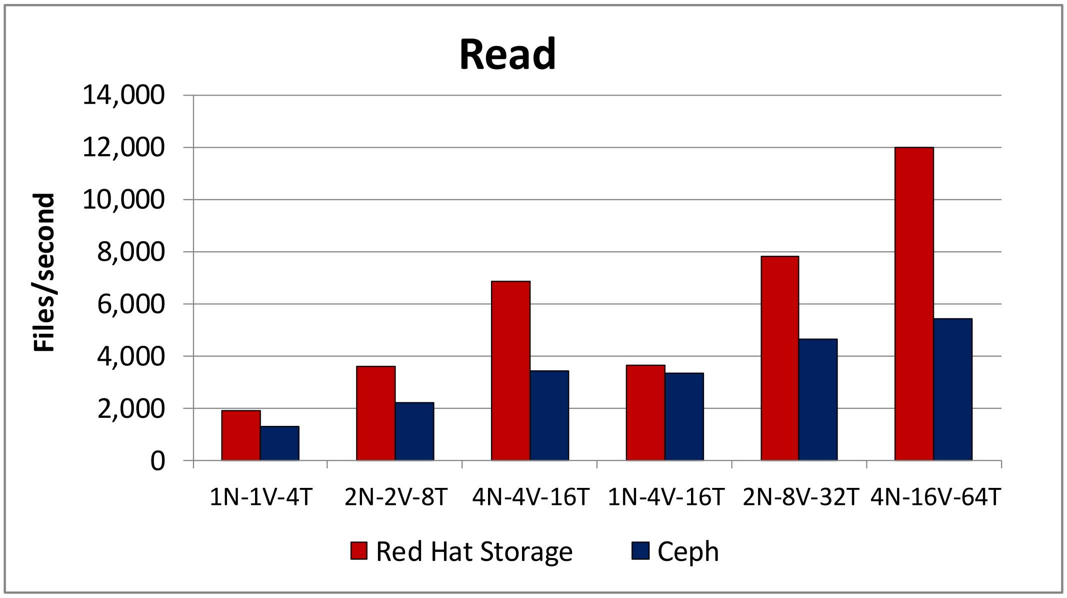

| Create | | | | | | |
|---|---|---|---|---|---|---|
| | 1N-1V-4T | 2N-2V-8T | 4N-4V-16T | 1N-4V-16T | 2N-8V-32T | 4N-16V-64T |
| Red Hat Storage | 4,152 | 7,779 | 11,893 | 10,143 | 17,016 | 15,977 |
| Ceph | 4,459 | 6,495 | 7,081 | 6,829 | 7,609 | 7,131 |
| Red Hat win | -6.9% | 19.8% | 68.0% | 48.5% | 123.6% | 124.0% |

Figure 4: Throughput, in files/second for the storage solutions completing the create operation at various node, VM, and thread counts.

## Read test results

Completing the read operation, Red Hat Storage handled up to 120.7 percent more files per second than Ceph Storage. As we increased nodes, VMs, and thread counts, Red Hat Storage continued to deliver increased performance for the workload (see Figure 5). For specific throughput data for this operation, see Figure 6.

Figure 5: Throughput comparison of the storage solutions completing the read operation at various node, VM, and thread counts.



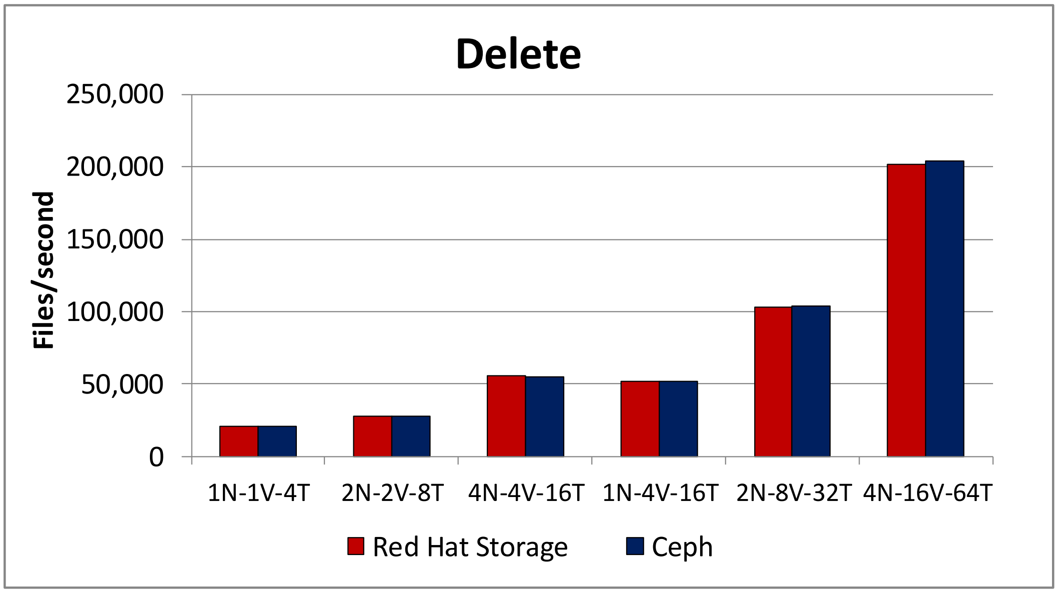

| Read | | | | | | |
|---|---|---|---|---|---|---|
| | 1N-1V-4T | 2N-2V-8T | 4N-4V-16T | 1N-4V-16T | 2N-8V-32T | 4N-16V-64T |
| Red Hat Storage | 1,879 | 3,612 | 6,845 | 3,636 | 7,814 | 11,991 |
| Ceph | 1,307 | 2,191 | 3,416 | 3,348 | 4,650 | 5,434 |
| Red Hat win | 43.8% | 64.9% | 100.4% | 8.6% | 68.0% | 120.7% |

Figure 6: Throughput, in files/second for the storage solutions completing the read operation at various node, VM, and thread counts.

## Delete test results

Completing the delete operation, Red Hat Storage performed comparably to Ceph storage at each node, VM, and thread count (see Figure 7). For specific throughput data for this operation, see Figure 8.

Figure 7: Throughput comparison of the storage solutions completing the delete operation at various node, VM, and thread counts.



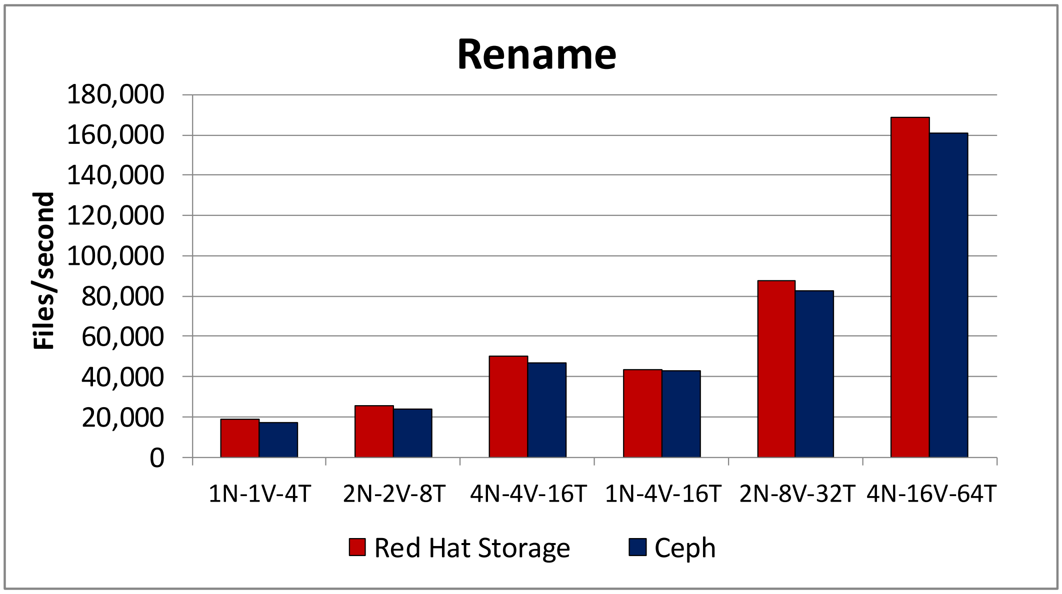

| Delete | | | | | | |
|---|---|---|---|---|---|---|
| | 1N-1V-4T | 2N-2V-8T | 4N-4V-16T | 1N-4V-16T | 2N-8V-32T | 4N-16V-64T |
| Red Hat Storage | 20,882 | 28,035 | 55,244 | 51,840 | 103,365 | 201,940 |
| Ceph | 20,716 | 27,442 | 54,716 | 52,121 | 103,464 | 203,749 |
| Red Hat win | 0.8% | 2.2% | 1.0% | -0.5% | -0.1% | -0.9% |

Figure 8: Throughput, in files/second for the storage solutions completing the delete operation at various node, VM, and thread counts.

## Rename test results

Completing the rename operation, Red Hat Storage handled up to 11.1 percent more files per second than Ceph Storage. As we increased nodes, VMs, and thread counts, Red Hat Storage continued to deliver increased performance for the workload (see Figure 9). For specific throughput data for this operation, see Figure 10.

Figure 9: Throughput comparison of the storage solutions completing the rename operation at various node, VM, and thread counts.



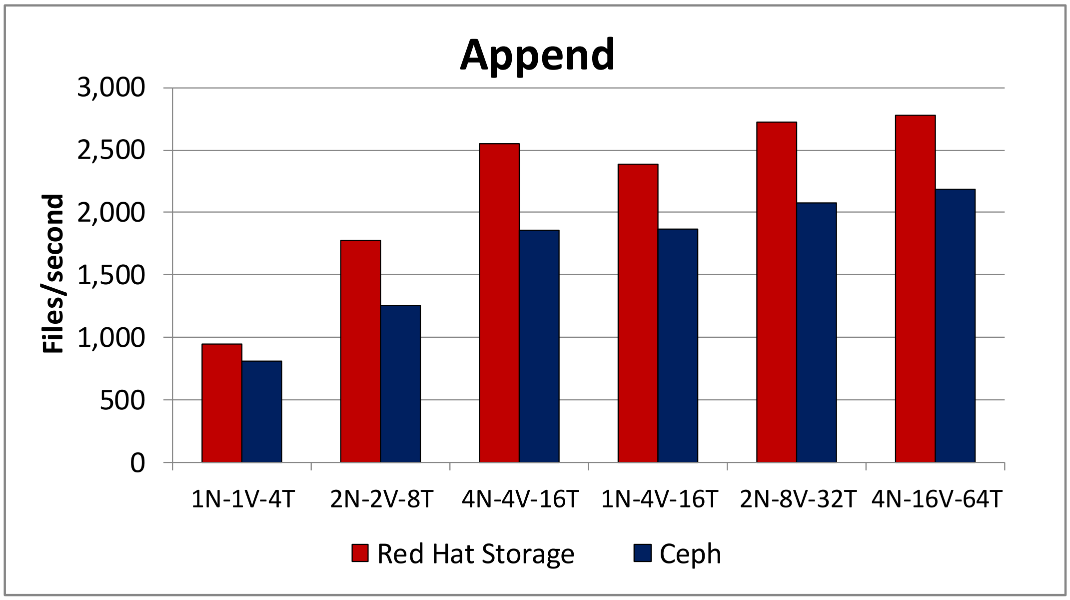

| Rename | | | | | | |
|---|---|---|---|---|---|---|
| | 1N-1V-4T | 2N-2V-8T | 4N-4V-16T | 1N-4V-16T | 2N-8V-32T | 4N-16V-64T |
| Red Hat Storage | 18,727 | 25,433 | 50,296 | 43,454 | 87,455 | 169,015 |
| Ceph | 16,863 | 23,782 | 46,981 | 42,863 | 82,828 | 161,126 |
| Red Hat win | 11.1% | 6.9% | 7.1% | 1.4% | 5.6% | 4.9% |

Figure 10: Throughput, in files/second for the storage solutions completing the rename operation at various node, VM, and thread counts.

## Append test results

Completing the append operation, Red Hat Storage handled up to 41.4 percent more files per second than Ceph Storage. As we increased nodes, VMs, and thread counts, Red Hat Storage continued to deliver increased performance for the workload (see Figure 11). For specific throughput data for this operation, see Figure 12.

Figure 11: Throughput comparison of the storage solutions completing the append operation at various node, VM, and thread counts.




| Append | | | | | | |
|---|---|---|---|---|---|---|
| | 1N-1V-4T | 2N-2V-8T | 4N-4V-16T | 1N-4V-16T | 2N-8V-32T | 4N-16V-64T |
| Red Hat Storage | 944 | 1,777 | 2,548 | 2,383 | 2,727 | 2,779 |
| Ceph | 805 | 1,257 | 1,856 | 1,866 | 2,075 | 2,185 |
| Red Hat win | 17.3% | 41.4% | 37.3% | 27.7% | 31.4% | 27.2% |

Figure 12: Throughput, in files/second for the storage solutions completing the append operation at various node, VM, and thread counts.

# WHAT WE TESTED

## About Red Hat Storage Server

Red Hat Storage Server is a software-based, or according to Red Hat "software-defined," storage platform to manage big, semi-structured, and unstructured data growth while maintaining performance, capacity, and availability to meet demanding enterprise storage requirements. Running on open-source software, it collects compute and network resources in addition to storage capacity, on both physical infrastructure and cloud environments to independently scale beyond the limitations of each type of environment. Along with the ability to deploy on-premises or in a cloud environment, Red Hat Storage Server has flexible deployment options to meet various business needs. For more information about Red Hat Storage, visit http://www.redhat.com/products/storage-server/.

## About Ceph Storage

Ceph Storage is an object-based storage system, separating objects from the underlying storage hardware using Reliable Autonomic Distributed Object Store

(RADOS). According to Ceph, the RADOS foundation ensures flexibility in data storage by allowing applications to use object, block, or file system interfaces simultaneously.

For more information about Ceph Storage, visit http://ceph.com/ceph-storage/.

## IN CONCLUSION

OpenStack cloud environments demand strong storage performance to handle the requests of end users. Software-based distributed storage can provide this performance while also providing much needed flexibility for storage resources.

In our tests, we found that Red Hat Storage Server better handled small-file IO workloads than did Ceph Storage, handling up to two times the number of files per second in some instances. The ***smallfile*** tool we used simulated users performing actions on their files to show the kind of end-user performance you could expect using both solutions at various node, VM, and thread counts.

These results show that Red Hat Storage Server can provide equivalent or better performance than Ceph Storage for similar workloads in OpenStack cloud environments, which can help users better access the files they keep in the cloud.

# APPENDIX A – SYSTEM CONFIGURATION INFORMATION

Figure 13 provides detailed configuration information for the systems we used in our tests.

| System | Dell™ PowerEdge™ C8220X (storage node) | Dell PowerEdge C8220 (compute node) |
|---|---|---|
| **Power supplies** | | |
| Total number | 2 | 2 |
| Vendor and model number | Dell B07B | Dell B07B |
| Wattage of each (W) | 2800 | 2800 |
| **Cooling fans** | | |
| Total number | 6 (chassis fans) | 6 (chassis fans) |
| Vendor and model number | Delta Electronics, Inc. PFC1212DE | Delta Electronics, Inc. PFC1212DE |
| Dimensions (h x w) of each | 5" x 5" x 1.5" | 5" x 5" x 1.5" |
| Volts | 12 | 12 |
| Amps | 4.80 | 4.80 |
| **General** | | |
| Number of processor packages | 2 | 2 |
| Number of cores per processor | 8 | 8 |
| Number of hardware threads per core | 2 | 2 |
| System power management policy | N/A | N/A |
| **CPU** | | |
| Vendor | Intel® | Intel |
| Name | Xeon® | Xeon |
| Model number | E5-2650 | E5-2650 |
| Stepping | C2 | C2 |
| Socket type | LGA2011 | LGA2011 |
| Core frequency (GHz) | 2.00 | 2.00 |
| Bus frequency | 4,000 | 4,000 |
| L1 cache | 32 KB + 32 KB (per core) | 32 KB + 32 KB (per core) |
| L2 cache | 256 KB (per core) | 256 KB (per core) |
| L3 cache | 20 MB | 20 MB |
| **Memory module(s)** | | |
| Total RAM in system (GB) | 16 | 128 |
| Vendor and model number | Samsung® M393B5273DH0-CK0 | Samsung M393B1K70DH0-CK0 |
| Type | PC3-12800R | PC3-12800R |
| Speed (MHz) | 1,600 | 1,600 |
| Size (GB) | 4 | 8 |
| Number of RAM module(s) | 4 | 8 |
| Chip organization | Double-sided | Double-sided |
| Rank | Dual | Dual |

| System | Dell™ PowerEdge™ C8220X (storage node) | Dell PowerEdge C8220 (compute node) |
|---|---|---|
| **Operating system** | | |
| Name | Red Hat Enterprise Linux® | Red Hat Enterprise Linux |
| Build number | 6.4 | 6.5 Beta |
| File system | xfs | ext4 |
| Kernel | 2.6.32-358.18.1.el6.x86_64 | 2.6.32-415.el6.x86_64 |
| Language | English | English |
| **Graphics** | | |
| Vendor and model number | ASPEED AST2300 | ASPEED AST2300 |
| Graphics memory (MB) | 16 | 16 |
| **RAID controller 1** | | |
| Vendor and model number | Intel C600 | Intel C600 |
| Cache size | N/A | N/A |
| **RAID controller 2** | | |
| Vendor and model number | LSI 9265-8i | N/A |
| Cache size | 1 GB | N/A |
| **Hard drive** | | |
| Vendor and model number | Dell 9RZ168-136 | Dell 9RZ168-136 |
| Number of disks in system | 2 | 2 |
| Size (GB) | 1,000 | 1,000 |
| Buffer size (MB) | 32 | 32 |
| RPM | 7,200 | 7,200 |
| Type | SATA 6.0 Gb/s | SATA 6.0 Gb/s |
| **Hard drive 2** | | |
| Vendor and model number | Dell 9TG066-150 | N/A |
| Number of disks in system | 8 | N/A |
| Size (GB) | 600 | N/A |
| Buffer size (MB) | 64 | N/A |
| RPM | 10,000 | N/A |
| Type | SAS 6.0 Gb/s | N/A |
| **Ethernet adapters** | | |
| **First network adapter** | | |
| Vendor and model number | Intel I350-BT2 | Intel I350-BT2 |
| Type | Integrated | Integrated |
| **Second network adapter** | | |
| Vendor and model number | Mellanox MAX383A | Mellanox MAX383A |
| Type | 10/40 GbE | 10/40 GbE |
| **USB ports** | | |
| Number | 2 | 2 |
| Type | 2.0 | 2.0 |

**Figure 13: Configuration information for our test systems.**

# APPENDIX B – TEST SETUP OVERVIEW

## Compute nodes and OpenStack controller

We installed four compute server nodes with Red Hat Enterprise Linux 6.5 Beta to be used for the OpenStack cloud. Each compute node contained two hard disks, which we configured in a RAID 1 mirror, where we installed the operating system. We used a separate server node to serve as our OpenStack controller, on which we ran all OpenStack services (Neutron, Cinder, Horizon, Keystone, MySQL™) other than nova-compute, which ran on the compute nodes. Figure 14 shows our test configuration.

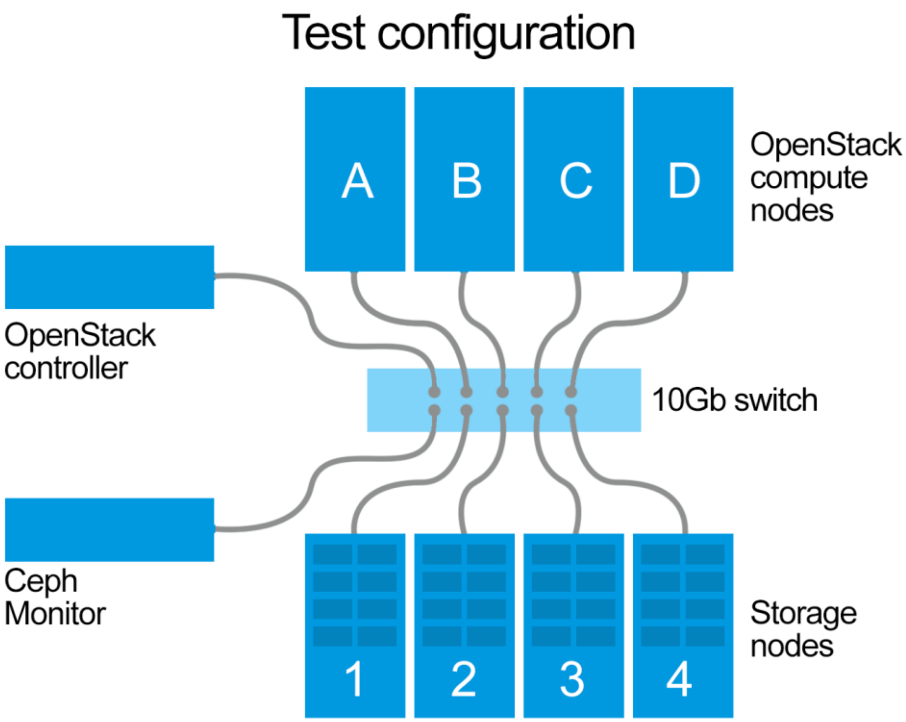



**Figure 14: Test hardware configuration.**

## Storage server configuration

Each of the four storage nodes contained two 1TB 7,200RPM SATA disks, which we configured in a RAID 1 mirror. On this RAID 1 set, we created two logical volumes. On the first, we installed Red Hat Storage 2.1 Beta for Red Hat Storage tests. On the second logical volume, we installed Red Hat Enterprise Linux 6.4 and the necessary Ceph Storage packages for the Ceph Storage tests (ceph version 0.67.4). To switch between storage platforms, we used GRUB to choose the boot volume, and booted the storage nodes into the correct environment, either Red Hat Storage, or Ceph Storage. These configurations remained constant amongst the four storage nodes.

Each of the four storage nodes also contained eight 600GB 10K RPM SAS disks. We configured these disks to be our data disks for testing, and varied our approach based on each platform's best practices and recommendations. For Red Hat Storage, we configured these eight disks in an eight disk RAID 6 volume and presented the volume to Red Hat Storage. Figure 15 shows the node configuration for Red Hat Storage tests.

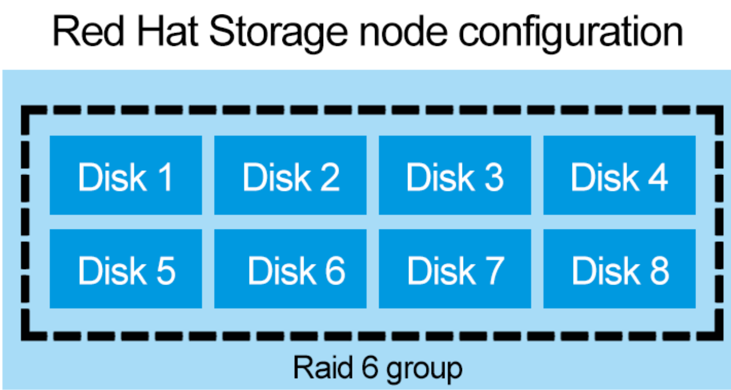



**Figure 15: Red Hat Storage node configuration.**

For Ceph Storage, we configured eight RAID 0 volumes (one for each physical disk) and presented all of them to Ceph Storage, whereby it could then use an independent OSD on each physical disk, per Ceph Storage best practices. These configurations remained constant amongst the four storage nodes. Figure 16 shows the node configuration for our Ceph tests.

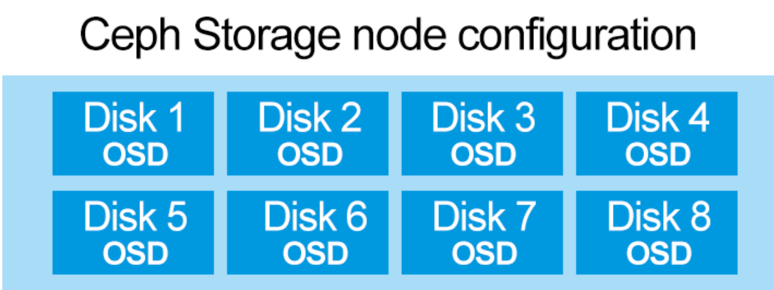



**Figure 16: Ceph Storage node configuration.**

Figure 17 details the software versions we used in our tests.

| Servers | Operating system | Additional software |
| --- | --- | --- |
| OpenStack Controller | Red Hat Enterprise Linux 6.5 Beta | RDO 2013.2 b3 |
| OpenStack Compute nodes | Red Hat Enterprise Linux 6.5 Beta | qemu-kvm-rhev-0.12.1.2-2.411<br>glusterfs-api-3.4.0.34rhs-1<br>librbd1-0.67.4-0 |
| Storage nodes (Red Hat Storage tests) | Red Hat Storage 2.1 Beta | glusterfs-server-3.4.0.19rhs-2 |
| Storage nodes (Ceph Storage tests) | Red Hat Enterprise Linux 6.4 | ceph-0.67.4-0 |

**Figure 17: Software versions we used in our tests.**

# APPENDIX C – DETAILED CONFIGURATION STEPS

In this section, we review in detail the steps we followed on the various machines to install and configure the various components. Commands are presented with no shading, while file contents or output is presented with gray shading.

## Configuring Red Hat Network Beta repositories

1. On each machine that will use Red Hat Enterprise Linux Beta, configure the RHN Beta repositories.

```
subscription-manager repos --enable=rhel-6-server-beta-rpms
subscription-manager repos --enable=rhel-6-server-optional-beta-rpms
```

## Configuring networking on all servers

1. Install the necessary rpms by using the following commands:

```
yum install -y openssh-clients wget acpid cpuspeed tuned sysstat sysfsutils
```

2. Bring these devices down using the following commands:

```
ifconfig p2p1 down
ifconfig ib0 down
```

3. Remove the ifcfg files using the following commands:

```
cd /etc/sysconfig/network-scripts/
rm -f ifcfg-p[0-9]p[0-9] ifcfg-ib[0-9]
```

4. Configure the Mellanox components using the following commands:

```
modprobe -r mlx4_en mlx4_ib mlx4_core
sed -i '/mlx4_core/,+1d' /etc/udev/rules.d/70-persistent-net.rules
echo "install mlx4_core /sbin/modprobe --ignore-install mlx4_core msi_x=1
enable_64b_cqe_eqe=1 port_type_array=2 && /sbin/modprobe mlx4_en" >
/etc/modprobe.d/mlx4.conf
modprobe mlx4_core
```

5. Disable SELinux using the following commands:

```
sed -i 's/SELINUX=.*/SELINUX=disabled/' /etc/selinux/config
reboot
```

6. Edit the /etc/hosts file on every host using the following command. Run vi and edit the hosts file.

```
vi /etc/hosts
```

We used the following /etc/hosts file:

```
127.0.0.1   localhost localhost.localdomain localhost4 localhost4.localdomain4
::1         localhost localhost.localdomain localhost6 localhost6.localdomain6

192.168.43.10     rdo-cont.test.lan rdo-cont
192.168.43.12     cephmon.test.lan cephmon

192.168.43.101    compute1.test.lan compute1
192.168.43.102    compute2.test.lan compute2
192.168.43.103    compute3.test.lan compute3
192.168.43.104    compute4.test.lan compute4

192.168.43.201    storage1.test.lan storage1
```

```
192.168.43.202    storage2.test.lan storage2
192.168.43.203    storage3.test.lan storage3
192.168.43.204    storage4.test.lan storage4
```

## Configuring additional networking – OpenStack controller

1. Edit the network configuration for the first NIC using the following command. Run vi and edit the ifcfg-em1 file.

```
vi ifcfg-em1
```

We used the following settings:

```
DEVICE=em1
TYPE=Ethernet
ONBOOT=yes
IPADDR=192.168.43.10
PREFIX=24
MTU=9000
```

2. Edit the network configuration for the second NIC using the following command. Run vi and edit the ifcfg-em2 file.

```
vi ifcfg-em2
```

We used the following settings:

```
DEVICE=em2
TYPE=Ethernet
ONBOOT=yes
MTU=9000
```

3. Set up passwordless ssh access for all relevant nodes from the OpenStack controller using the following commands:

```
ssh-keygen
ssh-copy-id cephmon
ssh-copy-id compute1
ssh-copy-id compute2
ssh-copy-id compute3
ssh-copy-id compute4
ssh-copy-id storage1
ssh-copy-id storage2
ssh-copy-id storage3
ssh-copy-id storage4
```

4. Configure DNS using the following command:

```
echo "nameserver 192.168.43.1" > /etc/resolv.conf
service network restart
```

## Configuring additional networking – OpenStack compute nodes

1. Edit the network configuration for the first NIC using the following command. Run vi and edit the ifcfg-em1 file.

```
vi ifcfg-em1
```

We used the following settings:

```
DEVICE=em1
TYPE=Ethernet
ONBOOT=no
IPADDR=192.168.43.101
PREFIX=24
MTU=9000
```

2. Edit the network configuration for the second NIC using the following command. Run vi and edit the ifcfg-em2 file.

```
vi ifcfg-em2
```

We used the following settings.

```
DEVICE=em2
TYPE=Ethernet
ONBOOT=yes
MTU=9000
```

3. Edit the network configuration for the third NIC using the following commands. Run vi and edit the ifcfg-eth0 file.

```
cp -p ifcfg-em1 ifcfg-eth0
vi ifcfg-eth0
```

We used the following settings:

```
DEVICE=eth0
TYPE=Ethernet
ONBOOT=yes
IPADDR=192.168.43.101
PREFIX=24
MTU=9000
```

4. Configure DNS using the following command:

```
echo "nameserver 192.168.43.1" > /etc/resolv.conf
service network restart
```

## Installing OpenStack

1. On the OpenStack controller machine, install the RDO rpms using the following commands:

```
yum install -y http://rdo.fedorapeople.org/openstack-havana/rdo-release-havana.rpm
yum install -y http://download.fedoraproject.org/pub/epel/6/i386/epel-release-6-
8.noarch.rpm
```

2. On the OpenStack controller machine, install PackStack using the following commands:

```
yum install -y openstack-packstack
packstack --gen-answer-file=packstack-answer-havana.txt
cp packstack-answer-havana.txt packstack-answer-havana.txt.orig
```

3. Edit the PackStack configuration file. Below we show the revisions we made to our PackStack configuration file from the original default file.

```
vi packstack-answer-havana.txt

#### revisions
diff packstack-answer-havana.txt packstack-answer-havana.txt.orig
43c43
< CONFIG_NTP_SERVERS=<NTP SERVER HERE>
---
> CONFIG_NTP_SERVERS=
142c142
< CONFIG_NOVA_COMPUTE_HOSTS=192.168.43.101,192.168.43.102,192.168.43.103,192.168.43.104
---
> CONFIG_NOVA_COMPUTE_HOSTS=192.168.43.10
220c220
< CONFIG_NEUTRON_L3_EXT_BRIDGE=provider
---
> CONFIG_NEUTRON_L3_EXT_BRIDGE=br-ex
249c249
```

```
< CONFIG_NEUTRON_OVS_TENANT_NETWORK_TYPE=vlan
---
> CONFIG_NEUTRON_OVS_TENANT_NETWORK_TYPE=local
253c253
< CONFIG_NEUTRON_OVS_VLAN_RANGES=inter-vlan:1200:1205
---
> CONFIG_NEUTRON_OVS_VLAN_RANGES=
257c257
< CONFIG_NEUTRON_OVS_BRIDGE_MAPPINGS=inter-vlan:br-inst
---
> CONFIG_NEUTRON_OVS_BRIDGE_MAPPINGS=
261c261
< CONFIG_NEUTRON_OVS_BRIDGE_IFACES=br-inst:em2
---
> CONFIG_NEUTRON_OVS_BRIDGE_IFACES=
```

4. On the OpenStack controller, run PackStack using the following command:

```
packstack --answer-file=packstack-answer-havana.txt
```

The output should be similar to the following:

```
Welcome to Installer setup utility

Installing:
Clean Up...                                            [ DONE ]
Adding pre install manifest entries...                 [ DONE ]
Installing time synchronization via NTP...             [ DONE ]
Setting up ssh keys...                                 [ DONE ]
Adding MySQL manifest entries...                       [ DONE ]
Adding QPID manifest entries...                        [ DONE ]
Adding Keystone manifest entries...                    [ DONE ]
Adding Glance Keystone manifest entries...             [ DONE ]
Adding Glance manifest entries...                      [ DONE ]
Installing dependencies for Cinder...                  [ DONE ]
Adding Cinder Keystone manifest entries...             [ DONE ]
Adding Cinder manifest entries...                      [ DONE ]
Checking if the Cinder server has a cinder-volumes vg...[ DONE ]
Adding Nova API manifest entries...                    [ DONE ]
Adding Nova Keystone manifest entries...               [ DONE ]
Adding Nova Cert manifest entries...                   [ DONE ]
Adding Nova Conductor manifest entries...              [ DONE ]
Adding Nova Compute manifest entries...                [ DONE ]
Adding Nova Scheduler manifest entries...              [ DONE ]
Adding Nova VNC Proxy manifest entries...              [ DONE ]
Adding Nova Common manifest entries...                 [ DONE ]
Adding Openstack Network-related Nova manifest entries...[ DONE ]
Adding Neutron API manifest entries...                 [ DONE ]
Adding Neutron Keystone manifest entries...            [ DONE ]
Adding Neutron L3 manifest entries...                  [ DONE ]
Adding Neutron L2 Agent manifest entries...            [ DONE ]
Adding Neutron DHCP Agent manifest entries...          [ DONE ]
Adding Neutron Metadata Agent manifest entries...      [ DONE ]
Adding OpenStack Client manifest entries...            [ DONE ]
Adding Horizon manifest entries...                     [ DONE ]
Adding Ceilometer manifest entries...                  [ DONE ]
Adding Ceilometer Keystone manifest entries...         [ DONE ]
Preparing servers...                                   [ DONE ]
Adding post install manifest entries...                [ DONE ]
```

```
Installing Dependencies...                              [ DONE ]
Copying Puppet modules and manifests...                 [ DONE ]
Applying Puppet manifests...
Applying 192.168.43.104_prescript.pp
Applying 192.168.43.103_prescript.pp
Applying 192.168.43.10_prescript.pp
Applying 192.168.43.101_prescript.pp
Applying 192.168.43.102_prescript.pp
192.168.43.104_prescript.pp :             [ DONE ]
192.168.43.10_prescript.pp :               [ DONE ]
192.168.43.102_prescript.pp :             [ DONE ]
192.168.43.103_prescript.pp :             [ DONE ]
192.168.43.101_prescript.pp :             [ DONE ]
Applying 192.168.43.104_ntpd.pp
Applying 192.168.43.103_ntpd.pp
Applying 192.168.43.10_ntpd.pp
Applying 192.168.43.101_ntpd.pp
Applying 192.168.43.102_ntpd.pp
192.168.43.101_ntpd.pp :                       [ DONE ]
192.168.43.104_ntpd.pp :                       [ DONE ]
192.168.43.10_ntpd.pp :                         [ DONE ]
192.168.43.103_ntpd.pp :                       [ DONE ]
192.168.43.102_ntpd.pp :                       [ DONE ]
Applying 192.168.43.10_mysql.pp
Applying 192.168.43.10_qpid.pp
192.168.43.10_mysql.pp :                       [ DONE ]
192.168.43.10_qpid.pp :                         [ DONE ]
Applying 192.168.43.10_keystone.pp
Applying 192.168.43.10_glance.pp
Applying 192.168.43.10_cinder.pp
192.168.43.10_keystone.pp :                 [ DONE ]
192.168.43.10_glance.pp :                     [ DONE ]
192.168.43.10_cinder.pp :                     [ DONE ]
Applying 192.168.43.10_api_nova.pp
192.168.43.10_api_nova.pp :                 [ DONE ]
Applying 192.168.43.10_nova.pp
Applying 192.168.43.101_nova.pp
Applying 192.168.43.102_nova.pp
Applying 192.168.43.103_nova.pp
Applying 192.168.43.104_nova.pp
192.168.43.10_nova.pp :                         [ DONE ]
192.168.43.103_nova.pp :                       [ DONE ]
192.168.43.104_nova.pp :                       [ DONE ]
192.168.43.102_nova.pp :                       [ DONE ]
192.168.43.101_nova.pp :                       [ DONE ]
Applying 192.168.43.10_neutron.pp
Applying 192.168.43.104_neutron.pp
Applying 192.168.43.103_neutron.pp
Applying 192.168.43.102_neutron.pp
Applying 192.168.43.101_neutron.pp
192.168.43.103_neutron.pp :                 [ DONE ]
192.168.43.101_neutron.pp :                 [ DONE ]
192.168.43.104_neutron.pp :                 [ DONE ]
192.168.43.102_neutron.pp :                 [ DONE ]
192.168.43.10_neutron.pp :                   [ DONE ]
Applying 192.168.43.10_osclient.pp
Applying 192.168.43.10_horizon.pp
```

```
Applying 192.168.43.10_ceilometer.pp
192.168.43.10_osclient.pp :                      [ DONE ]
192.168.43.10_ceilometer.pp :                  [ DONE ]
192.168.43.10_horizon.pp :                        [ DONE ]
Applying 192.168.43.104_postscript.pp
Applying 192.168.43.103_postscript.pp
Applying 192.168.43.10_postscript.pp
Applying 192.168.43.101_postscript.pp
Applying 192.168.43.102_postscript.pp
192.168.43.10_postscript.pp :                  [ DONE ]
192.168.43.102_postscript.pp :               [ DONE ]
192.168.43.104_postscript.pp :               [ DONE ]
192.168.43.101_postscript.pp :               [ DONE ]
192.168.43.103_postscript.pp :               [ DONE ]
                            [ DONE ]

 **** Installation completed successfully ******


Additional information:
 * Did not create a cinder volume group, one already existed
 * To use the command line tools you need to source the file /root/keystonerc_admin
created on 192.168.43.10
 * To use the console, browse to http://192.168.43.10/dashboard
 * The installation log file is available at: /var/tmp/packstack/20131001-030053-
rzecgC/openstack-setup.log
```

## Configuring OpenStack

### Configuring Neutron

```
#### NOTE: THIS IS WORKAROUND FOR RDO AND RHEL6.5 AT THE TIME OF WRITING
#### DO ON ALL OPENSTACK SERVERS
yum downgrade iproute
#### NOTE: THIS FIXES A BUG WITH UNSUPPORTED HARDWARE VLAN OFFLOAD
ovs-vsctl set interface em2 other-config:enable-vlan-splinters=true

#### DO ALL THESE ON THE CONTROLLER
source ~/keystonerc_admin
neutron net-create priv_net
Created a new network:
+---------------------------+--------------------------------------+
| Field                     | Value                                |
+---------------------------+--------------------------------------+
| admin_state_up            | True                                 |
| id                        | 1cf06d0d-7bda-4665-90ea-92faf11071ca |
| name                      | priv_net                             |
| provider:network_type     | vlan                                 |
| provider:physical_network | inter-vlan                           |
| provider:segmentation_id  | 1200                                 |
| shared                    | False                                |
| status                    | ACTIVE                               |
| subnets                   |                                      |
| tenant_id                 | 1fbb0466632b42008ffc9e7a2f7f0f6f     |
+---------------------------+--------------------------------------+

neutron subnet-create priv_net 10.0.0.0/24 --name priv_subnet
```

```
Created a new subnet:
c
| Field            | Value                                        |
+------------------+----------------------------------------------+
| allocation_pools | {"start": "10.0.0.2", "end": "10.0.0.254"} |
| cidr             | 10.0.0.0/24                                  |
| dns_nameservers  |                                              |
| enable_dhcp      | True                                         |
| gateway_ip       | 10.0.0.1                                     |
| host_routes      |                                              |
| id               | 81523105-f9e0-40c4-9b6f-6eefd216c712         |
| ip_version       | 4                                            |
| name             | priv_subnet                                  |
| network_id       | 1cf06d0d-7bda-4665-90ea-92faf11071ca         |
| tenant_id        | 1fbb0466632b42008ffc9e7a2f7f0f6f             |
+------------------+----------------------------------------------+

neutron net-create ext_net --provider:network_type vlan --provider:physical_network
inter-vlan --provider:segmentation_id 1000 --router:external=True
Created a new network:
+---------------------------+--------------------------------------+
| admin_state_up            | True                                 |
| id                        | aab3138e-dbc6-452a-ba00-4c615a928988 |
| name                      | ext_net                              |
| provider:network_type     | vlan                                 |
| provider:physical_network | inter-vlan                           |
| provider:segmentation_id  | 1000                                 |
| router:external           | True                                 |
| shared                    | False                                |
| status                    | ACTIVE                               |
| subnets                   |                                      |
| tenant_id                 | 1fbb0466632b42008ffc9e7a2f7f0f6f     |
+---------------------------+--------------------------------------+

neutron subnet-create ext_net --allocation-pool start=10.35.1.100,end=10.35.1.150 --
gateway 10.35.1.1 10.35.1.0/24 -- --enable_dhcp=False
Created a new subnet:
+------------------+--------------------------------------------------+
| Field            | Value                                            |
+------------------+--------------------------------------------------+
| allocation_pools | {"start": "10.35.1.100", "end": "10.35.1.150"} |
| cidr             | 10.35.1.0/24                                     |
| dns_nameservers  |                                                  |
| enable_dhcp      | False                                            |
| gateway_ip       | 10.35.1.1                                        |
| host_routes      |                                                  |
| id               | 93788a04-e9c2-4752-86a9-59a201925230             |
| ip_version       | 4                                                |
| name             |                                                  |
| network_id       | aab3138e-dbc6-452a-ba00-4c615a928988             |
| tenant_id        | 1fbb0466632b42008ffc9e7a2f7f0f6f                 |
+------------------+--------------------------------------------------+


neutron router-create router1
Created a new router:
+-----------------------+--------------------------------------+
```

```
| Field                 | Value                                |
+-----------------------+--------------------------------------+
| admin_state_up        | True                                 |
| external_gateway_info |                                      |
| id                    | 0cab02ce-800b-43b0-90c9-cc202dd19b72 |
| name                  | router1                              |
| status                | ACTIVE                               |
| tenant_id             | 1fbb0466632b42008ffc9e7a2f7f0f6f     |
+-----------------------+--------------------------------------+

neutron router-gateway-set router1 ext_net
Set gateway for router router1

neutron router-interface-add router1 priv_subnet
Added interface 51ce34ea-3980-47bf-8ece-f4b2ce17fdcd to router router1.

neutron security-group-rule-create --protocol icmp --direction ingress default
Created a new security_group_rule:
+-------------------+--------------------------------------+
| Field             | Value                                |
+-------------------+--------------------------------------+
| direction         | ingress                              |
| ethertype         | IPv4                                 |
| id                | ac2e33ca-741e-4b59-aec7-830a6332b79a |
| port_range_max    |                                      |
| port_range_min    |                                      |
| protocol          | icmp                                 |
| remote_group_id   |                                      |
| remote_ip_prefix  |                                      |
| security_group_id | ff8c2ff5-a9db-4ec0-bf3e-d0ac249d8fe4 |
| tenant_id         | 1fbb0466632b42008ffc9e7a2f7f0f6f     |
+-------------------+--------------------------------------+

neutron security-group-rule-create --protocol tcp --port-range-min 22 --port-range-max 22
--direction ingress default
Created a new security_group_rule:
+-------------------+--------------------------------------+
| Field             | Value                                |
+-------------------+--------------------------------------+
| direction         | ingress                              |
| ethertype         | IPv4                                 |
| id                | 29e3394f-6e53-4b9d-87cc-e0cbf6c3dbb7 |
| port_range_max    | 22                                   |
| port_range_min    | 22                                   |
| protocol          | tcp                                  |
| remote_group_id   |                                      |
| remote_ip_prefix  |                                      |
| security_group_id | ff8c2ff5-a9db-4ec0-bf3e-d0ac249d8fe4 |
| tenant_id         | 1fbb0466632b42008ffc9e7a2f7f0f6f     |
+-------------------+--------------------------------------+


rm -f floatingip_list.txt; for i in `seq 1 16`; do neutron floatingip-create ext_net |
awk "/floating_ip_address/{print \$4\"\tvm$i\"}" | tee -a floatingip_list.txt ; done
10.35.1.101 vm1
10.35.1.102 vm2
10.35.1.103 vm3
```

```
10.35.1.104 vm4
10.35.1.105 vm5
10.35.1.106 vm6
10.35.1.107 vm7
10.35.1.108 vm8
10.35.1.109 vm9
10.35.1.110 vm10
10.35.1.111 vm11
10.35.1.112 vm12
10.35.1.113 vm13
10.35.1.114 vm14
10.35.1.115 vm15
10.35.1.116 vm16

cat floatingip_list.txt >> /etc/hosts

#### PREPARE KEYS (on the controller only)
nova keypair-add GUEST_KEY > GUEST_KEY.pem && chmod 600 GUEST_KEY.pem
```

### Configuring Availability Zones

```
#### DO ALL THESE ON THE CONTROLLER
nova availability-zone-list

nova aggregate-create compaggr1 compzone1
nova aggregate-create compaggr2 compzone2
nova aggregate-create compaggr3 compzone3
nova aggregate-create compaggr4 compzone4

nova aggregate-add-host compaggr1 compute1.test.lan
nova aggregate-add-host compaggr2 compute2.test.lan
nova aggregate-add-host compaggr3 compute3.test.lan
nova aggregate-add-host compaggr4 compute4.test.lan

nova availability-zone-list
```

### Uploading image and creating flavor

```
glance image-create --name 'rhel-6.4-smallfile' --disk-format qcow2 --container-format
bare --min-disk 4 --min-ram 256 --is-public True --copy-from
http://<imgserver>/share/Linux/RHEL/6.4/rhel64-smallfile.img --checksum
0c115efc9dca2fc157b03b990522ab0f

nova flavor-create m1.smallfile 100 4096 5 1
```

## Updates

### Updating KVM

Update these packages on the compute nodes:

```
yum install -y kvm_update/qemu-*.rpm

Installed:
  qemu-img-rhev.x86_64 2:0.12.1.2-2.411.el6
  qemu-kvm-rhev.x86_64 2:0.12.1.2-2.411.el6
  qemu-kvm-rhev-tools.x86_64 2:0.12.1.2-2.411.el6

Replaced:
```

```
  qemu-img.x86_64 2:0.12.1.2-2.398.el6
  qemu-kvm.x86_64 2:0.12.1.2-2.398.el6
```

### Updating Gluster Client

Update these packages on the compute nodes:

```
umount -a -t fuse.glusterfs
yum install -y gluster_update/glusterfs*rpm

Installed:
  glusterfs-libs-3.4.0.34rhs-1.el6.x86_64
  glusterfs-3.4.0.34rhs-1.el6.x86_64
  glusterfs-api-3.4.0.34rhs-1.el6.x86_64
  glusterfs-fuse-3.4.0.34rhs-1.el6.x86_64
```

## Configuring Cinder for Red Hat Storage

```
#### ON CONTROLLER
source ~/keystonerc_admin
openstack-config --set /etc/cinder/cinder.conf DEFAULT enabled_backends GLUSTER
openstack-config --set /etc/cinder/cinder.conf GLUSTER volume_backend_name GLUSTER
openstack-config --set /etc/cinder/cinder.conf GLUSTER volume_driver
cinder.volume.drivers.glusterfs.GlusterfsDriver
openstack-config --set /etc/cinder/cinder.conf GLUSTER glusterfs_shares_config
/etc/cinder/glusterfs_shares

echo "storage1:/rhosfs" > /etc/cinder/glusterfs_shares

#### Do these 2 commands on all compute nodes and controller/cinder ####
openstack-config --set /etc/nova/nova.conf DEFAULT qemu_allowed_storage_drivers gluster
openstack-config --set /etc/nova/nova.conf DEFAULT debug False

#### ON CONTROLLER
for i in api scheduler volume; do sudo service openstack-cinder-${i} stop; done
for i in api scheduler volume; do sudo service openstack-cinder-${i} start; done

cinder type-create gluster
cinder type-key gluster set volume_backend_name=GLUSTER
cinder extra-specs-list
```

## Installing Ceph Storage and configuring Cinder for Ceph Storage

```
#### Install repos on ceph monitor and all storage nodes then install main ceph packages
yum install -y http://ceph.com/rpm-dumpling/el6/noarch/ceph-release-1-0.el6.noarch.rpm
yum install -y ceph

#### on ceph monitor (hostname: cephmon)
yum install -y ceph-deploy

ceph-deploy new cephmon
ceph-deploy mon create cephmon
ceph-deploy gatherkeys cephmon

ceph-deploy disk list storage{1,2,3,4}
ceph-deploy disk zap storage{1,2,3,4}:sd{a,b,c,d,e,f,g,h}
ceph-deploy osd create storage{1,2,3,4}:sd{a,b,c,d,e,f,g,h}
```

```
ceph-deploy disk list storage{1,2,3,4}
```

```
[ceph_deploy.cli][INFO  ] Invoked (1.2.7): /usr/bin/ceph-deploy disk list storage1
storage2 storage3 storage4
[ceph_deploy.sudo_pushy][DEBUG ] will use a remote connection without sudo
[ceph_deploy.osd][INFO  ] Distro info: RedHatEnterpriseServer 6.4 Santiago
[ceph_deploy.osd][DEBUG ] Listing disks on storage1...
[storage1][INFO  ] Running command: ceph-disk list
[storage1][INFO  ] /dev/sda :
[storage1][INFO  ]  /dev/sda1 ceph data, active, cluster ceph, osd.0, journal /dev/sda2
[storage1][INFO  ]  /dev/sda2 ceph journal, for /dev/sda1
[storage1][INFO  ] /dev/sdb :
[storage1][INFO  ]  /dev/sdb1 ceph data, active, cluster ceph, osd.1, journal /dev/sdb2
[storage1][INFO  ]  /dev/sdb2 ceph journal, for /dev/sdb1
[storage1][INFO  ] /dev/sdc :
[storage1][INFO  ]  /dev/sdc1 ceph data, active, cluster ceph, osd.2, journal /dev/sdc2
[storage1][INFO  ]  /dev/sdc2 ceph journal, for /dev/sdc1
[storage1][INFO  ] /dev/sdd :
[storage1][INFO  ]  /dev/sdd1 ceph data, active, cluster ceph, osd.3, journal /dev/sdd2
[storage1][INFO  ]  /dev/sdd2 ceph journal, for /dev/sdd1
[storage1][INFO  ] /dev/sde :
[storage1][INFO  ]  /dev/sde1 ceph data, active, cluster ceph, osd.4, journal /dev/sde2
[storage1][INFO  ]  /dev/sde2 ceph journal, for /dev/sde1
[storage1][INFO  ] /dev/sdf :
[storage1][INFO  ]  /dev/sdf1 ceph data, active, cluster ceph, osd.5, journal /dev/sdf2
[storage1][INFO  ]  /dev/sdf2 ceph journal, for /dev/sdf1
[storage1][INFO  ] /dev/sdg :
[storage1][INFO  ]  /dev/sdg1 ceph data, active, cluster ceph, osd.6, journal /dev/sdg2
[storage1][INFO  ]  /dev/sdg2 ceph journal, for /dev/sdg1
[storage1][INFO  ] /dev/sdh :
[storage1][INFO  ]  /dev/sdh1 ceph data, active, cluster ceph, osd.7, journal /dev/sdh2
[storage1][INFO  ]  /dev/sdh2 ceph journal, for /dev/sdh1
[storage1][INFO  ] /dev/sdi other, isw_raid_member
[storage1][INFO  ] /dev/sdj other, isw_raid_member
[ceph_deploy.sudo_pushy][DEBUG ] will use a remote connection without sudo
[ceph_deploy.osd][INFO  ] Distro info: RedHatEnterpriseServer 6.4 Santiago
[ceph_deploy.osd][DEBUG ] Listing disks on storage2...
[storage2][INFO  ] Running command: ceph-disk list
[storage2][INFO  ] /dev/sda :
[storage2][INFO  ]  /dev/sda1 ceph data, active, cluster ceph, osd.8, journal /dev/sda2
[storage2][INFO  ]  /dev/sda2 ceph journal, for /dev/sda1
[storage2][INFO  ] /dev/sdb :
[storage2][INFO  ]  /dev/sdb1 ceph data, active, cluster ceph, osd.9, journal /dev/sdb2
[storage2][INFO  ]  /dev/sdb2 ceph journal, for /dev/sdb1
[storage2][INFO  ] /dev/sdc :
[storage2][INFO  ]  /dev/sdc1 ceph data, active, cluster ceph, osd.10, journal /dev/sdc2
[storage2][INFO  ]  /dev/sdc2 ceph journal, for /dev/sdc1
[storage2][INFO  ] /dev/sdd :
[storage2][INFO  ]  /dev/sdd1 ceph data, active, cluster ceph, osd.11, journal /dev/sdd2
[storage2][INFO  ]  /dev/sdd2 ceph journal, for /dev/sdd1
[storage2][INFO  ] /dev/sde :
[storage2][INFO  ]  /dev/sde1 ceph data, active, cluster ceph, osd.12, journal /dev/sde2
[storage2][INFO  ]  /dev/sde2 ceph journal, for /dev/sde1
[storage2][INFO  ] /dev/sdf :
[storage2][INFO  ]  /dev/sdf1 ceph data, active, cluster ceph, osd.13, journal /dev/sdf2
[storage2][INFO  ]  /dev/sdf2 ceph journal, for /dev/sdf1
[storage2][INFO  ] /dev/sdg :
```

```
[storage2][INFO  ]  /dev/sdg1 ceph data, active, cluster ceph, osd.14, journal /dev/sdg2
[storage2][INFO  ]  /dev/sdg2 ceph journal, for /dev/sdg1
[storage2][INFO  ] /dev/sdh :
[storage2][INFO  ]  /dev/sdh1 ceph data, active, cluster ceph, osd.15, journal /dev/sdh2
[storage2][INFO  ]  /dev/sdh2 ceph journal, for /dev/sdh1
[storage2][INFO  ] /dev/sdi other, isw_raid_member
[storage2][INFO  ] /dev/sdj other, isw_raid_member
[ceph_deploy.sudo_pushy][DEBUG ] will use a remote connection without sudo
[ceph_deploy.osd][INFO  ] Distro info: RedHatEnterpriseServer 6.4 Santiago
[ceph_deploy.osd][DEBUG ] Listing disks on storage3...
[storage3][INFO  ] Running command: ceph-disk list
[storage3][INFO  ] /dev/sda :
[storage3][INFO  ]  /dev/sda1 ceph data, active, cluster ceph, osd.16, journal /dev/sda2
[storage3][INFO  ]  /dev/sda2 ceph journal, for /dev/sda1
[storage3][INFO  ] /dev/sdb :
[storage3][INFO  ]  /dev/sdb1 ceph data, active, cluster ceph, osd.17, journal /dev/sdb2
[storage3][INFO  ]  /dev/sdb2 ceph journal, for /dev/sdb1
[storage3][INFO  ] /dev/sdc :
[storage3][INFO  ]  /dev/sdc1 ceph data, active, cluster ceph, osd.18, journal /dev/sdc2
[storage3][INFO  ]  /dev/sdc2 ceph journal, for /dev/sdc1
[storage3][INFO  ] /dev/sdd :
[storage3][INFO  ]  /dev/sdd1 ceph data, active, cluster ceph, osd.19, journal /dev/sdd2
[storage3][INFO  ]  /dev/sdd2 ceph journal, for /dev/sdd1
[storage3][INFO  ] /dev/sde :
[storage3][INFO  ]  /dev/sde1 ceph data, active, cluster ceph, osd.20, journal /dev/sde2
[storage3][INFO  ]  /dev/sde2 ceph journal, for /dev/sde1
[storage3][INFO  ] /dev/sdf :
[storage3][INFO  ]  /dev/sdf1 ceph data, active, cluster ceph, osd.21, journal /dev/sdf2
[storage3][INFO  ]  /dev/sdf2 ceph journal, for /dev/sdf1
[storage3][INFO  ] /dev/sdg :
[storage3][INFO  ]  /dev/sdg1 ceph data, active, cluster ceph, osd.22, journal /dev/sdg2
[storage3][INFO  ]  /dev/sdg2 ceph journal, for /dev/sdg1
[storage3][INFO  ] /dev/sdh :
[storage3][INFO  ]  /dev/sdh1 ceph data, active, cluster ceph, osd.23, journal /dev/sdh2
[storage3][INFO  ]  /dev/sdh2 ceph journal, for /dev/sdh1
[storage3][INFO  ] /dev/sdi other, isw_raid_member
[storage3][INFO  ] /dev/sdj other, isw_raid_member
[ceph_deploy.sudo_pushy][DEBUG ] will use a remote connection without sudo
[ceph_deploy.osd][INFO  ] Distro info: RedHatEnterpriseServer 6.4 Santiago
[ceph_deploy.osd][DEBUG ] Listing disks on storage4...
[storage4][INFO  ] Running command: ceph-disk list
[storage4][INFO  ] /dev/sda :
[storage4][INFO  ]  /dev/sda1 ceph data, active, cluster ceph, osd.24, journal /dev/sda2
[storage4][INFO  ]  /dev/sda2 ceph journal, for /dev/sda1
[storage4][INFO  ] /dev/sdb :
[storage4][INFO  ]  /dev/sdb1 ceph data, active, cluster ceph, osd.25, journal /dev/sdb2
[storage4][INFO  ]  /dev/sdb2 ceph journal, for /dev/sdb1
[storage4][INFO  ] /dev/sdc :
[storage4][INFO  ]  /dev/sdc1 ceph data, active, cluster ceph, osd.26, journal /dev/sdc2
[storage4][INFO  ]  /dev/sdc2 ceph journal, for /dev/sdc1
[storage4][INFO  ] /dev/sdd :
[storage4][INFO  ]  /dev/sdd1 ceph data, active, cluster ceph, osd.27, journal /dev/sdd2
[storage4][INFO  ]  /dev/sdd2 ceph journal, for /dev/sdd1
[storage4][INFO  ] /dev/sde :
[storage4][INFO  ]  /dev/sde1 ceph data, active, cluster ceph, osd.28, journal /dev/sde2
[storage4][INFO  ]  /dev/sde2 ceph journal, for /dev/sde1
[storage4][INFO  ] /dev/sdf :
```

```
[storage4][INFO  ]  /dev/sdf1 ceph data, active, cluster ceph, osd.29, journal /dev/sdf2
[storage4][INFO  ]  /dev/sdf2 ceph journal, for /dev/sdf1
[storage4][INFO  ] /dev/sdg :
[storage4][INFO  ]  /dev/sdg1 ceph data, active, cluster ceph, osd.30, journal /dev/sdg2
[storage4][INFO  ]  /dev/sdg2 ceph journal, for /dev/sdg1
[storage4][INFO  ] /dev/sdh :
[storage4][INFO  ]  /dev/sdh1 ceph data, active, cluster ceph, osd.31, journal /dev/sdh2
[storage4][INFO  ]  /dev/sdh2 ceph journal, for /dev/sdh1
[storage4][INFO  ] /dev/sdi other, isw_raid_member
[storage4][INFO  ] /dev/sdj other, isw_raid_member
```

### Configuring Ceph pools for OpenStack

```
ceph osd pool create volumes 1600
ceph osd pool create images 128
```

### Configuring Cinder for Ceph Storage

```
#### all openstack systems - controller, compute
yum install -y http://ceph.com/rpm-dumpling/el6/noarch/ceph-release-1-0.el6.noarch.rpm

#### on cinder server
yum install -y ceph
mkdir /usr/lib64/qemu
ln -s /usr/lib64/librbd.so.1 /usr/lib64/qemu/librbd.so.1

#### on compute nodes
yum install -y librbd1
mkdir /usr/lib64/qemu
ln -s /usr/lib64/librbd.so.1 /usr/lib64/qemu/librbd.so.1

#### on ceph monitor
ssh rdo-cont tee /etc/ceph/ceph.conf </etc/ceph/ceph.conf
for i in `seq 1 4`; do ssh compute$i tee /etc/ceph/ceph.conf </etc/ceph/ceph.conf ; done
ceph auth get-or-create client.volumes mon 'allow r' osd 'allow class-read object_prefix
rbd_children, allow rwx pool=volumes, allow rx pool=images'
ceph auth get-or-create client.images mon 'allow r' osd 'allow class-read object_prefix
rbd_children, allow rwx pool=images'

ceph auth get-or-create client.images | ssh rdo-cont tee
/etc/ceph/ceph.client.images.keyring
ssh rdo-cont chown glance:glance /etc/ceph/ceph.client.images.keyring
ceph auth get-or-create client.volumes | ssh rdo-cont tee
/etc/ceph/ceph.client.volumes.keyring
ssh rdo-cont chown cinder:cinder /etc/ceph/ceph.client.volumes.keyring

ceph auth get-key client.volumes | tee client.volumes.key

uuidgen > secret.uuid
cat > secret.xml <<EOF
<secret ephemeral='no' private='no'>
  <uuid>`cat secret.uuid`</uuid>
  <usage type='ceph'>
    <name>client.volumes secret</name>
  </usage>
</secret>
EOF
```

```
for i in `seq 1 4`; do cat secret.xml | ssh compute$i "tee secret.xml ; virsh secret-
define --file secret.xml ; virsh secret-set-value --secret $(cat secret.uuid) --base64
$(cat client.volumes.key) && rm client.volumes.key secret.xml" ; done


#### on cinder server
source ~/keystonerc_admin
for i in api scheduler volume; do sudo service openstack-cinder-${i} stop; done

openstack-config --set /etc/cinder/cinder.conf DEFAULT enabled_backends CEPH
openstack-config --set /etc/cinder/cinder.conf CEPH volume_backend_name CEPH
openstack-config --set /etc/cinder/cinder.conf CEPH volume_driver
cinder.volume.drivers.rbd.RBDDriver
openstack-config --set /etc/cinder/cinder.conf CEPH rbd_pool volumes
openstack-config --set /etc/cinder/cinder.conf CEPH glance_api_version 2
openstack-config --set /etc/cinder/cinder.conf CEPH rbd_user volumes
openstack-config --set /etc/cinder/cinder.conf CEPH rbd_secret_uuid `cat secret.uuid`

openstack-config --set /etc/nova/nova.conf DEFAULT disk_cachemodes
"'network=writethrough'"
openstack-config --del /etc/nova/nova.conf DEFAULT disk_cachemodes
service openstack-nova-compute restart


openstack-config --set /etc/ceph/ceph.conf client rbd_cache_max_dirty 0

for i in api scheduler volume; do sudo service openstack-cinder-${i} start; done

cinder type-create ceph
cinder type-key ceph set volume_backend_name=CEPH
cinder extra-specs-list
```

### Ceph crush map

```
# begin crush map

# devices
device 0 osd.0
device 1 osd.1
device 2 osd.2
device 3 osd.3
device 4 osd.4
device 5 osd.5
device 6 osd.6
device 7 osd.7
device 8 osd.8
device 9 osd.9
device 10 osd.10
device 11 osd.11
device 12 osd.12
device 13 osd.13
device 14 osd.14
device 15 osd.15
device 16 osd.16
device 17 osd.17
device 18 osd.18
```

```
device 19 osd.19
device 20 osd.20
device 21 osd.21
device 22 osd.22
device 23 osd.23
device 24 osd.24
device 25 osd.25
device 26 osd.26
device 27 osd.27
device 28 osd.28
device 29 osd.29
device 30 osd.30
device 31 osd.31

# types
type 0 osd
type 1 host
type 2 rack
type 3 row
type 4 room
type 5 datacenter
type 6 root

# buckets
host storage1 {
      id -2       # do not change unnecessarily
      # weight 4.320
      alg straw
      hash 0      # rjenkins1
      item osd.0 weight 0.540
      item osd.1 weight 0.540
      item osd.2 weight 0.540
      item osd.3 weight 0.540
      item osd.4 weight 0.540
      item osd.5 weight 0.540
      item osd.6 weight 0.540
      item osd.7 weight 0.540
}
host storage2 {
      id -3       # do not change unnecessarily
      # weight 4.320
      alg straw
      hash 0      # rjenkins1
      item osd.8 weight 0.540
      item osd.9 weight 0.540
      item osd.10 weight 0.540
      item osd.11 weight 0.540
      item osd.12 weight 0.540
      item osd.13 weight 0.540
      item osd.14 weight 0.540
      item osd.15 weight 0.540
}
host storage3 {
      id -4       # do not change unnecessarily
      # weight 4.320
      alg straw
      hash 0      # rjenkins1
```

```
      item osd.16 weight 0.540
      item osd.17 weight 0.540
      item osd.18 weight 0.540
      item osd.19 weight 0.540
      item osd.20 weight 0.540
      item osd.21 weight 0.540
      item osd.22 weight 0.540
      item osd.23 weight 0.540
}
host storage4 {
      id -5       # do not change unnecessarily
      # weight 4.320
      alg straw
      hash 0      # rjenkins1
      item osd.24 weight 0.540
      item osd.25 weight 0.540
      item osd.26 weight 0.540
      item osd.27 weight 0.540
      item osd.28 weight 0.540
      item osd.29 weight 0.540
      item osd.30 weight 0.540
      item osd.31 weight 0.540
}
root default {
      id -1       # do not change unnecessarily
      # weight 17.280
      alg straw
      hash 0      # rjenkins1
      item storage1 weight 4.320
      item storage2 weight 4.320
      item storage3 weight 4.320
      item storage4 weight 4.320
}

# rules
rule data {
      ruleset 0
      type replicated
      min_size 1
      max_size 10
      step take default
      step chooseleaf firstn 0 type host
      step emit
}
rule metadata {
      ruleset 1
      type replicated
      min_size 1
      max_size 10
      step take default
      step chooseleaf firstn 0 type host
      step emit
}
rule rbd {
      ruleset 2
      type replicated
      min_size 1
```

```
      max_size 10
      step take default
      step chooseleaf firstn 0 type host
      step emit
}

# end crush map
```

## Starting VMs and creating cinder disks

Below, we include a sample script to start VMs and create cinder volumes.

```
#!/bin/bash
COUNT=0
COMPUTE_NODES=4
if [ "${2}" = "" ]; then
  COUNT=${1:-0}
else
  COMPUTE_NODES=${1:-1}
  COUNT=${2}
fi

DELAY=1
#VOLTYPE=gluster
VOLTYPE=ceph
VOLSIZE=40

PRIV_NET_ID=`neutron net-list -F id -F name -f csv --quote none | awk -F',' 
'/priv_net/{print $1}'`

for i in `seq 1 $COUNT`;
do
  VM=vm$i
  nova boot --image rhel-6.4-smallfile --flavor m1.smallfile --key_name GUEST_KEY --
availability-zone compzone`expr \( ${i} - 1 \) % ${COMPUTE_NODES} + 1`  --nic net-
id=${PRIV_NET_ID} ${VM}
  sleep $DELAY
  DEVICE_ID=`nova list --name $VM | awk "/$VM/{ print \\$2 }"`
  #sleep $DELAY
  PORT_ID=`neutron port-list -- --device_id ${DEVICE_ID} | awk '/ip_address/{print $2}'`
  FLOATIP=`resolveip -s $VM`
  FLOATING_ID=`neutron floatingip-list | awk "/$FLOATIP/{ print \\$2 }"`
  until [ `nova list --name $VM | awk "/$VM/{ print \\$6 }"` = ACTIVE ]; do
    sleep $DELAY
  done
  neutron floatingip-associate $FLOATING_ID $PORT_ID
  sleep $DELAY
  echo -n "Pinging ${VM}..."
  until ping -qc 1 ${VM} 1> /dev/null 2> /dev/null ;
  do
    sleep 1
    echo -n "."
  done
  echo "success!"
done

sleep 30
```

```
for i in `seq 1 $COUNT`;
do
  VM=vm$i
  sleep $DELAY
  until [ `jobs | wc -l` -lt 2 ]; do
    sleep 1
  done
  if [ "`cinder list --display-name ${VM}_${VOLTYPE}vol | awk '/available/{print $2}'`" =
"" ]; then
    cinder create --volume_type $VOLTYPE --display_name ${VM}_${VOLTYPE}vol $VOLSIZE
    sleep $DELAY
    while [ "`cinder list --display-name ${VM}_${VOLTYPE}vol | awk '/available/{print
$2}'`" = "" ]; do
      sleep 1
    done
    nova volume-attach ${VM} `cinder list --display-name ${VM}_${VOLTYPE}vol | awk
'/available/{print $2}'` /dev/vdb
    sleep $DELAY
    BLOCKS=`expr $VOLSIZE \* 1024`
    sleep $DELAY
    ssh -i GUEST_KEY.pem -o StrictHostKeyChecking=no ${VM} "dd if=/dev/zero of=/dev/vdb
bs=1M count=$BLOCKS ; sync"
    ./run_all_storage.sh "sync"
    sleep $DELAY
  else
    nova volume-attach ${VM} `cinder list --display-name ${VM}_${VOLTYPE}vol | awk
'/available/{print $2}'` /dev/vdb
  fi
done
wait
```

## Installing and configuring the base VM

Our VM base image used a 4GB qcow2 virtual disk, 4,096 MB of RAM, and two vCPUs. We installed Red Hat Enterprise Linux 6.4, chose to connect the virtual NIC automatically, and performed a custom disk layout. The disk layout configuration included one standard partition where we used the whole disk with the ext4 filesystem, mountpoint=/, and no swap. We chose the minimal package selection on installation. Below we show the additional steps we followed for the base VM, as well as installed and updated packages.

```
#### Disable selinux.
sed -i 's/SELINUX=.*/SELINUX=disabled/' /etc/selinux/config

#### Disable firewall.
iptables -F
/etc/init.d/iptables save
chkconfig iptables off

#### Subscribe system to RHN:
subscription-manager register
subscription-manager refresh

#### Install updates:
yum update -y
```

```
Installed:
  kernel.x86_64 0:2.6.32-358.18.1.el6

Updated:
  bash.x86_64 0:4.1.2-15.el6_4
  chkconfig.x86_64 0:1.3.49.3-2.el6_4.1
  coreutils.x86_64 0:8.4-19.el6_4.2
  coreutils-libs.x86_64 0:8.4-19.el6_4.2
  curl.x86_64 0:7.19.7-37.el6_4
  db4.x86_64 0:4.7.25-18.el6_4
  db4-utils.x86_64 0:4.7.25-18.el6_4
  dbus-glib.x86_64 0:0.86-6.el6_4
  dhclient.x86_64 12:4.1.1-34.P1.el6_4.1
  dhcp-common.x86_64 12:4.1.1-34.P1.el6_4.1
  dmidecode.x86_64 1:2.11-2.el6_1
  e2fsprogs.x86_64 0:1.41.12-14.el6_4.2
  e2fsprogs-libs.x86_64 0:1.41.12-14.el6_4.2
  glibc.x86_64 0:2.12-1.107.el6_4.4
  glibc-common.x86_64 0:2.12-1.107.el6_4.4
  gzip.x86_64 0:1.3.12-19.el6_4
  initscripts.x86_64 0:9.03.38-1.el6_4.2
  iputils.x86_64 0:20071127-17.el6_4.2
  kernel-firmware.noarch 0:2.6.32-358.18.1.el6
  krb5-libs.x86_64 0:1.10.3-10.el6_4.6
  libblkid.x86_64 0:2.17.2-12.9.el6_4.3
  libcom_err.x86_64 0:1.41.12-14.el6_4.2
  libcurl.x86_64 0:7.19.7-37.el6_4
  libnl.x86_64 0:1.1.4-1.el6_4
  libselinux.x86_64 0:2.0.94-5.3.el6_4.1
  libselinux-utils.x86_64 0:2.0.94-5.3.el6_4.1
  libss.x86_64 0:1.41.12-14.el6_4.2
  libuuid.x86_64 0:2.17.2-12.9.el6_4.3
  libxml2.x86_64 0:2.7.6-12.el6_4.1
  libxml2-python.x86_64 0:2.7.6-12.el6_4.1
  module-init-tools.x86_64 0:3.9-21.el6_4
  mysql-libs.x86_64 0:5.1.69-1.el6_4
  nspr.x86_64 0:4.9.5-2.el6_4
  nss.x86_64 0:3.14.3-4.el6_4
  nss-softokn.x86_64 0:3.14.3-3.el6_4
  nss-softokn-freebl.x86_64 0:3.14.3-3.el6_4
  nss-sysinit.x86_64 0:3.14.3-4.el6_4
  nss-tools.x86_64 0:3.14.3-4.el6_4
  nss-util.x86_64 0:3.14.3-3.el6_4
  openldap.x86_64 0:2.4.23-32.el6_4.1
  openssl.x86_64 0:1.0.0-27.el6_4.2
  python.x86_64 0:2.6.6-37.el6_4
  python-dmidecode.x86_64 0:3.10.13-3.el6_4
  python-libs.x86_64 0:2.6.6-37.el6_4
  python-rhsm.x86_64 0:1.8.17-1.el6_4
  rhn-check.noarch 0:1.0.0.1-8.el6
  rhn-client-tools.noarch 0:1.0.0.1-8.el6
  rhn-setup.noarch 0:1.0.0.1-8.el6
  rhnlib.noarch 0:2.5.22-15.el6
  rsyslog.x86_64 0:5.8.10-7.el6_4
  selinux-policy.noarch 0:3.7.19-195.el6_4.12
  selinux-policy-targeted.noarch 0:3.7.19-195.el6_4.12
  setup.noarch 0:2.8.14-20.el6_4.1
```

```
  subscription-manager.x86_64 0:1.8.22-1.el6_4
  tzdata.noarch 0:2013c-2.el6
  upstart.x86_64 0:0.6.5-12.el6_4.1
  util-linux-ng.x86_64 0:2.17.2-12.9.el6_4.3
  yum-rhn-plugin.noarch 0:0.9.1-49.el6

#### after updates reboot
reboot

#### Edit /boot/grub/grub.conf
vi /boot/grub/grub.conf

# grub.conf generated by anaconda
#
# Note that you do not have to rerun grub after making changes to this file
# NOTICE:  You do not have a /boot partition.  This means that
#          all kernel and initrd paths are relative to /, eg.
#          root (hd0,0)
#          kernel /boot/vmlinuz-version ro root=/dev/vda1
#          initrd /boot/initrd-[generic-]version.img
#boot=/dev/vda
default=0
timeout=3
splashimage=(hd0,0)/boot/grub/splash.xpm.gz
hiddenmenu
title Red Hat Enterprise Linux Server (2.6.32-358.18.1.el6.x86_64)
        root (hd0,0)
        kernel /boot/vmlinuz-2.6.32-358.18.1.el6.x86_64 ro root=/dev/vda1 console=tty0
console=ttyS0
        initrd /boot/initramfs-2.6.32-358.18.1.el6.x86_64.img
title Red Hat Enterprise Linux (2.6.32-358.el6.x86_64)
        root (hd0,0)
        kernel /boot/vmlinuz-2.6.32-358.el6.x86_64 ro root=/dev/vda1 console=tty0
console=ttyS0
        initrd /boot/initramfs-2.6.32-358.el6.x86_64.img

#### Reboot after grub edit
reboot

#### Install additional packages:
yum install -y acpid openssh-clients sysstat tuned vim wget

Installed:
  acpid.x86_64 0:1.0.10-2.1.el6
  openssh-clients.x86_64 0:5.3p1-84.1.el6
  sysstat.x86_64 0:9.0.4-20.el6
  tuned.noarch 0:0.2.19-11.el6.1
  vim-enhanced.x86_64 2:7.2.411-1.8.el6
  wget.x86_64 0:1.12-1.8.el6

Dependency Installed:
  gpm-libs.x86_64 0:1.20.6-12.el6
  libedit.x86_64 0:2.11-4.20080712cvs.1.el6
  perl.x86_64 4:5.10.1-131.el6_4
  perl-Module-Pluggable.x86_64 1:3.90-131.el6_4
  perl-Pod-Escapes.x86_64 1:1.04-131.el6_4
  perl-Pod-Simple.x86_64 1:3.13-131.el6_4
```

```
  perl-libs.x86_64 4:5.10.1-131.el6_4
  perl-version.x86_64 3:0.77-131.el6_4
  vim-common.x86_64 2:7.2.411-1.8.el6

#### Add Common channel:
rhn-channel --add --channel=rhel-x86_64-server-rh-common-6

#### Install and configure cloud-init packages:
yum install -y cloud-init

Installed:
  cloud-init.noarch 0:0.7.1-2.el6

Dependency Installed:
  PyYAML.x86_64 0:3.10-3.1.el6
  audit-libs-python.x86_64 0:2.2-2.el6
  libcgroup.x86_64 0:0.37-7.2.el6_4
  libselinux-python.x86_64 0:2.0.94-5.3.el6_4.1
  libsemanage-python.x86_64 0:2.0.43-4.2.el6
  libyaml.x86_64 0:0.1.3-1.1.el6
  make.x86_64 1:3.81-20.el6
  policycoreutils-python.x86_64 0:2.0.83-19.30.el6
  python-argparse.noarch 0:1.2.1-2.1.el6
  python-boto.noarch 0:2.5.2-1.1.el6
  python-cheetah.x86_64 0:2.4.1-1.el6
  python-configobj.noarch 0:4.6.0-3.el6
  python-markdown.noarch 0:2.0.1-3.1.el6
  python-prettytable.noarch 0:0.6.1-1.el6
  python-pygments.noarch 0:1.1.1-1.el6
  python-setuptools.noarch 0:0.6.10-3.el6
  setools-libs.x86_64 0:3.3.7-4.el6
  setools-libs-python.x86_64 0:3.3.7-4.el6

#### Edit datasources file
vim /etc/cloud/cloud.cfg.d/10_datasources.cfg

datasource:
  Ec2:
    timeout : 10
    max_wait : 30
    metadata_urls:
     - http://10.0.0.1:9697

  MAAS:
    timeout : 10
    max_wait : 30

  NoCloud:
    seedfrom: None


#### Edit cloud.cfg file
vim /etc/cloud/cloud.cfg

#### Change...
ssh_pwauth:   0
```

```
#### To...
ssh_pwauth:   1
datasource_list: ["ConfigDrive", "Ec2", "NoCloud"]


#### Install Git:
yum install -y git

Installed:
  git.x86_64 0:1.7.1-3.el6_4.1

Dependency Installed:
  perl-Error.noarch 1:0.17015-4.el6
  perl-Git.noarch 0:1.7.1-3.el6_4.1


#### Download smallfile:
git clone https://github.com/bengland2/smallfile.git
Initialized empty Git repository in /root/smallfile/.git/


#### Configure local NFS server (used for smallfile thread sync):
yum install -y nfs-utils
mkdir /mnt/nfsshare
mkdir /mnt/nfsexport

vim /etc/exports

/mnt/nfsshare   *(rw)


#### Disable NIC udev rename:
ln -s /dev/null /etc/udev/rules.d/75-persistent-net-generator.rules
rm -f /etc/udev/rules.d/70-persistent-net.rules
vim /etc/sysconfig/network-scripts/ifcfg-eth0

DEVICE=eth0
ONBOOT=yes
BOOTPROTO=dhcp
```

## Tuning the guest

```
vi /etc/tune-profile/virtual-guest/ktune.sh

#### Change this line:
multiply_disk_readahead 4

#### To:
multiply_disk_readahead 16

#### Apply profile:
tuned-adm profile virtual-guest

reboot
```

## Cleaning up and preparing for qcow2 compact

```
yum clean all
rm -rf /var/tmp/*
rm -rf /tmp/*
dd if=/dev/zero of=/zerofile.tmp bs=64k ; sync ; rm -f /zerofile.tmp ; sync
```

## Running the tests

Each VM used its own cinder volume, backed by either Red Hat Storage or Ceph Storage, for each test. We formatted the virtual disk using ext4. The disk was reformatted before running the tests for every VM/node count combination. All filesystem caches were cleared on the storage nodes, compute nodes, and VMs before each smallfile operation type.

```
mkfs.ext4 /dev/vdb
```

Example smallfile command:

python smallfile_cli.py --top /mnt/test --network-sync-dir /mnt/nfsexport/smf --remote-pgm-dir /root/smallfile --response-times Y --hash-into-dirs Y --file-size-distribution exponential --fsync Y --pause 10 --threads 4 --file-size 64 --files 32768 --host-set vm1 --operation create

We used the following three scripts to run ***smallfile*** tests on the VMs from a remote system.

prepare_vms_smallfile.sh :

```
#!/bin/bash
COUNT=${1:-1}
for i in `seq 1 $COUNT`;
do
  ssh -i GUEST_KEY.pem vm$i "sync ; umount /dev/vdb ; rmdir /mnt/test ; mkfs.ext4
/dev/vdb ; tune2fs -c -1 /dev/vdb ; sync ; mkdir /mnt/test ; mount /dev/vdb /m
nt/test ; sync "
  scp -i GUEST_KEY.pem -c arcfour -r /root/nfsrpms vm$i:/root/
  ssh -i GUEST_KEY.pem vm$i "rpm -ivh /root/nfsrpms/*.rpm ; sync ; rm -rf /root/nfsrpms"
  scp -i GUEST_KEY.pem -c arcfour -r /root/smallfile vm$i:/root/
done
```

smallfile_test.sh :

```
#!/bin/sh
export RSH="ssh -i GUEST_KEY.pem"
FILESIZE=${2}
FILES=${3}
VM_COUNT=${4}
COMPUTE_COUNT=${5}
THREAD_COUNT=${6:-1}
FSYNC=${7:-N}
PAUSE=${8:-10}
THREADS=`expr $VM_COUNT \* $THREAD_COUNT`
STORAGE_COUNT=4
INTERVAL=5
CEPH_MON=0

SMF_RUNPATH="/root/smallfile"
SMF_TESTDIR="/mnt/test"
SMF_NETSHARE="/mnt/nfsexport"
```

```
SMF_NETMOUNT="192.168.43.11:/mnt/nfsshare"
SMF_OPERATIONS="create append read rename delete-renamed"

if [ "${1}" = "ceph" ]; then
  CEPH_MON=1
fi

# Initialize SMF host list
SMF_HOSTS="vm1"
for i in `seq 2 ${VM_COUNT}`; do
  SMF_HOSTS="${SMF_HOSTS},vm$i"
done

# SMF command
SMF_CMD="python ${SMF_RUNPATH}/smallfile_cli.py --top ${SMF_TESTDIR} --network-sync-dir
${SMF_NETSHARE}/smf --remote-pgm-dir ${SMF_RUNPATH}  --response-times Y --hash-into-dirs
Y --file-size-distribution exponential --fsync ${FSYNC} --pause ${PAUSE} --threads
${THREAD_COUNT} --file-size ${FILESIZE} --files ${FILES} --host-set ${SMF_HOSTS}"

# Prepare and cleanup
umount ${SMF_NETSHARE} ; mkdir ${SMF_NETSHARE} ; mount -o actimeo=1 ${SMF_NETMOUNT}
${SMF_NETSHARE}
rm -rf ${SMF_TESTDIR}/file_srcdir ; rm -rf ${SMF_TESTDIR}/file_dstdir ; mkdir
${SMF_TESTDIR}/file_srcdir ; mkdir ${SMF_TESTDIR}/file_dstdir
for i in `seq 1 ${VM_COUNT}`; do
  ssh -i GUEST_KEY.pem vm$i "umount ${SMF_NETSHARE} ; mkdir ${SMF_NETSHARE} ; mount -o
actimeo=1 ${SMF_NETMOUNT} ${SMF_NETSHARE}"
  ssh -i GUEST_KEY.pem vm$i "rm -rf ${SMF_TESTDIR}/file_srcdir ; rm -rf
${SMF_TESTDIR}/file_dstdir ; mkdir ${SMF_TESTDIR}/file_srcdir ; mkdir
${SMF_TESTDIR}/file_dstdir"
done
rm -rf ${SMF_NETSHARE}/smf

# BEGIN OPERATION: Do everything for each operation type
for OPERATION in ${SMF_OPERATIONS}; do

# Prepare results folder and filename
SMF_RESULT="smallfile_${1}_${FILESIZE}k_${FILES}f_${OPERATION}_${COMPUTE_COUNT}HV_${VM_CO
UNT}VM_${THREADS}T"
RESULT_DIR=${SMF_RESULT}
mkdir ${RESULT_DIR}

# Cleanup and drop caches
for i in `seq 1 ${VM_COUNT}`; do
  ssh -i GUEST_KEY.pem vm${i} "sync ; echo 3 > /proc/sys/vm/drop_caches ; sync"
done

for i in `seq 1 ${COMPUTE_COUNT}`; do
  ssh compute${i} "sync ; echo 3 > /proc/sys/vm/drop_caches ; sync"
done

for i in `seq 1 ${STORAGE_COUNT}`; do
  ssh storage${i} "sync ; echo 3 > /proc/sys/vm/drop_caches ; sync"
done
sleep $INTERVAL

# Start statistics collection
```

```
echo Starting stat collection...
for i in `seq 1 ${VM_COUNT}`; do
  ssh -i GUEST_KEY.pem vm${i} "pkill vmstat ; vmstat -n ${INTERVAL} | sed -e 's/^[
\t]*//' -e '3d'" > ${RESULT_DIR}/vmstat_vm${i}.log &
  ssh -i GUEST_KEY.pem vm${i} "pkill sar ; rm -f /root/sar_*.bin ; sar -o sar_vm${i}.bin
${INTERVAL} > /dev/null" &
done

for i in `seq 1 ${COMPUTE_COUNT}`; do
  ssh compute${i} "pkill vmstat ; vmstat -n ${INTERVAL} | sed -e 's/^[ \t]*//' -e '3d'" >
${RESULT_DIR}/vmstat_compute${i}.log &
  ssh compute${i} "pkill sar ; rm -f /root/sar_*.bin ; sar -o sar_compute${i}.bin
${INTERVAL} > /dev/null" &
done

for i in `seq 1 ${STORAGE_COUNT}`; do
  ssh storage${i} "pkill vmstat ; vmstat -n ${INTERVAL} | sed -e 's/^[ \t]*//' -e '3d'" >
${RESULT_DIR}/vmstat_storage${i}.log &
  ssh storage${i} "pkill sar ; rm -f /root/sar_*.bin ; sar -o sar_storage${i}.bin
${INTERVAL} > /dev/null" &
done

if [ $CEPH_MON -eq 1 ]; then
  ssh cephmon "pkill vmstat ; vmstat -n ${INTERVAL} | sed -e 's/^[ \t]*//' -e '3d'" >
${RESULT_DIR}/vmstat_cephmon.log &
  ssh cephmon "pkill sar ; rm -f /root/sar_*.bin ; sar -o sar_cephmon.bin ${INTERVAL} >
/dev/null" &
  ssh cephmon "pkill python ; nohup python -u /usr/bin/ceph -w > ceph_status.log &"
fi

# Run smallfile
sleep `expr $INTERVAL \* 2`

echo Running smallfile test: $SMF_RESULT

$SMF_CMD --operation $OPERATION | tee ${RESULT_DIR}/${SMF_RESULT}.txt

sleep `expr $INTERVAL \* 2`

# Stop statistics collection
for i in `seq 1 ${VM_COUNT}`; do
  ssh -i GUEST_KEY.pem vm${i} "pkill vmstat ; pkill iostat ; pkill sar"
done
for i in `seq 1 ${COMPUTE_COUNT}`; do
  ssh compute${i} "pkill vmstat ; pkill iostat ; pkill sar"
done
for i in `seq 1 ${STORAGE_COUNT}`; do
  ssh storage${i} "pkill vmstat ; pkill iostat ; pkill sar"
done
if [ $CEPH_MON -eq 1 ]; then
  ssh cephmon "pkill vmstat ; pkill iostat ; pkill python ; killall -w sar"
  scp cephmon:/root/sar_*.bin ${RESULT_DIR}/
  scp cephmon:/root/ceph_status.log ${RESULT_DIR}/
fi

wait
```

```
# Copy stat files
for i in `seq 1 ${VM_COUNT}`; do
  scp -i GUEST_KEY.pem vm${i}:/root/sar_*.bin ${RESULT_DIR}/
done
for i in `seq 1 ${COMPUTE_COUNT}`; do
  scp compute${i}:/root/sar_*.bin ${RESULT_DIR}/
done
for i in `seq 1 ${STORAGE_COUNT}`; do
  scp storage${i}:/root/sar_*.bin ${RESULT_DIR}/
done
cp -fv ${SMF_NETSHARE}/smf/*.csv ${RESULT_DIR}/

echo Test $SMF_RESULT complete!
sleep $INTERVAL

#END OPERATION
done
```

run_smallfile.sh :

```
#!/bin/bash
COMPUTE_COUNT=$1
VM_COUNT=$2
THREAD_COUNT=${3:-4}
FILESIZE=${4:-64}
FILES=${5:-16384}
FSYNC=${6:-N}
PAUSE=${7:-10}
STORAGE_TYPE=gluster
#STORAGE_TYPE=ceph

./prepare_vms_smallfile.sh $VM_COUNT
sleep 1
time ./smallfile_test.sh $STORAGE_TYPE $FILESIZE 32768 $VM_COUNT $COMPUTE_COUNT 4 N 10
```

Example output from a ***smallfile*** test:

```
smallfile version 2.1
                           hosts in test : ['vm1']
                   top test directory(s) : ['/mnt/test']
                               operation : create
                            files/thread : 32768
                                 threads : 4
                        record size (KB) : 0
                          file size (KB) : 64
                  file size distribution : random exponential
                           files per dir : 100
                            dirs per dir : 10
              threads share directories? : N
                         filename prefix :
                         filename suffix :
             hash file number into dir.? : Y
                     fsync after modify? : N
          pause between files (microsec) : 10
                    finish all requests? : Y
                              stonewall? : Y
                 measure response times? : Y
                            verify read? : Y
                                verbose? : False
```

```
                          log to stderr? : False
               permute host directories? : N
                remote program directory : /root/smallfile
               network thread sync. dir. : /mnt/nfsexport/smf
host = vm1, thread = 00, elapsed sec. = 28.767661, total files = 32768, total_records = 32769, status = ok
host = vm1, thread = 01, elapsed sec. = 28.708897, total files = 32768, total_records = 32769, status = ok
host = vm1, thread = 02, elapsed sec. = 28.722196, total files = 32700, total_records = 32701, status = ok
host = vm1, thread = 03, elapsed sec. = 28.724667, total files = 32500, total_records = 32501, status = ok
total threads = 4
total files = 130736
total data =     7.980 GB
 99.74% of requested files processed, minimum is  70.00
4544.547420 files/sec
4544.686465 IOPS
28.767661 sec elapsed time, 284.042904 MB/sec
```

# ABOUT PRINCIPLED TECHNOLOGIES

Principled Technologies, Inc.
1007 Slater Road, Suite 300
Durham, NC, 27703
www.principledtechnologies.com

We provide industry-leading technology assessment and fact-based marketing services. We bring to every assignment extensive experience with and expertise in all aspects of technology testing and analysis, from researching new technologies, to developing new methodologies, to testing with existing and new tools.

When the assessment is complete, we know how to present the results to a broad range of target audiences. We provide our clients with the materials they need, from market-focused data to use in their own collateral to custom sales aids, such as test reports, performance assessments, and white papers. Every document reflects the results of our trusted independent analysis.

We provide customized services that focus on our clients' individual requirements. Whether the technology involves hardware, software, Web sites, or services, we offer the experience, expertise, and tools to help our clients assess how it will fare against its competition, its performance, its market readiness, and its quality and reliability.

Our founders, Mark L. Van Name and Bill Catchings, have worked together in technology assessment for over 20 years. As journalists, they published over a thousand articles on a wide array of technology subjects. They created and led the Ziff-Davis Benchmark Operation, which developed such industry-standard benchmarks as Ziff Davis Media's Winstone and WebBench. They founded and led eTesting Labs, and after the acquisition of that company by Lionbridge Technologies were the head and CTO of VeriTest.