Red Hat Performance Briefs

Red Hat OpenStack Platform on Hyper-Converged Red Hat Ceph Storage

Cinder Volume Performance at Scale

Performance and Scale Engineering

Version 1.0 May 2017

RHEL 7.3
RHOSP 10
RHCS 2.1

Table of Contents

100 East Davie Street
Raleigh NC 27601 USA
Phone: +1 919 754 3700
Phone: 888 733 4281
Fax: +1 919 754 3701
PO Box 13588
Research Triangle Park NC 27709 USA

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

Chapter 1. Executive Summary

This document describes large-scale I/O characterization testing performed by the Red Hat Performance and Scale Engineering group on:

- Red Hat Enterprise Linux (RHEL) 7.3
- Red Hat OpenStack Platform (RHOSP) 10
- Red Hat Ceph Storage (RHCS) 2.1

It uses Glance images, Nova instances, and Cinder volumes on over 540 TB of Ceph storage configured in a hyper-converged RHOSP-RHCS infrastructure. Eight servers acting as Ceph OSD nodes and RHOSP compute nodes as well as three servers acting as Ceph monitors and RHOSP controllers were used to exercise as many as 512 RHOSP instances. FIO (Flexible I/O) benchmarks measured application latency percentiles as a function of scale and uperf network request-response times were measured to characterize the impact of combining disk and network workloads.

The purpose of this testing was not to achieve world-record performance by extensive tuning, but to document and understand the user experience of RHOSP in a HCI using Ceph storage at scale with respect to performance and specifically to understand issues customers might encounter with such a configuration in production.

Some deployment issues with RHOSP were encountered and documented as bugzilla reports. These problems have either been fixed or are being fixed in an upcoming release. See Appendix: Issues for details.

Chapter 2. Test Environment

This section describes the hardware, software, RHCS, and RHOSP configurations used for the systems under test.

2.1. Hardware

Table 1 identifies the specifications of the hardware used in testing.

Table 1. Hardware Configuration

RHCS OSD Node, RHOSP Compute	Supermicro SSG-6048R-E1CR36H with 256GB RAM, 2-socket (28) Intel Xeon E5-2660 v4 @2.00GHz (56 threads/server) Dual-port 40-GbE (SFP+) - Intel XL710 LSI 3108 disk controller (2) P3700 800GB NVMe (journals) (34) 2TB 7.2k SATA (OSDs) (2) 500GB 7.2k SATA (system)
RHCS Monitor, RHOSP Controller	Supermicro SSG-6048R-E1CR36H with 256GB RAM, 2-socket (28) Intel Xeon E5-2660 v4 @2.00GHz (56 threads/server) Dual-port 40-GbE (SFP+) - Intel XL710 (2) 500GB 7.2k SATA (system)
RHOSP Director	Dell PowerEdge R620 with 64GB RAM, 2-socket (12) Intel Xeon E5-2620 v4 @2.20GHz (24 threads/server) (4) 1TB SATA (1) Intel X520 Dual Port 10-GbE NIC
RHOSP Instance (VM)	1 CPU, 1GB Memory, 20GB system disk
Network	Juniper QFX5200 Switches, 100-GbE uplinks

2.2. Software

Table 2 lists the versions of relevant software packages used in testing.

Table 2. Software Configuration

RHCS OSD Nodes, RHOSP Computes (8)	RHEL 7.3 RHOSP 10 RHCS 2.1 tuned active profile: *virtual-host*
RHCS Monitors, RHOSP Controllers (3)	RHEL 7.3 RHOSP 10 RHCS 2.1 tuned active profile: *throughput-performance*

RHOSP Director (1)	RHEL 7.3 RHOSP 10
RHOSP Instances (512)	RHEL 7.3 fio-2.14 tuned active profile: *virtual-guest*

See Appendix: Tuned Profiles for details of the profiles referenced in the Software Configuration table.

2.3. Hyper-Converged OpenStack and Ceph

2.3.1. Undercloud Deployment

The RHOSP 10 director was installed and configured using Ansible playbooks to deploy the undercloud and RHOSP configuration files. The main.yaml file used by the playbook to deploy the undercloud can be viewed in Appendix: Undercloud Deployment Configuration Files.

2.3.2. Pre Overcloud Deployment

The RHOSP-RHCS hyper-converged cluster was deployed using the RHOSP director node and procedures outlined in the following documents.

- HCI RHOSP 10 and RHCS 2 Reference Architecture
- TripleO - Deployment Best Pratices and Debugging Tips

See Appendix: References for more information regarding the playbook or these documents.

Ironic Node Cleaning

To configure ironic to auto-wipe the metadata on overcloud nodes, the ironic.conf file was modified to include the following:

```
[conductor]
automated_clean = True
[deploy]
erase_devices_priority = 0
erase_devices_metadata_priority = 10
[neutron]
cleaning_network_uuid = <ctlplane_UUID>
```

followed by a restart of the openstack-ironic-conductor service. Once done, the ironic nodes will have all disk partitions removed each time their provisioning state changes from manageable to available. This guarantees that no previously existing system disk can be discovered and used when the OSPd reboots a node. The provisioning state can be manually changed using the ironic node-set-provision-state command.

```
ironic node-set-provision-state <ironic_UUID> manage
ironic node-set-provision-state <ironic_UUID> provide
```

Device Persistence and /dev/disk/by-path/

Due to issues with the devices on multiple controllers not being discovered in the same order across reboots, we altered the ceph.yaml file to use /dev/disk/by-path softlinks rather than the device names. This embeds the PCI slot and disk unit number within a specific controller, was stable across reboots, and simplified OSD and system disk specification because it was constant across hosts in the configuration. A puppet-ceph patch was created to allow the director to use /dev/disk/by-path softlinks for the installation. See related bugs in Appendix: Issues for details.

Ironic Root Device Hinting

Knowing that each of the servers in the configuration have two 500GB drives on one controller and 36 2TB drives on another, root device hinting specified one of the 500GB drives by using the smallest device hint. The Configure Boot step was executed again after introspection

```
openstack baremetal configure boot --root-device=smallest
```

which instructs ironic to use the WWN of one of the 500GB drives for the root device when deployed.

Host Memory Reservation

The supplied nova_mem_cpu_calc.py script was used to determine optimal amount of OSD-compute memory reserved away from Nova to guarantee a sufficient amount for the host operating system and Ceph.

```
 $ ./nova_mem_cpu_calc.py 256 56 36 1 0.05
 Inputs:
 - Total host RAM in GB: 256
 - Total host cores: 56
 - Ceph OSDs per host: 36
 - Average guest memory size in GB: 1
 - Average guest CPU utilization: 5%
{nbsp}
 Results:
 - number of guests allowed based on memory = 98
 - number of guest vCPUs allowed = 400
 - nova.conf reserved_host_memory = 157000 MB
 - nova.conf cpu_allocation_ratio = 7.142857
{nbsp}
 Compare "guest vCPUs allowed" to "guests allowed based on memory" for actual guest count
```

The values for *reserved_host_memory* and *cpu_allocation_ratio* were entered into the compute.yaml file used in the overcloud deployment for inclusion into the nova.conf file on all ironic nodes.

2.3.3. Overcloud Deployment

Overcloud deployment completed in approximately 60 minutes.

```
openstack overcloud deploy --templates \
-r ~/custom-roles.yaml \
-e /usr/share/openstack-tripleo-heat-templates/environments/puppet-pacemaker.yaml \
-e /usr/share/openstack-tripleo-heat-templates/environments/network-isolation.yaml \
-e /usr/share/openstack-tripleo-heat-templates/environments/storage-environment.yaml \
-e ~/network-environment.yaml \
-e ~/ceph.yaml \
-e ~/compute.yaml \
-e ~/layout.yaml
```

The configuration files modified to accommodate the Scale Lab environment and used in the deployment of the overcloud are available in Appendix: Overcloud Deployment Configuration Files.

The deployment configured eight OSD-compute nodes and three monitor-controller nodes. Each RHCS OSD node allocated 34 2TB local disks as OSDs (272 total) and two NVMe SSDs for journaling. Each RHOSP instance had a 100GB cinder volume created, attached, pre-allocated, XFS formatted, and mounted locally for all I/O tests.

Neutron was configured with the following:

- the control plane on a 10-GbE public interface with Jumbo Frames (MTU=9000)
- RHCS on a 40-GbE interface with Jumbo Frames (MTU=9000)
- tenant traffic using a 40-GbE interface with Jumbo Frames (MTU=9000)
- isolation using VXLAN

2.3.4. Post Overcloud Deployment

To improve repeatability of the throughput test results to the thinly provisioned RBD volumes, blocks are pre-allocated using dd prior to formatting them with an XFS file system. This is done because RHCS does not allocate or initialize the RBD volume when it is created so the first write to a Cinder block will incur greater overhead than subsequent writes where allocation is not required.

Ceph PG Settings

The number of placement groups (PGs) for the pool (pg_num) as well as the number of PGs to use when calculating data placement (pgp_num) were adjusted using the recommended PGs per Pool Calculator specifying OpenStack as the Ceph use case with no erasure coding or cache tiering. Because this testing focused on Cinder volume performance, no cinder-backup pool was used and the

percentage of data for that pool was added to that of Cinder volumes. Testing used a total 272 OSDs so the PG counts applied to the storage pools were calculated as seen in Table 3.

Table 3. RHCS PG Calculations

Pool Name	Pool Type	Replica Size	# OSDs	% Data	Target PGs/OSD	PG Count
volumes	Replicated	3	272	78	200	**16384**
vms	Replicated	3	272	15	200	**2048**
images	Replicated	3	272	7	200	**1024**

Chapter 3. Test Design

All disk I/O tests are designed to:

- represent steady state behavior by using random I/O test durations of 10 min + 10 sec/guest to ensure run times are sufficient to fully utilize guest memory and force cache flushes to storage.
- ensure data travels the full I/O path between persistent storage and application by:
  - using vm.dirty_expire_centisecs in conjunction with vm.dirty_ratio.
  - dropping all (server, compute, guest) caches prior to start of every test.
  - ensuring workload generators do not read or write the same file or offset in a file more than once, reducing opportunities for caching in memory.
  - using O_DIRECT with random I/O to bypass guest caching.
- use a data set an order of magnitude larger than aggregate writeback cache size to ensure data is written to disk by end of test.
- push I/O to its limits (excluding low instance count random I/O tests due to rate limiting) with no regard for system operational safety margins that would normally be taken into consideration in production environments.

To produce more repeatable test results and avoid RHCS caching data in memory, the following changes were made to the test environment:

1. cinder volume sizes were 100GB to avoid fitting within memory cache
2. FIO iodepth (how many I/Os it issues to the OS at any given time) was set to 4
3. FIO ioengine was set to libaio

3.1. Performance Statistics

Performance statistics were collected during testing using the Pbench Benchmarking and Performance Analysis tool. `pbench-fio` was used to execute disk I/O workloads and `pbench-uperf` was used to test network speeds and auto start and stop performance statistic collections (iostat, sar) before and after network tests. Additionally, pbench auto-generates graphs for the duration of the test for each statistical tool configured to execute.

3.2. Network Testing

`pbench-uperf` was used to execute both stream and request-response network workloads. The network flow was designed to correspond to the manner in which RHCS works, where OSD hosts are both sending and receiving requests to which they must respond. In the command syntax:

```
pbench-uperf -t rr -p tcp -i 8 -r 60 -m 4096,131072,4194304 -p tcp -C
host1,host2,host3,...,hostN -S hostN,host1,host2,...,hostN-1
```

host[k] sends traffic to host[k-1 mod N] using 4K, 128K and 4M request/response sizes.

3.3. Large File Random I/O Testing

Large-file random multi-stream reads and writes using an FIO 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 RHOSP on RHCS. Additionally, FIO can rate limit throughput, run time and IOPS of individual processes to measure instance scaling throughput as well as the OSD node's aggregate RHOSP instance capacity.

```
/usr/local/bin/fio --client=/root/vms.list.512 --max-jobs=512 --output-format=json
fio.job
```

This is the FIO job file used with the previous FIO command where run time is calculated at 10 minutes plus 10 seconds per instance.

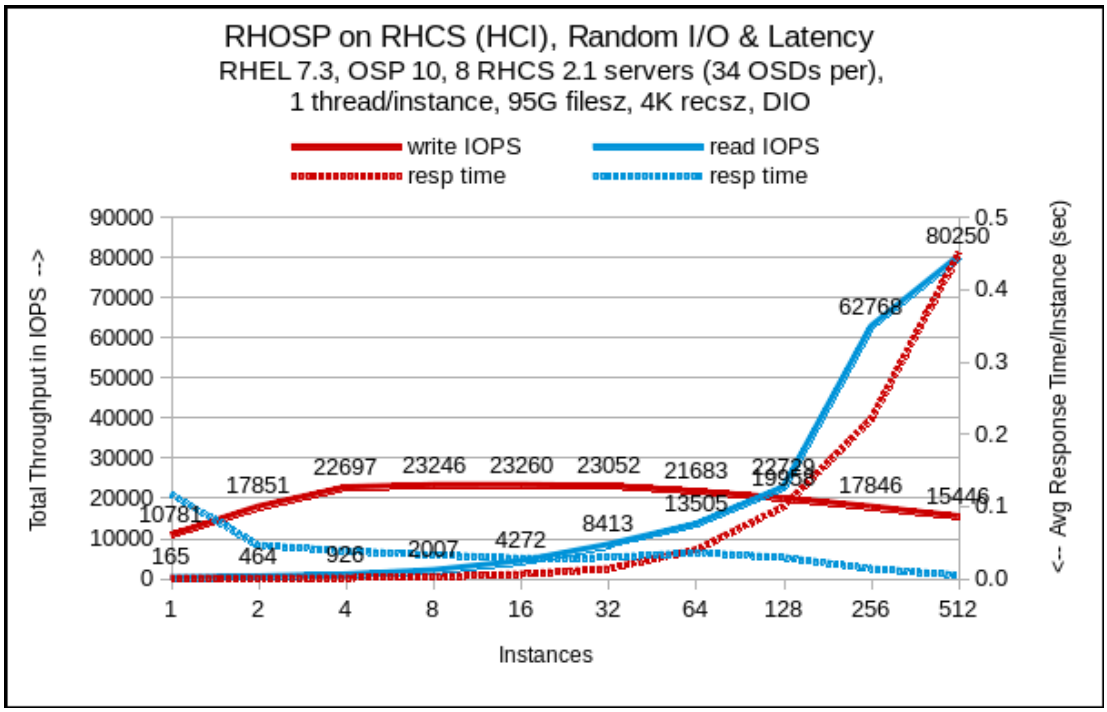
```
[global]
ioengine=libaio
bs=4k
iodepth=4
direct=1
fsync_on_close=1
time_based=1
runtime=5720
clocksource=clock_gettime
ramp_time=10
startdelay=20
[fio]
rw=randread
size=95g
write_bw_log=fio
write_iops_log=fio
write_lat_log=fio
write_hist_log=fio
numjobs=1
per_job_logs=1
log_avg_msec=60000
log_hist_msec=60000
directory=/mnt/ceph/fio
```

Chapter 4. Test Results

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

4.1. Scaling Random I/O

The following figure graphs the effect of varying instance counts on random read and write IOPS and 95th percentile latencies.



Scaling RHOSP Instances - Random IOPS and Latencies

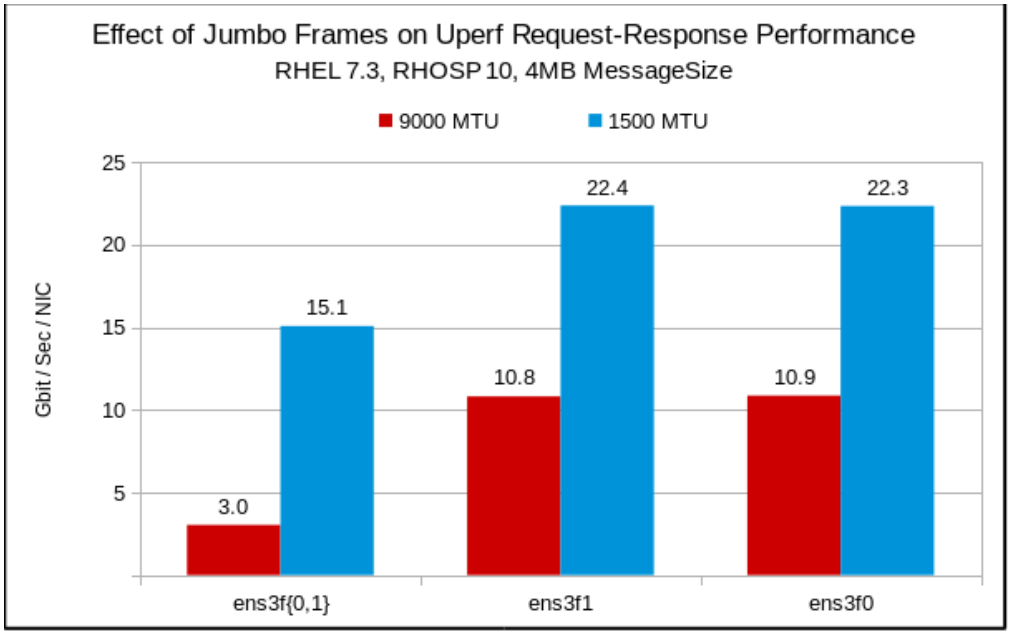
The I/O scaling tests are designed to push as much I/O as possible and are performed using increasing instance counts. All multi-instance tests ensured even distribution across compute nodes.

This workload writes or reads 95GB to Ceph-backed cinder storage using a 4K transfer size for 10-min plus 10-sec per instance so while low counts will run for 10-min, 512 instances run for 95-min. Given the transfer size, random write IOPS at 16 instances peak at 93 MB/sec of application writes but because Ceph writes with 3x replication, that equates to 280 MB/sec of network write traffic, or 35 MB/sec/server.

4.2. Network I/O Performance

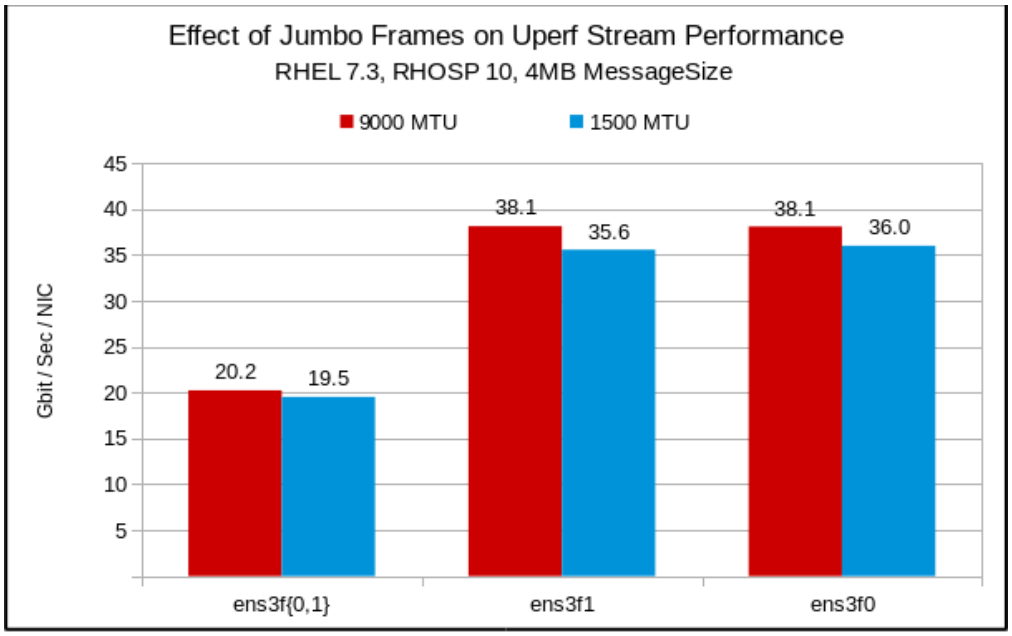
4.2.1. Jumbo Frames - Effect on Throughput

Jumbo frames had a detrimental effect on a request-response workload using a 4MB message size.



Effect of Jumbo Frames on Uperf Request-Response Performance

However a uperf stream workload benefits, albeit slightly, from having jumbo frames configured.

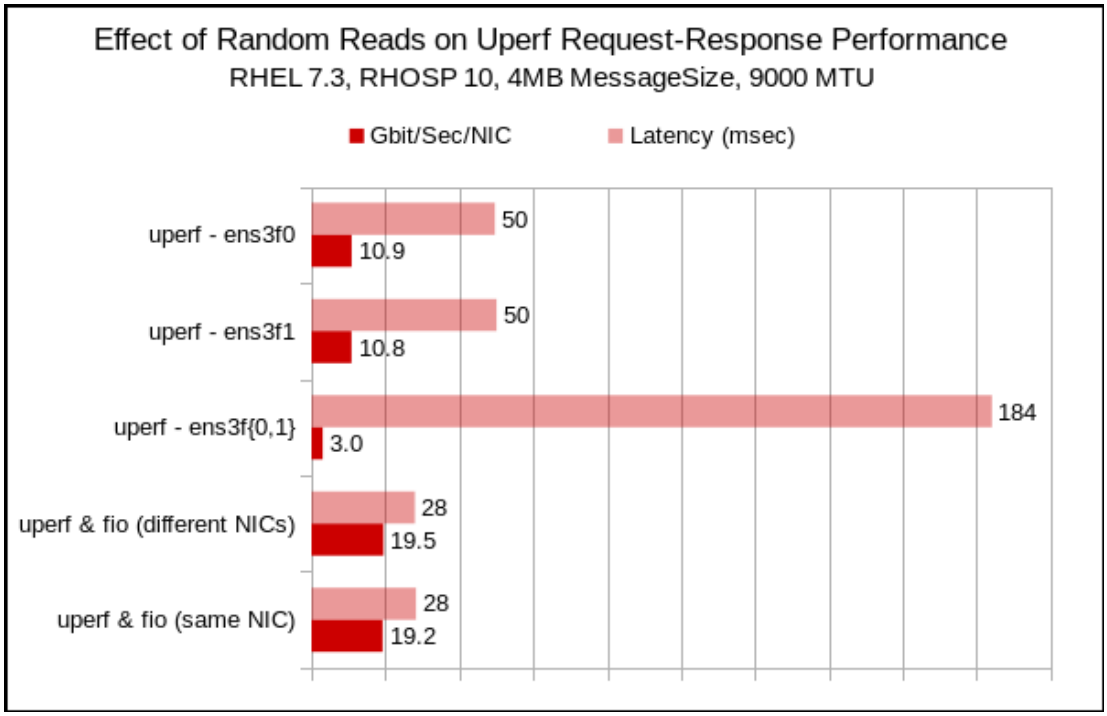


Effect of Jumbo Frames on Uperf Stream Performance

Although a stream workload produces better throughput regardless of MTU rate, the request-response test allows the user to measure response latency and is more representative of real-world network traffic.

4.2.2. Disk I/O Load - Effect on Throughput

This section examines network throughput and latency to determine the impact of a random read workload on network performance. This image graphs network throughput using a 4MB message size and jumbo frames.



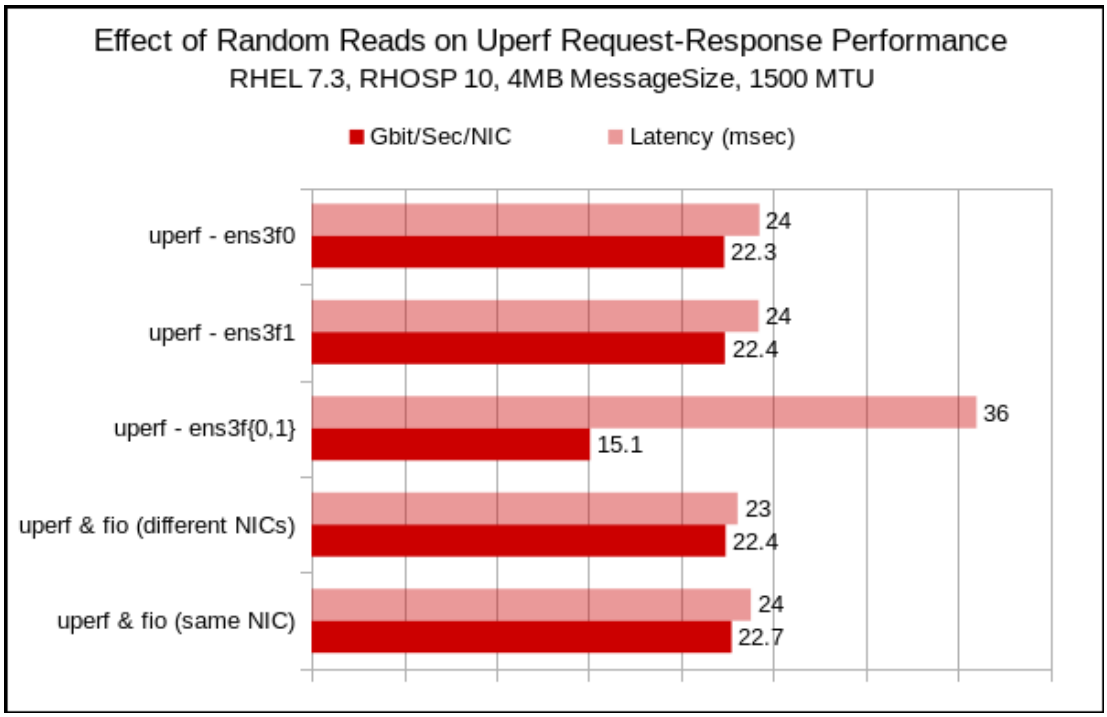
Effect of Random Reads on Uperf Request-Response Performance (9000 MTU)

Note that:

- almost 11 Gbit/sec/NIC is achieved on individual 40-GbE NIC ports but combined the throughput decreases over 70% due to the boost in latency
- adding a random read disk load to the uperf tests almost doubles the network throughput reported by uperf

The first thought WRT the above observations was to repeat without using jumbo frames, having seen its effect on uperf request-response throughput. The second thought was to repeat using the *network-latency* tuned profile rather than *virtual-host* which is applied due to the OSD nodes being hypervisors in the HCI environment.

The next image graphs the results of the same testing without jumbo frames. It not only shows a 50% gain in individual NIC port throughput compared to tests at 9000 MTU, but the throughput loss observed when testing on both NIC ports is reduced from >70% to 32%. Additionally, the combined network and disk IO workload has no impact on network throughput.



Effect of Random Reads on Uperf Request-Response Performance (1500 MTU)

4.2.3. Tuned Profile - Effect on Throughput

From the RHEL 7 documentation on tuning with Tuned profiles, the virtual-host profile focuses on optimizing performance in RHEL 7 virtualization hosts. It:

- favors performance over power savings by setting intel_pstate and max_perf_pct to 100
- decreases the swappiness of virtual memory
- enables transparent huge pages and writes dirty pages back to disk more frequently
- uses cpupower to set the performance cpufreq governor
- sets kernel.sched_min_granularity_ns to 10 µsec, kernel.sched_wakeup_granularity_ns to 15 µsec, kernel.sched_migration_cost to 5 µsec, and vm.dirty_ratio to 40%

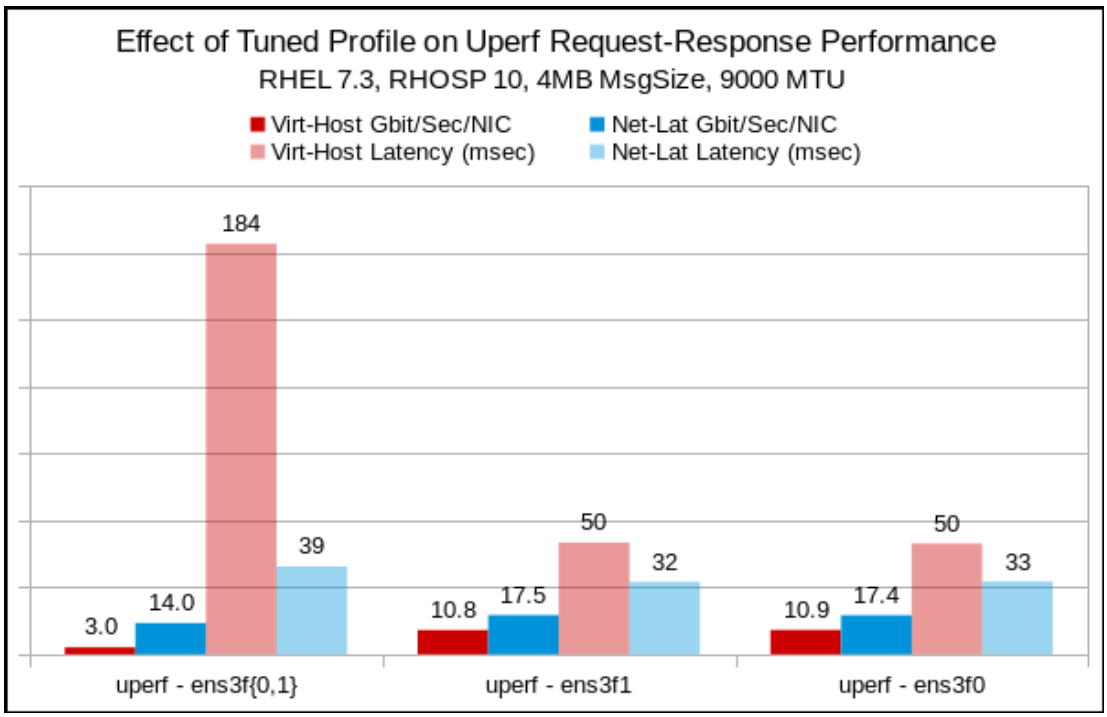
while the network-latency profile is focused on lowering network latency. It:

- favors performance over power savings by setting intel_pstate and min_perf_pct to 100
- disables transparent huge pages and automatic NUMA balancing
- uses cpupower to set the performance cpufreq governor
- requests a cpu_dma_latency value of 1
- sets busy_read and busy_poll times to 50 µsec , and tcp_fastopen to 3

See Appendix C for the exact settings applied by each profile.

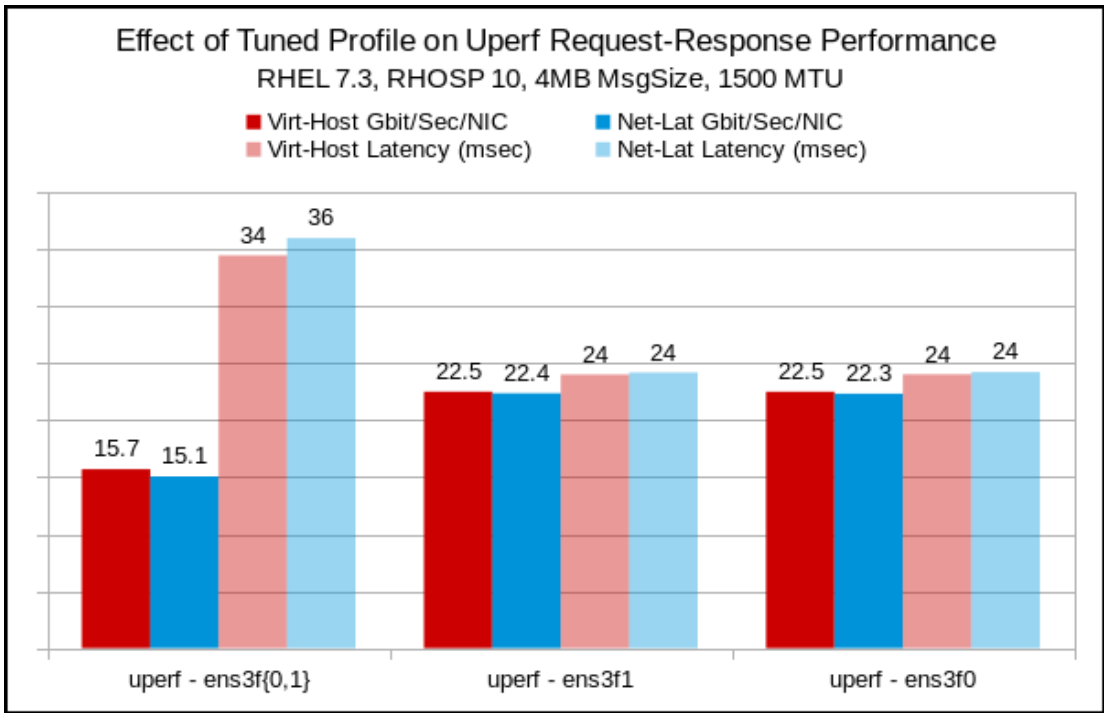
The results of comparing virtual-host with the network-latency tuned profile are in the next images. They show a similar effect as the non jumbo frames results inasmuch as the individual NIC port

throughput is roughly 60% better than that using the virtual-host profile and the combined NIC port workload is only 24% less than the individual NIC port throughput.



Effect of Tuned Profiles on Uperf Request-Response Performance (9000 MTU)

Moreover, if the above test using the network-latency profile is repeated without jumbo frames configured, we see similar results to those observed when using the virtual-host profile without jumbo frames except that there is still a drop in throughput when using both NIC ports.



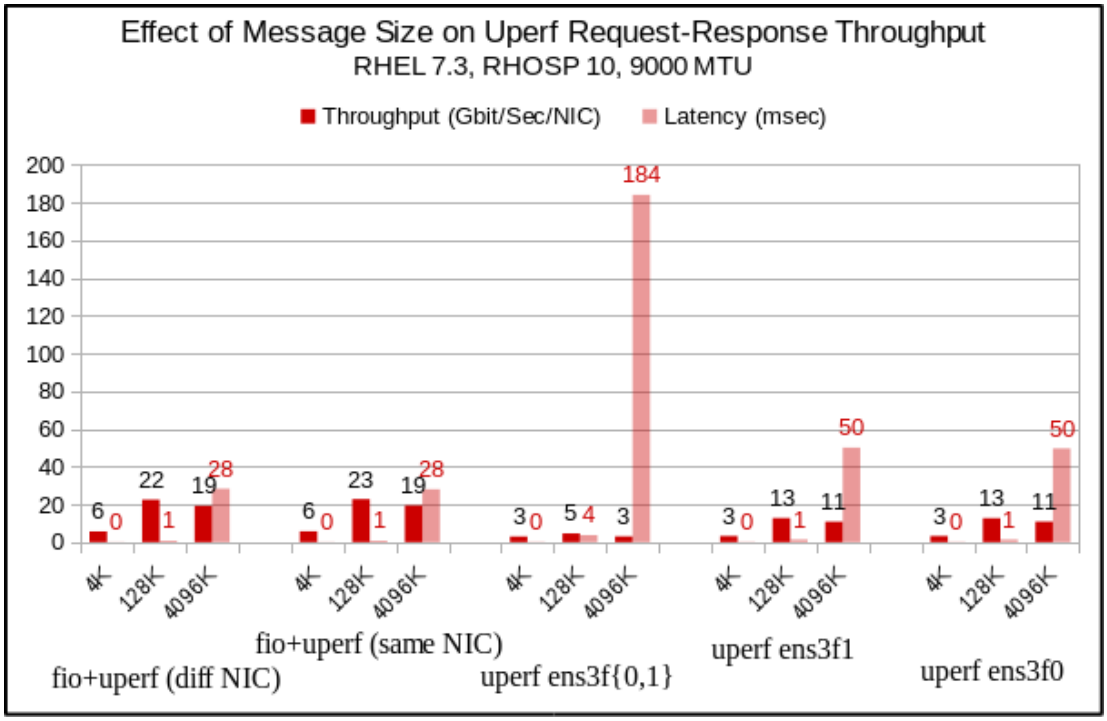
Effect of Tuned Profiles on Uperf Request-Response Performance (1500 MTU)

This confirms that between the MTU rate and the tuned profile, jumbo frames has the largest, albeit detrimental, impact on network request-response throughput and latency using a 4MB message size. The issue of both NIC ports failing to produce the throughput of a single port should be tested further to understand the unexpected results.

4.2.4. Message Size - Effect on Throughput

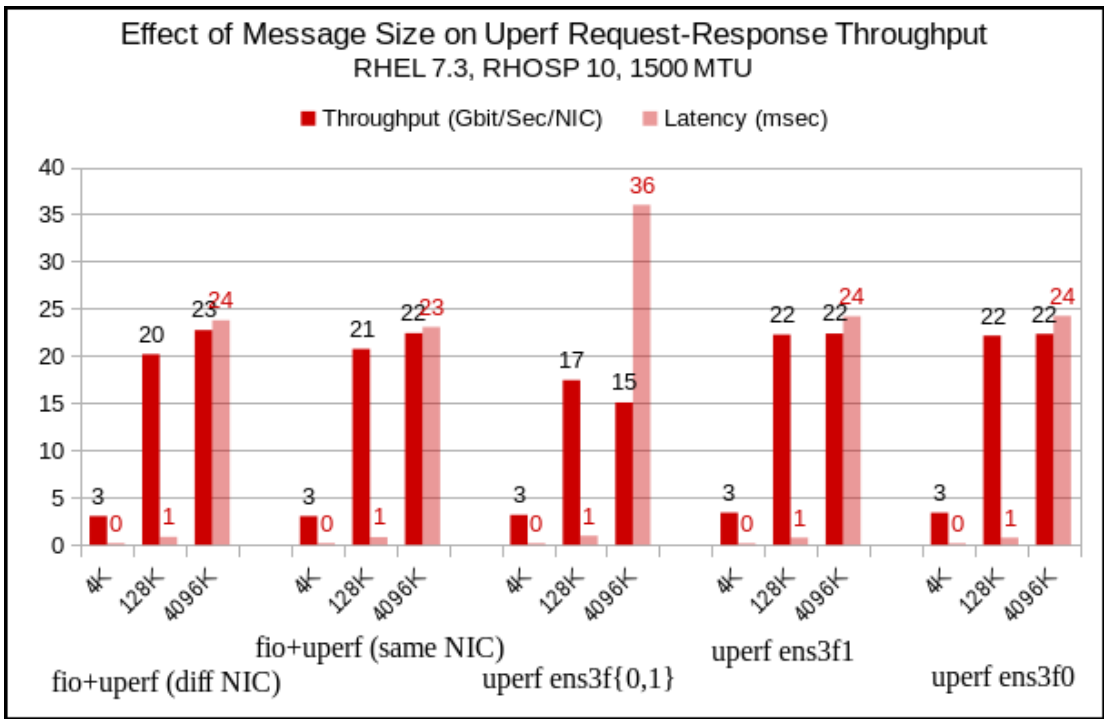
Uperf Request-Response

The performance impact of the message size in a request-response network load is highlighted in the next two images which graph network throughput and latency using varying message sizes in the same test scenarios with and without jumbo frames configured. Note the increased network latencies using a 4MB message size.



Effect of Message Size on Uperf Request-Response Performance (9000 MTU)

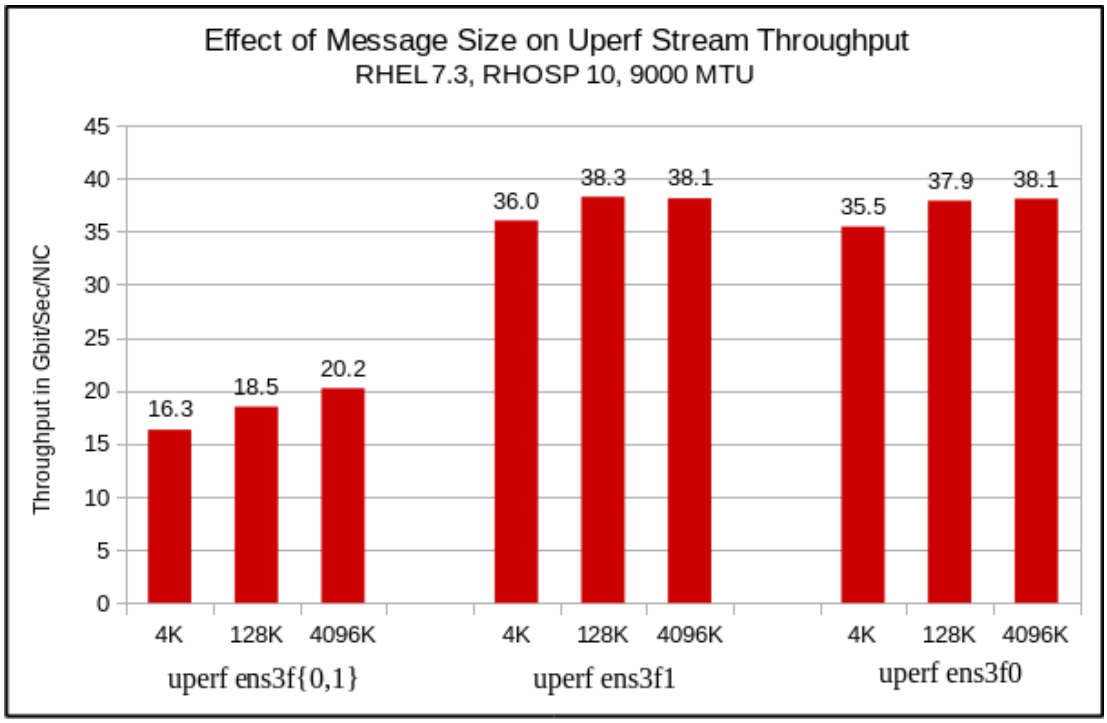
The same tests without jumbo frames continue to highlight a latency boost (albeit less) when using both NIC ports and a 4MB message size.



Effect of Message Size on Uperf Request-Response Performance (1500 MTU)

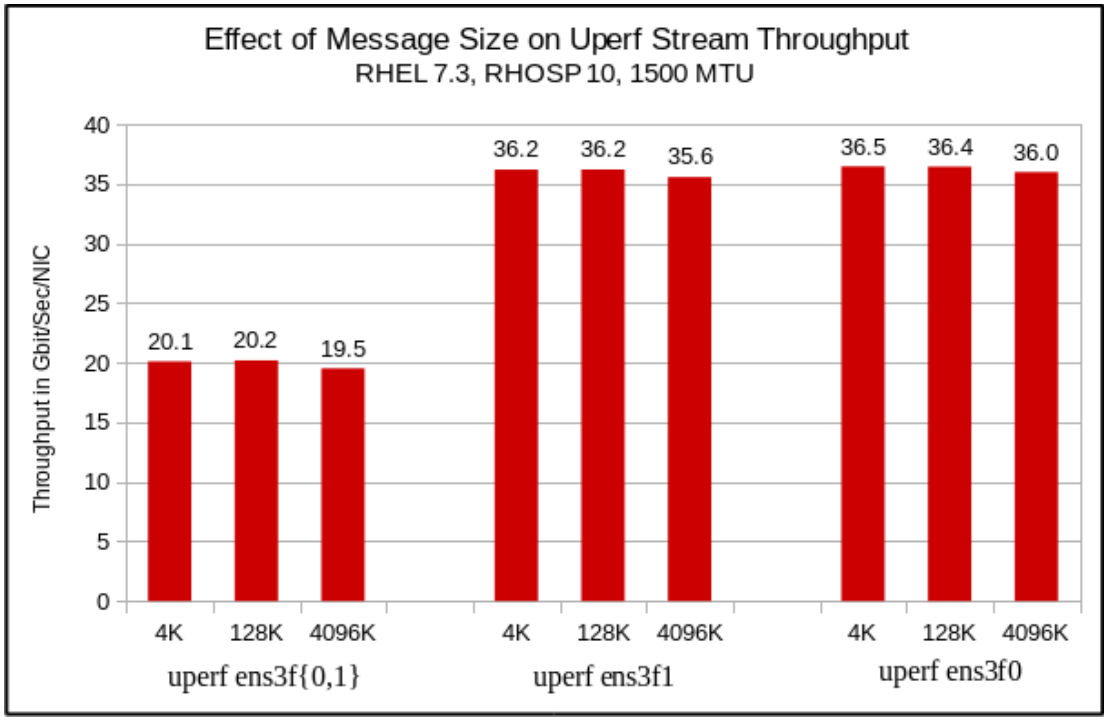
Uperf Stream

These images graph the performance impact of message size in a uperf stream network load with and without jumbo frames.



Effect of Message Size on Uperf Stream Performance (9000 MTU)

Without jumbo frames, message size has much less impact, especially when both NIC ports are used.

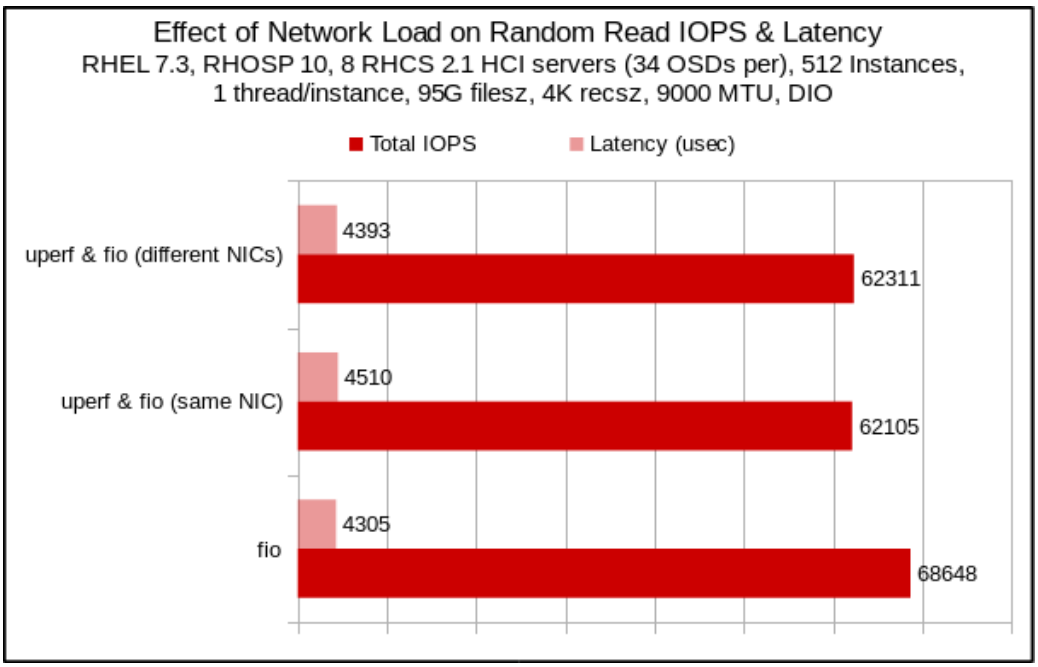


Effect of Message Size on Uperf Stream Performance (1500 MTU)

4.3. Disk I/O Performance

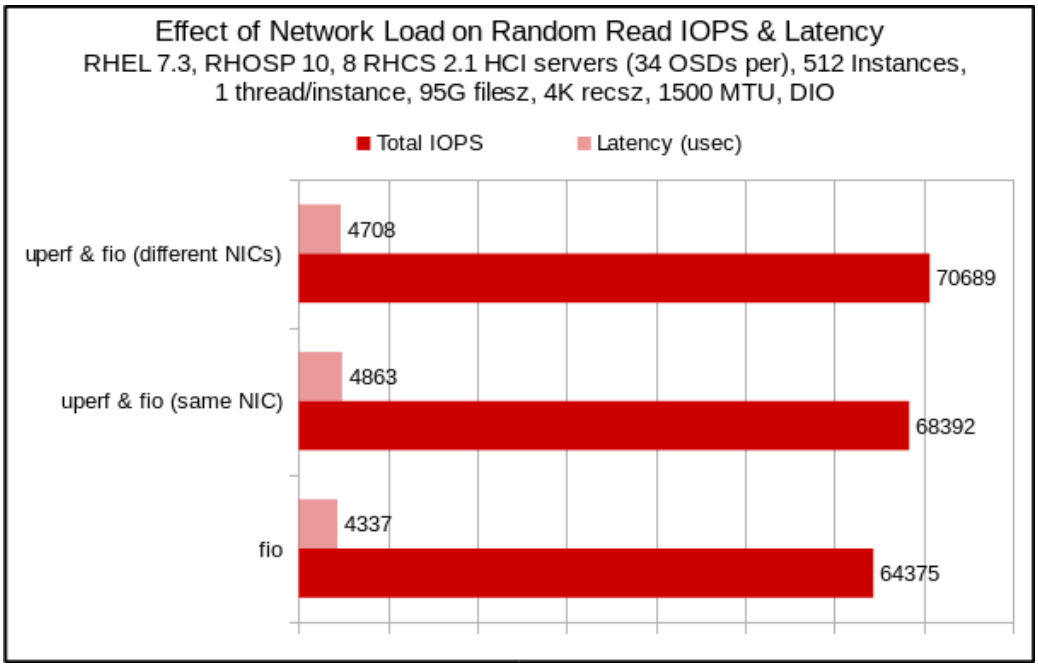
4.3.1. Network Load and Random Read Performance

Random read IOPS and latency are seen in this graph which indicates the disk I/O loses roughly 10% of its throughput performance when combined with a network load.



Effect of Network Load on Random I/O Performance (9000 MTU)

But without jumbo frames configured on the 40-GbE NIC ports, the picture is quite different.



Effect of Network Load on Random I/O Performance (1500 MTU)

Note in the results without jumbo frames that even though the latency for FIO alone is lower, the throughput is as well. This indicates the issue is not overwhelmed storage or network congestion. It is suspected that the power management settings in BIOS are altering performance settings dynamically during testing. The effect of such changes are evident in the previous images graphing the effects of tuned profiles which also modify power management settings.

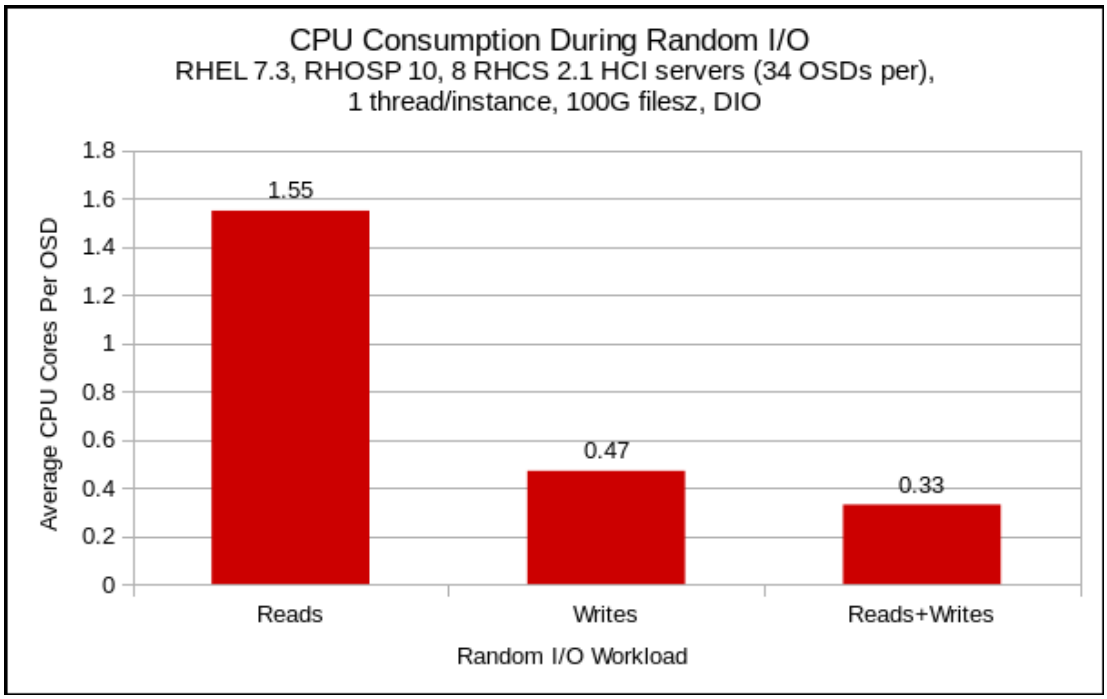
These results are with uperf using a 4MB message size. The effect of the request-response message size on uperf throughput was highlighted in the previous section.

4.4. Resource Consumption

4.4.1. CPU

This testing was performed to support the single core per HDD OSD guideline regarding CPU resource planning in a HCI.

This image graphs the average CPU usage levels during random I/O testing.

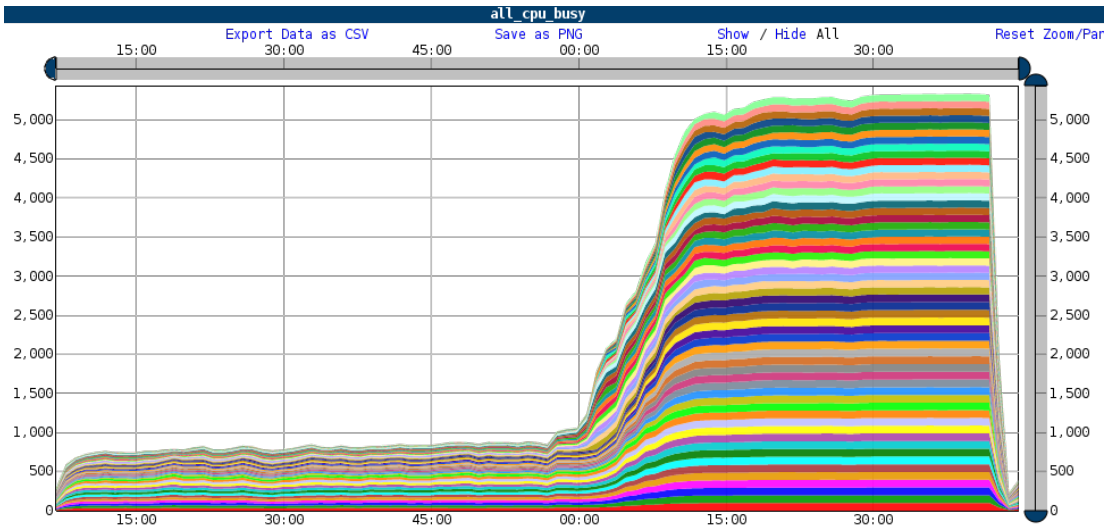


CPU Usage During Random I/O Workloads

Theoretically, the combined usage levels of the read and write tests support the single core per OSD rule of thumb regarding HCI CPU consumption. However, the actual CPU usage during an even mix of random writes and reads indicates at most an average 12 cores busy, approximately one third core per OSD. Further testing of different read:write ratios would be required to see if these levels remained consistent.

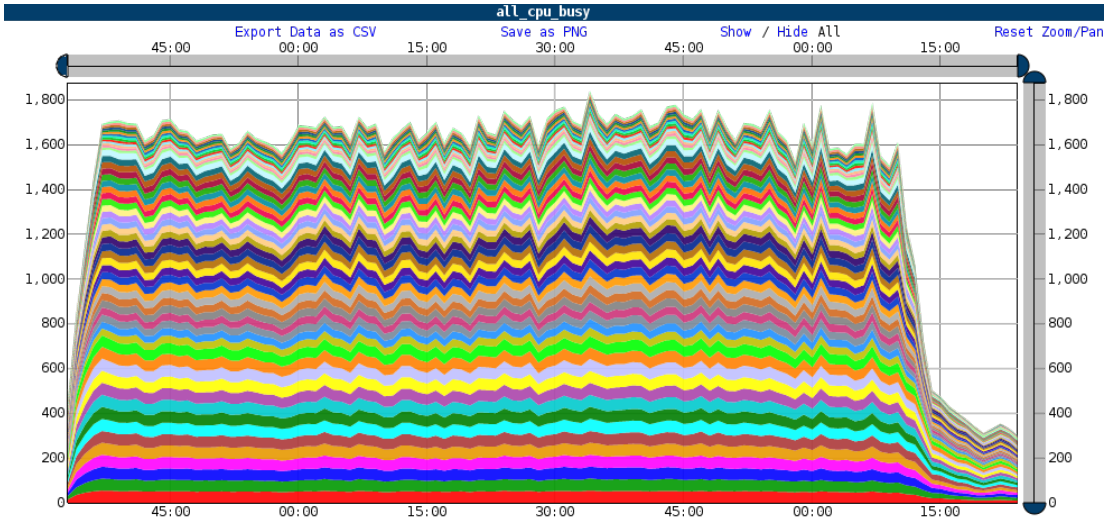
The next three images are the specific levels of CPU usage throughout the test duration that were used to average the overall results per workload.

CPU usage levels during 512 thread random reads show roughly 53 cores busy, or approximately 1.5 cores used by each of the 34 OSDs.



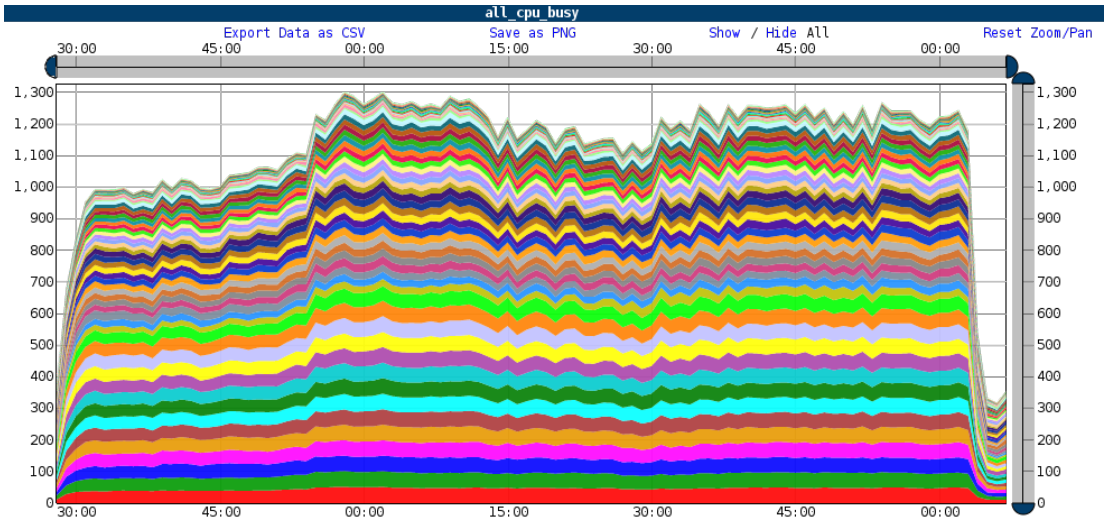
Random Reads CPU Usage

Whereas the CPU usage levels during 512 thread random writes in the image below indicate an average of 16 cores busy, resulting in a half core used per OSD.



Random Writes CPU Usage

The combined usage levels support the single core per OSD rule of thumb regarding HCI CPU consumption. However, the actual CPU usage during an even mix of random writes and reads indicates at most an average 12 cores busy, approximately one third core per OSD.

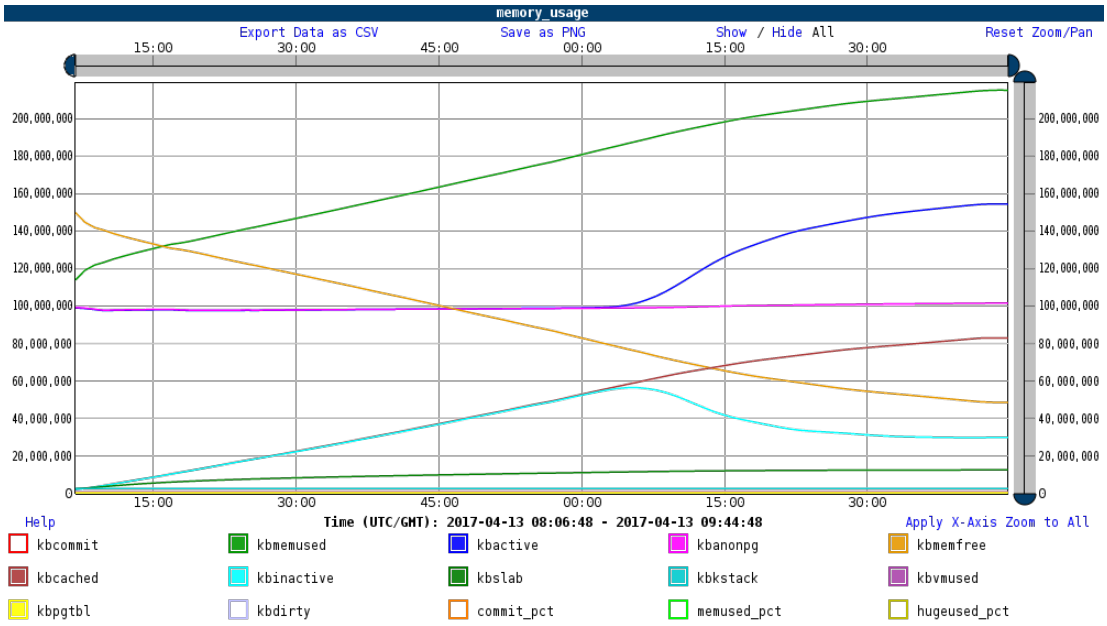


Random Read-Write CPU Usage

4.4.2. Memory

This testing was performed to validate the 3GB/OSD reserved host memory guideline. The memory consumption results indicate an average of approximately 5GB/OSD during random reads which does not verify the existing guideline specifically and as such further testing with reduced memory would be required.

The memory usage during 512 thread random reads indicates that even though all caches are dropped prior to testing, given time it will still engage as seen below where the kilobytes of inactive pages dip while active pages increase.

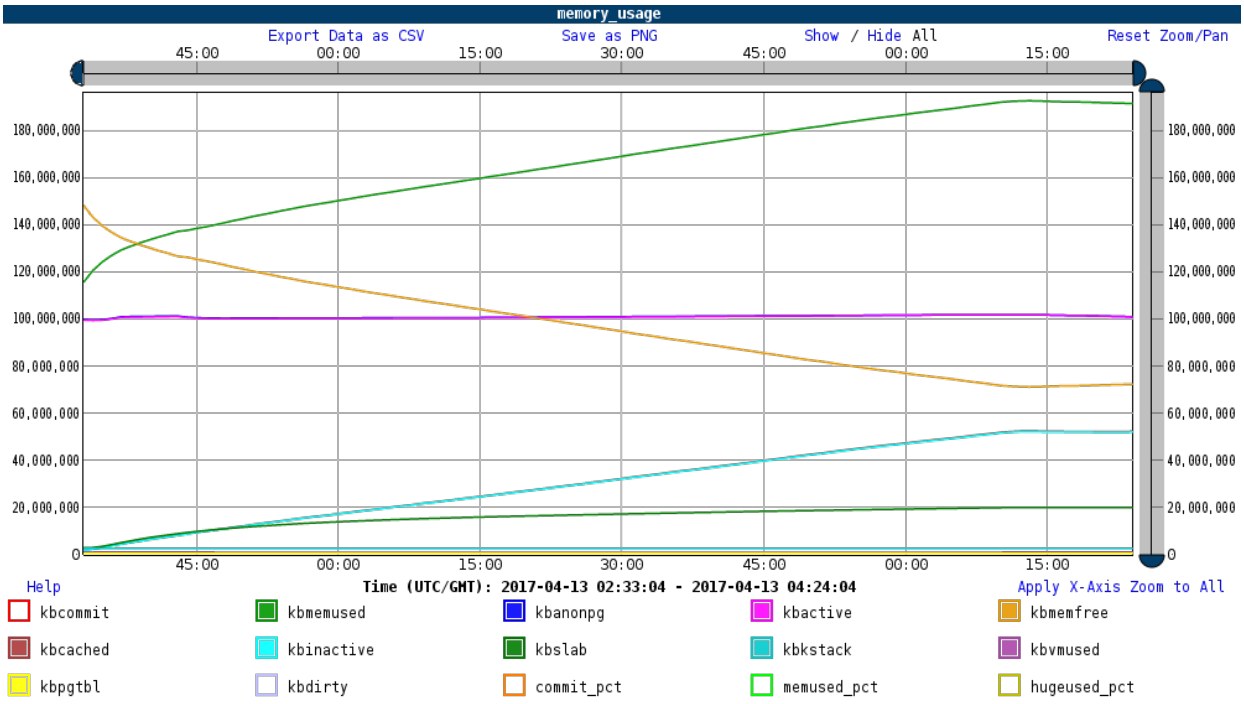


Random Read Memory Usage

Because the scaling test run times are a fixed amount per instance, read caching is only observed in the larger instance count tests as they generate more I/O for a longer period.

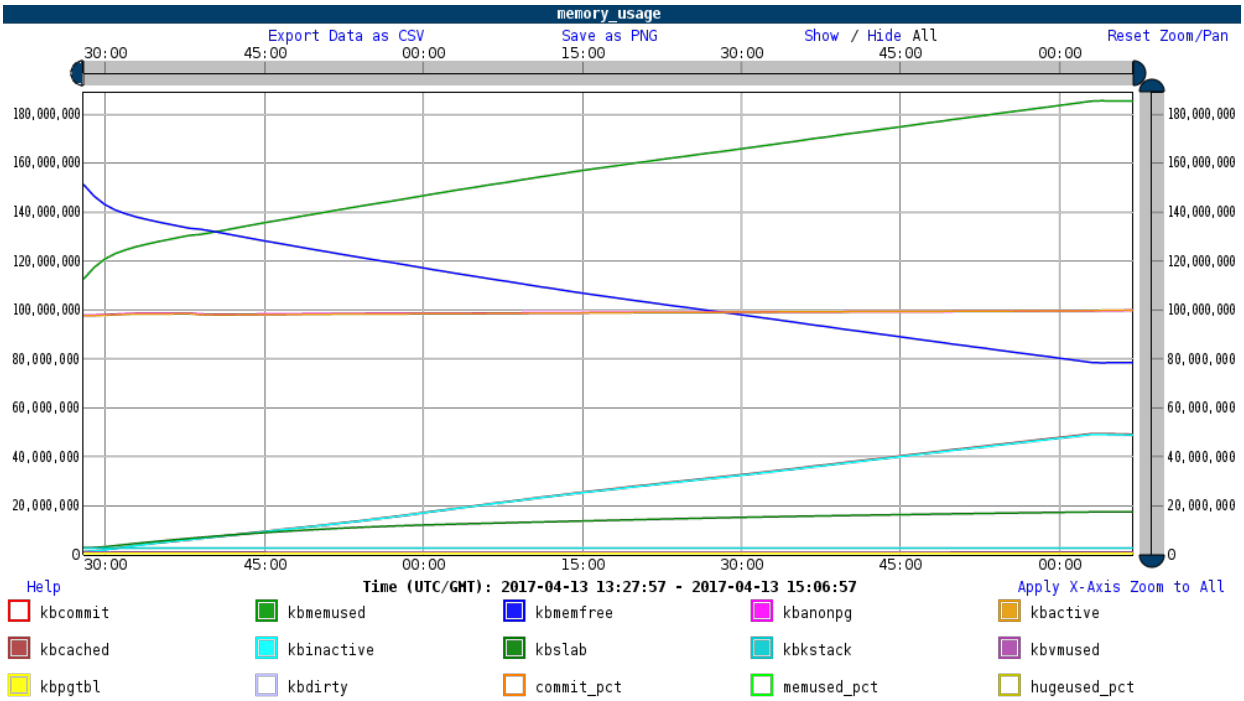
The memory usage during 512 thread random writes shows no such caching as evident in the image below where the KB of cached pages never deviates from that of inactive pages.

The memory consumption results indicate an average of approximately 5GB/OSD during random reads which does not verify the 3GB/OSD reserved host memory guideline specifically and as such further testing with reduced memory would be required.



Random Write Memory Usage

The memory usage pattern for a mixed random read-write workload matches that of random writes, primarily due to using a 50:50 read-write ratio but because Ceph writes must do 3x replication, 75% of the I/O is writing.



Random Read-Write Memory Usage

Appendix A: References

1. *HCI RHOSP 10 and RHCS 2 Reference Architecture* - https://access.redhat.com/documentation/en-us/reference_architectures/2017/html-single/hyper-converged_red_hat_openstack_platform_10_and_red_hat_ceph_storage_2/
2. *TripleO: Deployment Best Practices and Debugging Tips* - https://docs.google.com/document/d/1wAdx-QBCbLVZ7ZRs0Ca1T-mTZL1oAwv1khBpo_KePb8/edit
3. *Ceph PGs per Pool Calculator* - https://access.redhat.com/labs/cephpgc/
4. *Pbench Benchmarking and Performance Analysis Framework* - http://distributed-system-analysis.github.io/pbench/

Appendix B: Undercloud Deployment Configuration Files

B.1. main.yaml

```
 # All variables needed to deploy an undercloud

rhos_release_rpm: https://url.corp.redhat.com/rhos-release

 # OSP/OSPd versioning and build
version: 10
rhos_release: 10-director
 #build: "ga"
build: "2017-03-03.1"

 # RDO
rdo_version: newton
rdo_rc: 3
rdo_trunk: false

deploy_additional_repos: false
repos:
  rhel-7-server:
    baseurl: http://walkabout.foobar.com/released/RHEL-7/7.3/Server/x86_64/os/

 # bulk introspection
introspection: true

 # Dump version file to this directory
version_directory: /etc

 # Stack user password:
 # Make a new password:
 # python -c "from passlib.hash import sha512_crypt; print sha512_crypt.encrypt('password')"
stack_password:
$6$rounds=656000$3ZalBI1dHsb8kfJI$XbtraTo6qcAfAHR158Wf4mLoUrDQsTCIWSAZevFapJIq2OS/qAT
HQ39PwsJ6JusOBue3nxeNm5mTBM3m97EnQ.

 # undercloud control plane interface:
local_interface: em2

overcloud_ssl_endpoints: false
```

```
external_network_vip: 172.21.0.10

 # External private VLAN on undercloud:
deploy_external_private_vlan: true
external_vlan_device: em2.10
private_external_address: 172.21.0.1
private_external_netmask: 255.255.255.0
 # Tripleo maps external network port to noop.yml by default
allow_external_on_compute: false

 # neutron DNS:
dns_server: 10.16.36.29

 # instackenv:
instackenv_json: http://quads.scalelab.redhat.com/cloud/cloud12_instackenv.json
```

Appendix C: Overcloud Deployment Configuration Files

C.1. custom-roles.yaml

```
# Specifies which roles (groups of nodes) will be deployed
# Note this is used as an input to the various *.j2.yaml
# jinja2 templates, so that they are converted into *.yaml
# during the plan creation (via a mistral action/workflow).
#
# The format is a list, with the following format:
#
# * name: (string) mandatory, name of the role, must be unique
#
# CountDefault: (number) optional, default number of nodes, defaults to 0
# sets the default for the {{role.name}}Count parameter in overcloud.yaml
#
# HostnameFormatDefault: (string) optional default format string for hostname
# defaults to '%stackname%-{{role.name.lower()}}-%index%'
# sets the default for {{role.name}}HostnameFormat parameter in overcloud.yaml
#
# ServicesDefault: (list) optional default list of services to be deployed
# on the role, defaults to an empty list. Sets the default for the
# {{role.name}}Services parameter in overcloud.yaml

- name: Controller
  CountDefault: 1
  ServicesDefault:
    - OS::TripleO::Services::CACerts
    - OS::TripleO::Services::CephMon
    - OS::TripleO::Services::CephExternal
    - OS::TripleO::Services::CephRgw
    - OS::TripleO::Services::CinderApi
    - OS::TripleO::Services::CinderBackup
    - OS::TripleO::Services::CinderScheduler
    - OS::TripleO::Services::CinderVolume
    - OS::TripleO::Services::Core
    - OS::TripleO::Services::Kernel
    - OS::TripleO::Services::Keystone
    - OS::TripleO::Services::GlanceApi
    - OS::TripleO::Services::GlanceRegistry
    - OS::TripleO::Services::HeatApi
```

- OS::TripleO::Services::HeatApiCfn
- OS::TripleO::Services::HeatApiCloudwatch
- OS::TripleO::Services::HeatEngine
- OS::TripleO::Services::MySQL
- OS::TripleO::Services::NeutronDhcpAgent
- OS::TripleO::Services::NeutronL3Agent
- OS::TripleO::Services::NeutronMetadataAgent
- OS::TripleO::Services::NeutronApi
- OS::TripleO::Services::NeutronCorePlugin
- OS::TripleO::Services::NeutronOvsAgent
- OS::TripleO::Services::RabbitMQ
- OS::TripleO::Services::HAproxy
- OS::TripleO::Services::Keepalived
- OS::TripleO::Services::Memcached
- OS::TripleO::Services::Pacemaker
- OS::TripleO::Services::Redis
- OS::TripleO::Services::NovaConductor
- OS::TripleO::Services::MongoDb
- OS::TripleO::Services::NovaApi
- OS::TripleO::Services::NovaMetadata
- OS::TripleO::Services::NovaScheduler
- OS::TripleO::Services::NovaConsoleauth
- OS::TripleO::Services::NovaVncProxy
- OS::TripleO::Services::Ntp
- OS::TripleO::Services::SwiftProxy
- OS::TripleO::Services::SwiftStorage
- OS::TripleO::Services::SwiftRingBuilder
- OS::TripleO::Services::Snmp
- OS::TripleO::Services::Timezone
- OS::TripleO::Services::CeilometerApi
- OS::TripleO::Services::CeilometerCollector
- OS::TripleO::Services::CeilometerExpirer
- OS::TripleO::Services::CeilometerAgentCentral
- OS::TripleO::Services::CeilometerAgentNotification
- OS::TripleO::Services::Horizon
- OS::TripleO::Services::GnocchiApi
- OS::TripleO::Services::GnocchiMetricd
- OS::TripleO::Services::GnocchiStatsd
- OS::TripleO::Services::ManilaApi
- OS::TripleO::Services::ManilaScheduler
- OS::TripleO::Services::ManilaBackendGeneric
- OS::TripleO::Services::ManilaBackendNetapp
- OS::TripleO::Services::ManilaBackendCephFs
- OS::TripleO::Services::ManilaShare

```
    - OS::TripleO::Services::AodhApi
    - OS::TripleO::Services::AodhEvaluator
    - OS::TripleO::Services::AodhNotifier
    - OS::TripleO::Services::AodhListener
    - OS::TripleO::Services::SaharaApi
    - OS::TripleO::Services::SaharaEngine
    - OS::TripleO::Services::IronicApi
    - OS::TripleO::Services::IronicConductor
    - OS::TripleO::Services::NovaIronic
    - OS::TripleO::Services::TripleoPackages
    - OS::TripleO::Services::TripleoFirewall
    - OS::TripleO::Services::OpenDaylightApi
    - OS::TripleO::Services::OpenDaylightOvs
    - OS::TripleO::Services::SensuClient
    - OS::TripleO::Services::FluentdClient
    - OS::TripleO::Services::VipHosts

- name: Compute
  CountDefault: 1
  HostnameFormatDefault: '%stackname%-compute-%index%'
  ServicesDefault:
    - OS::TripleO::Services::CACerts
    - OS::TripleO::Services::CephClient
    - OS::TripleO::Services::CephExternal
    - OS::TripleO::Services::Timezone
    - OS::TripleO::Services::Ntp
    - OS::TripleO::Services::Snmp
    - OS::TripleO::Services::NovaCompute
    - OS::TripleO::Services::NovaLibvirt
    - OS::TripleO::Services::Kernel
    - OS::TripleO::Services::ComputeNeutronCorePlugin
    - OS::TripleO::Services::ComputeNeutronOvsAgent
    - OS::TripleO::Services::ComputeCeilometerAgent
    - OS::TripleO::Services::ComputeNeutronL3Agent
    - OS::TripleO::Services::ComputeNeutronMetadataAgent
    - OS::TripleO::Services::TripleoPackages
    - OS::TripleO::Services::TripleoFirewall
    - OS::TripleO::Services::NeutronSriovAgent
    - OS::TripleO::Services::OpenDaylightOvs
    - OS::TripleO::Services::SensuClient
    - OS::TripleO::Services::FluentdClient
    - OS::TripleO::Services::VipHosts

- name: BlockStorage
```

```
  ServicesDefault:
    - OS::TripleO::Services::CACerts
    - OS::TripleO::Services::BlockStorageCinderVolume
    - OS::TripleO::Services::Kernel
    - OS::TripleO::Services::Ntp
    - OS::TripleO::Services::Timezone
    - OS::TripleO::Services::Snmp
    - OS::TripleO::Services::TripleoPackages
    - OS::TripleO::Services::TripleoFirewall
    - OS::TripleO::Services::SensuClient
    - OS::TripleO::Services::FluentdClient
    - OS::TripleO::Services::VipHosts

- name: ObjectStorage
  ServicesDefault:
    - OS::TripleO::Services::CACerts
    - OS::TripleO::Services::Kernel
    - OS::TripleO::Services::Ntp
    - OS::TripleO::Services::SwiftStorage
    - OS::TripleO::Services::SwiftRingBuilder
    - OS::TripleO::Services::Snmp
    - OS::TripleO::Services::Timezone
    - OS::TripleO::Services::TripleoPackages
    - OS::TripleO::Services::TripleoFirewall
    - OS::TripleO::Services::SensuClient
    - OS::TripleO::Services::FluentdClient
    - OS::TripleO::Services::VipHosts

- name: CephStorage
  ServicesDefault:
    - OS::TripleO::Services::CACerts
    - OS::TripleO::Services::CephOSD
    - OS::TripleO::Services::Kernel
    - OS::TripleO::Services::Ntp
    - OS::TripleO::Services::Snmp
    - OS::TripleO::Services::Timezone
    - OS::TripleO::Services::TripleoPackages
    - OS::TripleO::Services::TripleoFirewall
    - OS::TripleO::Services::SensuClient
    - OS::TripleO::Services::FluentdClient
    - OS::TripleO::Services::VipHosts

- name: OsdCompute
  CountDefault: 0
```

```
  HostnameFormatDefault: '%stackname%-osd-compute-%index%'
  ServicesDefault:
    - OS::TripleO::Services::CephOSD
    - OS::TripleO::Services::CACerts
    - OS::TripleO::Services::CephClient
    - OS::TripleO::Services::CephExternal
    - OS::TripleO::Services::Timezone
    - OS::TripleO::Services::Ntp
    - OS::TripleO::Services::Snmp
    - OS::TripleO::Services::NovaCompute
    - OS::TripleO::Services::NovaLibvirt
    - OS::TripleO::Services::Kernel
    - OS::TripleO::Services::ComputeNeutronCorePlugin
    - OS::TripleO::Services::ComputeNeutronOvsAgent
    - OS::TripleO::Services::ComputeCeilometerAgent
    - OS::TripleO::Services::ComputeNeutronL3Agent
    - OS::TripleO::Services::ComputeNeutronMetadataAgent
    - OS::TripleO::Services::TripleoPackages
    - OS::TripleO::Services::TripleoFirewall
    - OS::TripleO::Services::NeutronSriovAgent
    - OS::TripleO::Services::OpenDaylightOvs
    - OS::TripleO::Services::SensuClient
    - OS::TripleO::Services::FluentdClient
    - OS::TripleO::Services::VipHosts
```

C.2. layout.yaml

```
resource_registry:
  OS::TripleO::OsdCompute::Ports::InternalApiPort: /usr/share/openstack-tripleo-heat-
templates/network/ports/internal_api.yaml
  OS::TripleO::OsdCompute::Ports::TenantPort: /usr/share/openstack-tripleo-heat-
templates/network/ports/tenant.yaml
  OS::TripleO::OsdCompute::Ports::StoragePort: /usr/share/openstack-tripleo-heat-
templates/network/ports/storage.yaml
  OS::TripleO::OsdCompute::Ports::StorageMgmtPort: /usr/share/openstack-tripleo-heat-
templates/network/ports/storage_mgmt.yaml
  OS::TripleO::OsdCompute::Ports::ExternalPort: /usr/share/openstack-tripleo-heat-
templates/network/ports/noop.yaml

parameter_defaults:
  NtpServer: 10.5.26.10

  ControllerCount: 3
  ComputeCount: 0
```

```
  CephStorageCount: 0
  OsdComputeCount: 8

  OvercloudControlFlavor: control
  OvercloudOsdComputeFlavor: ceph-storage
```

C.3. network-environment.yaml

```
resource_registry:
  OS::TripleO::Controller::Net::SoftwareConfig: /home/stack/nic-configs/controller-nics.yaml
  OS::TripleO::OsdCompute::Net::SoftwareConfig: /home/stack/nic-configs/compute-nics.yaml
parameter_defaults:
  NeutronExternalNetworkBridge: """"
  InternalApiNetCidr: 172.16.0.0/24
  TenantNetCidr: 172.17.0.0/24
  StorageNetCidr: 172.18.0.0/24
  StorageMgmtNetCidr: 172.19.0.0/24
  ManagementNetCidr: 172.20.0.0/24
  ExternalNetCidr: 172.21.0.0/24
  InternalApiAllocationPools: [{'start': '172.16.0.10', 'end': '172.16.0.200'}]
  TenantAllocationPools: [{'start': '172.17.0.10', 'end': '172.17.0.200'}]
  StorageAllocationPools: [{'start': '172.18.0.10', 'end': '172.18.0.200'}]
  StorageMgmtAllocationPools: [{'start': '172.19.0.10', 'end': '172.19.0.200'}]
  ManagementAllocationPools: [{'start': '172.20.0.10', 'end': '172.20.0.200'}]
  ExternalAllocationPools: [{'start': '172.21.0.10', 'end': '172.21.0.100'}]
  # Set to the router gateway on the external network
  ExternalInterfaceDefaultRoute: 172.21.0.1
  PublicVirtualFixedIPs: [{'ip_address':'172.21.0.10'}]
  # Gateway router for the provisioning network (or Undercloud IP)
  ControlPlaneDefaultRoute: 192.0.2.1
  # The IP address of the EC2 metadata server. Generally the IP of the Undercloud
  EC2MetadataIp: 192.0.2.1
  # Define the DNS servers (maximum 2) for the overcloud nodes
  DnsServers: ["10.122.64.161"]
  InternalApiNetworkVlanID: 301
  StorageNetworkVlanID: 302
  StorageMgmtNetworkVlanID: 303
  TenantNetworkVlanID: 304
  ManagementNetworkVlanID: 305
  ExternalNetworkVlanID: 10
```

C.4. compute.yaml

```
parameter_defaults:
```

```
  ExtraConfig:
    nova::compute::reserved_host_memory: 157000
    nova::cpu_allocation_ratio: 7.142857
```

C.5. ceph.yaml

```
parameter_defaults:
  ExtraConfig:
    ceph::profile::params::fsid: eb2bb192-b1c9-11e6-9205-525400330666
    ceph::profile::params::osd_pool_default_pg_num: 256
    ceph::profile::params::osd_pool_default_pgp_num: 256
    ceph::profile::params::osd_pool_default_size: 3
    ceph::profile::params::osd_pool_default_min_size: 2
    ceph::profile::params::osd_recovery_max_active: 1
    ceph::profile::params::osd_max_backfills: 1
    ceph::profile::params::osd_recovery_op_priority: 1
  OsdComputeExtraConfig:
    ceph::profile::params::osd_journal_size: 5120
    ceph::profile::params::osds:
      '/dev/disk/by-path/pci-0000:03:00.0-scsi-0:2:0:0':
        journal: '/dev/nvme0n1'
      '/dev/disk/by-path/pci-0000:03:00.0-scsi-0:2:1:0':
        journal: '/dev/nvme0n1'
      '/dev/disk/by-path/pci-0000:03:00.0-scsi-0:2:2:0':
        journal: '/dev/nvme0n1'
      '/dev/disk/by-path/pci-0000:03:00.0-scsi-0:2:3:0':
        journal: '/dev/nvme0n1'
      '/dev/disk/by-path/pci-0000:03:00.0-scsi-0:2:4:0':
        journal: '/dev/nvme0n1'
      '/dev/disk/by-path/pci-0000:03:00.0-scsi-0:2:5:0':
        journal: '/dev/nvme0n1'
      '/dev/disk/by-path/pci-0000:03:00.0-scsi-0:2:6:0':
        journal: '/dev/nvme0n1'
      '/dev/disk/by-path/pci-0000:03:00.0-scsi-0:2:7:0':
        journal: '/dev/nvme0n1'
      '/dev/disk/by-path/pci-0000:03:00.0-scsi-0:2:8:0':
        journal: '/dev/nvme0n1'
      '/dev/disk/by-path/pci-0000:03:00.0-scsi-0:2:9:0':
        journal: '/dev/nvme0n1'
      '/dev/disk/by-path/pci-0000:03:00.0-scsi-0:2:10:0':
        journal: '/dev/nvme0n1'
      '/dev/disk/by-path/pci-0000:03:00.0-scsi-0:2:11:0':
        journal: '/dev/nvme0n1'
      '/dev/disk/by-path/pci-0000:03:00.0-scsi-0:2:12:0':
```

```
    journal: '/dev/nvme0n1'
  '/dev/disk/by-path/pci-0000:03:00.0-scsi-0:2:13:0':
    journal: '/dev/nvme0n1'
  '/dev/disk/by-path/pci-0000:03:00.0-scsi-0:2:14:0':
    journal: '/dev/nvme0n1'
  '/dev/disk/by-path/pci-0000:03:00.0-scsi-0:2:15:0':
    journal: '/dev/nvme0n1'
  '/dev/disk/by-path/pci-0000:03:00.0-scsi-0:2:16:0':
    journal: '/dev/nvme0n1'
  '/dev/disk/by-path/pci-0000:03:00.0-scsi-0:2:17:0':
    journal: '/dev/nvme1n1'
  '/dev/disk/by-path/pci-0000:03:00.0-scsi-0:2:18:0':
    journal: '/dev/nvme1n1'
  '/dev/disk/by-path/pci-0000:03:00.0-scsi-0:2:19:0':
    journal: '/dev/nvme1n1'
  '/dev/disk/by-path/pci-0000:03:00.0-scsi-0:2:20:0':
    journal: '/dev/nvme1n1'
  '/dev/disk/by-path/pci-0000:03:00.0-scsi-0:2:21:0':
    journal: '/dev/nvme1n1'
  '/dev/disk/by-path/pci-0000:03:00.0-scsi-0:2:22:0':
    journal: '/dev/nvme1n1'
  '/dev/disk/by-path/pci-0000:03:00.0-scsi-0:2:23:0':
    journal: '/dev/nvme1n1'
  '/dev/disk/by-path/pci-0000:03:00.0-scsi-0:2:24:0':
    journal: '/dev/nvme1n1'
  '/dev/disk/by-path/pci-0000:03:00.0-scsi-0:2:25:0':
    journal: '/dev/nvme1n1'
  '/dev/disk/by-path/pci-0000:03:00.0-scsi-0:2:26:0':
    journal: '/dev/nvme1n1'
  '/dev/disk/by-path/pci-0000:03:00.0-scsi-0:2:27:0':
    journal: '/dev/nvme1n1'
  '/dev/disk/by-path/pci-0000:03:00.0-scsi-0:2:28:0':
    journal: '/dev/nvme1n1'
  '/dev/disk/by-path/pci-0000:03:00.0-scsi-0:2:29:0':
    journal: '/dev/nvme1n1'
  '/dev/disk/by-path/pci-0000:03:00.0-scsi-0:2:30:0':
    journal: '/dev/nvme1n1'
  '/dev/disk/by-path/pci-0000:03:00.0-scsi-0:2:31:0':
    journal: '/dev/nvme1n1'
  '/dev/disk/by-path/pci-0000:03:00.0-scsi-0:2:32:0':
    journal: '/dev/nvme1n1'
  '/dev/disk/by-path/pci-0000:03:00.0-scsi-0:2:33:0':
    journal: '/dev/nvme1n1'
```

Appendix D: Tuned Profiles

Below are the system settings applied by tuned for each profile including the default profile throughput-performance.

D.1. throughput-performance

[cpu]
governor = performance
energy_perf_bias = performance
[disk]
readahead = 4096
[sysctl]
kernel.sched_min_granularity_ns = 10000000
kernel.sched_wakeup_granularity_ns = 15000000
kernel.sched_migration_cost_ns = 500000
vm.dirty_ratio = 40
vm.dirty_background_ratio = 10
vm.swappiness = 10
[vm]
transparent hugepages = enabled

D.2. virtual-host

[cpu]
governor = performance
energy_perf_bias = performance
min_perf_pct = 100
[disk]
readahead = 4096
[sysctl]
kernel.sched_migration_cost_ns = 5000000
kernel.sched_min_granularity_ns = 10000000
kernel.sched_wakeup_granularity_ns = 15000000
vm.dirty_ratio = 40
vm.dirty_background_ratio = 5
vm.swappiness = 10

D.3. network-latency

[cpu]
force_latency = 1
governor = performance

```
energy_perf_bias = performance
min_perf_pct = 100
[sysctl]
kernel.numa_balancing = 0
kernel.sched_migration_cost_ns = 5000000
kernel.sched_min_granularity_ns = 10000000
net.core.busy_read = 50
net.core.busy_poll = 50
net.ipv4.tcp_fastopen = 3
vm.dirty_ratio = 10
vm.dirty_background_ratio = 3
vm.swappiness = 10
[vm]
transparent_hugepages = never
```

Appendix E: Issues

Some RHOSP HCI deployment issues were found and documented as bugzilla bug reports. All issues encountered are included in the following table.

Bug ID	Status	Description
bz 1430588	Open	gnocchi with ceph storage driver creates many objects
tripleo bug 1676926	Open	aodh-dbsync fails during tripleo deployment
bz 1444909	Open	puppet-ceph usage of /dev/disk/by-path device names
bz 1438572	RFE	complex ceph OSD specification of SSD journals
bz 1438590	RFE	allow specification of OSD block devices by rule
bz 1445451	Open	introspection does not report /dev/disk/by-path device name

With the preceding exceptions, the test environment installed and performed well with respect to hardware utilization for the workloads in this configuration.


redhat.