

Test Configuration (BlueStore LVM simple)

- Hardware - baremetal, no HCI
 - 5x RBD OSD servers (5x HDD, 1 NVMe)
 - 3x ceph-MON / ceph-MGR
- RHEL 7.6 (kernel-3.10.0-957)
- RHCS 3.2 (12.2.8-84.el7cp)
- ceph-ansible-3.2.4-1.el7cp
- FIO v3.12
- all.yml

```
ceph_repository_type: cdn
ceph_origin: repository
ceph_repository: rhcs
ceph_rhcs_version: 3
monitor_interface: ens1f1
public_network: 192.168.10.0/24
cluster_network: 172.17.50.0/24
osd_objectstore: bluestore
osd_memory_target: 16000000000
ceph_docker_image: rhceph
ceph_docker_image_tag: ceph-3.2-rhel-7-containers-candidate-51832-20190205161527
ceph_docker_registry: brew-pulp-docker01.web.prod.ext.phx2.redhat.com:8888
containerized_deployment: true
```
- osds.yml

```
osd_scenario: lvm
devices:
  - /dev/nvme0n1
osds_per_device: 4
```

RBD Configuration

- 2x 50GB RBD volumes per OSD server
- Each volume is mapped back to the OSD server acting as clients

RBD Workload (FIO)

- Large-file, random, multi-stream reads and writes are performed using the [pbench-fio](#) based workload to execute I/O processes from each client in parallel using options to start and stop all threads at approximately the same time allowing aggregation of the per-thread results to achieve system-wide IOPS throughput. This testing executed the following fio command on five concurrent clients to execute (random or sequential) writes & reads for 10+ minutes.

```
fio --client=clients.5 --max-jobs=5 --output-format=json fio.job
```

The fio.job file used:

```
[global]
ioengine=libaio
bs=4k
iodepth=4
```

```

direct=1
fsync_on_close=1
time_based=1
runtime=650
clocksource=clock_gettime
ramp_time=10
startdelay=0

[pbfio]
rw=randwrite
size=75g
write_bw_log=fio
write_iops_log=fio
write_lat_log=fio
write_hist_log=fio
numjobs=10
per_job_logs=1
log_avg_msec=10000
log_hist_msec=10000
filename=/mnt0/ceph/file0:/mnt0/ceph/file1:/mnt0/ceph/file2:/mnt0/ceph/file3:/mnt0/ceph/file4:/mnt1/ceph/file0:/mnt1/ceph/file1:/mnt1/ceph/file2:/mnt1/ceph/file3:/mnt1/ceph/file4

```

- Random I/O tests
 - use the *libaio* ioengine
 - sync file contents at closing
 - bypass cache via O_DIRECT
- The metrics measured are the total IOPS and the average latency per thread.

Results

With 5 CPU cores per OSD on these older systems, parity was achieved with non-containerized Ceph, essentially the same as containerized except for CGroup limits and Seccomp

CPU Limits	Write IOPS	Write 95% Latency (usec)	Read IOPS	Read 95% Latency (usec)
1 (default)	6957	93508	46675	22157
3	19066 (+274%)	44092	160229 (+343%)	1502
5	31233 (+449%)	14241	262242 (+562%)	1840
none	31215 (+449%)	13323	265929 (+570%)	1843
none, no seccomp	31031 (+446%)	13413	270955 (+581%)	1872

NOTE: The percentage deltas are relative to the default CPU quota=1