Red Hat Performance Briefs

RHEL OpenStack Platform on Red Hat Storage

Cinder Volume Performance

Performance Engineering

Version 1.0

June 2014

100 East Davie Street
Raleigh NC 27601 USA
Phone: +1 919 754 4950
Fax: +1 919 800 3804

The GPG fingerprint of the security@redhat.com key is:
CA 20 86 86 2B D6 9D FC 65 F6 EC C4 21 91 80 CD DB 42 A6 0E

Table of Contents

1 Executive Summary

This paper describes performance testing performed by the Red Hat Performance Engineering group on a Red Hat Enterprise Linux OpenStack Platform (RHEL OSP) 4.0 configuration using Cinder volumes stored on Red Hat Storage (RHS) 2.1. It focuses on steady-state performance of OpenStack instance created file systems on Cinder volumes as a function of the number of instances. The collected data is used to characterize and compare the performance of a RAID6 2-way replication based RHS configuration with that of a hardware equivalent JBOD 3-way replication configuration, using both FUSE and libgfapi access.

This document does not focus on the Cinder provisioning process (i.e. the Cinder provided services for configuring Cinder block storage volumes).

This work is partially related to two articles published by Principled Technologies (PT) using OpenStack RDO Havana for which Red Hat provided RHS configuration recommendations. These articles compare performance of Ceph to RHS with libgfapi using a somewhat different methodology in workloads and configuration. See the References section of this document for further details.

2 Test Environment

The hardware and sofware configurations used for the systems under test (SUT).

2.1 Hardware

RAID6 RHS Server	Dell PowerEdge R510, 48GB RAM (12) Intel Xeon CPU X5650 @2.67GHz (1) Intel 82599ES dual-port 10-GbE NIC Dell PERC H700 (MegaRAID) controller (12) 7200-RPM SAS drives in RAID6 volume 256 KB stripe width, writeback caching enabled
JBOD RHS Server	Dell PowerEdge R710xd, 48GB RAM (12) Intel Xeon CPU E5-2620 @2.00GHz (1) Intel 82599EB dual-port 10-GbE NIC Dell PERC H710P (MegaRAID) controller (12) 7200-RPM SAS drives (JBOD) writeback caching enabled
OSP Controller **OSP Compute**	Dell PowerEdge R510, 48GB RAM (12) Intel Xeon CPU X5650 @2.67GHz (1) Intel 82599ES dual-port 10-GbE NIC

Table 1: Hardware Configuration

2.2 Software

RAID6 RHS Servers (4)	kernel-2.6.32-431.3.1.el6 (RHEL 6.5), 2 Westmere sockets glusterfs-3.4.0.58rhs-1.el6rhs (RHS 2.1U2) tuned active profile: *rhs-virtualization* (see *Appendix B: Tuned Profiles* for details)
JBOD RHS Servers (3)	2.6.32-358.28.1.el6 (RHEL 6.4), 2 Westmere sockets glusterfs-3.4.0.49rhs-1.el6rhs (RHS 2.1U2) tuned active profile: *rhs-virtualization* (see *Appendix B: Tuned Profiles* for details)
OSP Controller (1)	kernel-2.6.32-431.el6 (RHEL 6.5), 2 Westmere sockets glusterfs-3.4.0.59rhs.1.el6rhs (RHS 2.1U2) openstack-nova-2013.2.2.2.el6ost openstack-glance-2013.2.2.2.el6ost openstack-packstack-2013.2.1.0.25.dev987.el6ost openstack-cinder-2013.2.2.1.el6ost openstack-neutron-2013.2.2.1.el6ost openstack-cinder-2013.2.2.1.el6ost openstack-keystone-2013.2.2.1.el6ost openstack-utils-2013.2.3.el6ost tuned active profile: *virtual-host* (see *Appendix B: Tuned Profiles* for details)
OSP Computes (4)	kernel-2.6.32-431.el6 (RHEL 6.5), 2 Westmere sockets glusterfs-3.4.0.59rhs.1.el6rhs (RHS 2.1U2) tuned active profile: *virtual-host* (see *Appendix B: Tuned Profiles* for details)
OSP Instances (128)	kernel-2.6.32-431.5.1.el6 (RHEL 6.5) 1 CPU, 1 GB, 20GB system disk tuned active profile: *virtual-guest* (see *Appendix B: Tuned Profiles* for details)

Table 2: Software Configuration

The Kernel Samepage Merging (KSM) tuning daemon was disabled on all nodes.

```
# service ksmtuned stop
# chkconfig ksmtuned off
# service ksm stop
# chkconfig ksm off
```

2.3 Gluster

One 4-brick replica-2 gluster volume was used for all RAID6 testing while a 36-brick replica-3 volume was used for JBOD tests. At this writing, JBOD with 3-way gluster replication is not officially supported but is under consideration for inclusion in a future release.

Each OSP instance had a 40GB cinder volume (gluster container) created, attached, EXT4 formatted, and mounted as /dev/vdb for all I/O tests.

The *virt* gluster profile was applied to each of the volumes used in testing.

```
# gluster volume set <vol> group virt
```

To see what settings are applied by this command, reference **/var/lib/glusterd/groups/virt** on any of the RHS servers.

Although disabled by the *virt* profile, the stat-prefetch option provides optimal read performance with libgfapi to avoid gluster executing a STAT RPC prior to each READ RPC.

```
# gluster volume set <vol> stat-prefetch on
```

By default, not every server in a replicated volume participates in reads. Additionally, RHEL OSP hot-plugs Cinder volumes to the instances so all the Gluster files are not opened at the same time which can cause Gluster to do some load balancing based on volume response time. A read-hash-mode setting of 2 was applied to better balance reads across all servers, provided there are enough files to keep all servers busy.

```
# gluster volume set <vol> read-hash-mode 2
```

See *Appendix A: Gluster Volume Tuning* for the specific list of configured gluster volume options.

Ensure that *rpc-auth-allow-insecure* is enabled in **/etc/glusterfs/glusterd.vol** and the glusterd service is restarted on all servers.

2.4 OpenStack

RHEL OSP 4.0 was installed and configured using a packstack answer file, the specific contents of which are available in *Appendix C: Packstack Answer File*. It configures one OpenStack controller node (nova, neutron, glance, keystone, etc.) and four compute nodes.

3 Test Workloads

This project used a set of workloads that covers a wide spectrum of the possible workloads applicable to a Cinder volume.

1. Large-file sequential multi-stream reads and writes by the 'iozone -+m' benchmark
   - using a 4GB file per server with a 64KB record size
   - with each instance executes a single I/O thread
   - where all random I/O tests use O_DIRECT
2. Large-file random multi-stream reads and writes using an fio (Flexible I/O benchmark) based workload for testing pure random I/O on Cinder volumes. This workload executes a random I/O process inside each instance in parallel using options to start and stop all threads at approximately the same time. This allows aggregation of the per-thread results to achieve system-wide IOPS throughput for OpenStack on RHS. Additionally, fio can rate limit throughput, runtime and IOPS of individual processes to measure:
   - instance scaling throughput
   - the server's aggregate OSP instance capacity
3. Small-file multi-stream using the smallfile benchmark. Sequential operations include:.
   - create -- create a file and write data to it
   - append -- open an existing file and append data to it
   - read -- drop cache and read an existing file
   - rename -- rename a file
   - delete_renamed -- delete a previously renamed file

4 Test Results, Scaling OpenStack

I/O scaling tests were performed using increasing instance counts with both FUSE and libgfapi. All multi-instance tests ensured even distribution across compute nodes. The results of all testing are in the following sections.

4.1 RAID6 vs. JBOD

All testing was performed in one phase using RHS comprised of 12-disk RAID6 volumes from each of four Gluster servers (48 total disks in 4 bricks) and in a second phase using storage comprised of 12 disks configured as JBOD from each of three Gluster servers (36 total disks in 36 bricks). All comparison testing used libgfapi.

4.1.1 Large File I/O

Clustered iozone was executed on all participating instances to both RAID6 and JBOD RHS storage.

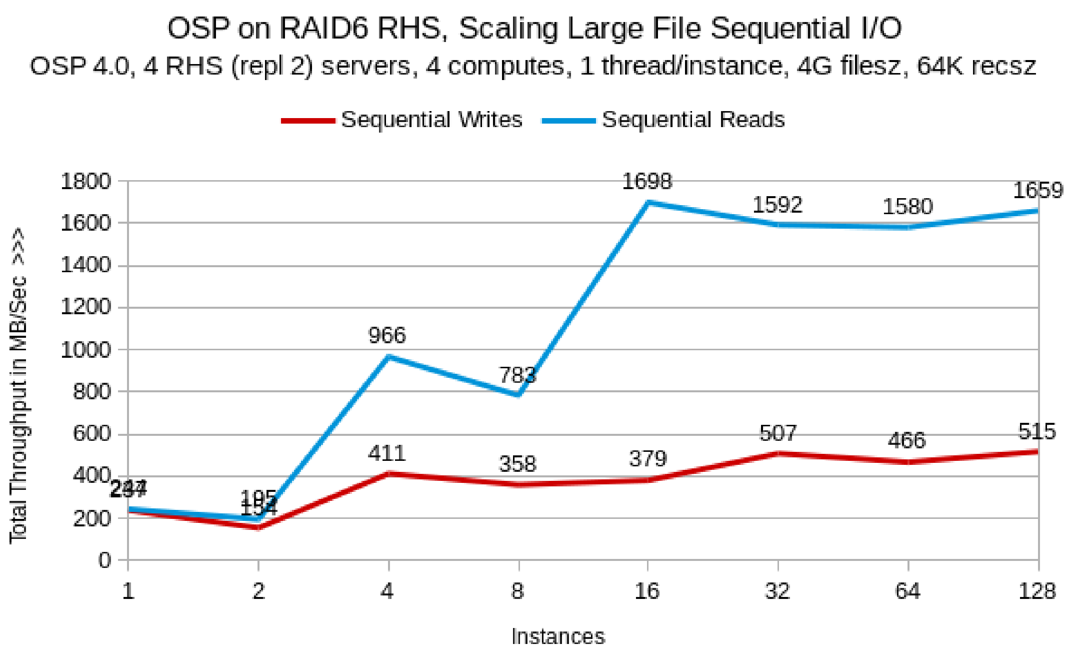

Figure 4.1: Scaling Large File Sequential I/O on RAID6

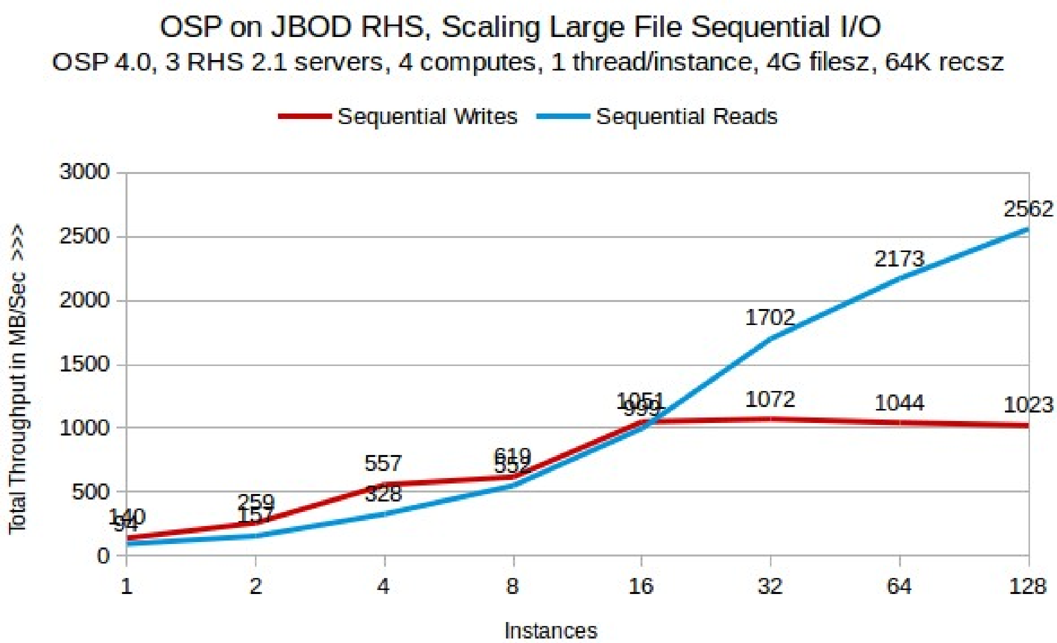

Figure 4.2: Scaling Large File Sequential I/O on JBOD

The graph in Figure *4.3* normalizes the large file sequential data shown in the previous graphs and compares it in total MB/second per RHS server.

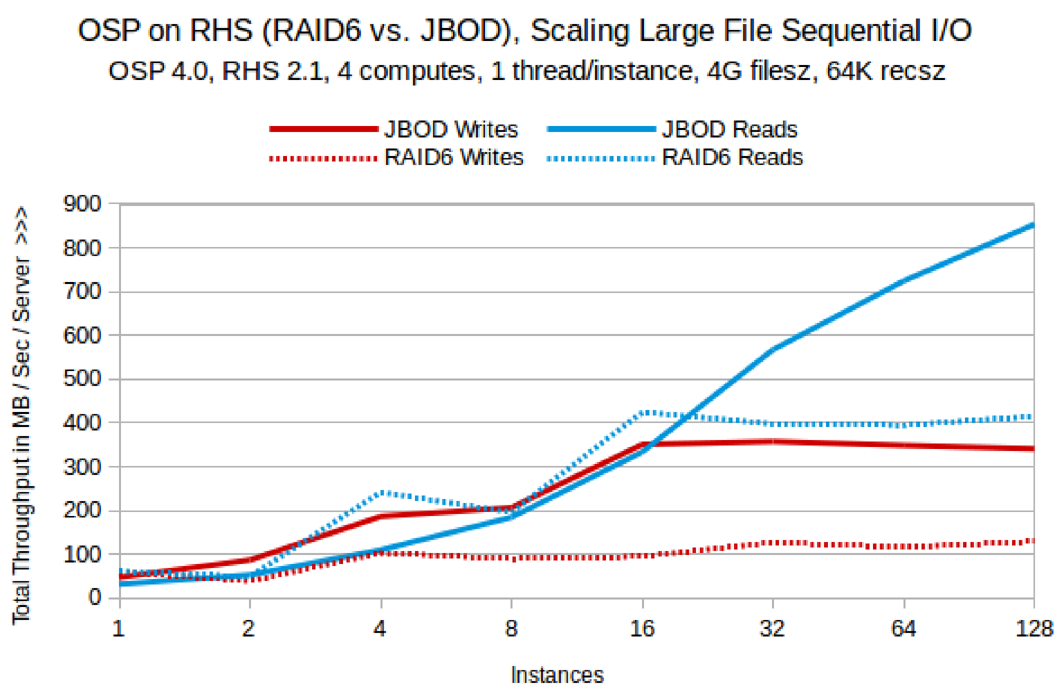

Figure 4.3: RAID6 vs. JBOD - Scaling Large File Sequential I/O

Underlying JBOD storage improves the performance of sequential reads, continuing to scale reasonably well beyond 16 instances where RAID6 made no further progress.

4.1.2 Small File I/O

The small-file benchmark was configured to sequentially execute the following workload:

- operations: create, append, read (w/prior cache drop), rename, delete-renamed
- one thread per OSP instance
- 30000 64KB files, 100 files per directory
- stonewalling disabled

Figure *4.4* graphs the results of RHS on RAID6 (measured in total files processed per second) for file creates, appends and reads.

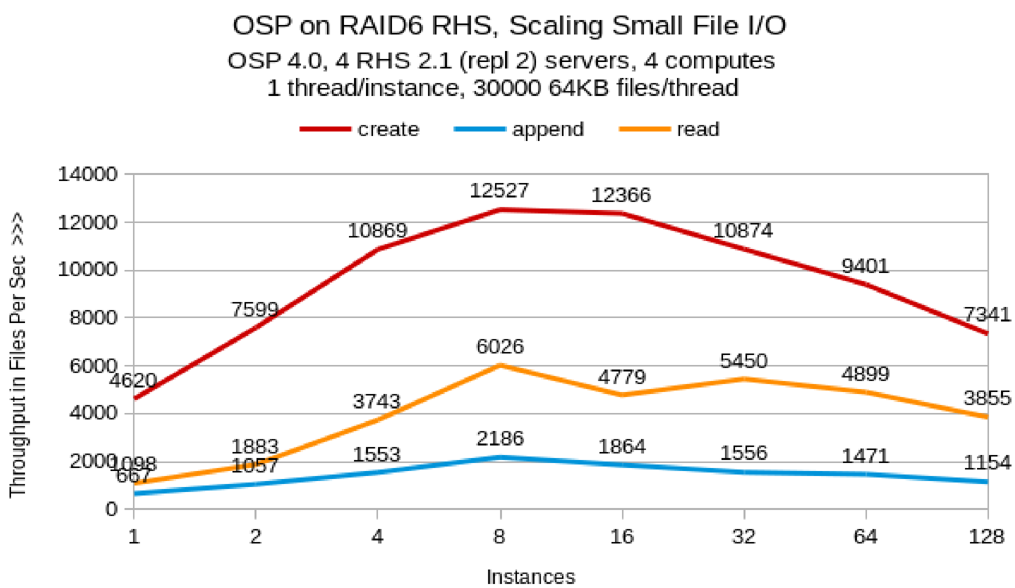

Figure 4.4: Scaling Small File I/O on RAID6

Figure *4.5* graphs the results for file creates, appends and reads on RHS JBOD.

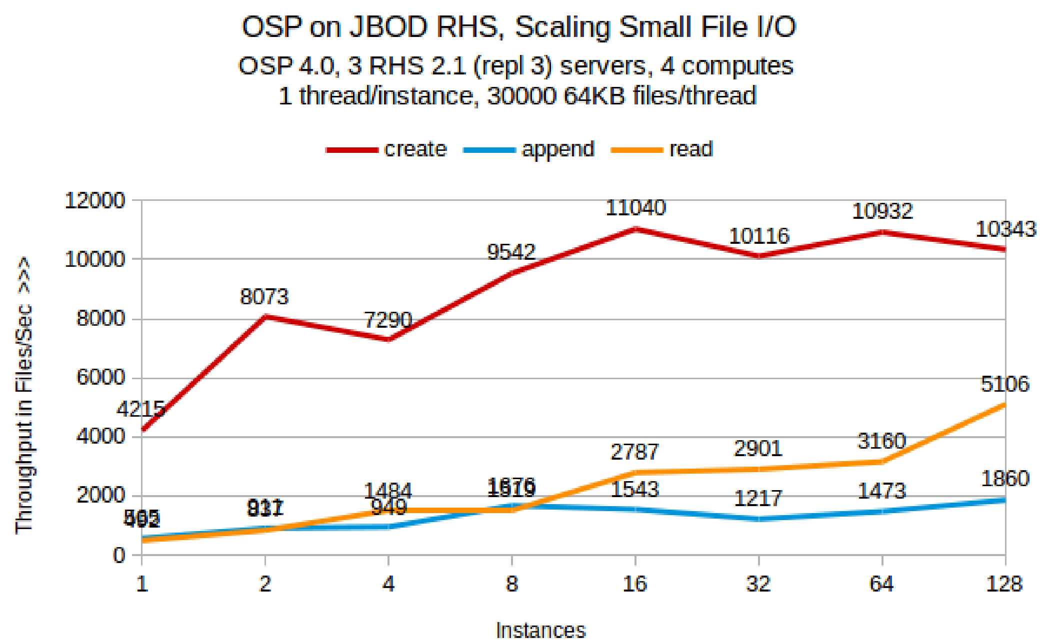

Figure 4.5: Scaling Small File I/O on JBOD

Figure *4.6* normalizes the small file data shown in the previous graphs and compares them in files/second per RHS server. JBOD provides a sequential write advantage maintaining throughput beyond 16 instances where RAID6 scaling dropped off.

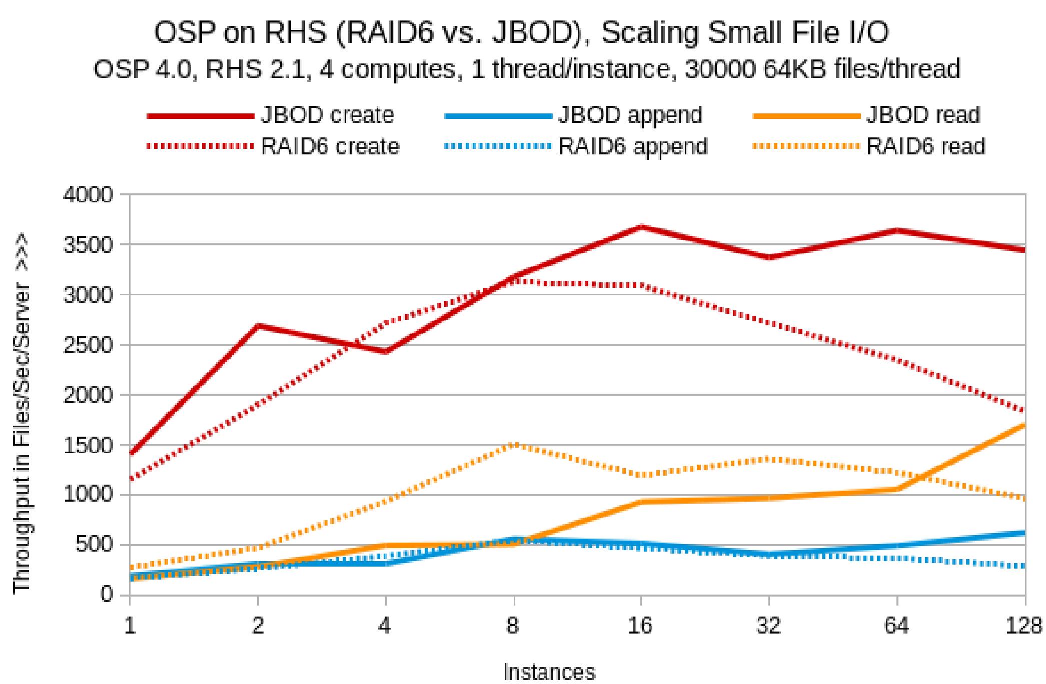

Figure 4.6: RAID6 vs. JBOD - Scaling Small File I/O

4.1.3 Flexible I/O

The images below graph the results of sequential and random large file tests on RHS over RAID6. All instances were rate limited by both runtime and small/medium/large total IOPS per instance settings. Each test ran for a base time of ten minutes plus ten seconds for each participating instance. All results were measured in total IOPS.

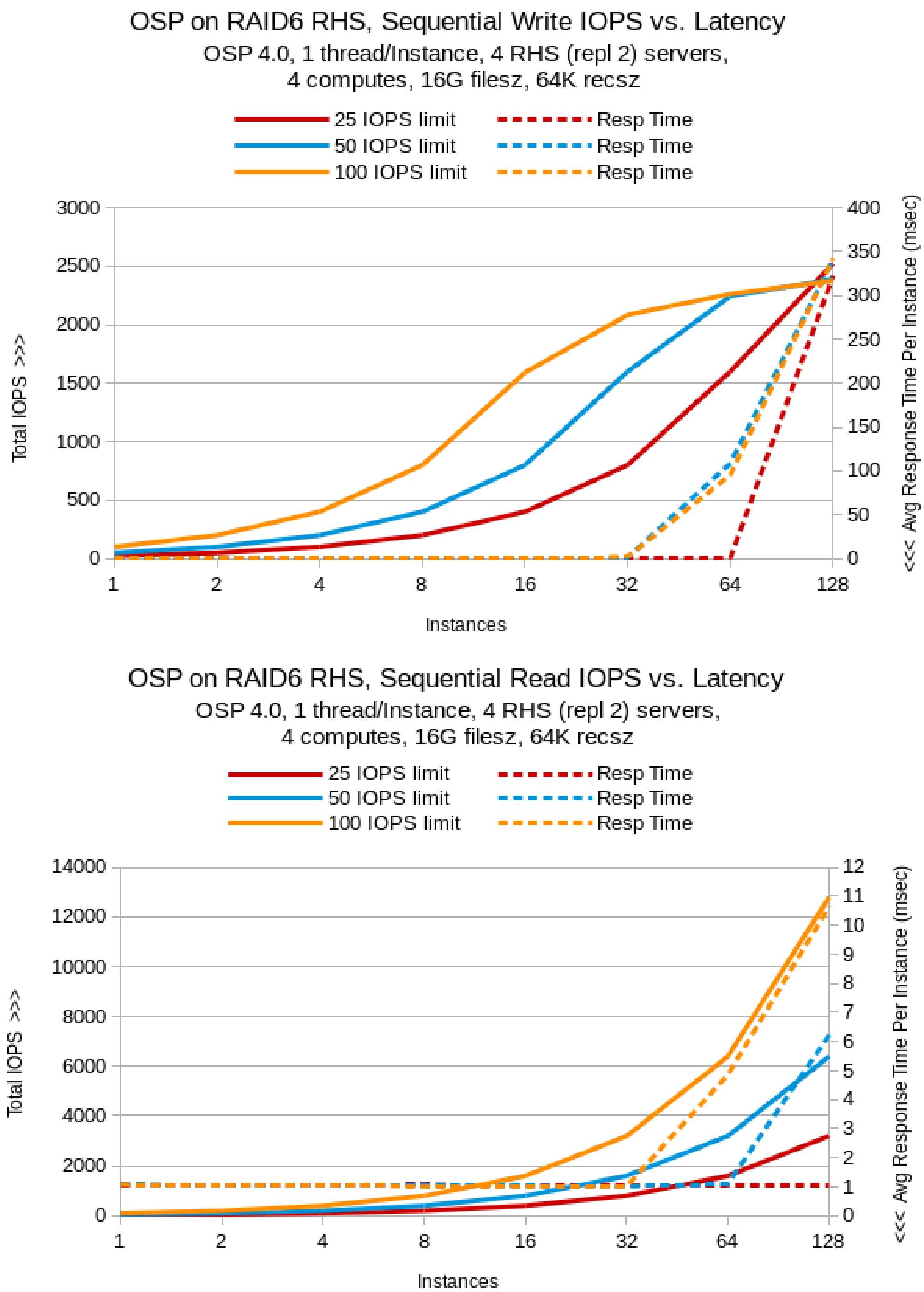

Figure 4.7: Sequential IOPS & Latency on RAID6

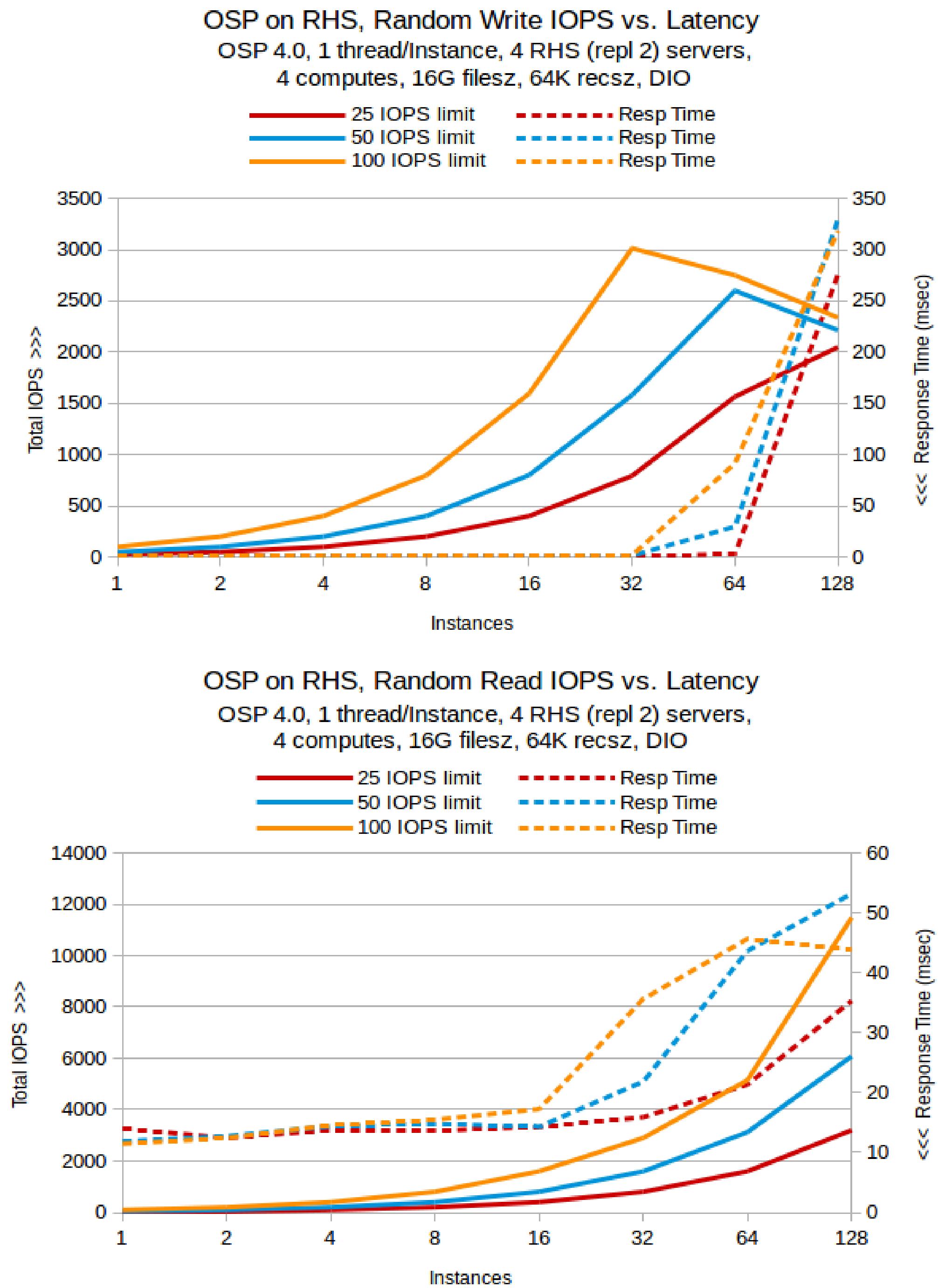

Figure 4.8: Random IOPS & Latency on RAID6

The graphs in Figures 4.9 and 4.10 normalize the RAID6 sequential I/O data shown in Figure 4.7, comparing it to the results of the large IOPS limit tests on JBOD measured in total IOPS per RHS server. Note the effect instance response time (95th percentile submission to completion latency) has on RAID6 sequential writes in Figure 4.9.

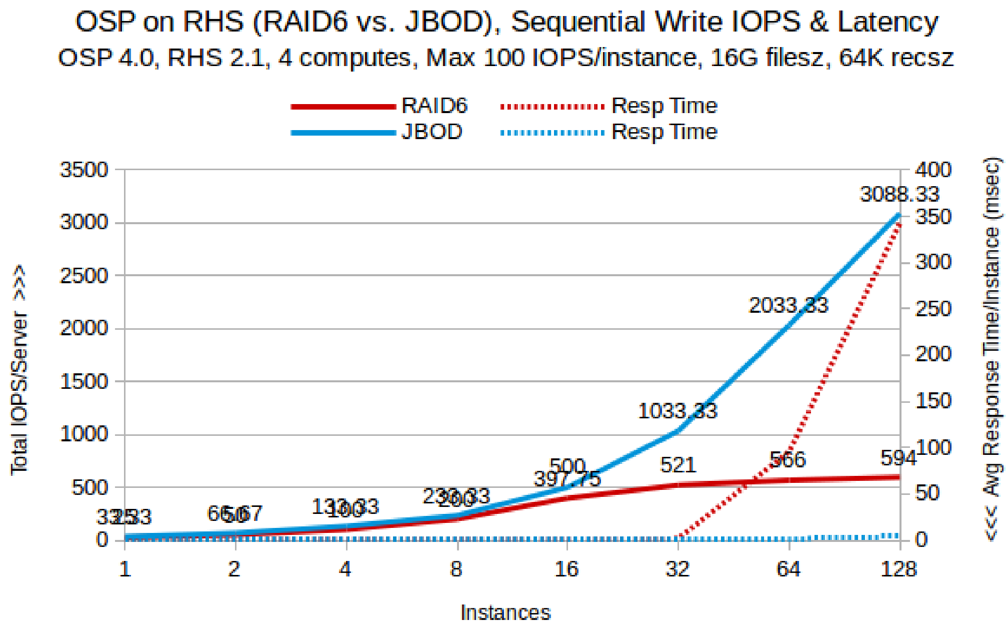

Figure 4.9: RAID6 vs. JBOD - Sequential Write IOPS & Latency

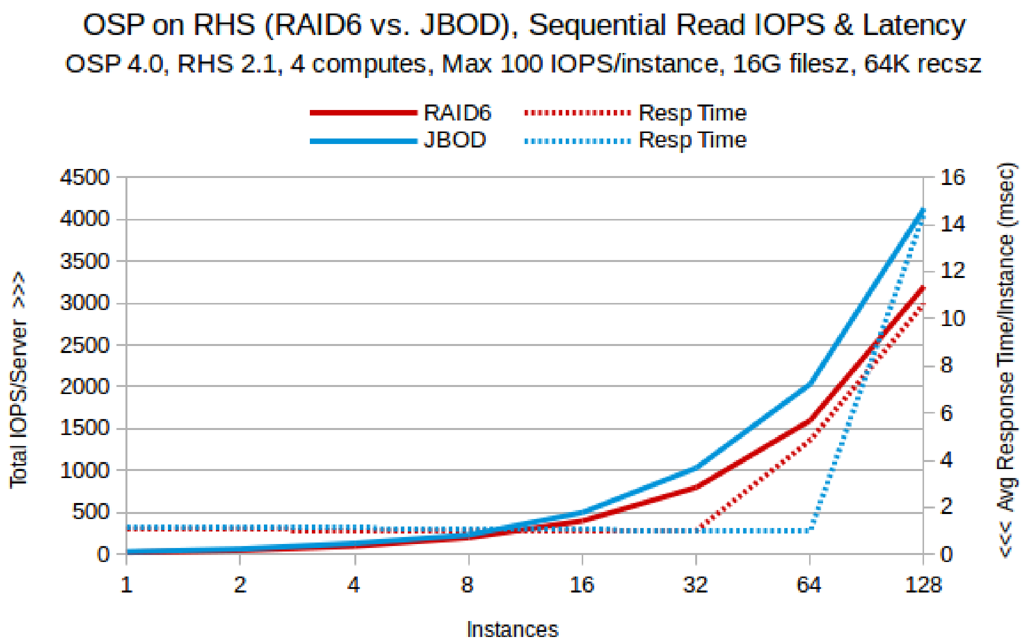

Figure 4.10: RAID6 vs. JBOD - Sequential Read IOPS & Latency

Figures 4.11 and 4.12 normalize the RAID6 random I/O data previously displayed in Figure 4.8, comparing it to the results of the large IOPS limit tests on JBOD measured in total IOPS per RHS server.

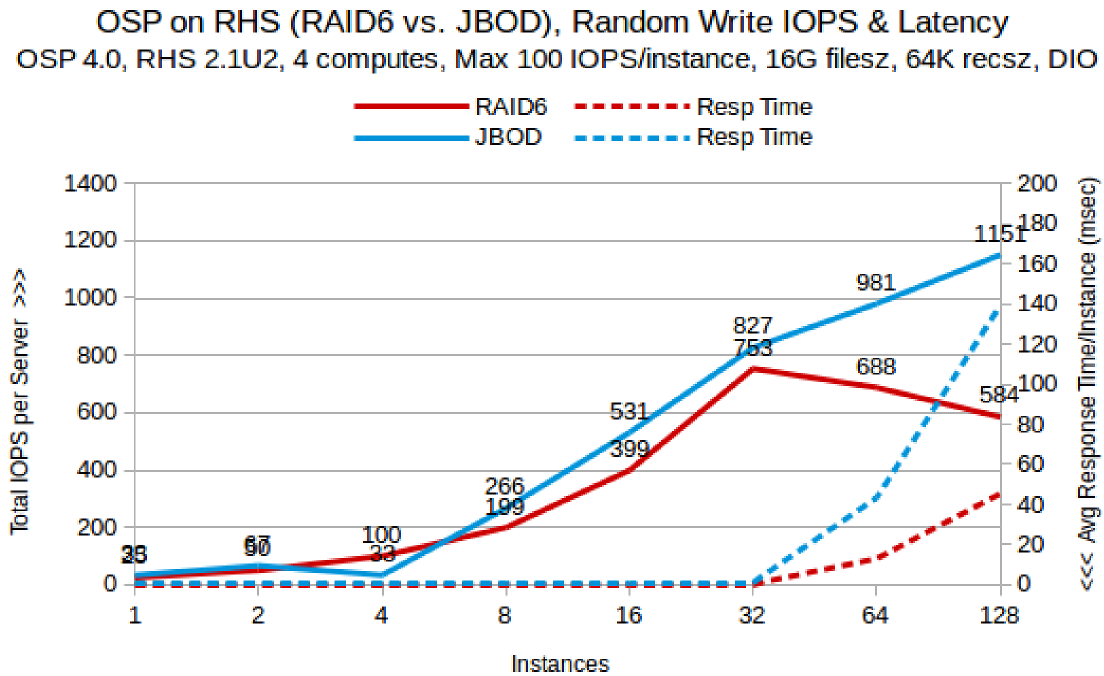

Figure 4.11: RAID6 vs. JBOD - Random Write IOPS & Latency

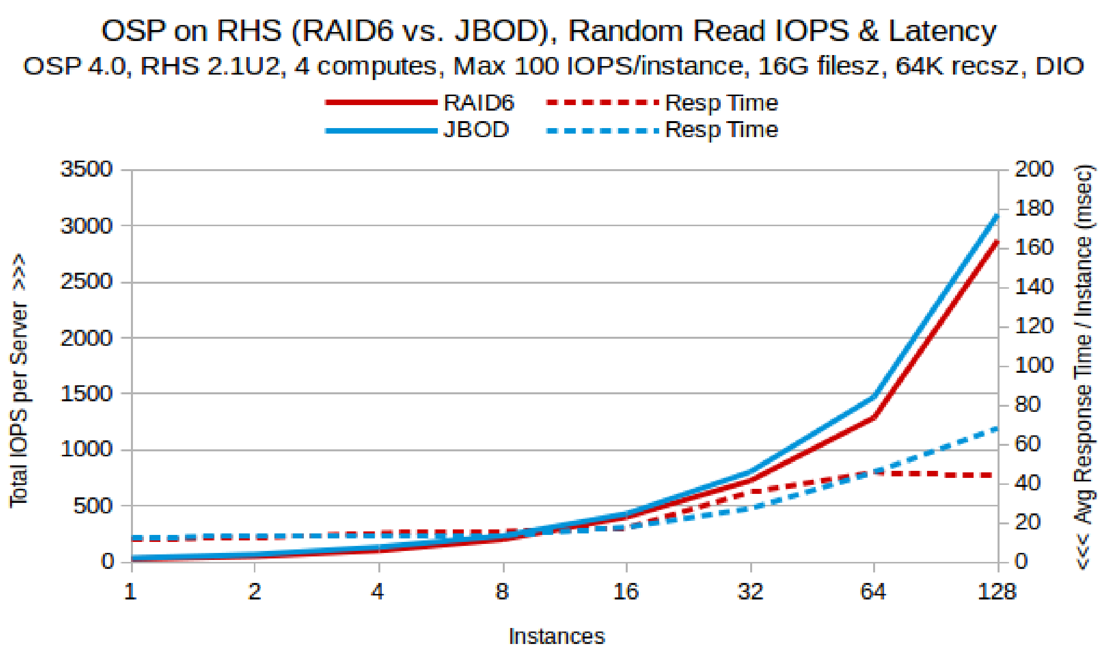

Figure 4.12: RAID6 vs. JBOD - Random Read IOPS & Latency

In both of the previous comparisons, the JBOD configuration allows write performance to continue scaling beyond 32 instances where RAID6 performance did not.

4.2 FUSE vs. Libgfapi

4.2.1 Large File I/O

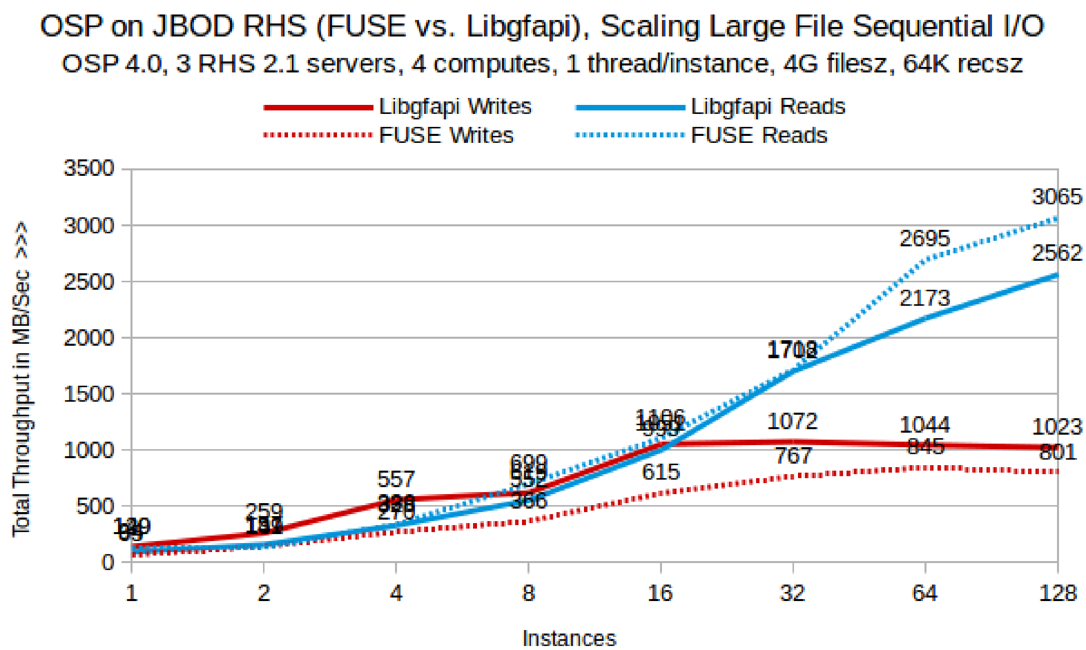

Figure 4.13: FUSE vs. Libgfapi - Large File Sequential I/O

4.2.2 Small File I/O

JBOD provides a sequential write advantage maintaining throughput beyond 16 instances where RAID6 scaling dropped off.

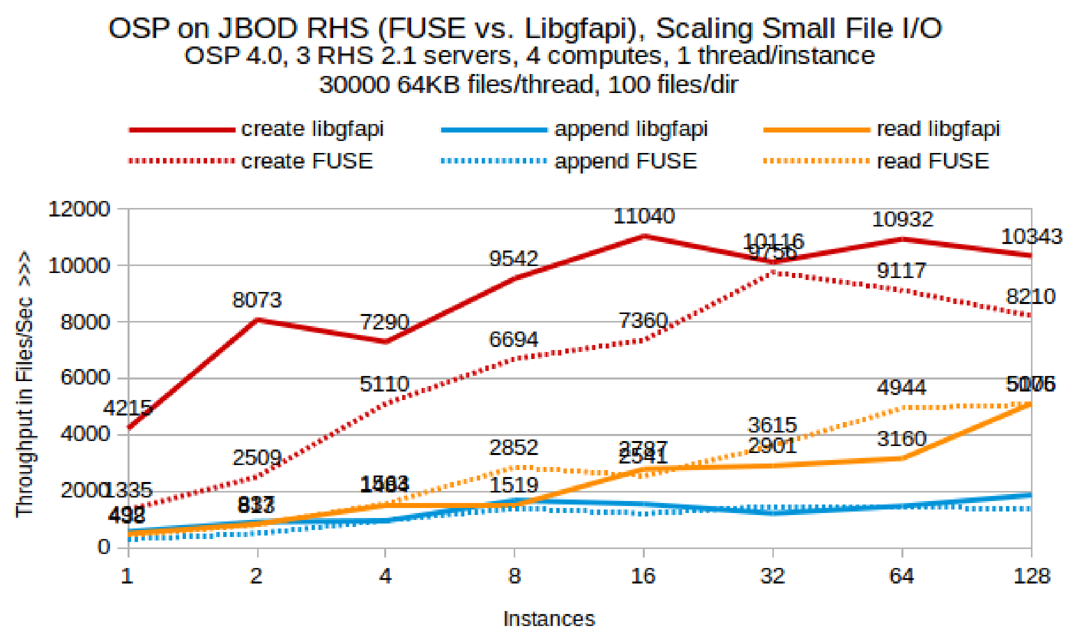

Figure 4.14: FUSE vs. Libgfapi - Small File I/O

4.2.3 Flexible I/O

Note the effect of instance response time on libgfapi random writes.

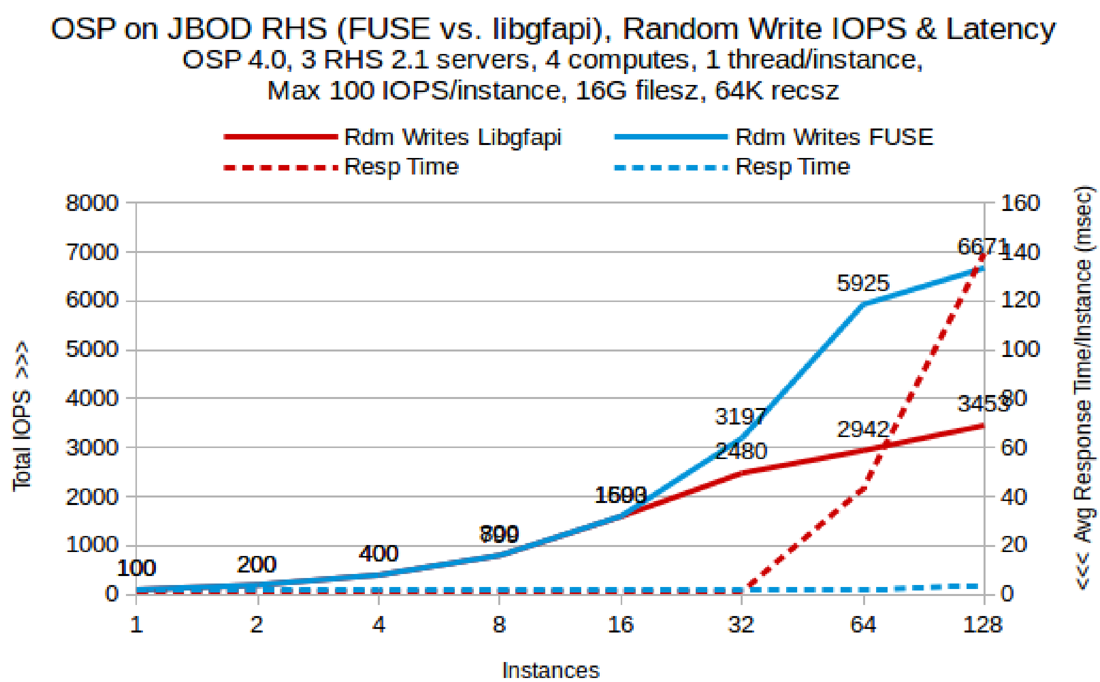

Figure 4.15: FUSE vs. Libgfapi - Random Write IOPS & Latency

5 Performance Tuning Effects

5.1 Server Queue Depths

Initial testing used m1.large sized instances (4CPU, 8GB) but it was determined that the maximum 16 instances that could be created within the resource limitations of the four compute nodes were not pushing server queue depths enough to make use of all of the disks behind the gluster volume. The instances were recreated using the m1.small flavor (1CPU, 2GB) to accommodate 64 in total resulting in greatly improved random I/O results. Note the difference in queue depths over time during random I/O tests when the instance count increases from 16 to 64.

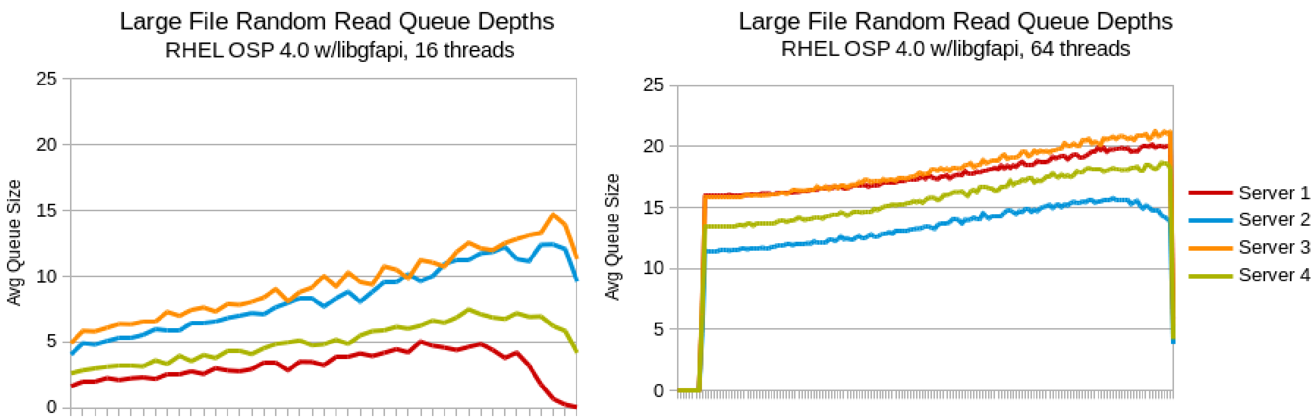

Figure 5.1: Effect of Server Queue Depths

5.2 Instance Sizing

All reads (excluding 4 instance sequential) appear to benefit from the additional I/O processing power while only greater instance counts show any gain at all in write throughput performance. Both the sequential and random results highlight how reads benefit from the additional threads (instances) using the underlying storage. Each instance executed a single I/O thread. Sequential I/O is measured in MB per second while random I/O is measured in IOPS.

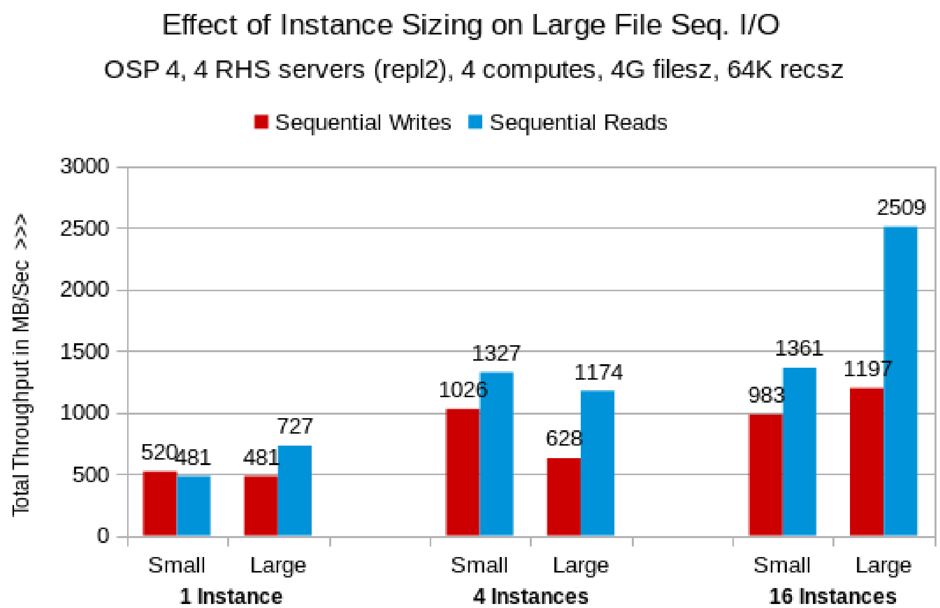

Figure 5.2: Effect of Instance Size on Large File Seq. I/O

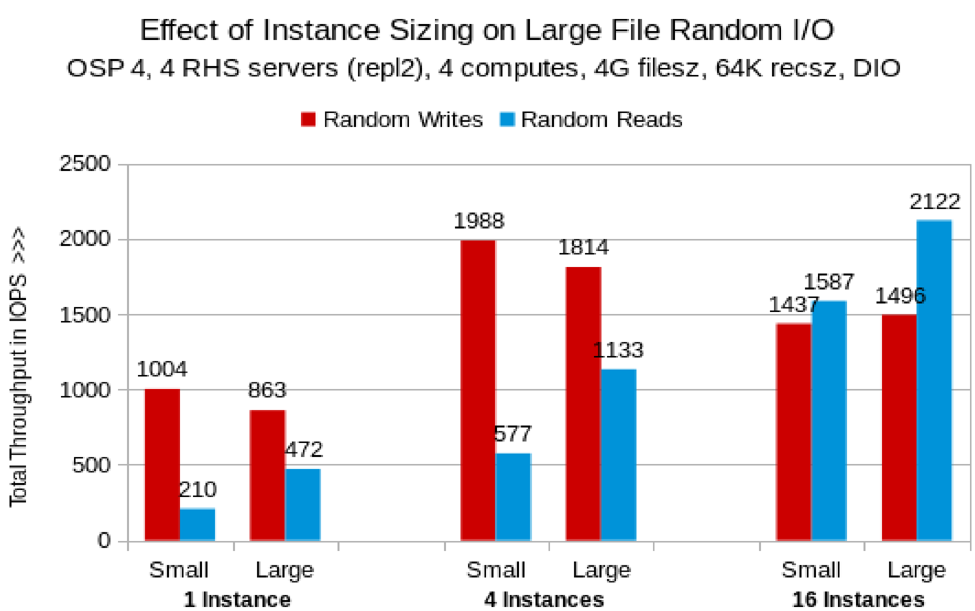

Figure 5.3: Effect of Instance Size on Large File Random I/O

The four instance random writes stand out due to the servers buffering writes (regardless of O_DIRECT use) providing no advantage for higher thread counts.

5.3 Server Block Device Readahead

Tests were executed to compare the default readahead setting of 4096 KB to a 4x setting of 16 MB. Sequential results are measured in MB/second while random I/O is measured in IOPS.

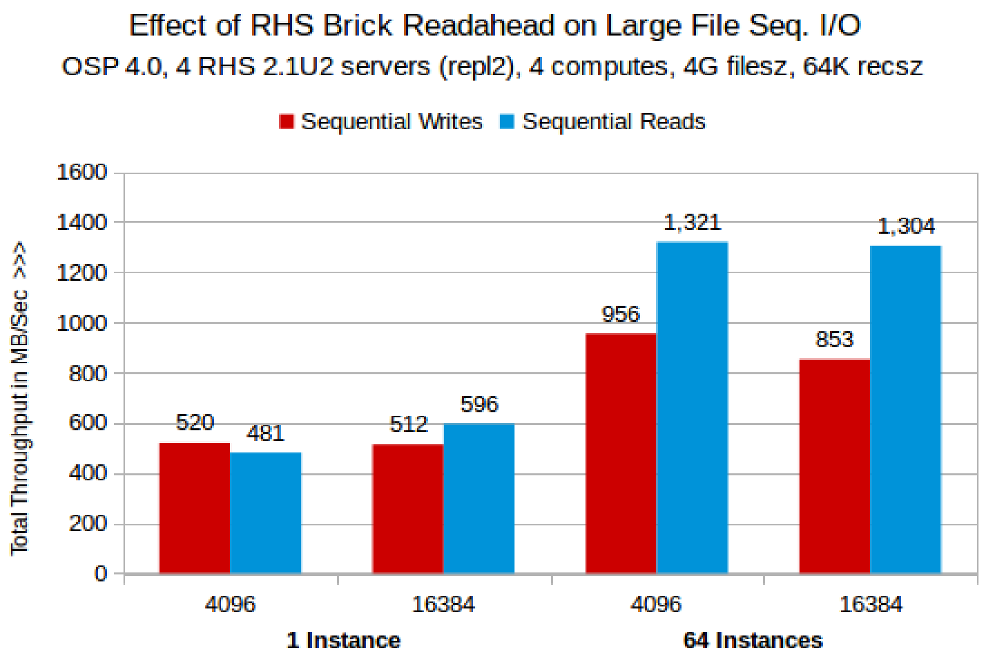

Figure 5.4: Effect of Device Readahead on Large File Seq. I/O

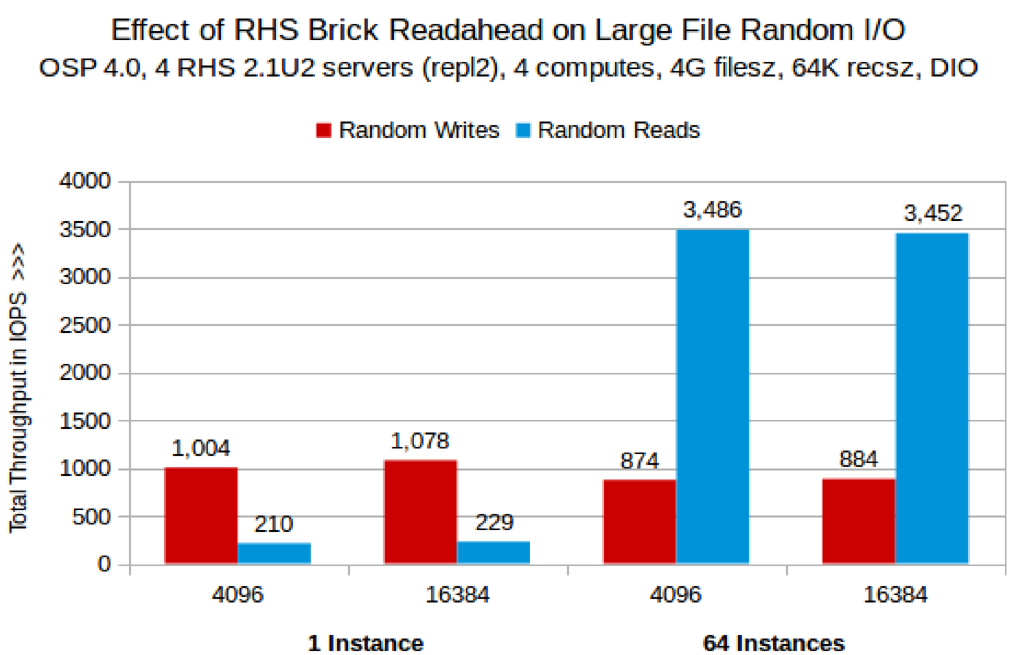

Figure 5.5: Effect of Device Readahead on Large File Random I/O

Block device readahead results indicate while single thread sequential reads were able to take some advantage, most I/O is unaffected by the increased gluster brick device read_ahead_kb setting.

5.4 Compute Node sched_migration_cost

Only single threaded I/O showed any variation in results when modifying the value of sched_migration_cost on the compute nodes from 5 million, as set by the virtual-host tuned profile (see *Appendix B: Tuned Profiles* for details). Although this default appears to have some disadvantage in single thread sequential reads, all modifications level out when multiple threads are applied and the default ends up remaining optimal for both sequential and random I/O. Sequential results are measured in MB/second while random I/O is measured in IOPS.

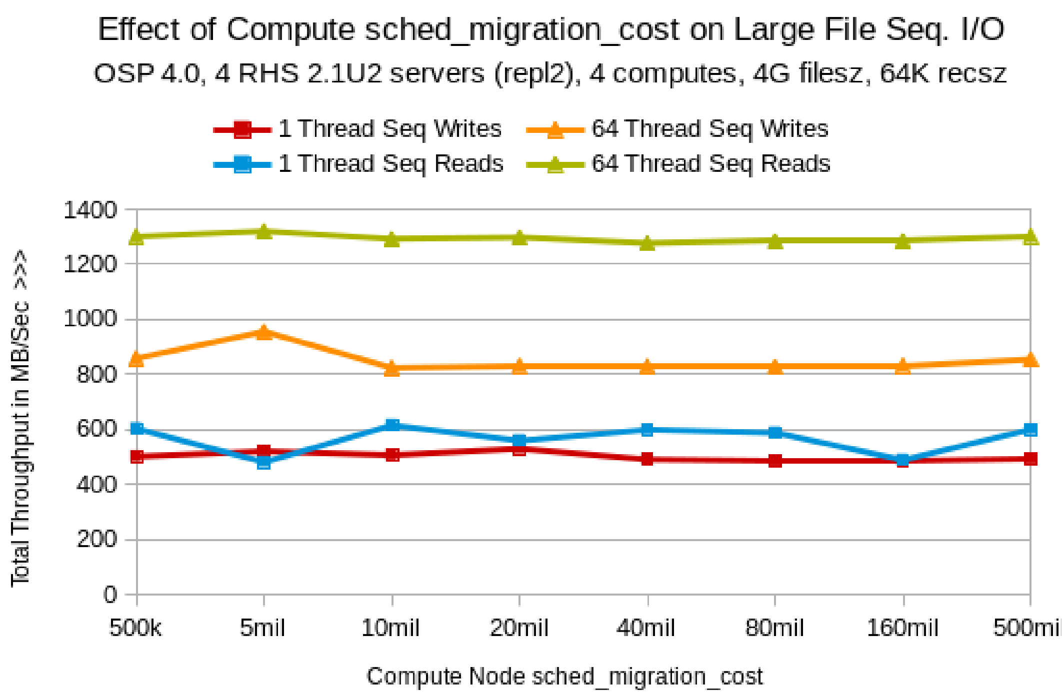

Figure 5.6: Effect of sched_migration_cost on Large File Seq. I/O

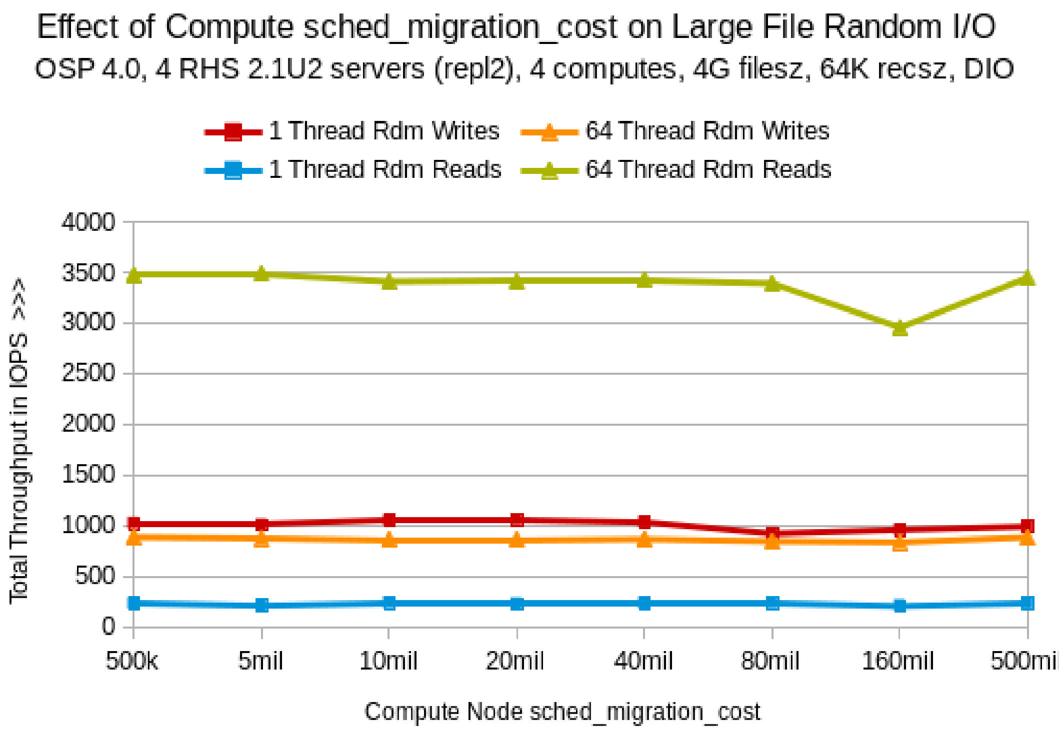

Figure 5.7: Effect of sched_migration_cost on Large File Random I/O

6 References

1. Large-file Sequential Performance [Nov.2013]
2. Small-file Performance [March 2014]

Appendix A: Gluster Volume Tuning

The following options were applied to the gluster volume either by the *virt* profile or individually.

cluster.entry-self-heal: off

cluster.data-self-heal: off

cluster.metadata-self-heal: off

server.allow-insecure: on

performance.client-io-threads: on

network.remote-dio: on

cluster.eager-lock: enable

performance.stat-prefetch: on

performance.io-cache: off

performance.read-ahead: off

performance.quick-read: off

cluster.read-hash-mode: 2

Appendix B: Tuned Profiles

Below are the system settings applied by *tuned* for each profile.

rhs-virtualization

I/O scheduler = deadline
CPU governor = performance
block device readahead = 4096 KB
filesystem barriers = disabled
kernel.sched_min_granularity_ns = 10000000
kernel.sched_wakeup_granularity_ns = 15000000
vm.swappiness = 10
vm.dirty_background_ratio = 1
net.ipv4.neigh.default.gc_thresh2 = 4096
net.ipv4.neigh.default.gc_thresh3 = 8192
net.ipv4.neigh.default.gc_interval = 60
net.core.netdev_max_backlog = 2048

net.ipv4.tcp_max_syn_backlog = 2048
net.ipv4.tcp_fin_timeout = 30
net.core.somaxconn = 512

virtual-guest

I/O scheduler = deadline
CPU governor = performance
block device readahead = 4x default
transparent hugepages = enabled
filesystem barriers = disabled
queue depths = 0.25x default
kernel.sched_min_granularity_ns = 10000000
kernel.sched_wakeup_granularity_ns = 15000000
vm.swappiness = 30
vm.dirty_ratio = 40

virtual-host

I/O scheduler = deadline
CPU governor = performance
block device readahead = 4x default
transparent hugepages = enabled
filesystem barriers = disabled
kernel.sched_min_granularity_ns = 10000000
kernel.sched_wakeup_granularity_ns = 15000000
kernel.sched_migration_cost = 5000000
vm.swappiness = 10
vm.dirty_ratio = 10
vm.dirty_background_ratio = 5

Appendix C: Packstack Answer File

```
[general]
CONFIG_DEBUG_MODE=n
CONFIG_SSH_KEY=/root/.ssh/id_rsa.pub
CONFIG_MYSQL_INSTALL=y
CONFIG_GLANCE_INSTALL=y
CONFIG_CINDER_INSTALL=y
CONFIG_NOVA_INSTALL=y
CONFIG_NEUTRON_INSTALL=y
CONFIG_HORIZON_INSTALL=y
CONFIG_SWIFT_INSTALL=n
CONFIG_CEILOMETER_INSTALL=y
CONFIG_HEAT_INSTALL=n
CONFIG_CLIENT_INSTALL=y
CONFIG_NTP_SERVERS=17.11.159.254,17.11.255.2,17.11.255.3
CONFIG_NAGIOS_INSTALL=n
EXCLUDE_SERVERS=
CONFIG_MYSQL_HOST=17.11.154.120
CONFIG_MYSQL_USER=root
CONFIG_MYSQL_PW=password
CONFIG_QPID_HOST=17.11.154.120
CONFIG_QPID_ENABLE_SSL=n
CONFIG_QPID_ENABLE_AUTH=n
CONFIG_QPID_NSS_CERTDB_PW=password
CONFIG_QPID_SSL_PORT=5671
CONFIG_QPID_SSL_CERT_FILE=/etc/pki/tls/certs/qpid_selfcert.pem
CONFIG_QPID_SSL_KEY_FILE=/etc/pki/tls/private/qpid_selfkey.pem
CONFIG_QPID_SSL_SELF_SIGNED=y
CONFIG_QPID_AUTH_USER=qpid_user
CONFIG_QPID_AUTH_PASSWORD=password
CONFIG_KEYSTONE_HOST=17.11.154.120
CONFIG_KEYSTONE_DB_PW=password
CONFIG_KEYSTONE_ADMIN_TOKEN=9a4d45dc558742099f8011b5ba8d7869
CONFIG_KEYSTONE_ADMIN_PW=password
CONFIG_KEYSTONE_DEMO_PW=password
CONFIG_KEYSTONE_TOKEN_FORMAT=PKI
CONFIG_GLANCE_HOST=17.11.154.120
CONFIG_GLANCE_DB_PW=password
CONFIG_GLANCE_KS_PW=password
CONFIG_CINDER_HOST=17.11.154.120
CONFIG_CINDER_DB_PW=password
CONFIG_CINDER_KS_PW=password
CONFIG_CINDER_BACKEND=gluster
CONFIG_CINDER_VOLUMES_CREATE=n
CONFIG_CINDER_VOLUMES_SIZE=20G
CONFIG_CINDER_GLUSTER_MOUNTS='172.17.40.33:/jbod3'
CONFIG_CINDER_NFS_MOUNTS=
CONFIG_NOVA_API_HOST=17.11.154.120
CONFIG_NOVA_CERT_HOST=17.11.154.120
CONFIG_NOVA_VNCPROXY_HOST=17.11.154.120
CONFIG_NOVA_COMPUTE_HOSTS=17.11.154.123,17.11.154.129,17.11.154.126,17.11.15
```

```
4.132,17.11.154.135
CONFIG_NOVA_CONDUCTOR_HOST=17.11.154.120
CONFIG_NOVA_DB_PW=password
CONFIG_NOVA_KS_PW=password
CONFIG_NOVA_SCHED_HOST=17.11.154.120
CONFIG_NOVA_SCHED_CPU_ALLOC_RATIO=16.0
CONFIG_NOVA_SCHED_RAM_ALLOC_RATIO=1.5
CONFIG_NOVA_COMPUTE_PRIVIF=p2p2
CONFIG_NOVA_NETWORK_HOSTS=17.11.154.120
CONFIG_NOVA_NETWORK_MANAGER=nova.network.manager.FlatDHCPManager
CONFIG_NOVA_NETWORK_PUBIF=em1
CONFIG_NOVA_NETWORK_PRIVIF=p2p2
CONFIG_NOVA_NETWORK_FIXEDRANGE=191.168.32.0/24
CONFIG_NOVA_NETWORK_FLOATRANGE=10.3.4.0/22
CONFIG_NOVA_NETWORK_DEFAULTFLOATINGPOOL=nova
CONFIG_NOVA_NETWORK_AUTOASSIGNFLOATINGIP=n
CONFIG_NOVA_NETWORK_VLAN_START=100
CONFIG_NOVA_NETWORK_NUMBER=1
CONFIG_NOVA_NETWORK_SIZE=255
CONFIG_NEUTRON_SERVER_HOST=17.11.154.120
CONFIG_NEUTRON_KS_PW=password
CONFIG_NEUTRON_DB_PW=password
CONFIG_NEUTRON_L3_HOSTS=17.11.154.120
CONFIG_NEUTRON_L3_EXT_BRIDGE=br-ex
CONFIG_NEUTRON_DHCP_HOSTS=17.11.154.120
CONFIG_NEUTRON_LBAAS_HOSTS=
CONFIG_NEUTRON_L2_PLUGIN=openvswitch
CONFIG_NEUTRON_METADATA_HOSTS=17.11.154.120
CONFIG_NEUTRON_METADATA_PW=password
CONFIG_NEUTRON_LB_TENANT_NETWORK_TYPE=local
CONFIG_NEUTRON_LB_VLAN_RANGES=
CONFIG_NEUTRON_LB_INTERFACE_MAPPINGS=
CONFIG_NEUTRON_OVS_TENANT_NETWORK_TYPE=gre
CONFIG_NEUTRON_OVS_VLAN_RANGES=
CONFIG_NEUTRON_OVS_BRIDGE_MAPPINGS=
CONFIG_NEUTRON_OVS_BRIDGE_IFACES=
CONFIG_NEUTRON_OVS_TUNNEL_RANGES=1:1000
CONFIG_NEUTRON_OVS_TUNNEL_IF=p2p2
CONFIG_OSCLIENT_HOST=17.11.154.120
CONFIG_HORIZON_HOST=17.11.154.120
CONFIG_HORIZON_SSL=n
CONFIG_SSL_CERT=
CONFIG_SSL_KEY=
CONFIG_SWIFT_PROXY_HOSTS=17.11.154.120
CONFIG_SWIFT_KS_PW=password
CONFIG_SWIFT_STORAGE_HOSTS=17.11.154.120
CONFIG_SWIFT_STORAGE_ZONES=1
CONFIG_SWIFT_STORAGE_REPLICAS=1
CONFIG_SWIFT_STORAGE_FSTYPE=ext4
CONFIG_SWIFT_HASH=bc05f46001e442b6
CONFIG_SWIFT_STORAGE_SIZE=2G
CONFIG_PROVISION_DEMO=n
CONFIG_PROVISION_DEMO_FLOATRANGE=172.24.4.224/28
CONFIG_PROVISION_TEMPEST=n
CONFIG_PROVISION_TEMPEST_REPO_URI=https://github.com/openstack/tempest.git
```

```
CONFIG_PROVISION_TEMPEST_REPO_REVISION=master
CONFIG_PROVISION_ALL_IN_ONE_OVS_BRIDGE=n
CONFIG_HEAT_HOST=17.11.154.120
CONFIG_HEAT_DB_PW=password
CONFIG_HEAT_KS_PW=password
CONFIG_HEAT_CLOUDWATCH_INSTALL=n
CONFIG_HEAT_CFN_INSTALL=n
CONFIG_HEAT_CLOUDWATCH_HOST=17.11.154.120
CONFIG_HEAT_CFN_HOST=17.11.154.120
CONFIG_CEILOMETER_HOST=17.11.154.120
CONFIG_CEILOMETER_SECRET=8a8af0a389b04b02
CONFIG_CEILOMETER_KS_PW=password
CONFIG_NAGIOS_HOST=17.11.154.120
CONFIG_NAGIOS_PW=password
CONFIG_USE_EPEL=n
CONFIG_REPO=
CONFIG_RH_USER=
CONFIG_RH_PW=
CONFIG_RH_BETA_REPO=n
CONFIG_SATELLITE_URL=
CONFIG_SATELLITE_USER=
CONFIG_SATELLITE_PW=
CONFIG_SATELLITE_AKEY=
CONFIG_SATELLITE_CACERT=
CONFIG_SATELLITE_PROFILE=
CONFIG_SATELLITE_FLAGS=
CONFIG_SATELLITE_PROXY=
CONFIG_SATELLITE_PROXY_USER=
CONFIG_SATELLITE_PROXY_PW=
```