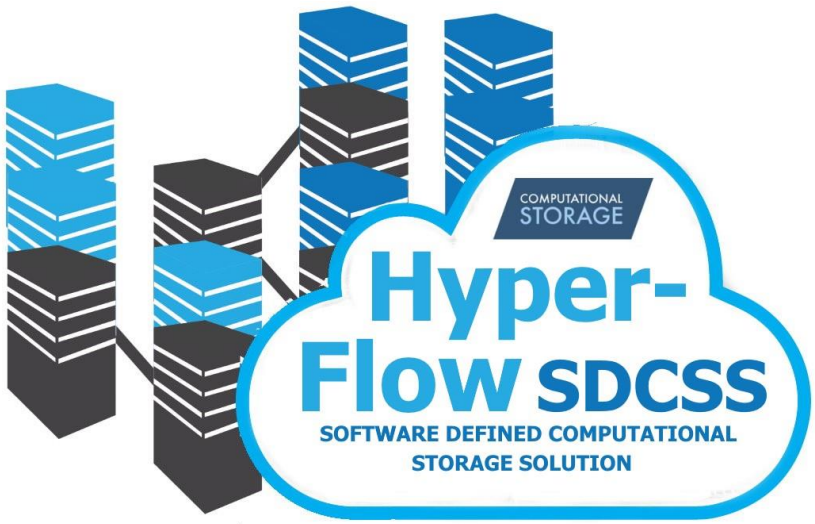
p +61 1300 113 112
e info@hyperscalers.com

HyperFlow Software Defined Computational Storage Solution (SDCSS)

Based on Ceph using Computational Storage NVMe by ScaleFlux

Reference guide

HYPERSCALERS



Friday, 2 September 2022

1 CONTENTS

Table of Figures

INTRODUCTION

The HyperFlow Software Defined Computational Storage Solution (SDCSS) by Hyperscalers [1] and ScaleFlux [2] was co-developed to fill a need experienced by many organisations for easy to consume, low cost yet blazingly fast NVMe based Computational Storage delivered in the context of the highly-available, mature, and flexible Block, File or Object storage services provided by Ceph.

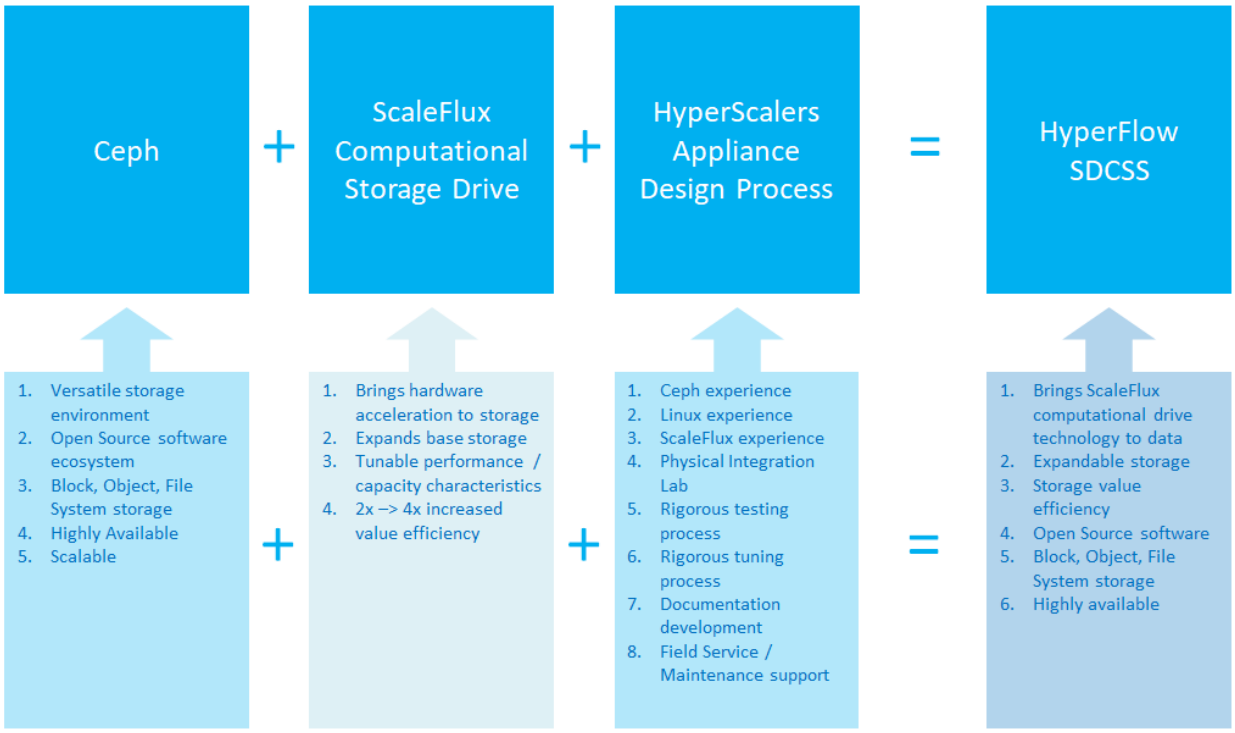


Figure 1 Values of the HyperFlow Software Defined Computational Storage Solution

Hyperscalers is the world's first open supply chain Original Equipment Manufacturer-OEM, solving Information Technology challenges through standardization of best practices and hyperscale inspired practices and efficiencies. Hyperscalers offers choice across two open hardware architectures:

- Hyperscale - high efficiency open compute equipment as used by macro service providers
- Tier 1 Original – conventional equipment as per established Tier 1 OEM suppliers.

Each architecture is complete with network, compute, storage, and converged GP GPU infrastructure elements, and is open / free from vendor lock-in.

Hyperscalers' appliance solutions are packaged complete with hardware, software and pre-built (customisable) configurations using an in-house IP Appliance Design Process and

validated in partnership with software manufacturer partners. Hyperscalers Lab as a Service (LaaS) provides a means for channel partners and their customers to test drive various appliances in order to prove which option is right for their business. Hyperscalers appliance solutions are ideally suited to IaaS, PaaS, SaaS and GPUaaS providers needing to hyperscale their services from anywhere.

< About all other technology partners >

ScaleFlux [2] is the pioneer in deploying Computational Storage at scale. Computational Storage is the foundation for modern, data-driven infrastructure that enables responsive performance, affordable scaling, and agile platforms for compute and storage I/O intensive applications. ScaleFlux is a well-funded startup and has leaders with proven experience across deployment of complex computing and solid-state storage solutions at scale

Computational Storage Drives are integrated into x86/Linux server and storage environments via an easy-to-install ScaleFlux software module. Host-based Flash Translation Layer and Flash Management technologies support consistent latency and performance characteristics. CSD Compute Engines are accessible to applications through APIs exposed by the ScaleFlux software module.

By simultaneously solving compute and storage I/O bottlenecks, CSD technology provides significant and proven run-time improvements to compute and data intensive applications.

ScaleFlux Computational Storage is the ideal foundation for highly scalable, reliable, and low-latency database infrastructure [3].

With data-path compression and decompression that is directly integrated with Flash storage, ScaleFlux delivers the most consistent transactional performance with the smallest Flash storage capacity footprint.

Ceph (16.2.7/ Pacific) [4] is an open-source storage platform that implements object storage on a single distributed computer cluster and provides interfaces for object, block and file-level storage. Ceph aims primarily for completely distributed operation without a single point of failure. Ceph storage manages data replication and is generally quite fault tolerant. As a result of its design, the system is both self-healing and self-managing. Hyperscalers developed this appliance with an all flash NVMe Ceph storage cluster using CSD (Computational Storage Drive) technology from Scaleflux [5] in QuantaGrid D53X-1U (S5X) servers from Hyperscalers [6].

Audience and Purpose

Engineers, Enthusiasts, Executives and IT professionals with a background in Computer Science/ Electronics/ Information Technology and with an understanding of Linux commands, Python language and basic electronics who intend to study, explore, deploy Ceph (16.2.7) cluster with CSD in Ubuntu 20.04.

The purpose of this document is to create a Ceph (16.2.7) cluster with CSD as object storage drives installed within QuantaGrid D53X-1U servers running the Ubuntu 20.04 operating system.

Digital IP Appliance Design Process

Hyperscalers has developed a Digital- IP-Appliance Design Process and an Appliance Optimizer Utility which we use in conjunction with each other to productise IT-appliances for Digital-IP owners needing to hyperscale their services quickly, reliably and at a fraction of traditional costs.

Appliance Optimizer Utility AOU

The Appliance Optimizer Utility (AOU) automates the discovery of appliance bottlenecks by pinging all layers in the proposed solution stack. A live dashboard unifies all key performance characteristics to provide a head-to-head performance assessment between all data-path layers in the appliance, and also as a comparison between holistic appliances.

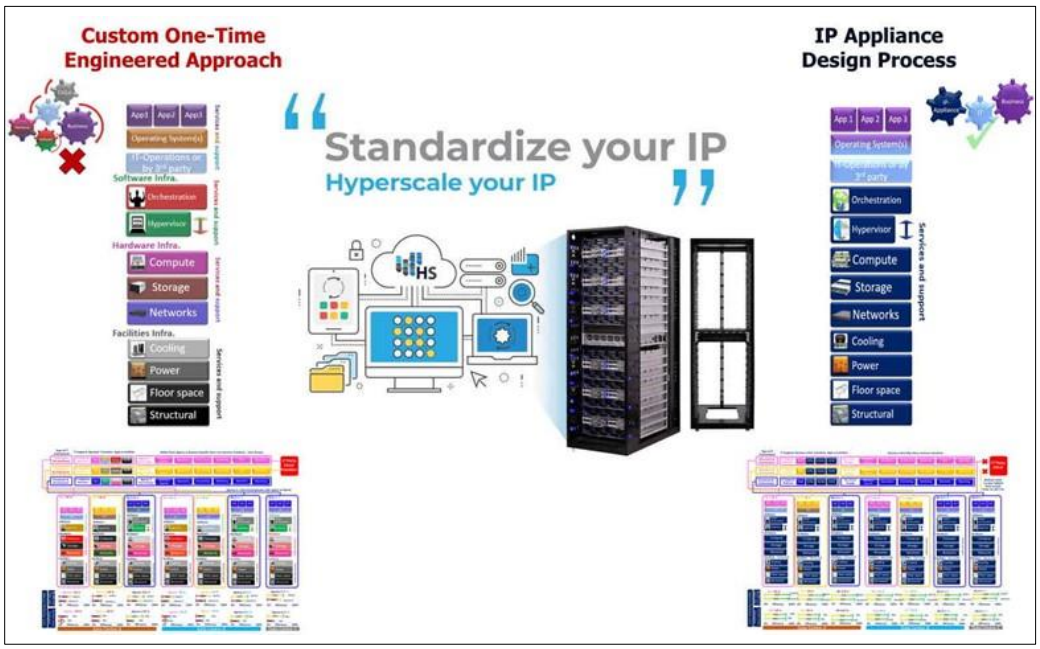


Figure 2 Digital IP-Appliance Design Process

Important Considerations

This appliance documentation is qualified and valid only for this hardware (11) and software (15) configuration.

Infrastructure Setup

The following figure shows the final appliance architecture that will be built upon completion of the configuration steps contained within this document. The requirements of this appliance are mentioned at (12)

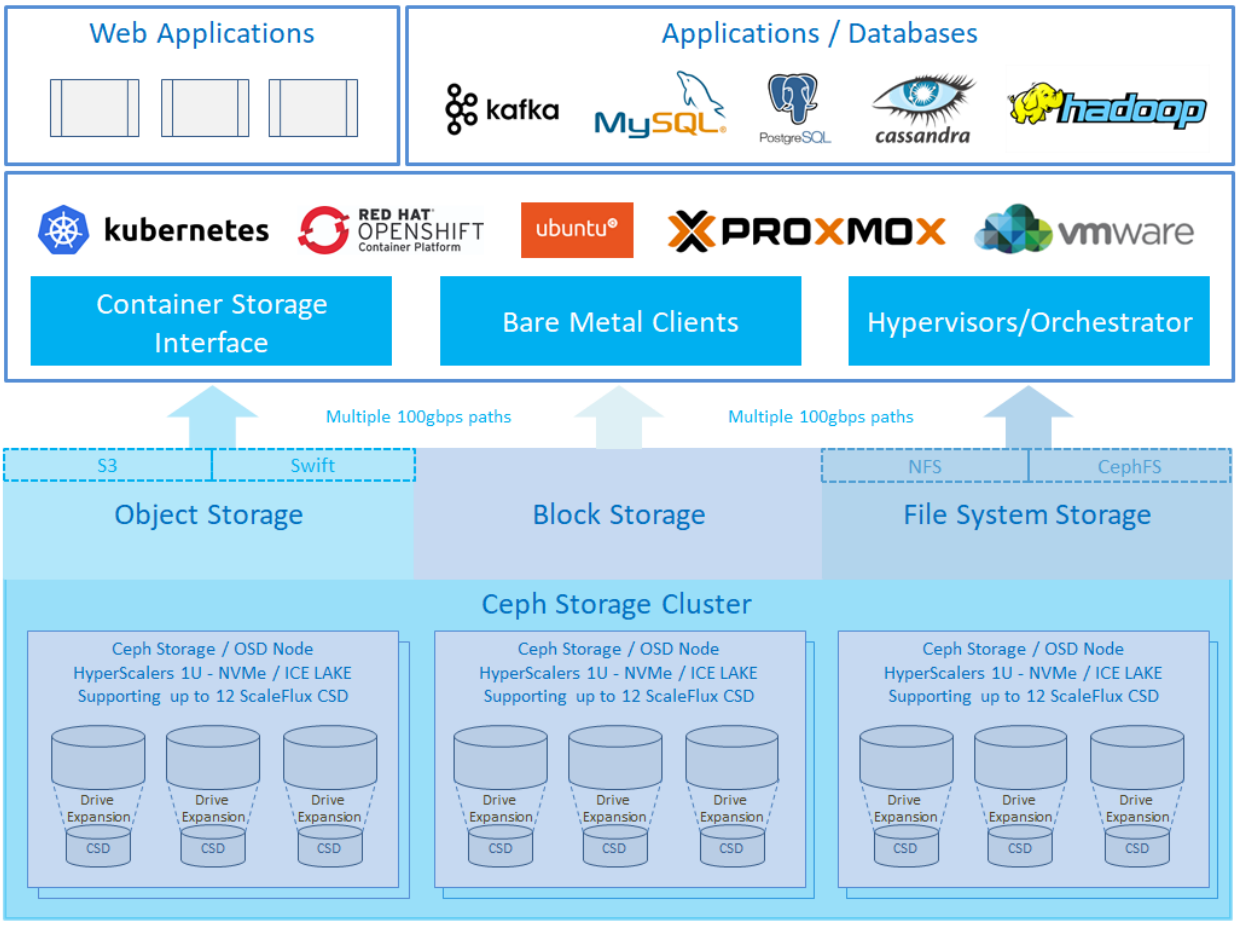


Figure 3 HyperFlow Software Defined Computational Storage Solution generic structure

Access and Default Credentials

The following credentials can change without notice upon network reset, maintenance, or any other circumstances. Please contact Hyperscalers for updated credentials to this appliance.

Ssh to `_admin` node – `root@192.168.18.151 /` Contact Hyperscalers

Ceph Dashboard - `https://192.168.18.151:8443/`

Credentials – Contact Hyperscalers

Ceph object gateway - `https://192.168.18.151:443`

Access key - Contact Hyperscalers

Secret key – Contact Hyperscalers

2 BASE PRODUCT DEPLOYMENT

Ceph (16.2.7/ Pacific) [4] is an open-source storage platform that implements object storage on a single distributed compute cluster and provides interfaces for object, block and file-level storage. Ceph aims primarily for completely distributed operation without a single point of failure. Ceph storage manages data replication and is generally quite fault tolerant. As a result of its design, the system is both self-healing and self-managing. In this appliance, Hyperscalers deployed an all flash NVMe ceph storage cluster with CSD 2000 drives (Computational Storage Drive) from Scaleflux [5] in QuantaGrid D53X-1U (S5X) server from Hyperscalers [6].

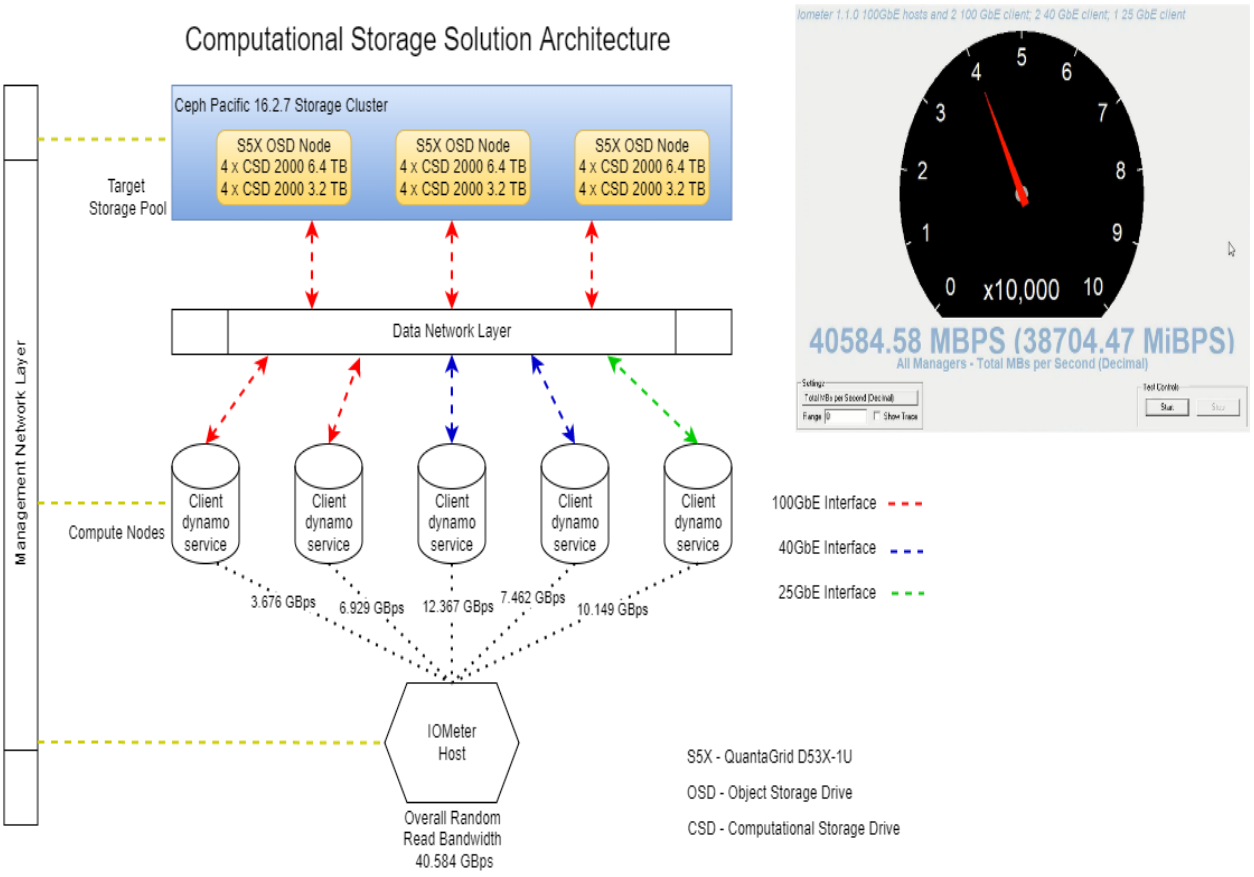


Figure 4 Computational Storage Solution Architecture

Introduction to CSD 2000

Scaleflux offers a CSD 2000 [5] storage drive with field programmable gate array to change the effective capacity which is ideally suited for this Ceph storage appliance. CSD 2000 performs with high throughput suited for hot storage and more effective capacity suited for cold storage applications using Ceph.

- **Form Factor** - PCIe AIC & U.2 Drive
- **Flash Capacity** - Up to 16TB Effective Capacity with data path compression (8TB raw)
- **Interface** - PCIe Gen3 x4
- **Compute Engines** - GZIP Compression / Decompression Customizable Database Engine Accelerator
- **Compute Capability** - Transparent Datapath Compression, Accelerated Performance Extended Capacity, Adjustable drive settings
- **Software Compatibility** - Linux OS 2.6 Kernel or later Only
- **Repository Support -** Ubuntu 16/18/20, RedHat/CentOS 6/7/8

Terminologies of a Ceph cluster

There are three services that form the backbone of the cluster [7]

- **ceph monitors** (`ceph-mon`) maintain maps of the cluster state and are also responsible for managing authentication between daemons and clients
- **managers** (`ceph-mgr`) are responsible for keeping track of runtime metrics and the current state of the Ceph cluster
- **object storage daemons** (`ceph-osd`) store data, handle data replication, recovery, rebalancing, and provide some ceph monitoring information.

Additionally, we can add further parts to the cluster to support different storage solutions

- **metadata servers** (`ceph-mds`) store metadata on behalf of the Ceph Filesystem
- **rados gateway** (`ceph-rgw`) is a Hypertext Transfer Protocol server for interacting with a Ceph Storage Cluster that provides interfaces compatible with OpenStack Swift and Amazon S3.

There are multiple ways of deploying these services. In this document, we will be deploying using the `cephadm` orchestrator [8].

Hardware Configuration

Server	Number of nodes	CPU	RAM	NIC Mezz	Storage card	PCIe NIC	Object Storage Drives	OS
QuantaGrid D53X-	3	Intel Xeon	64/2933x 4 units	Connect X	Null	Connect X 100 G	CSD	Ubuntu

1U (S5X) [6]		4310 T x 2		10/25G		(active)	2000 (6.4 TB) (U.2) x 4. CSD 2000 (3.2TB) (U.2) x 4 [5]	20.04 (4.8.0-43-generic)

Deployment

Hardware Deployment

There are a few hardware requirements for CSD 2000s and Ceph that needs to be considered while deploying Ceph/ Scaleflux appliance.

CSD 2000 requirements

CSD 2000 comes in add-in card and U.2 form factors. For add-in cards, a physical x8 CEM slot- connector is required. The slot must support PCIe Gen3 or above. The 2.5" U.2 form factors support SAS/SATA or PCIe using the same SFF-8639 connector, but not at the same time. Because the connector is the same, a 2.5" U.2 drive will mechanically fit in the slot no matter which interface is present. Therefore, it is critical to verify that the U.2 drive bay is wired for PCIe and not SAS/SATA. Furthermore, the U.2 slots must not be attached to a storage controller (e.g., a Broadcom Mega RAID device) that prevents the host operating system from accessing PCIe devices directly. PCIe switches or re-timers do not pose any issues.

CSD 2000 uses an "open channel" style interface that requires the installation of a driver. The driver is based on the NVMe driver, with additional logic added for Flash management. Because the driver caches the Flash translation layer in host memory, it will occupy a portion of DRAM. The following formula calculates the amount of host memory needed to install the driver. If there is insufficient memory, the driver will not be loaded.

$$\text{Logical Capacity (GB)} \times 0.2\% + 3.5\text{GB} = \text{Required System Memory per Drive}$$

For example, for a 3.2TB CSD 2000: $3200\text{ GB} \times 0.2\% + 3.5\text{ GB} = 9.9\text{ GB}$

When there are multiple drives installed in the system, multiply the number of drives by the memory required for a single drive to get the total amount of memory required.

For example, if there are 12 3.2TB CSD 2000 drives installed in the system.

9.9 GB x 12 ~= 120 GB

Ceph requirements

The following are a guideline to choose hardware for Ceph Pacific (16.2.7) installation.

Process	Criteria	Minimum Recommended
`ceph-osd`	Processor	• 1 core minimum • 1 core per 200-500 MB/s • 1 core per 1000-3000 IOPS • Results are before replication. • Results may vary with different CPU models and Ceph features. (erasure coding, compression, etc) • ARM processors specifically may require additional cores. • Actual performance depends on many factors including drives, net, and client throughput and latency. Benchmarking is highly recommended.
	RAM	• 4GB+ per daemon (more is better) • 2-4GB often functions (may be slow) • Less than 2GB not recommended
	Volume Storage	1x storage drive per daemon
	DB/WAL	1x SSD partition per daemon (optional)
	Network	1x 1GbE+ NICs (10GbE+ recommended)
`ceph-mon`	Processor	• 2 cores minimum
	RAM	2-4GB+ per daemon
	Disk Space	60 GB per daemon
	Network	1x 1GbE+ NICs
`ceph-mds`	Processor	• 2 cores minimum
	RAM	2GB+ per daemon
	Disk Space	1 MB per daemon

Process	Criteria	Minimum Recommended
	Network	1x 1GbE+ NICs

Installation of Software components

We will deploy the Ceph/ Scaleflux appliance in a freshly installed Ubuntu 20.04 in QuantaGrid D53X-1U. In summary, the appliance software deployment will involve

- Installation of operating system
- Installing drivers for CSD 2000.
- Preparing the CSD 2000 to be used as an Object Storage Drive in Ceph/ Scaleflux appliance.
- Installing the prerequisites for Ceph Pacific (16.2.7)
- Installation of Ceph

By the end of this document, we'll have these key services/ daemons holding this appliance together (**Error! Reference source not found.**).

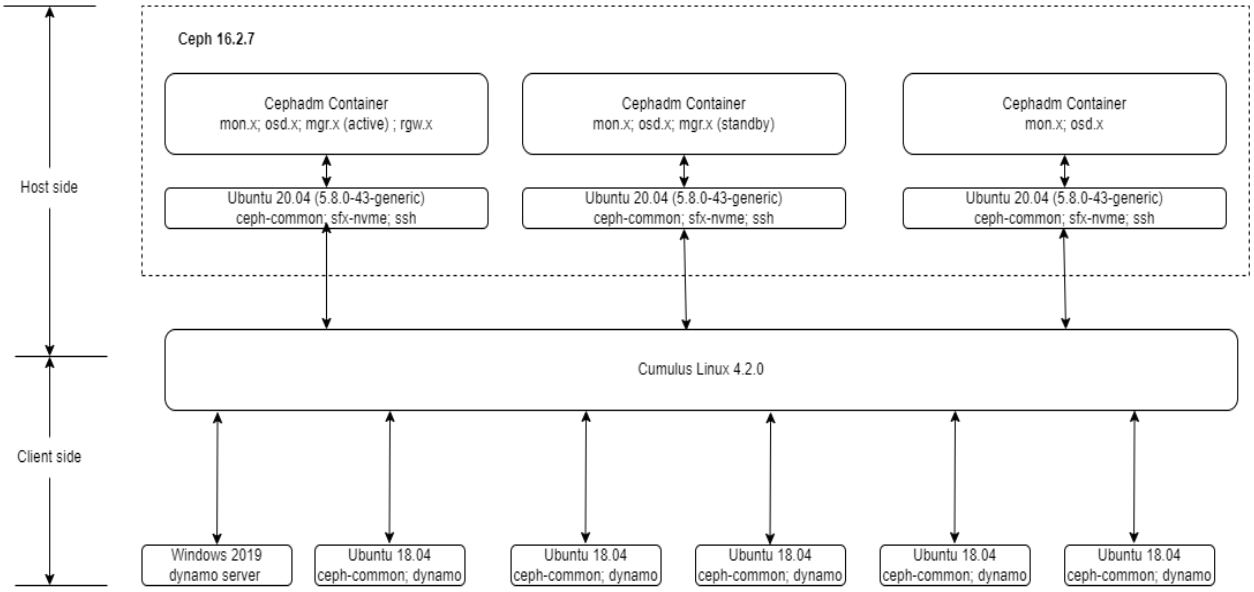


Figure 5 Key services/ daemons in Ceph/ Scaleflux appliance

Installation of operating system

We will begin by installing all the nodes (minimum of 3) with Ubuntu 20.04 [9]. While installing Ubuntu 20.04, ensure that "Download updates while installing Ubuntu" option is *unchecked* to avoid updating to unsupported kernel for CSD 2000s [5].

Installing drivers for CSD 2000

While the server restarts after installation of the operating system, get into BIOS and set the CPU to performance mode at Socket Configuration -> Pwr and Perf Profile -> High Performance. (Might change depending on the hardware manufacturer) (**Error! Reference source not found.**)

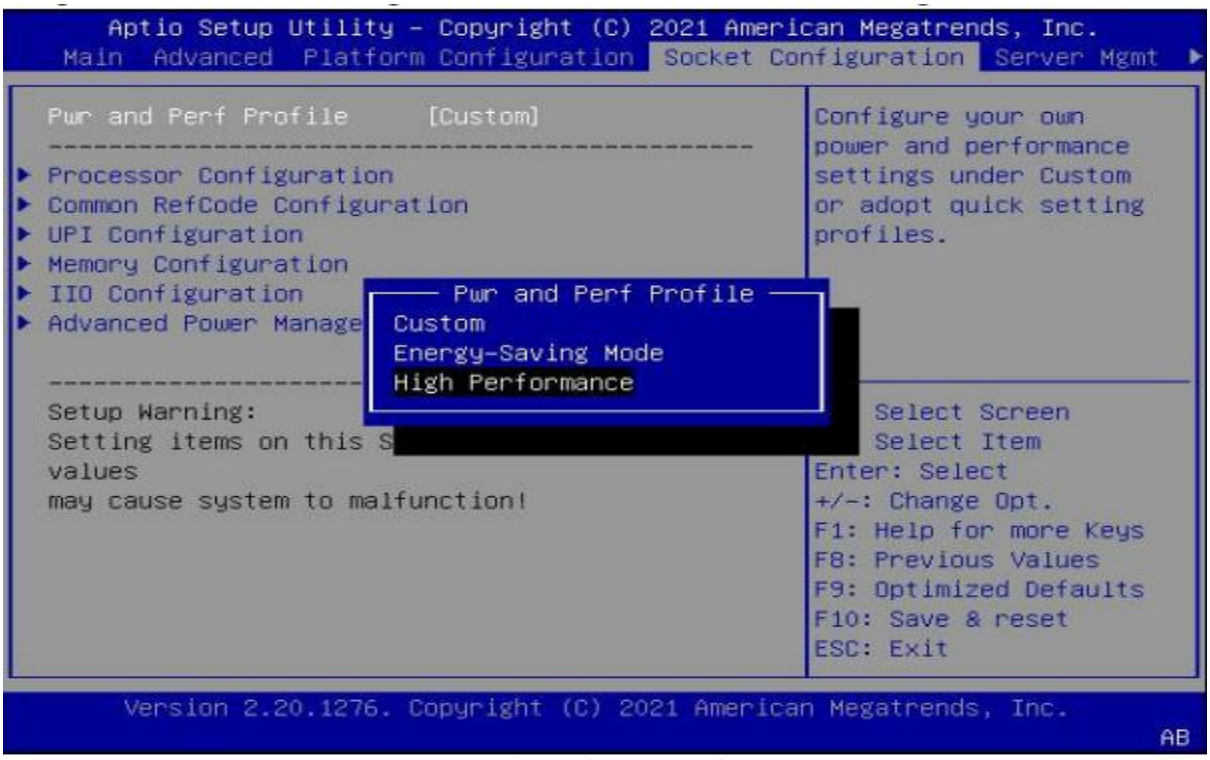


Figure 6 BIOS CPU Performance profile

With the fresh install of Ubuntu 20.04, open an elevated terminal and execute the following commands to install the drivers for CSD 2000. Lines 4-9 stops the CPU from entering idle state. Lines 10-13 installs the necessary drivers for CSD 2000, and Line 14 helps in verification of the drives attached to the server with the installed drivers. In this appliance, we'll be using all the CSD 2000 (3.2 TB and 6.4 TB) drives at 11.2 TB effective capacity. In our scenario, CSD 2000 3.2TB is at balanced performance and CSD 2000 6.4 TB is at maximum performance.

```
1.  #Preparing CSD2000s for use in Ceph
2.
3.  apt update
4.  uname -r
5.  apt-mark hold 5.8.0-43-generic # takes output from earlier command. Ensure scaleflux drivers
    exist for the kernel at https://packagecloud.io/scaleflux/sfx3x
6.  nano /etc/default/grub
7.  # Edit the grub with GRUB_CMDLINE_LINUX="crashkernel=auto rhgb quiet intel_idle.max_cstate=0
    processor.max_cstate=0 idle=poll"
8.  update-grub
```

```
9.  reboot
10. apt install curl
11. curl -s https://packagecloud.io/install/repositories/scaleflux/sfx3x/script.deb.sh | sudo
    bash # Works only for Debian based OS
12. apt search sfx3xdriver-src
13. sudo apt install sfx3xdriver-src
14. sfx-status
15. sfx-nvme sfx set-feature -f 0xdc /dev/sfxv[x] # Format CSD 2000
16. sfx-nvme sfx change-cap /dev/sfdv[x]n1 -c 11200 # change capacity of CSD 2000
```

Preparing CSD 2000 for use as Object Storage Drive

In order to utilize the CSD 2000 as object storage drive with Ceph Pacific, we need to precondition them with FIO before adding them to the cluster to get a stable performance.

There are two separate commands for sequential and random preconditioning of the drives. It is advisable to precondition the drives depending on the testing methodologies [10].

```
1.  # Preconditioning of CSD 2000
2.  apt install fio
3.  # Sequential preconditioning
4.
5.  fio --ioengine=libaio --direct=1 --group_reporting --name=baseline --thread --stonewall --
    new_group --fill_device=1 --rw=write --rwmixread=0 --bs=128k --numjobs=1 --iodepth=128 --
    loops=2 --buffer_compress_percentage=80 --refill_buffers --filename /dev/sfdv[X]n1
6.
7.  # random pre conditioning
8.
9.  fio --ioengine=libaio --direct=1 --group_reporting --name=baseline --thread --stonewall --
    new_group --fill_device=1 --rw=randrw --rwmixread=0 --bs=128k --numjobs=4 --iodepth=128 --
    loops=1 --buffer_compress_percentage=80 --refill_buffers --filename /dev/sfdv[X]n1
```

Installation of prerequisites for Ceph

In this appliance, we'll be using 3 nodes (QuantaGrid D53X-1U) to cluster and create the storage appliance. In the case of object storage drives, you can run multiple of them on the same host but using the same storage drive for multiple instances is a bad idea as the disk's Input/Output speed might limit the object storage drive daemons' performance.

Before you deploy Ceph, firewall settings or other resources have to be adjusted to open these ports

- 22 for secure shell
- 6789 for monitors
- 6800:7300 for object storage drives, managers, and metadata servers
- 8080 for dashboard
- 7480/80/443(with SSL) for rados object gateway

The following are the requirements within the operating system (Ubuntu 20.04) needed for deployment of Ceph storage cluster in every node [11]

- Python 3
- Systemd
- Podman or Docker for running containers [12]
- Time synchronization (such as chrony or network time protocol)
- Logical Volume Manager 2 for provisioning storage drives

For Ceph, network time protocol (line 12) helps in synchronizing the clustered nodes. It is *preferable* to use the clients and nodes as a root user. Ceph also relies on seamless secure shell connection to communicate and hold the cluster together, so we're creating private-public key pair and placing it on every host that are to be clustered to have password-less access between them [13]. In this deployment method (cephadm orchestrator), the first node of the cluster is considered as admin node. We install lvm2 package (line 19) as object storage drives are created using it.

```
1.  #Ceph Pre-requisites Install
2.  apt install ntp
3.  apt install net-tools
4.  apt-get install     ca-certificates     gnupg     lsb-release
5.  echo   "deb [arch=$(dpkg --print-architecture) signed-by=/usr/share/keyrings/docker-archive-
    keyring.gpg] https://download.docker.com/linux/ubuntu \ $(lsb_release -cs) stable" | sudo
    tee /etc/apt/sources.list.d/docker.list > /dev/null
6.  apt-get install docker-ce docker-ce-cli
7.  apt-get update
8.  apt-get install docker-ce docker-ce-cli containerd.io
9.  apt install openssh-server
10. nano /etc/ssh/sshd_config
11. # Edit the ssh config with PermitRootLogin yes
12. passwd # set/change root password for ssh access
13. ssh-keygen # Generates public-private key pair
14. nano /etc/hosts
15. # Add the hosts and their corrsponding ip address. Ensure hostname matches the actual
    hostname.
16. ssh-copy-id <host-name>
17. #This allows passwordless ssh access
18. apt install lvm2
```

Installation of Ceph

In this appliance, we'll follow curl-based installation of Ceph [11],

1. Open an elevated terminal
2. Pull cephadm file from the repository (line 4).
3. Upon pulling the cephadm file, make it as executable (line 5)
4. Add the release repo (line 7) (Example., Pacific) that is to be installed to the update repositories of Ubuntu.
5. Install cephadm (line 8)
6. Bootstrap the ceph with passing the monitor ip

The `bootstrap` command (line 9) will [8]:

- Create a monitor and manager daemon for the new cluster on the local host.
- Generate a new SSH key for the Ceph cluster and add it to the root user's `/root/.ssh/authorized_keys` file.
- Write a copy of the public key to `/etc/ceph/ceph.pub`.
- Write a minimal configuration file to `/etc/ceph/ceph.conf`. This file is needed to communicate with the new cluster.
- Write a copy of the `client.admin` administrative (privileged!) secret key to `/etc/ceph/ceph.client.admin.keyring`.
- Add the `_admin` label to the bootstrap host. By default, any host with this label will (also) get a copy of `/etc/ceph/ceph.conf` and `/etc/ceph/ceph.client.admin.keyring`.

7. Upon bootstrapping the cluster, one will be able to access the dashboard (with SSL) with the monitor passed on earlier at `https://monitor-ip:8443/`.
8. Installing `ceph-common` will allow us to access the cluster from outside the container.
9. The `ceph.pub` will need to be copied to all the nodes (in this case, three nodes) to hold the ceph cluster together.
10. `ceph-common` needs to be installed and `ceph.conf`, `ceph.client.admin.keyring` needs to be copied to `/etc/ceph` location at every node that are to clustered in order to view the cluster details in any given node.
11. Given that the nodes that are to be clustered have the pre-requisites satisfied and share a common SSH public key, one can add the host to the cluster through `ceph orch host add <host-name>` from `_admin` node.

```
1.  #Ceph Installation
2.
3.  #Navigate to any location of interest where you want the "cephadm" file to be placed
4.  curl --silent --remote-name --location https://github.com/ceph/ceph/raw/<release-
    name>/src/cephadm/cephadm
5.  chmod +x cephadm
6.
7.  # For help and available options use "./cephadm --help"
8.  ./cephadm add-repo --release <release-name>
9.  ./cephadm install
10. cephadm bootstrap --mon-ip <monitor-ip>
11. # creates a minimal ceph cluster with 1 monitor node and 1 manager node with dashboard url
    (with SSL) and its access credentials are presented as output
12.
13. cephadm install ceph-common # helps in accessing cluster details outside the "cephadm"
    container
14. ssh-copy-id -f -i /etc/ceph/ceph.pub <host-name>
15. ./cephadm prepare-host <host-name>
```

```
16. # checks the host for necessary pre-requisites
17. ceph orch host add <host-name>
18. # adds node to the cluster
19.  cephadm shell # To access the container shell
```

By default (in this method of installation) available Object Storage Drives (OSD) are picked up by the cluster and added as OSDs to the cluster through the service named `osd.all-available-devices`. To disable this behaviour, execute the following command in every node

```
ceph orch apply osd --all-available-devices unmanaged = true # Stops adding OSD automatically into the cluster in any given node
```

Upon adding all the nodes, with enough monitors and standby manager, a sample `ceph status` output, sample `ceph.conf` file and the dashboard (**Error! Reference source not found. Error! Reference source not found.**) of our appliance is shown.

```
root@cephnvme-QuantaGrid-D53X-1U-1S5X2000079:~# ceph status
  cluster:
    id:     12fde18a-bad5-11ec-80ac-2f401ebdd182
    health: HEALTH_OK
  services:
    mon: 3 daemons, quorum cephnvme-QuantaGrid-D53X-1U-1S5X2000079,cephnvmetwo-QuantaGrid-D53X-1U-1S5X2000079,cephnvemethree-QuantaGrid-D53X-1U-1S5X2000079 (age 7d)
    mgr: cephnvme-QuantaGrid-D53X-1U-1S5X2000079.iytztt(active, since 9d), standbys: cephnvmetwo-QuantaGrid-D53X-1U-1S5X2000079.hjrrsf
    osd: 24 osds: 24 up (since 27h), 24 in (since 9d)
  data:
    pools:   2 pools, 33 pgs
    objects: 797 objects, 1.0 GiB
    usage:   24 GiB used, 244 TiB / 244 TiB avail
    pgs:     33 active+clean
```

```
#Sample details of ceph.conf file
# minimal ceph.conf for 12fde18a-bad5-11ec-80ac-2f401ebdd182
[global]
        fsid = 12fde18a-bad5-11ec-80ac-2f401ebdd182
        mon_host = [v2:192.168.18.178:3300/0,v1:192.168.18.178:6789/0]
[v2:192.168.18.151:3300/0,v1:192.168.18.151:6789/0]
[v2:192.168.18.180:3300/0,v1:192.168.18.180:6789/0]
```

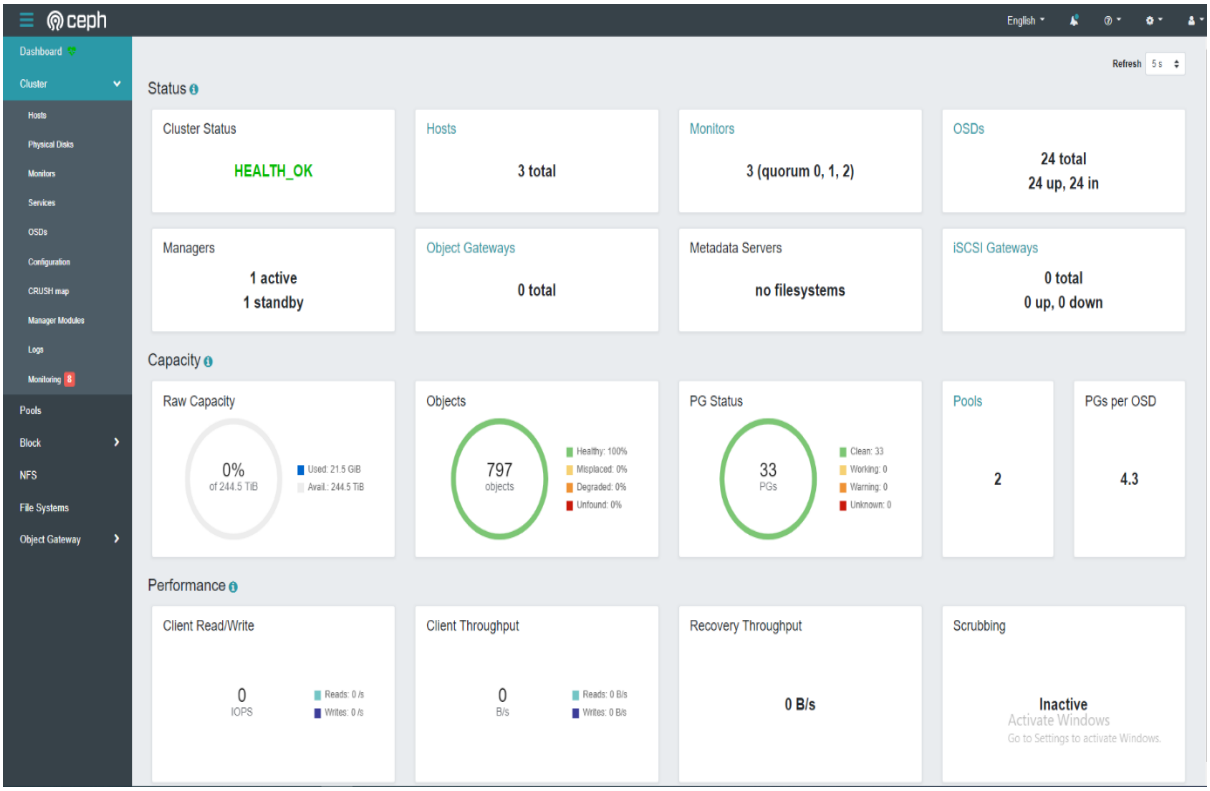


Figure 7 Ceph Dashboard

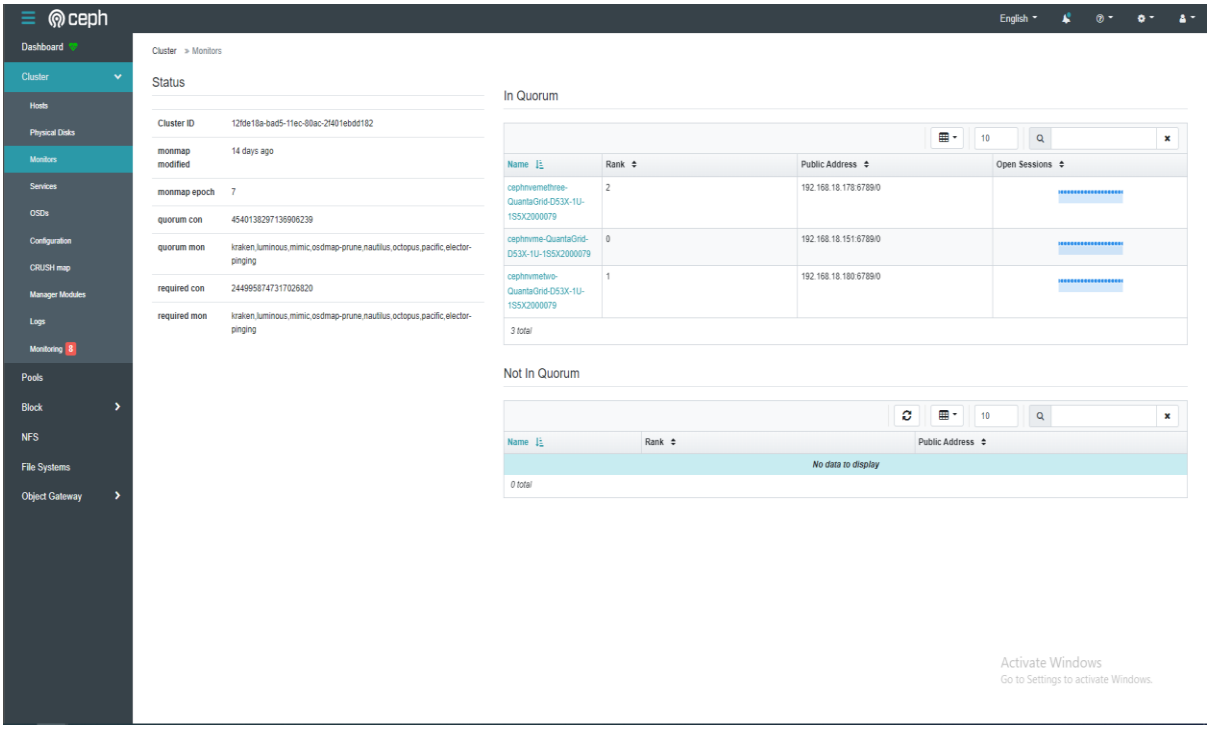


Figure 8 Monitors in Ceph cluster

3 CONFIGURE THE APPLIANCE

Ceph (16.2.7) offers three types of storage to its users, namely, object (Ideal for application development), block (Ideal for host/ VM), and file system storage (Ideal for client). In this document, we'll cover configuration of object and block storage features and testing of block, object storage.

Rados block device

There are *two ways* to create a pool in an existing Ceph cluster. One through dashboard and other through command line interface.

In the dashboard,

1. Select pools from the left panel and select create (**Error! Reference source not found.**)
2. Fill out the form with name, pool type, replication size.
3. Ensure that application is selected as rbd to create the pool with block device functionality and select "Create Pool" (**Error! Reference source not found.**).

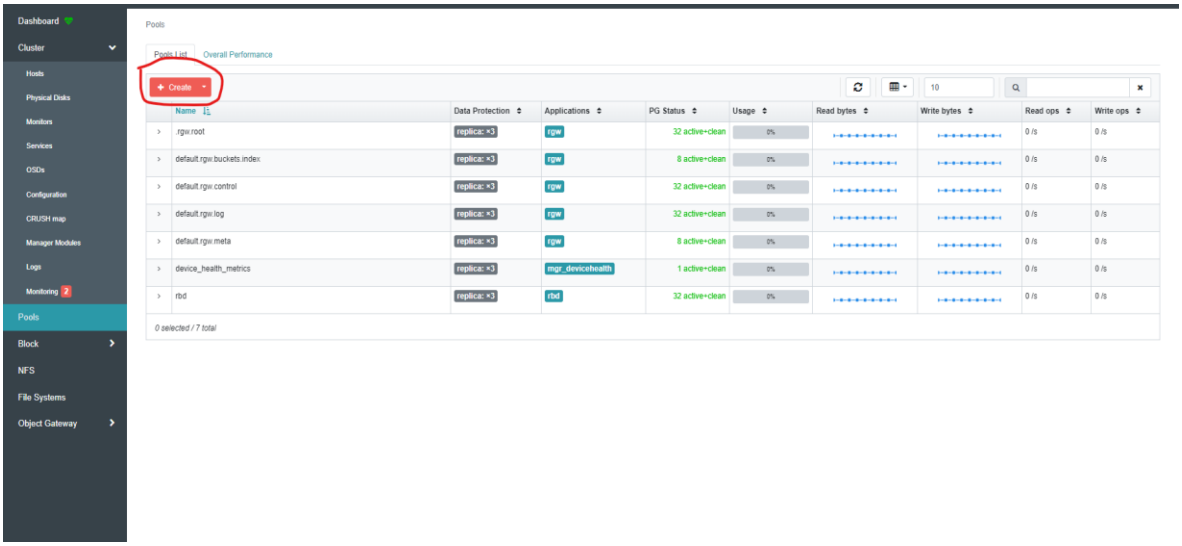


Figure 9 Interface to view / create pools

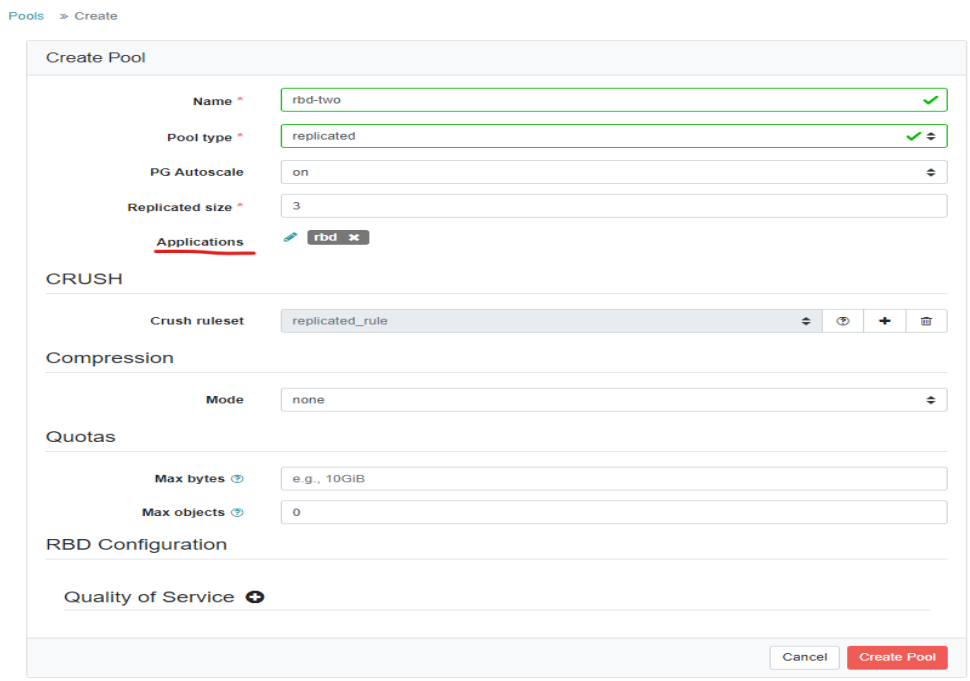


Figure 10 "Create Pool" form

To create an image that is to be mapped to the client,

1. Select Block -> Images from the left panel and select Create to open a form (**Error! Reference source not found.**).
2. Fill out the form with Name, block device pool that it needs to be associated with, size of the image and select "Create RBD" (**Error! Reference source not found.**).

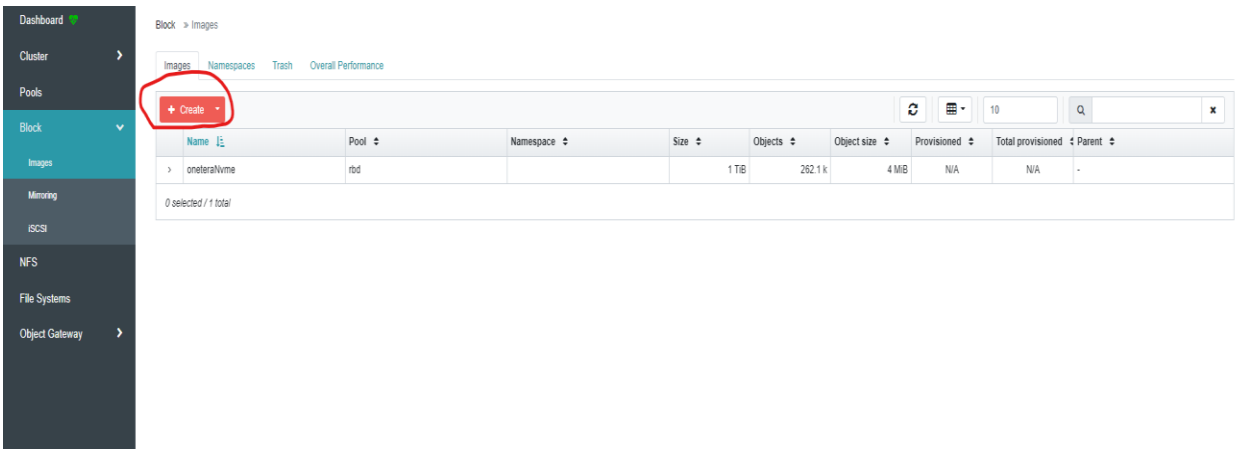


Figure 11 Interface to view/ create block image

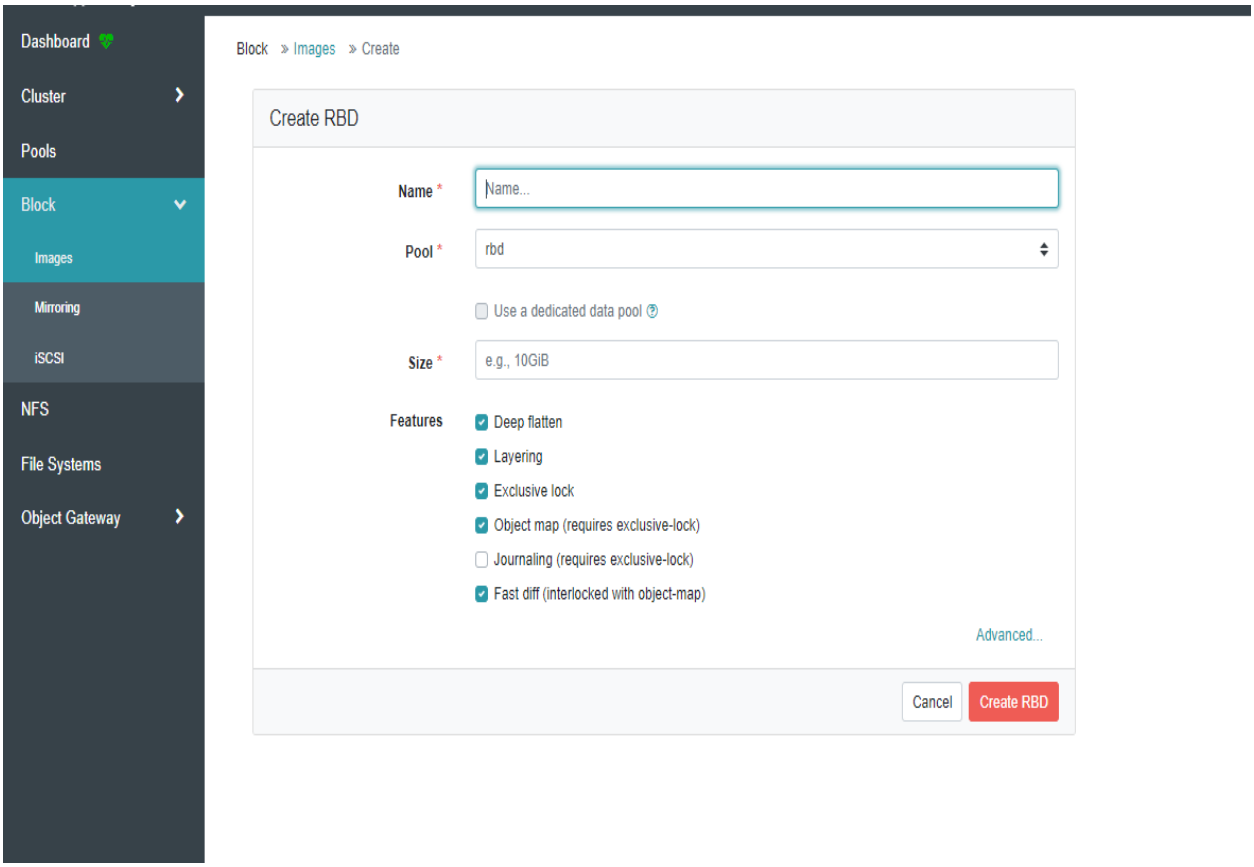


Figure 12 Create block image

In order to map the image of the block device to a client, one has to execute the following commands in the client's terminal. Please note one needs `ceph.conf` and `ceph.client.admin.keyring` to successfully map the block device image.

```
1. # In client node,
2. apt install ceph-common # Only if ceph-common was not installed earlier to the client
3. rbd map <pool-name> --name client.admin -m monitor-ip -k /path/to/ceph.client.admin.keyring
   -c /path/to/ceph.conf
4. mkfs.ext4 -m0 /dev/rbdX
```

To automatically map rados block device on boot,

```
1. # Automap block devices on boot. Ensure ceph.conf file to drives that are to be mapped is
   present at /etc/ceph/ceph.conf
2. nano /etc/ceph/rbdmap
3. pool-name/image-name       name=client.admin,keyring=/path/to/ceph.client.admin.keyring
4. systemctl enable rbdmap
5.
6. #To map /unmap devices
7. rbdmap map
8. rbdmap unmap
```

Through command line interface,

```
# To create a Rados Block Device(RBD)
# In Monitor node,
rbd pool init <pool-name>
# In client node,
apt install ceph-common # Only if ceph-common was not installed earlier
rbd create <pool-name> --size <pool-size> --image-feature layering -m mon-ip -k
/path/to/ceph.client.admin.keyring -c /path/to/ceph.conf
rbd map <pool-name> --name client.admin -m monitor-ip -k /path/to/ceph.client.admin.keyring
-c /path/to/ceph.conf
mkfs.ext4 -m0 /dev/rbdX
```

Object gateway

In order to create an object gateway [14] with SSL certificate, secure shell into one of the monitor nodes,

1. Create SSL certificate and key pair using openssl (line 4) (config in Addendum)
2. Concatenate certificate and key to a single file. (line 6-9)
3. To create the object gateway, execute `ceph orch apply rgw <gateway-name> --realm=<realm-name> --zone=<zone-name> --placement=<host-name>`
4. Upon execution of this command, object gateway will be deployed and start running at port 80 without SSL.
5. Wait for the object storage drive to rebalance with placement groups (PG) [15] [16] of object gateway.
6. Upon rebalancing, In the ceph dashboard (**Error! Reference source not found.**) select `Cluster -> Services -> rgw.<gateway-name> -> Edit`
7. In the pop-up window (**Error! Reference source not found.**), change the port to 443 and attach the concatenated ".pem" certificate file.
8. The service will restart and redeploy itself with the self-signed certificate.

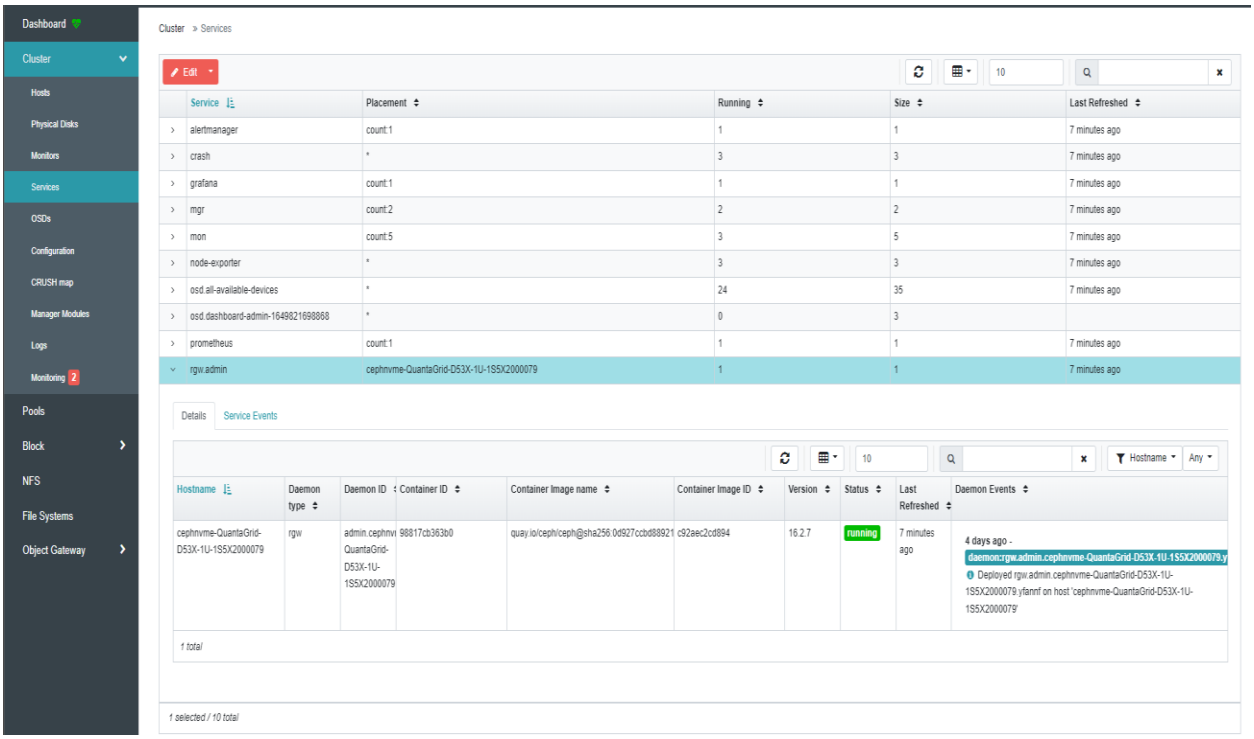


Figure 13 Edit rgw.<service-name>

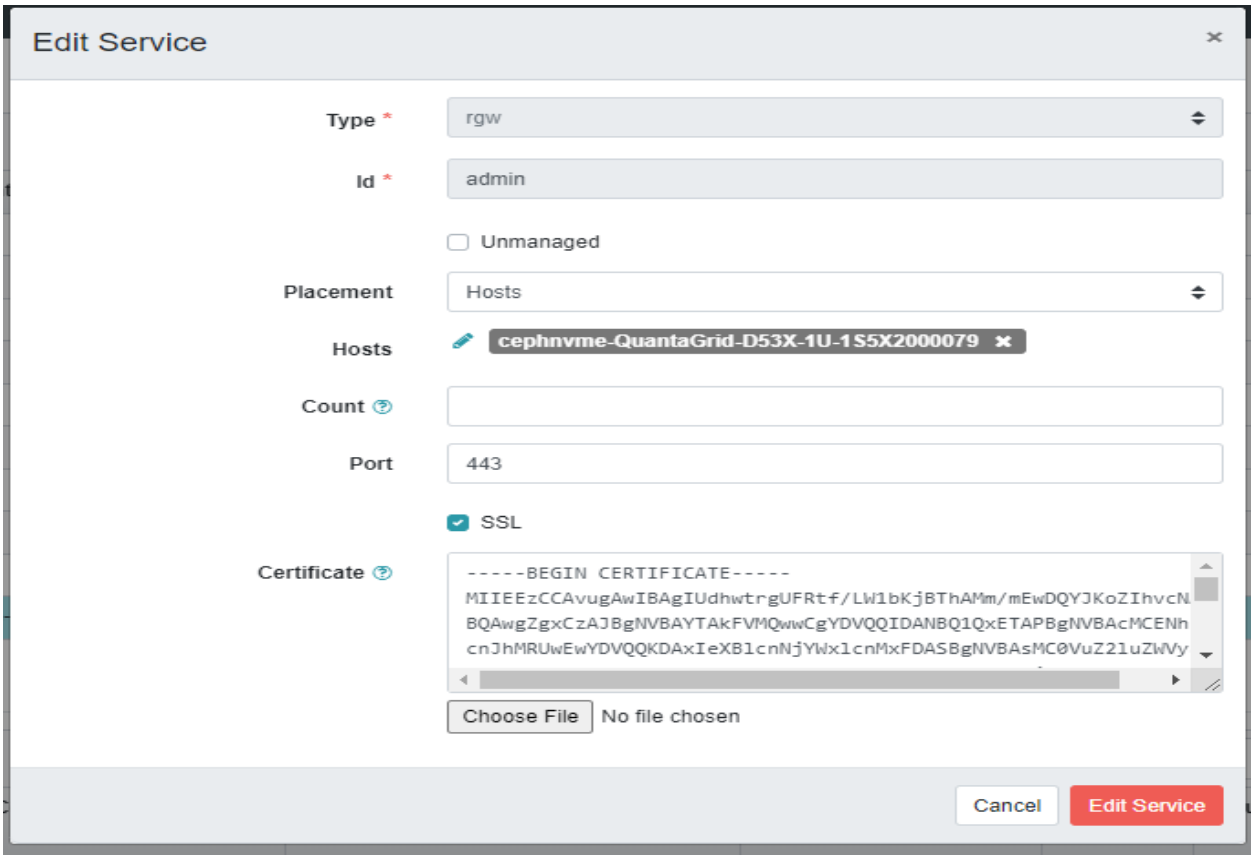


Figure 14 RGW Service edit pop-up

9. Set `ceph dashboard set-rgw-api-ssl-verify False` to view the object gateway daemon in the dashboard.
10. Verify `https://<placement-host-name-ip>:443` is reachable through curl and browser.

```
# To deploy object gateway with ssl

ssh <one-of-monitor-nodes>
openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout /etc/ssl/private/ceph-rgw-cert.key -out /etc/ssl/certs/ceph-rgw.crt # create a SSL certificate
# Navigate to any desired location
touch nvmeServer.pem
cat /etc/ssl/certs/ceph-rgw.crt >> /path/to/nvmeServer.pem
cat /etc/ssl/private/ceph-rgw-cert.key >> /path/to/nvmeServer.pem # concatenate key and certificate files
cat nvmeServer.pem # verify that key and certificate files are concatenated
ceph orch apply rgw admin --realm=default --zone=default --placement=<host-name>
# In Ceph Dashboard Cluster -> Services -> rgw.admin -> Edit
# Change port to 443 ; Tick the SSL  box ; Attach the nvmeServer.pem file
ceph dashboard set-rgw-api-ssl-verify False
curl -k https://<placement-host-name-ip>:443 # verify "anonymous" response from the ip
# Verify similar response from the browser
```

In order to create bucket for the object gateway,

1. In dashboard navigate to `Object gateway -> Buckets -> Create` (**Error! Reference source not found.**)
2. In the Create bucket panel, mention bucket name, owner and placement target to create the bucket. (**Error! Reference source not found.**)

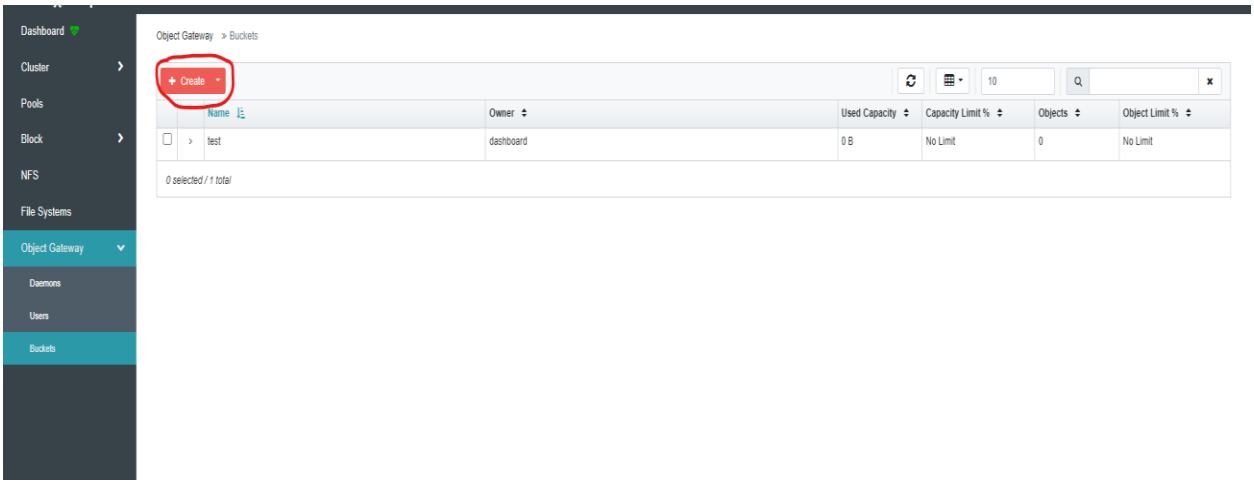


Figure 15 Interface to view/ create buckets

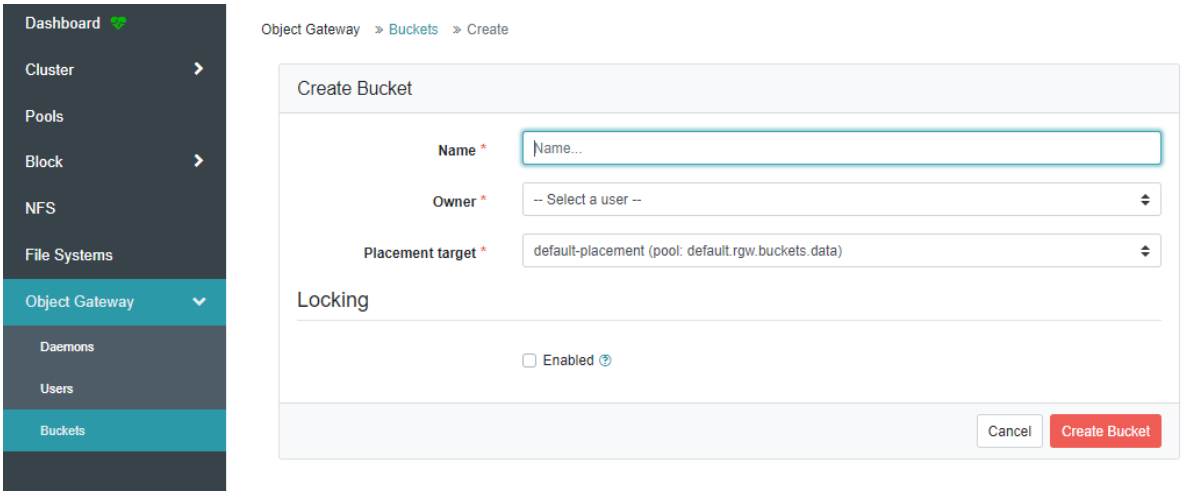


Figure 16 Create buckets

To access a specific bucket,

1. Execute `radosgw-admin user info --uid=<user-name>`, in any of the monitor nodes
2. Note down the access and secret key to the user, whom the bucket belongs to.
3. In postman, mention the ip `https://<placement-host-name-ip>:443/<bucket-name>`, secret key private key and type of bucket as S3. (**Error! Reference source not found.** and **Error! Reference source not found.**)

.

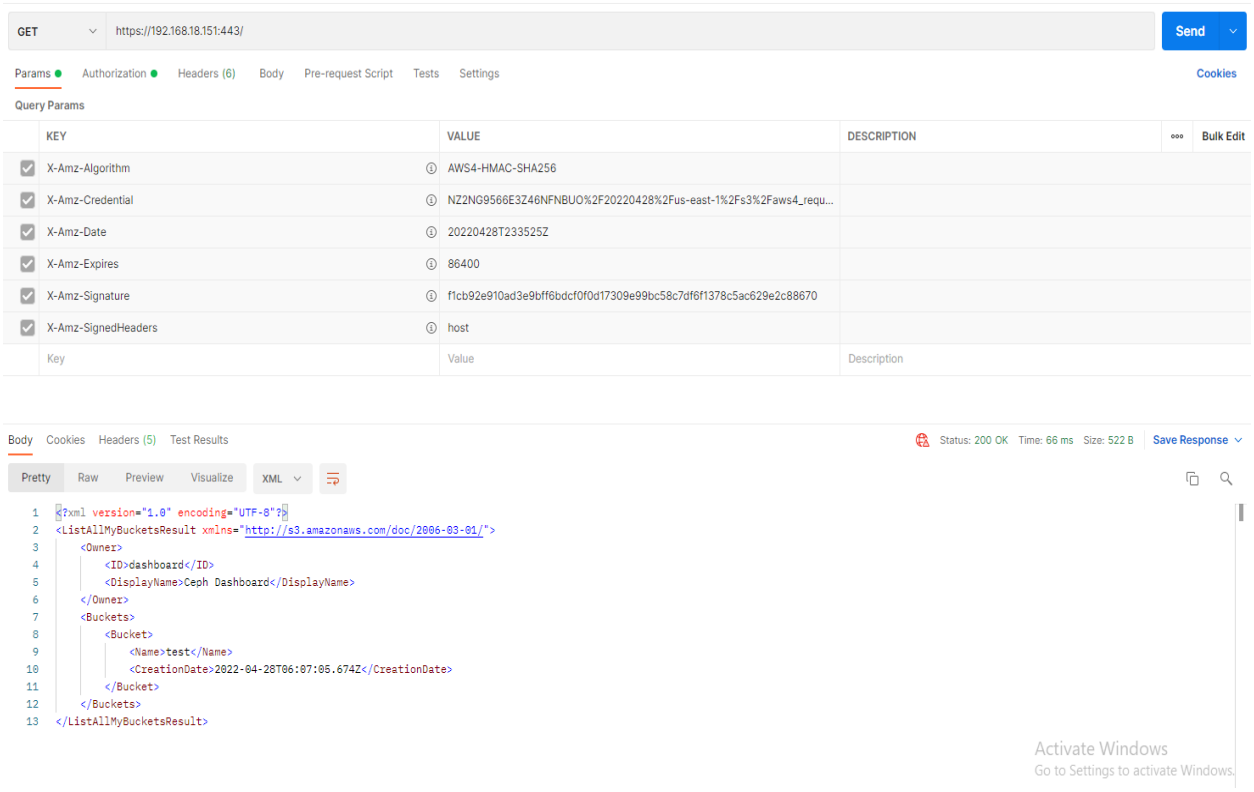


Figure 17 Generic object gateway response with access and secret keys

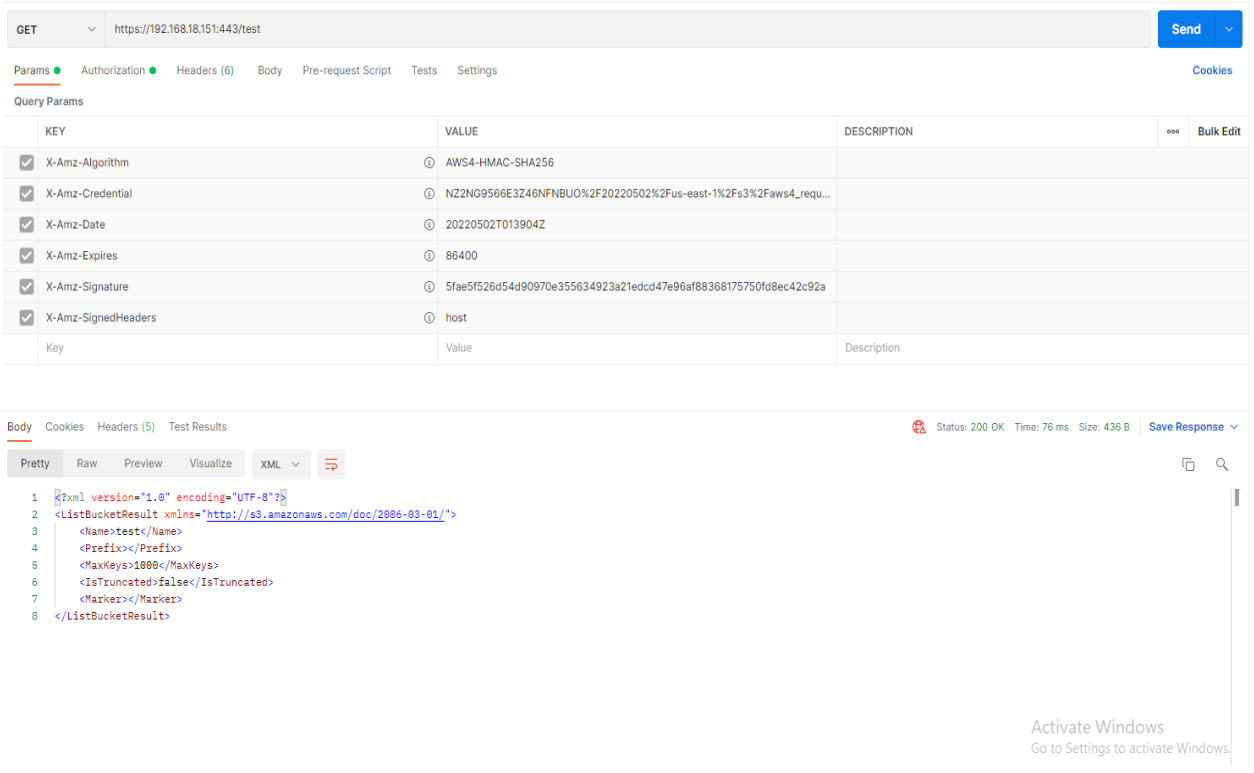


Figure 18 Accessing a specific bucket in the object gateway

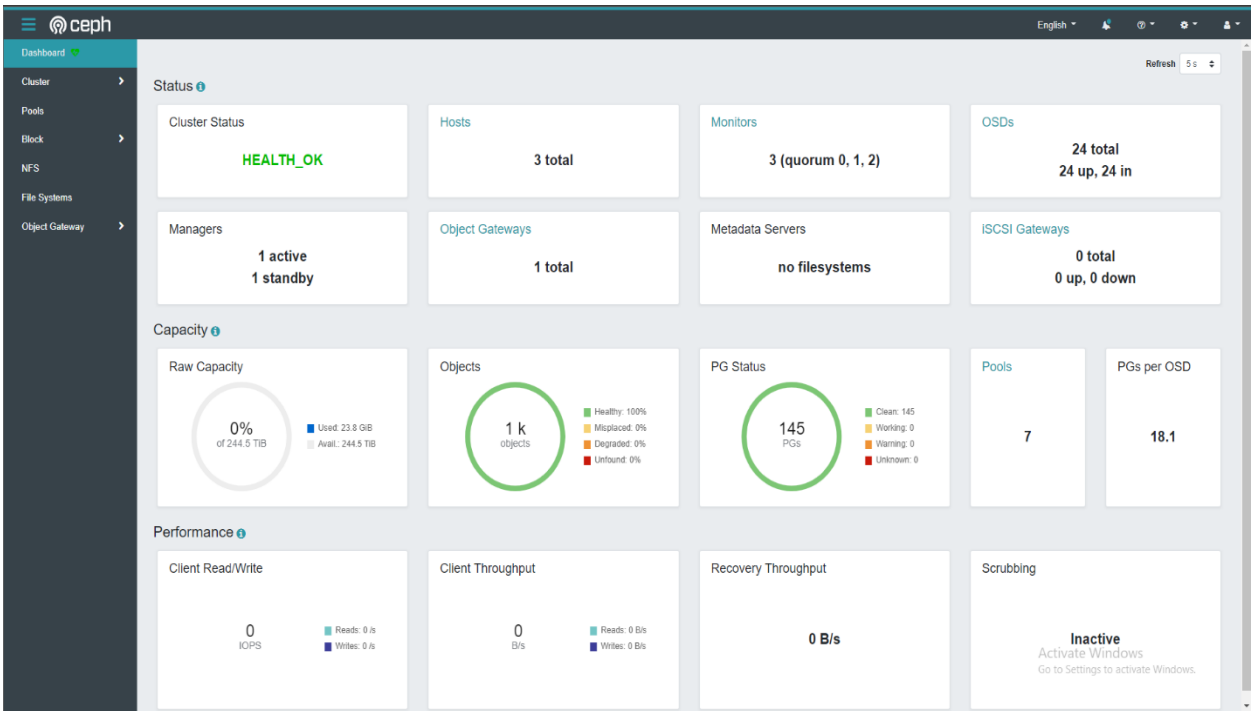


Figure 19 Ceph Dasboard after object gateway deployment

4 UPDATING THE APPLIANCE

Adding a host

To add a host to the cluster, execute the following commands (lines 1-8) in host to be added and lines 12-21 in the node with _admin tag

```
1.  apt install ntp
2.  apt install net-tools
3.  apt-get install     ca-certificates     gnupg     lsb-release
4.  echo   "deb [arch=$(dpkg --print-architecture) signed-by=/usr/share/keyrings/docker-archive-
    keyring.gpg] https://download.docker.com/linux/ubuntu \ $(lsb_release -cs) stable" | sudo
    tee /etc/apt/sources.list.d/docker.list > /dev/null
5.  apt-get install docker-ce docker-ce-cli
6.  apt-get update
7.  apt-get install docker-ce docker-ce-cli containerd.io
8.  apt install openssh-server
9.  nano /etc/ssh/sshd_config
10. # Edit the ssh config with PermitRootLogin yes
11. passwd # set/change root password for ssh access
12. nano /etc/hosts
13. # Add the hosts and their corrsponding ip address. Ensure hostname matches the actual
    hostname.
14. ssh-copy-id <host-name>
15. #This allows passwordless ssh access
16. apt install lvm2
```

```
17. ssh-copy-id -f -i /etc/ceph/ceph.pub <host-name>
18. ./cephadm prepare-host <host-name>
19. # checks the host for necessary pre-requisites
```

```
20. ceph orch host add <host-name>
21. # adds node to the cluster
```

Remove a host from the cluster

To remove a host from the cluster, execute the following commands (lines 2 -8) in the node that is to be removed and lines 9 and 11 in the node with _admin tag.

```
1.  # To remove a host from the cluster
2.  ssh <host-to-be-removed>
3.  systemctl stop <ceph-osd-service>
4.  ceph osd out osd.x
5.  ceph osd down osd.x
6.  ceph osd rm osd.x
7.  ceph osd crush rm osd.x
8.  ceph auth del osd.x
9.  ceph osd destroy x --yes-i-really-mean-it
10. Ctrl +D
11. ssh <_admin-tagged-node>
12. ceph orch host drain <host-name>
13. # Deactivates monitor and manager services, removes them and updates the monmap of the
    cluster
14. ceph orch host rm <host-name>
```

Adding OSD to the cluster

If the `osd.all-available-devices` service is running and a new drive is inserted into the node, the cluster will automatically add it as an object storage drive.

If the `osd.all-available-devices` service is not running, insert the drive to the node and execute the following commands in the node.

- `ceph osd create --data /dev/sdX node-name`

Remove OSD from the cluster

Execute the following commands in the node where the OSD is present,

```
systemctl stop <ceph-osd-service>
ceph osd out osd.x
ceph osd down osd.x
ceph osd rm osd.x
ceph osd crush rm osd.x
ceph auth del osd.x
ceph osd destroy x --yes-i-really-mean-it
```

Remove a pool

To remove a pool, execute the following commands in the node with `_admin` tag.

```
ceph tell mon.\* injectargs '--mon-allow-pool-delete=true'
ceph osd pool delete <pool-name> <pool-name> --yes-i-really-mean-it
ceph osd pool delete <pool-name> <pool-name> --yes-i-really-really-mean-it
```

Remove failed daemons

To remove failed daemons, execute the following commands in the node where the daemons have failed.

```
# To remove failed "cephadm" daemons

ceph health detail # Look for the failed daemons and their hosts
ssh <host-name>
cephadm rm-daemon --fsid <FSID> --name <daemon-name> --force
```

5 TESTING THE APPLIANCE

This document covers testing of block device of the Ceph/ Scaleflux appliance only.

Testing block device with one client node with 40Gb/s network

While testing the block device sequential and random read write performances, we used FIO tool [17] to perform these tests by modifying the commands from [10] and [18]. The client was mapped with the block device image by following the instructions at 24.

Sequential read

```
fio --name=io-test --ioengine=libaio --iodepth=32 --rw=read --bs=<block-size> --numjobs=16 --
refill_buffers --buffer_compress_percentage=80 --direct=1 --time_based --runtime=120 --
group_reporting --filename /dev/rbd0
```

Sequential write

```
fio --name=io-test --ioengine=libaio --iodepth=32 --rw=write --bs=<block-size> --numjobs=16 --
refill_buffers --buffer_compress_percentage=80 --direct=1 --time_based --runtime=120 --
group_reporting --filename /dev/rbd0
```

Random read-write

```
fio --name=io-test --ioengine=libaio --iodepth=32 --rw=randrw --rwmixread=<percentage-of-read> --
bs=<block-size> --numjobs=16 --refill_buffers --buffer_compress_percentage=80 --direct=1 --
time_based --runtime=120 --group_reporting --filename /dev/rbd0
```

The results to the final configuration of the appliance are tabulated below and screenshots to the some of the results are presented. By using CSD 2000 in our Ceph cluster, we were

able to achieve consistent sequential read (**Error! Reference source not found.**), sequential write (**Error! Reference source not found.**) and random read speeds (**Error! Reference source not found.**) with **17.069 GB/s, 3.237 GB/s, 13.375 GB/s** respectively. In `randrw`, tests the results reported at `rwmixread = 0` shows the write speeds and all other values of `rwmixread` shows read speeds.

Table 1 Ceph with 4 CSD2000 6.4 TB + 4 CSD 2000 3.2TB per node with capacity at 11.2T

block size	seqread (GB/s)	seqwrite (GB/s)	randrw rwmixread =0 (GB/s)	randrw rwmixread =25 (GB/s)	randrw rwmixread =50 (GB/s)	randrw rwmixread =75 (GB/s)	randrw rwmixread= 100 (GB/s)
64kB	7.704	2.89007	1.87785	0.42265	0.82818	1.32038	3.42721
128kB	12.091	2.69533	2.43104	0.54356	1.08177	1.78797	5.59075
512kB	16.692	2.95213	3.10514	0.7704	1.65101	3.02382	11.021
1024kB	**17.069**	3.237	2.71566	0.85065	1.9367	3.69364	**13.375**

```
root@cephadmin-S100-X1S1N-1S1NZZZ0ST0:~# fio --name=io-test --ioengine=libaio --iodepth=32 --rw=read --rwmixwrite=1 --bs=1024k  --numjobs=16 --refill_buffers --buffer_compress_percentage=80
io-test: (g=0): rw=read, bs=(R) 1024KiB-1024KiB, (W) 1024KiB-1024KiB, (T) 1024KiB-1024KiB, ioengine=libaio, iodepth=32
...
fio-3.1
Starting 16 processes
Jobs: 16 (f=16): [R(16)][100.0%][r=16.8GiB/s,w=0KiB/s][r=17.2k,w=0 IOPS][eta 00m:00s]
io-test: (groupid=0, jobs=16): err= 0: pid=11769: Thu Apr 14 11:07:35 2022
   read: IOPS=17.1k, BW=16.7GiB/s (17.0GB/s)(2010GiB/120037msec)
    slat (usec): min=13, max=26220, avg=28.76, stdev=138.94
    clat (usec): min=31, max=98617, avg=29824.71, stdev=10264.48
     lat (usec): min=149, max=115338, avg=29853.74, stdev=10263.69
    clat percentiles (usec):
     |  1.00th=[ 3523],  5.00th=[ 9765], 10.00th=[15926], 20.00th=[21365],
     | 30.00th=[25035], 40.00th=[28181], 50.00th=[30802], 60.00th=[33817],
     | 70.00th=[36439], 80.00th=[39060], 90.00th=[42206], 95.00th=[44303],
     | 99.00th=[47973], 99.50th=[49546], 99.90th=[52691], 99.95th=[54789],
     | 99.99th=[72877]
   bw (  MiB/s): min=  754, max= 1293, per=6.26%, avg=1072.90, stdev=58.81, samples=3840
   iops        : min=  754, max= 1293, avg=1072.76, stdev=58.80, samples=3840
  lat (usec)   : 50=0.01%, 100=0.01%, 250=0.01%, 500=0.01%, 750=0.01%
  lat (usec)   : 1000=0.02%
  lat (msec)   : 2=0.24%, 4=0.98%, 10=3.85%, 20=11.81%, 50=82.70%
  lat (msec)   : 100=0.36%
  cpu          : usr=0.37%, sys=2.87%, ctx=591625, majf=0, minf=131237
  IO depths    : 1=0.1%, 2=0.1%, 4=0.1%, 8=0.1%, 16=0.1%, 32=100.0%, >=64=0.0%
     submit    : 0=0.0%, 4=100.0%, 8=0.0%, 16=0.0%, 32=0.0%, 64=0.0%, >=64=0.0%
     complete  : 0=0.0%, 4=100.0%, 8=0.0%, 16=0.0%, 32=0.1%, 64=0.0%, >=64=0.0%
     issued rwt: total=2058063,0,0, short=0,0,0, dropped=0,0,0
     latency   : target=0, window=0, percentile=100.00%, depth=32

Run status group 0 (all jobs):
   READ: bw=16.7GiB/s (17.0GB/s), 16.7GiB/s-16.7GiB/s (17.0GB/s-17.0GB/s), io=2010GiB (2158GB), run=120037-120037msec

Disk stats (read/write):
  rbd0: ios=566106/0, merge=1485355/0, ticks=16529573/0, in_queue=15398416, util=100.00%
```

Figure 20 Sequential read Block size 1024 KB

```
root@cephadmin-S100-X1S1N-1S1NZZZ0ST0:~# fio --name=io-test --ioengine=libaio --iodepth=32 --rw=write --bs=1024k  --numj
obs=16 --refill_buffers --buffer_compress_percentage=80 --direct=1 --time_based --runtime=120 --group_reporting --filena
me /dev/rbd0
io-test: (g=0): rw=write, bs=(R) 1024KiB-1024KiB, (W) 1024KiB-1024KiB, (T) 1024KiB-1024KiB, ioengine=libaio, iodepth=32
...
fio-3.1
Starting 16 processes
Jobs: 16 (f=16): [W(16)][100.0%][r=0KiB/s,w=3088MiB/s][r=0,w=3088 IOPS][eta 00m:00s]
io-test: (groupid=0, jobs=16): err= 0: pid=16255: Tue May  3 10:03:34 2022
  write: IOPS=3087, BW=3087MiB/s (3237MB/s)(363GiB/120235msec)
    slat (usec): min=8, max=33363, avg=21.82, stdev=114.35
    clat (msec): min=4, max=720, avg=165.55, stdev=184.69
     lat (msec): min=4, max=720, avg=165.57, stdev=184.69
    clat percentiles (msec):
     |  1.00th=[   22],  5.00th=[   27], 10.00th=[   31], 20.00th=[   36],
     | 30.00th=[   41], 40.00th=[   46], 50.00th=[   53], 60.00th=[   63],
     | 70.00th=[  188], 80.00th=[  422], 90.00th=[  464], 95.00th=[  498],
     | 99.00th=[  550], 99.50th=[  558], 99.90th=[  592], 99.95th=[  609],
     | 99.99th=[  676]
   bw (  KiB/s): min=33032, max=428032, per=6.29%, avg=198711.49, stdev=61625.46, samples=3840
   iops        : min=   32, max=  418, avg=193.74, stdev=60.17, samples=3840
  lat (msec)   : 10=0.01%, 20=0.70%, 50=46.21%, 100=21.55%, 250=1.77%
  lat (msec)   : 500=25.29%, 750=4.48%
  cpu          : usr=3.30%, sys=0.37%, ctx=100204, majf=0, minf=180
  IO depths    : 1=0.1%, 2=0.1%, 4=0.1%, 8=0.1%, 16=0.1%, 32=99.9%, >=64=0.0%
     submit    : 0=0.0%, 4=100.0%, 8=0.0%, 16=0.0%, 32=0.0%, 64=0.0%, >=64=0.0%
     complete  : 0=0.0%, 4=100.0%, 8=0.0%, 16=0.0%, 32=0.1%, 64=0.0%, >=64=0.0%
     issued rwt: total=0,371212,0, short=0,0,0, dropped=0,0,0
     latency   : target=0, window=0, percentile=100.00%, depth=32

Run status group 0 (all jobs):
  WRITE: bw=3087MiB/s (3237MB/s), 3087MiB/s-3087MiB/s (3237MB/s-3237MB/s), io=363GiB (389GB), run=120235-120235msec

Disk stats (read/write):
  rbd0: ios=0/102053, merge=0/267780, ticks=0/16438951, in_queue=16235072, util=100.00%
```

Figure 21 Sequential Write Block size 1024 KB

```
root@cephadmin-S100-X1S1N-1S1NZZZ0ST0:~# fio --name=io-test --ioengine=libaio --iodepth=32 --rw=randrw --rwmixread=100 --bs=1024k  --numjobs=16 --refill_buffers --buffer_compress_percentage=80
io-test: (g=0): rw=randrw, bs=(R) 1024KiB-1024KiB, (W) 1024KiB-1024KiB, (T) 1024KiB-1024KiB, ioengine=libaio, iodepth=32
...
fio-3.1
Starting 16 processes
Jobs: 16 (f=16): [r(16)][100.0%][r=12.5GiB/s,w=0KiB/s][r=12.8k,w=0 IOPS][eta 00m:00s]
io-test: (groupid=0, jobs=16): err= 0: pid=27659: Fri Apr 22 14:33:00 2022
   read: IOPS=12.8k, BW=12.5GiB/s (13.4GB/s)(1502GiB/120069msec)
    slat (usec): min=14, max=20445, avg=1225.61, stdev=1932.94
    clat (usec): min=366, max=195545, avg=38717.82, stdev=11255.15
     lat (usec): min=384, max=196735, avg=39943.88, stdev=11545.34
    clat percentiles (msec):
     |  1.00th=[   14],  5.00th=[   21], 10.00th=[   25], 20.00th=[   30],
     | 30.00th=[   33], 40.00th=[   36], 50.00th=[   39], 60.00th=[   42],
     | 70.00th=[   45], 80.00th=[   48], 90.00th=[   53], 95.00th=[   57],
     | 99.00th=[   67], 99.50th=[   74], 99.90th=[  102], 99.95th=[  112],
     | 99.99th=[  134]
   bw (  KiB/s): min=607422, max=1315398, per=6.28%, avg=823609.22, stdev=56656.30, samples=3840
   iops        : min=  593, max= 1284, avg=804.04, stdev=55.30, samples=3840
  lat (usec)   : 500=0.01%, 750=0.01%, 1000=0.01%
  lat (msec)   : 2=0.01%, 4=0.01%, 10=0.19%, 20=4.03%, 50=81.62%
  lat (msec)   : 100=14.05%, 250=0.11%
  cpu          : usr=0.25%, sys=2.14%, ctx=961711, majf=0, minf=163
  IO depths    : 1=0.1%, 2=0.1%, 4=0.1%, 8=0.1%, 16=0.1%, 32=100.0%, >=64=0.0%
     submit    : 0=0.0%, 4=100.0%, 8=0.0%, 16=0.0%, 32=0.0%, 64=0.0%, >=64=0.0%
     complete  : 0=0.0%, 4=100.0%, 8=0.0%, 16=0.0%, 32=0.1%, 64=0.0%, >=64=0.0%
     issued rwt: total=1538351,0,0, short=0,0,0, dropped=0,0,0
     latency   : target=0, window=0, percentile=100.00%, depth=32

Run status group 0 (all jobs):
   READ: bw=12.5GiB/s (13.4GB/s), 12.5GiB/s-12.5GiB/s (13.4GB/s-13.4GB/s), io=1502GiB (1613GB), run=120069-120069msec

Disk stats (read/write):
  rbd0: ios=1535037/0, merge=378/0, ticks=30286637/0, in_queue=27216076, util=99.97%
```

Figure 22 Random read with Block size 1024 KB

Testing block device with five client nodes

To coherently run tests on five machines, we used IO meter [19] to perform tests on the clients. In these tests, all the client machines were running Ubuntu 18.04 with dynamo service of IO meter sending data to a Windows machine running the IO meter server. Out of the five clients, two clients were connected using a 25 Gb/s network interface and three clients were connected using 40 Gb/s network interface. All the clients were mapped with

the same block device (`rbd0`) by following the instructions at 24. The Table 2 below shows the block device speeds in every client and their latencies across various timings.

With link aggregation (combining two or more network interfaces to function as a single link), we were able to achieve **40.584 GB/s** in random read tests with 1024 kB block size (Table 2). To aggregate links [20], please execute the following commands in every host.

1. Install ifenslave

```
sudo apt-get install ifenslave
```

2. Add loop, lp, rtc and bonding to /etc/modules location

```
# /etc/modules: kernel modules to load at boot time.
#
# This file contains the names of kernel modules that should be loaded
# at boot time, one per line. Lines beginning with "#" are ignored.

sfxvdriver
sfvv
sfxv_bd_dev
loop
lp
rtc
bonding
```

3. Stop networking

```
sudo stop networking
```

4. Load the kernel modules

```
sudo modprobe bonding
```

5. Edit /etc/network/interface (sample shown below)

```
# ens1f0np0 is manually configured, and slave to the "bond0" bonded NIC
auto ens1f0np0
iface ens1f0np0 inet manual
    bond-master bond0
    bond-primary ens1f0np0

# ens1f1np1 ditto, thus creating a 2-link bond.
auto ens1f1np1
iface ens1f1np1 inet manual
    bond-master bond0

# bond0 is the bonding NIC and can be used like any other normal NIC.
# bond0 is configured using static network information.
auto bond0
iface bond0 inet static
    address 192.168.18.180
    gateway 192.168.18.1
    netmask 255.255.255.0
    bond-mode balance-rr
    bond-miimon 100
```

```
    bond-slaves none
```

Table 2 100GbE Link aggregated hosts and two 100 GbE client; two 40 GbE client; one 25 GbE client

Test Description														
100GbE Link aggregated hosts and 2 100 GbE client; 2 40 GbE client; 1 25 GbE client														
Target Name	**Read MBps (Decimal)**	**200 to 500 uS**	**0.5 to 1 mS**	**1 to 2 mS**	**2 to 5 mS**	**5 to 10 mS**	**10 to 15 mS**	**15 to 20 mS**	**20 to 30 mS**	**30 to 50 mS**	**50 to 100 mS**	**100 to 200 mS**	**200 to 500 mS**	**Network interface speed (Gb/s)**
All	**40584.58331**	250314	161047 5	31510006	13386298	2881058	1751382	366271	50354	4042	2612	743	78	
cephadmin-S100-X1S1N-1S1NZZZ0ST0	**12367.28216**	234627	496441	145959 6	482681 8	247924	4806	1216	1102	1070	596	32	0	40
Worker 1	12367.28216	234627	496441	145959 6	482681 8	247924	4806	1216	1102	1070	596	32	0	
rbd0	12367.28216	234627	496441	145959 6	482681 8	247924	4806	1216	1102	1070	596	32	0	
cephosd4-QuantaPlex-T41S-2U	**6929.076276**	6046	456968	596527	675263	445000	382224	74977	5291	337	364	98	0	100
Worker 1	6929.076276	6046	456968	596527	675263	445000	382224	74977	5291	337	364	98	0	

rbd0	6929.076276	6046	456968	596527	675263	445000	382224	74977	5291	337	364	98	0	
cephclientthree-QuantaPlex-T42SP-2U-LBG-4-21S5SMA0110	**10149.65352**	6575	56386	122730	599103 1	195655	4425	609	864	922	518	44	0	25
Worker 1	10149.65352	6575	56386	122730	599103 1	195655	4425	609	864	922	518	44	0	
rbd0	10149.65352	6575	56386	122730	599103 1	195655	4425	609	864	922	518	44	0	
clientfive-D51B-1U-dual-1G-LoM	**7462.399917**	3066	600283	926709	159963 3	555667	402161	160850	14602	706	825	376	16	40
Worker 1	7462.399917	3066	600283	926709	159963 3	555667	402161	160850	14602	706	825	376	16	
rbd0	7462.399917	3066	600283	926709	159963 3	555667	402161	160850	14602	706	825	376	16	
ubuntu-QuantaGrid-D52B-1U	**3676.171439**	0	397	45444	293553	143681 2	957766	128619	28495	1007	309	193	62	100
Worker 1	3676.171439	0	397	4544	2935	143681	957766	128619	28495	1007	309	193	62	

				4	53	2								
rbd0	3676.171439	0	397	4544	2935	143681 2	957766	128619	28495	1007	309	193	62	

Initial setup on Windows host side,

6. Download IO meter for Windows from [19].
7. Unzip the downloaded file to any desired location and execute `IOmeter.exe` file.

Setup on client side,

8. Download IO meter for linux from [19].
9. Execute the following commands in an elevated terminal to connect to the IO meter server.

```
chmod u+x /path/to/dynamo
./dynamo -i <io-meter-server-ip> -m <io-meter-dynamo-service-host-ip>
```

Moving to the Windows host side,

10. One should see the client pop-up on the server side in `topology` panel
11. Select the drive to be tested (here `rbd0`) in `disk targets` tab (Figure)
12. Mention the test parameter in the application which includes IO depth (here 32) in `disk targets` tab (Figure).
13. Mention or create the tests in `access specification` tab (Figure).
14. Mention number of cpus (here 16) and description to the test in `test setup` tab (Figure).

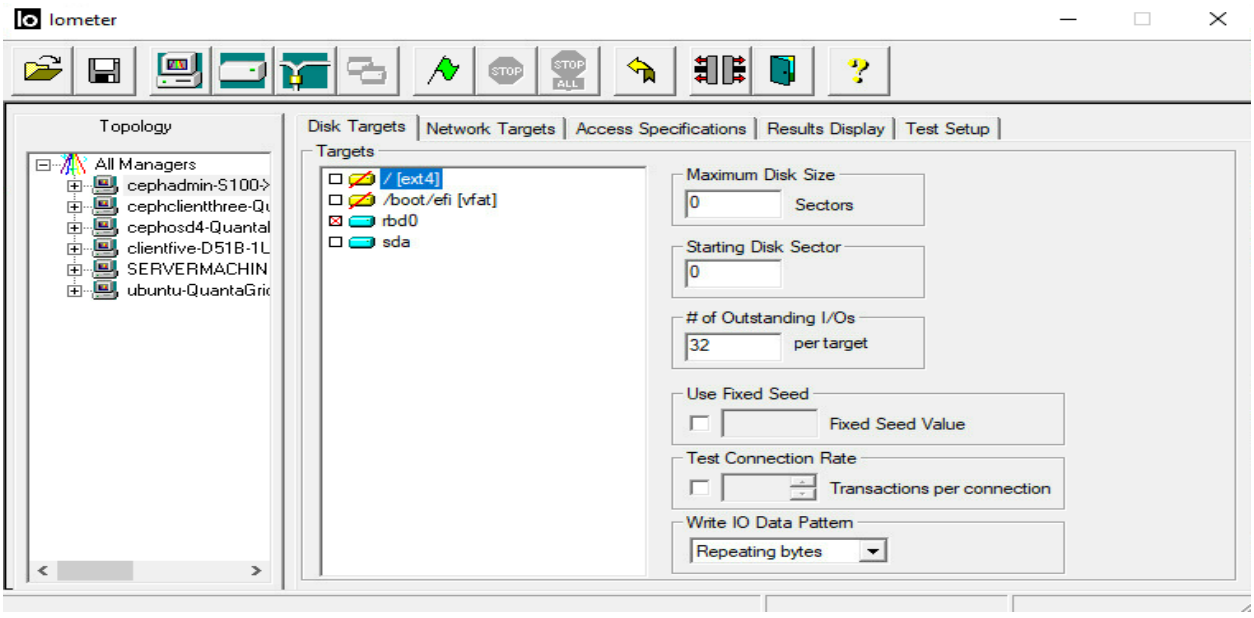


Figure 22 Disk targets tab

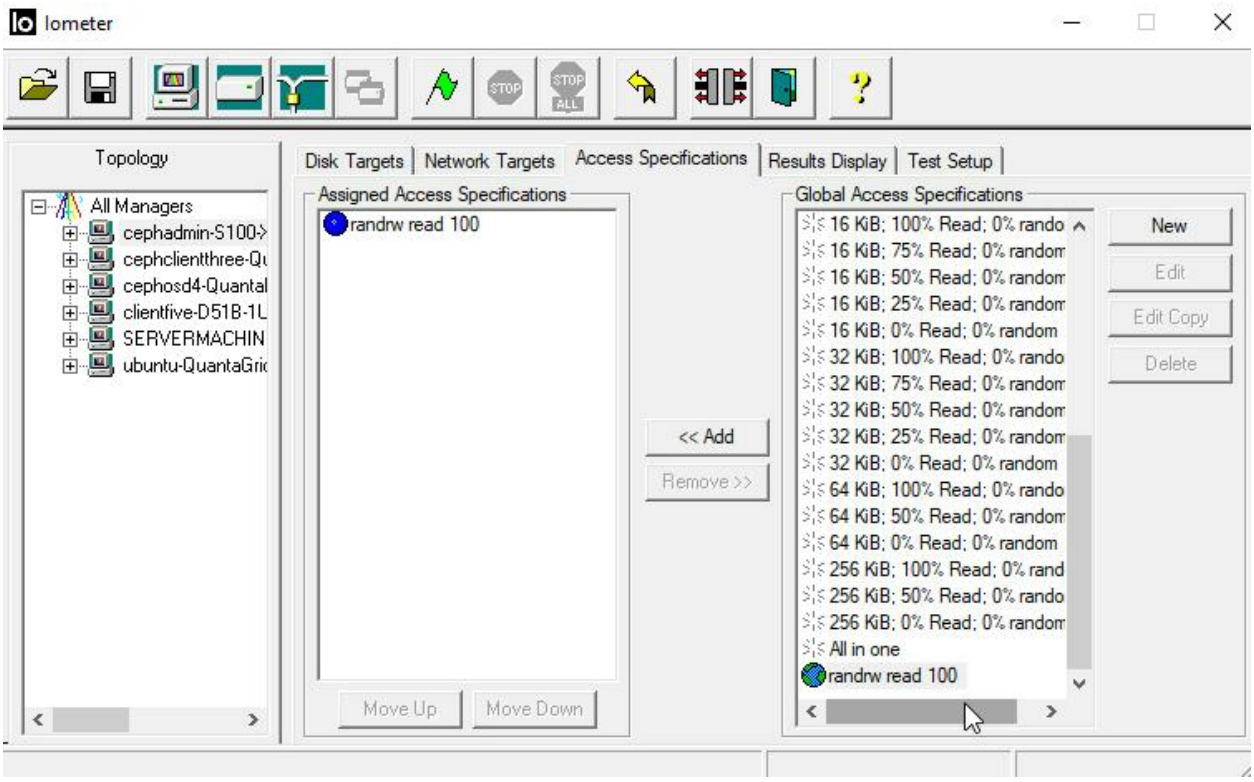


Figure 23 Access Specifications tab

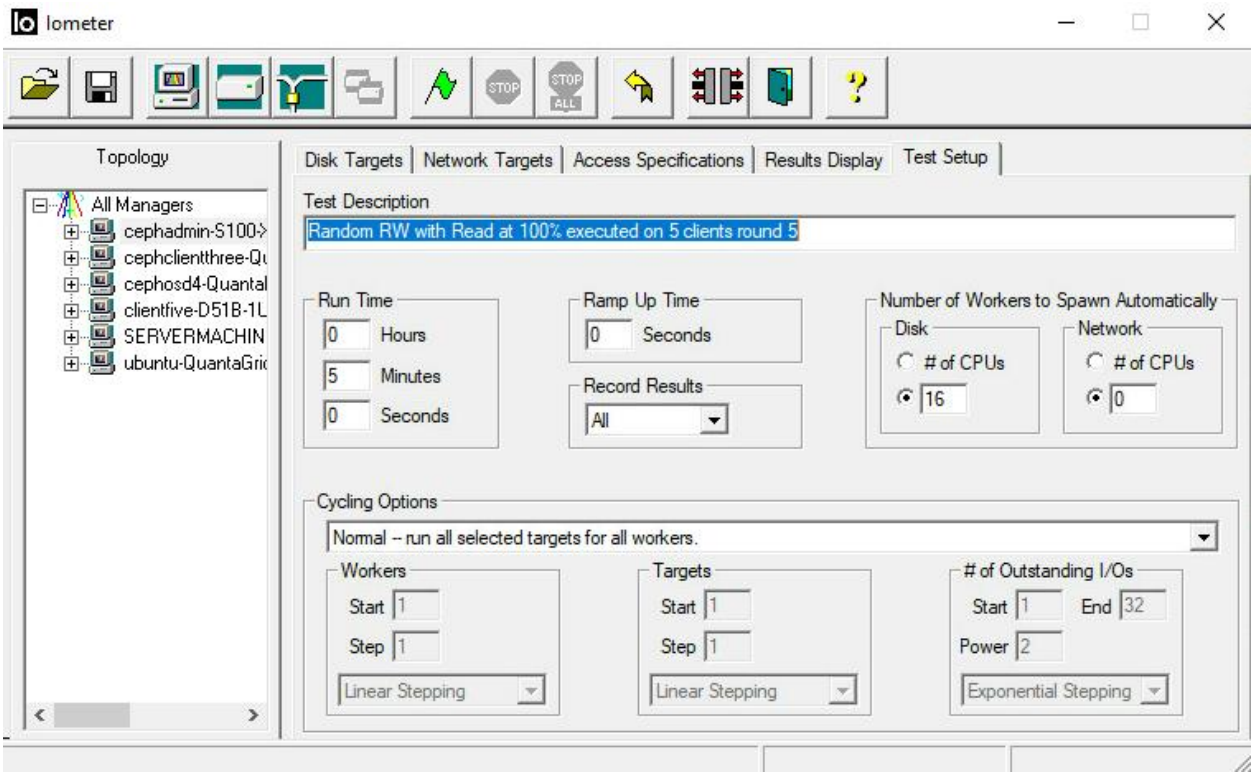


Figure 24 Test setup tab

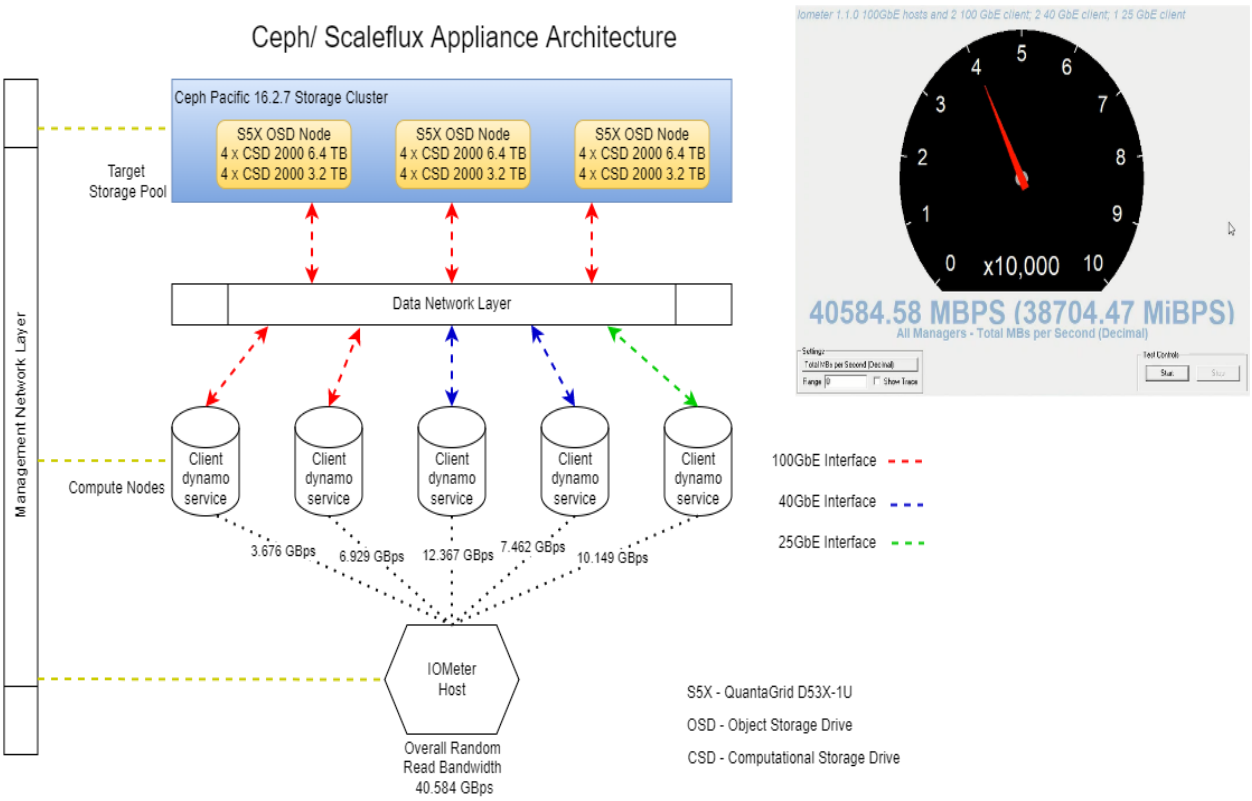


Figure 25 Ceph/ Scaleflux appliance performance with five clients overview

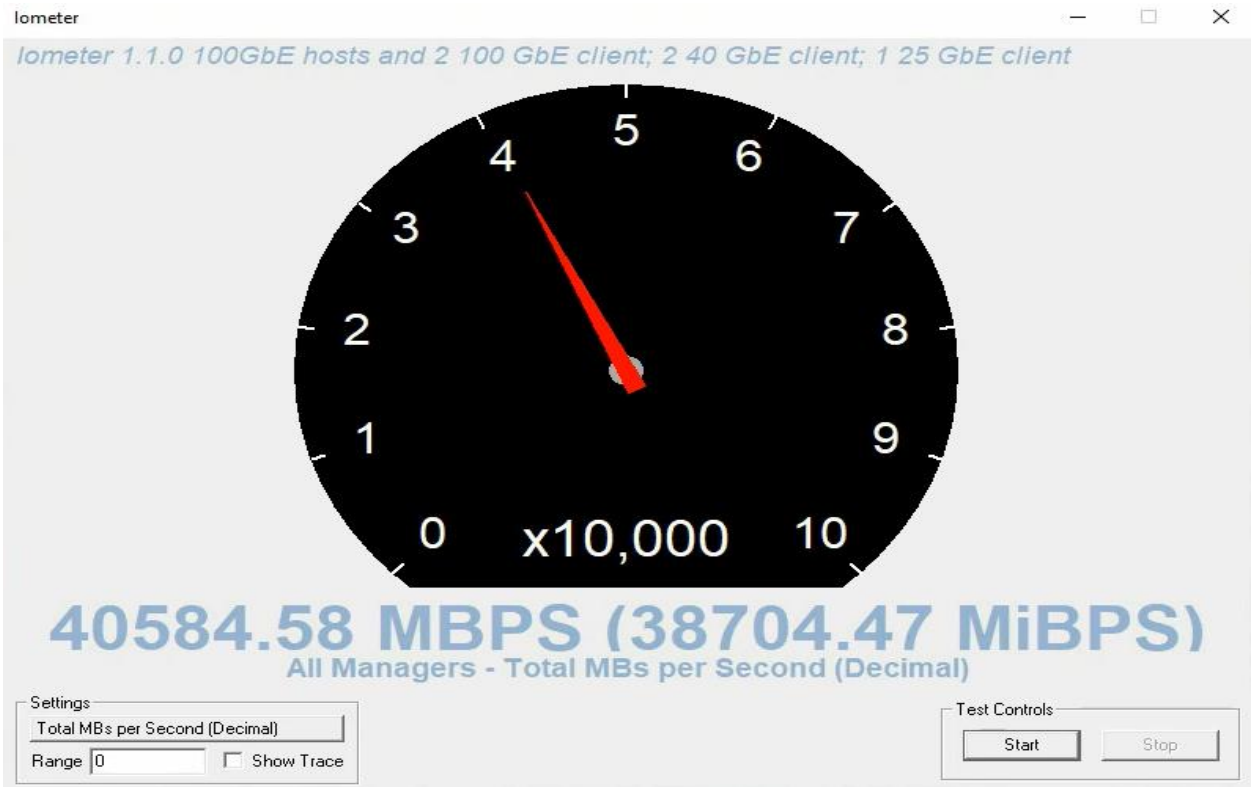


Figure 26 Average combined throughput after link aggregation

We can understand that changing the network configuration, i.e., combining two interfaces of 100GbE NIC, of the hosts helps significantly in improving the random read speeds (25 % improvement) of the appliance. We can expect similar improvements on adding hosts to the cluster or changing the network interface card.

Erasure coding

Ceph offers erasure coding in its pools for cold storage to maximize the usable storage capacity of the cluster. In this document, we'll be testing our pools in *jerasure* plugin (default). Erasure coding stores data through data chunks (denoted as *k*) and parity (denoted as *m*). Though erasure coding is available for object storage for a while, for block storage, it is still under active development and promoted as a *technical preview*. Erasure coding makes sense only when storing large amounts of data (archive). Erasure coding as a block pool with cache tiering works with acceptable performance only in an all-flash solution such as HyperFlow SDCSS. In this appliance, Hyperscalers deployed erasure coded block pool with cache tiering using a replicated cache tier and erasure coded base tier. One should be mindful of available CPU and RAM resources before deploying erasure coding in pools due to the computational complexity of the algorithm (with increase in *m*, you are increasing the order of the equation that is to be solved to store the parity data). Erasure coded pools require a minimum of *k* chunks of data to recover the data. In HyperFlow SDCSS, minimum *k* is 2 for all available plugins. An overview to how data is stored in erasure coded pool and a replicated pool to tolerate 3 concurrent object storage drive failure is shown below (Figure).

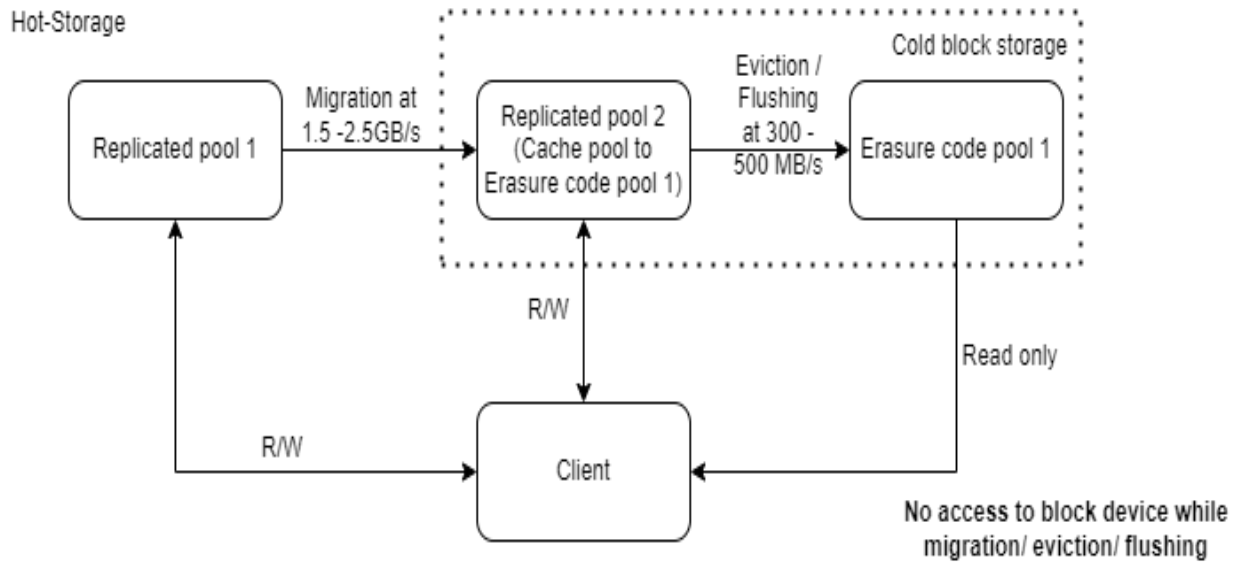


Figure 27 Cold storage overview in HyperFlow SDCSS

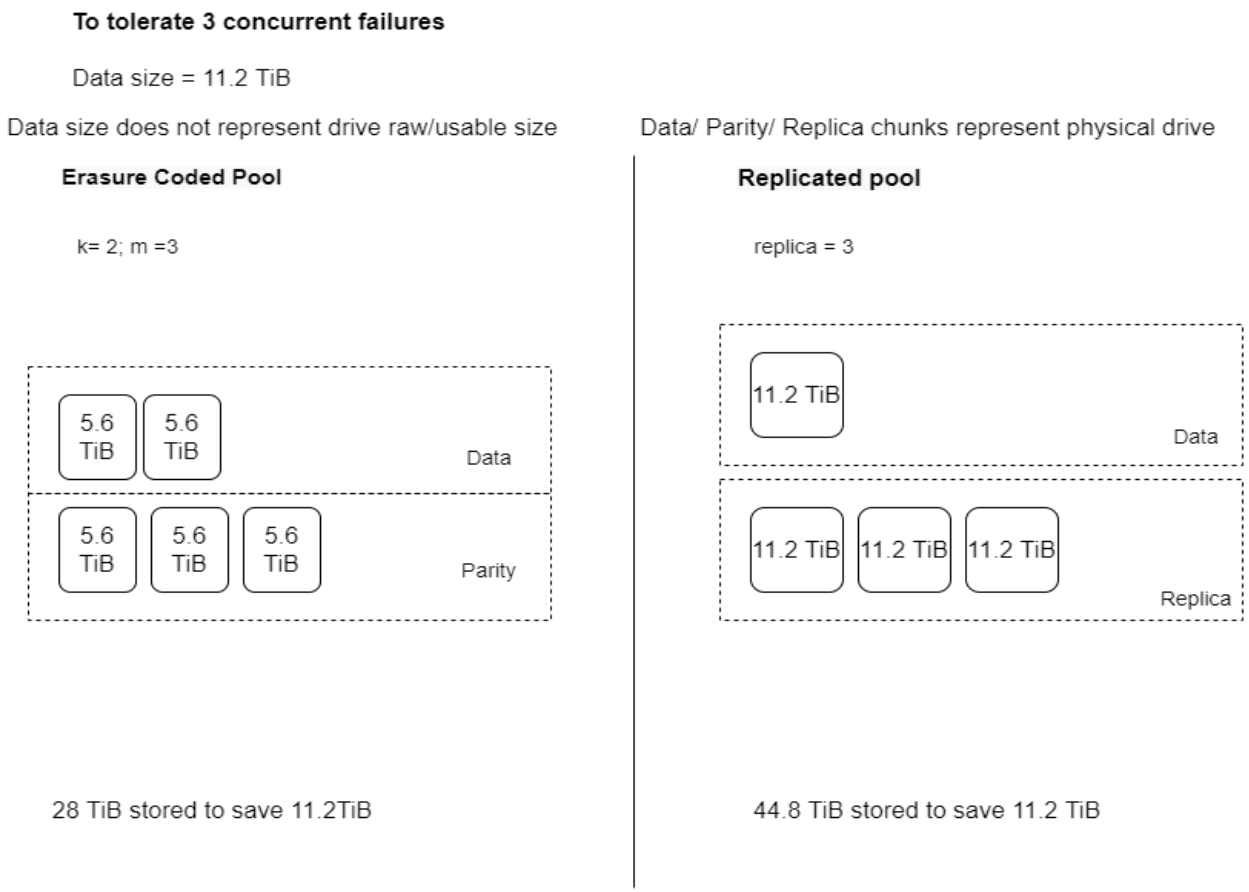


Figure 28 Erasure coded pool vs Replicated pool

To create erasure coded block pools with cache tiering,

```
1.  # Erasure Coding
2.  ceph osd pool create <erasure-coded-pool-name> PG_NUM PGP_NUM erasure default
3.  # PG numbers depend on values of k and m values related to erasure coding
4.  ceph osd pool create <cache-pool-name> PG_NUM PGP_NUM replicated
5.  ceph osd tier add <erasure-coded-pool-name> <cache-pool-name> --force-nonempty
6.  ceph osd tier cache-mode testpool writeback
7.  ceph osd pool set <cache-pool-name> hit_set_type bloom
8.  ceph osd tier set-overlay <erasure-coded-pool-name> <cache-pool-name>
9.  # to set up auto-eviction
10. ceph osd pool set {cachepool} target_max_objects {#objects}
11. ceph osd pool set {cachepool} target_max_bytes {#bytes}
12.
13. rados -p <cache-pool-name> cache-flush-evict-all # To free up cache
```

Object Storage tests

In order to test object storage gateway, we used warp benchmark from [21]. The test was conducted on a 100 Gbps client with warp client accessing the host. The command used to test the object gateway was

```
warp mixed --host 192.168.18.151:443 --access-key=NZ2NG9566E3Z46NFNBUO --secret-key=<secret-key> --
autoterm --tls --insecure
```

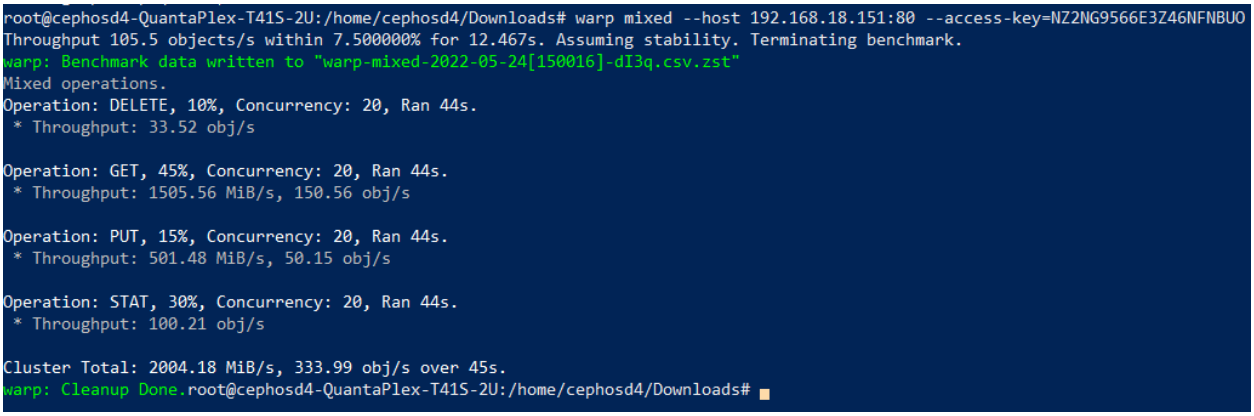


Figure 29 Speed test - object storage

6 IMPROVEMENTS AND BUGS

Improvements

These are the improvement / testing in scope of the appliance

1. DNS / Public SSL
2. Bucket level DNS access

Bugs

1. There's a known bug for Mezzanine 40Gb network card (Connect-X3) with Ubuntu 18.04

7 ADDENDUM

Guidelines in changing and monitoring extended capacity of CSD 2000

Extended capacity of CSD 2000 should be used only when data is known to be compressible with the amount of capacity extension set according to the expected compressibility of the data and the performance goals of the system. If extended capacity is deployed, capacity monitoring must also consider the true free space remaining on the drive [21]. The apparent free space (the space reported based on the advertised capacity) must continue to be monitored alongside the internal free space, since running out of either will result in out-of-space errors.

To monitor the free space of CSD 2000, `sysfs` can be used alongside Nagios and Prometheus. The following parameters are defined:

`/sys/block/sfdv(x)n(n)/sfx_smart_features/sfx_freespace`

This parameter prints two integers. The first integer is the provisioned capacity expressed in 512-byte sectors. The second integer is the amount of free space remaining, also expressed in 512-byte sectors. When the free space reaches zero, the drive is full, and any additional writes will result in out of space errors.

`/sys/block/sfdv(x)n(n)/sfx_smart_features/sfx_physical_size`

This parameter prints the amount of space physically used in the media by the user data. It is expressed in 512-byte sectors. When the value of this parameter reaches the provisioned capacity, the drive is full, and any additional writes will result in out of space errors.

`/sys/block/sfdv(x)n(n)/sfx_smart_features/sfx_logical_size`

This parameter prints the logical size of the user data. It is expressed in 512-byte sectors. The ratio of the logical size to the physical size yields the compression ratio.

`/sys/block/sfdv(x)n(n)/sfx_smart_features/sfx_comp_ratio`

Prints the ratio of the logical size to the physical size.

`/sys/block/sfdv(x)n(n)/sfx_smart_features/sfx_capacity_stat`

Returns data from all the above parameters with a helpful text header. This parameter is intended for manual inspection of the capacity statistics. To estimate the compressibility ratio for CSD 2000, the data that is to be stored in CSD 2000 was passed through compression estimator utility (received from Scaleflux) to find the compressibility ratio to be 4.4:1 (Figure 23).

```
gcc c_est.c -pthread -D_GNU_SOURCE -lz -o c_est
./c_est -d /mnt/ceph-block-device-threetera/Aagi/R\&D/Technologies/ScaleFlux-CSD/Dataset -t 32
```

We chose a nominal value as 3.5:1 and formatted the CSD 2000's effective capacity as 11.2 TB (Maximum performance for CSD 2000 with 6.4 TB and Balance performance for CSD 2000 with 3.2 TB) using the following command [21].

```
sfx-nvme sfx change-cap /dev/sfdv[x]n1 -c 11200 # change capacity of CSD 2000
```

Figure 23 Compression Estimate

The following image shows the extended capacity with compression ratio for CSD 2000 (3.2 TB and 6.4 TB) in Figure 24 and Figure 25 respectively.

Reference Settings Table – 4TiB CSD 2000

Compression Ratio	Maximum Performance		Balanced Performance / Capacity		Maximum Capacity	
	Provisioned Capacity (GB)	Advertised Capacity (GB)	Provisioned Capacity (GB)	Advertised Capacity (GB)	Provisioned Capacity (GB)	Advertised Capacity (GB)
1.1:1	Keep Defaults		3200	3520	3840	4224
1.2:1			3200	3840	3840	4608
1.3:1			3200	4160	3840	4992
1.4:1			3200	4480	3840	5376
1.5:1			3200	4800	3840	5760
2:1			3200	6400	3840	7680
2.5:1	3200	4000	3200	8000	3840	9600
3:1	3200	4800	3200	9600	3840	11520
3.5:1	3200	5600	3200	11200	3840	13440
4:1	3200	6400	3200	12800	3840	15360

Figure 24 Effective capacity guidelines for CSD 2000 4 TiB [21]

Reference Settings Table – 8TiB CSD 2000

Compression Ratio	Maximum Performance		Balanced Performance / Capacity		Maximum Capacity	
	Provisioned Capacity (GB)	Advertised Capacity (GB)	Provisioned Capacity (GB)	Advertised Capacity (GB)	Provisioned Capacity (GB)	Advertised Capacity (GB)
1.1:1	Keep Defaults		6400	7040	7680	8448
1.2:1			6400	7680	7680	9216
1.3:1			6400	8320	7680	9984
1.4:1			6400	8960	7680	10752
1.5:1			6400	9600	7680	11520
2:1			6400	12800	7680	15360
2.5:1	6400	8000	6400	16000	7680	19200
3:1	6400	9600	6400	19200	7680	23040
3.5:1	6400	11200	6400	22400	7680	26880
4:1	6400	12800	6400	25600	7680	30720

Figure 25 Effective Capacity guidelines for CSD 2000 8TiB [21]

OpenSSL Configuration

The following is the command and configuration file used for OpenSSL certificate creation

```
openssl req -new -x509 -nodes -days 730 -keyout private.key -out public.crt -config openssl.conf
```

```
1.  [req]
2.  distinguished_name = req_distinguished_name
3.  x509_extensions = v3_req
4.  prompt = no
5.
6.  [req_distinguished_name]
7.  C = AU
8.  ST = ACT
9.  L = Canberra
10. O = Hyperscalers
11. OU = Engineering
12. CN = HS
13.
14. [v3_req]
15. subjectAltName = @alt_names
16.
17. [alt_names]
18. IP.1 = 192.168.18.151
19. DNS.1 = cephnvme-QuantaGrid-D53X-1U-1S5X2000079
20. IP.2 = 192.168.18.173
21. DNS.2 = cephosd4
```

Commands cheat sheet

The following are the commands to setup the appliance completely with additional frequently used commands to help with initial setup of the appliance.

```
16. # Assumes a fresh install of Ubuntu OS without updates while installation
17. # Executed in an elevated terminal
18. #Preparing CSD2000s for use in Ceph
19.
20. apt update
21. uname -r
22. apt-mark hold 5.8.0-43-generic # takes output from earlier command. Ensure scaleflux drivers
    exist for the kernel at https://packagecloud.io/scaleflux/sfx3x
23. nano /etc/default/grub
24.
25. # Edit the grub with GRUB_CMDLINE_LINUX="crashkernel=auto rhgb quiet intel_idle.max_cstate=0
    processor.max_cstate=0 idle=poll"
26. update-grub
27. reboot
28. apt install curl
29. curl -s https://packagecloud.io/install/repositories/scaleflux/sfx3x/script.deb.sh | sudo
    bash # Works only for Debian based OS
30. apt search sfx3xdriver-src
31. sudo apt install sfx3xdriver-src
32.
33. # Preconditioning of CSD 2000
34. apt install fio
35. # Sequential preconditioning
36.
```

```
fio --ioengine=libaio --direct=1 --group_reporting --name=baseline --thread --stonewall --new_group --fill_device=1 --rw=write --rwmixread=0 --bs=128k --numjobs=1 --iodepth=128 --loops=2 --buffer_compress_percentage=80 --refill_buffers --filename /dev/sfdv[X]n1

# random pre conditioning

fio --ioengine=libaio --direct=1 --group_reporting --name=baseline --thread --stonewall --new_group --fill_device=1 --rw=randrw --rwmixread=0 --bs=128k --numjobs=4 --iodepth=128 --loops=1 --buffer_compress_percentage=80 --refill_buffers --filename /dev/sfdv[X]n1

#Ceph Pre-requisites Install

apt install ntp
apt install net-tools
apt-get install     ca-certificates     gnupg     lsb-release #if you're using CSD 2000s
echo   "deb [arch=$(dpkg --print-architecture) signed-by=/usr/share/keyrings/docker-archive-keyring.gpg] https://download.docker.com/linux/ubuntu \ $(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
apt-get install docker-ce docker-ce-cli
apt-get update
apt-get install docker-ce docker-ce-cli containerd.io
apt install openssh-server
nano /etc/ssh/sshd_config
# Edit the ssh config with PermitRootLogin yes
passwd # set/change root password for ssh access
ssh-keygen # Generates public-private key pair
nano /etc/hosts
# Add the hosts and their corrsponding ip address. Ensure hostname matches the actual hostname.
ssh-copy-id <host-name>
#This allows passwordless ssh access
apt install lvm2

#Ceph Installation

#Navigate to any location of interest where you want the "cephadm" file to be placed
curl --silent --remote-name --location https://github.com/ceph/ceph/raw/<release-name>/src/cephadm/cephadm
chmod +x cephadm
# For help and available options use "./cephadm --help"
./cephadm add-repo --release <release-name>
./cephadm install
cephadm bootstrap --mon-ip <monitor-ip>
# creates a minimal ceph cluster with 1 monitor node and 1 manager node with dashboard url (with SSL) and its access credentials are presented as output
cephadm install ceph-common # helps in accessing cluster details outside the "cephadm" container
ssh-copy-id -f -i /etc/ceph/ceph.pub <host-name>
./cephadm prepare-host <host-name>
# checks the host for necessary pre-requisites
ceph orch host add <host-name>
# adds node to the cluster

cephadm shell # To access the container shell

# To remove a host from the cluster

systemctl stop <ceph-osd-service>
ceph osd out osd.x
ceph osd down osd.x
ceph osd rm osd.x
ceph osd crush rm osd.x
ceph auth del osd.x
ceph osd destroy x --yes-i-really-mean-it
ceph orch host drain <host-name>
```

```
92. # Deactivates monitor and manager services, removes them and updates the monmap of the
    cluster
93. ceph orch host rm <host-name>
94.
95. # To stop adding available OSD to the cluster
96. # By default (in this method of installation) available Object Storage Drives (OSD) are
    picked up by the cluster and added as OSDs to the cluster
97. ceph orch apply osd --all-available-devices unmanaged = true # Stops adding OSD
    automatically into the cluster in any given node
98.
99. # To remove an OSD from the cluster
100.
101.  systemctl stop <ceph-osd-service>
102.  ceph osd out osd.x
103.  ceph osd down osd.x
104.  ceph osd rm osd.x
105.  ceph osd crush rm osd.x
106.  ceph auth del osd.x
107.  ceph osd destroy x --yes-i-really-mean-it
108.
109.  # To create a Rados Block Device(RBD)
110.  # In Monitor node,
111.  rbd pool init <pool-name>
112.  # In client node,
113.  apt install ceph-common # Only if ceph-common was not installed earlier
114.  rbd create <pool-name> --size <pool-size> --image-feature layering -m mon-ip -k
    /path/to/ceph.client.admin.keyring -c /path/to/ceph.conf
115.  rbd map <pool-name> --name client.admin -m monitor-ip -k
    /path/to/ceph.client.admin.keyring -c /path/to/ceph.conf
116.  mkfs.ext4 -m0 /dev/rbdX
117.
118.
119.  # To remove an existing pool
120.
121.  ceph tell mon.\* injectargs '--mon-allow-pool-delete=true'
122.  ceph osd pool delete <pool-name> <pool-name> --yes-i-really-mean-it
123.  ceph osd pool delete <pool-name> <pool-name> --yes-i-really-really-mean-it
124.
125.  # To restore "device_health_metrics" in case of removal of all OSDs
126.
127.  ceph tell mon.\* injectargs '--mon-allow-pool-delete=true'
128.  ceph osd pool delete device_health_metrics device_health_metrics --yes-i-really-mean-it
129.  ceph osd pool delete device_health_metrics device_health_metrics --yes-i-really-really-
    mean-it
130.  ceph device scrape-health-metrics
131.
132.  # To remove failed "cephadm" daemons
133.
134.  ceph health detail # Look for the failed daemons and their hosts
135.  ssh <host-name>
136.  cephadm rm-daemon --fsid <FSID> --name <daemon-name> --force
137.
138.  # To deploy object gateway with ssl
139.
140.  ssh <one-of-monitor-nodes>
141.  openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout /etc/ssl/private/ceph-rgw-
    cert.key -out /etc/ssl/certs/ceph-rgw.crt # create a SSL certificate
142.  # Navigate to any desired location
143.  touch nvmeServer.pem
144.  cat /etc/ssl/certs/ceph-rgw.crt >> /home/cephnvme/nvmeServer.pem
145.  cat /etc/ssl/private/ceph-rgw-cert.key >> /home/cephnvme/nvmeServer.pem # concatenate key
    and certificate files
146.  cat nvmeServer.pem # verify that key and certificate files are concatenated
147.  ceph orch apply rgw admin --realm=default --zone=default --placement=<host-name>
148.  # In Ceph Dashboard Cluster -> Services -> rgw.admin -> Edit
```

```
# Change port to 443 ; Tick the SSL  box ; Attach the nvmeServer.pem file
ceph dashboard set-rgw-api-ssl-verify False
curl -k https://<placement-host-name-ip>:443 # verify "anonymous" response from the ip
# Verify similar response from the browser
```

Test results

The following tables are the speed test results with 1 CSD 2000 and 7 CSD 2000 per node from one client with 40 Gb/s network. In `randrw`, tests the results reported at `rwmixread = 0` shows the write speeds and all other values of `rwmixread` shows read speeds.

Table 3 Ceph with 1 CSD2000 6.4TB per node with capacity at 11.2T

block size	seqread (GB/s)	seqwrite (GB/s)	randrw rwmixread =0 (GB/s)	randrw rwmixread =25 (GB/s)	randrw rwmixread =50 (GB/s)	randrw rwmixread =75 (GB/s)	randrw rwmixread =100 (GB/s)
64kB	9.68992	1.76229	0.89024	0.28462	0.7704	1.4231	4.32922
128kB	12.198	1.84468	1.08391	0.46866	0.9951	1.90567	6.84586
512kB	15.515	1.99555	1.40812	0.40874	1.28828	3.05806	12.305
1024kB	17.334	1.98913	1.34285	0.4708	1.73875	3.71718	14.445

Table 4 Ceph with 4 CSD2000 6.4 TB + 3 CSD 2000 3.2TB per node with capacity at 11.2T

block size	seqread (GB/s)	seqwrite (GB/s)	randrw rwmixread =0 (GB/s)	randrw rwmixread =25 (GB/s)	randrw rwmixread =50 (GB/s)	randrw rwmixread =75 (GB/s)	randrw rwmixread =100 (GB/s)
64kB	7.33913	2.93501	1.88427	0.42907	0.80999	1.38886	3.62944
128kB	12.519	2.95855	2.4075	0.54035	1.09033	1.87464	5.56293
512kB	18.832	2.99386	2.83978	0.77468	1.66813	3.02382	15.836
1024kB	18.939	2.99172	2.68035	0.82711	1.92921	3.68401	17.655

IO meter tests

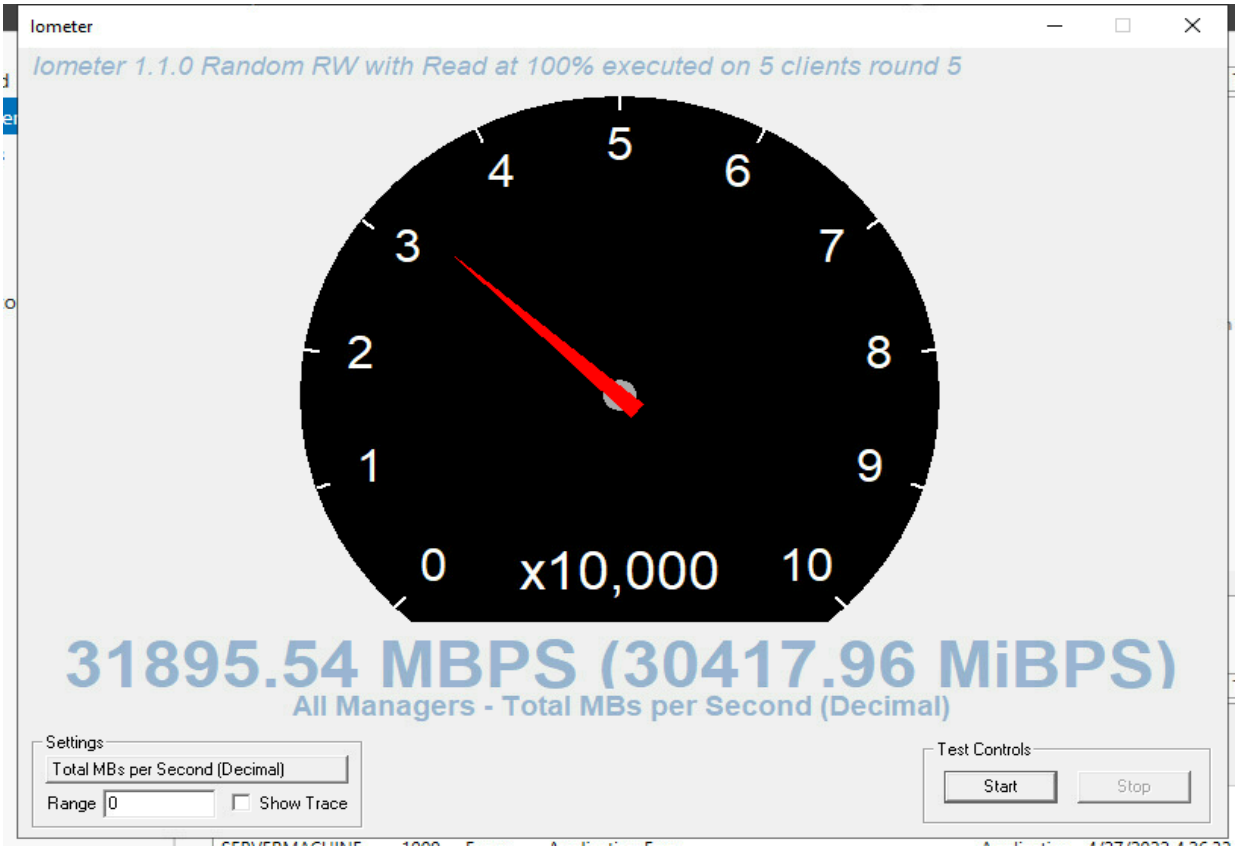


Figure 26 Average combined throughput of random read with 1024 KB block size before link aggregation

Table 5 Five client IO meter test Block Size 1024 KB without link aggregation

		Latencies of data (Number of packets)											
Target Name	**Read MBps (Decimal)**	**200 to 500 uS**	**0.5 to 1 mS**	**1 to 2 mS**	**2 to 5 mS**	**5 to 10 mS**	**10 to 15 mS**	**15 to 20 mS**	**20 to 30 mS**	**30 to 50 mS**	**50 to 100 mS**	**100 to 200 mS**	**Network interface speed (Gb/s)**
All	**31895.54061**	8486	1193634	1851086	956730	5774699	1651586	527660	200773	28913	2223	519	
cephadmin-S100-X1S1N-1S1NZZZ0ST0	**6506.051153**	1184	117927	206083	243758	3626950	45803	1180	629	626	476	65	40
Worker 1	6506.051153	1184	117927	206083	243758	3626950	45803	1180	629	626	476	65	
rbd0	6506.051153	1184	117927	206083	243758	3626950	45803	1180	629	626	476	65	
clientfive-D51B-1U-dual-1G-LoM	**6927.706683**	2826	589598	768849	136952	5883082	464251	163865	13998	523	476	141	40
Worker 1	6927.706683	2826	589598	768849	136952	5883082	464251	163865	13998	523	476	141	
rbd0	6927.706683	2826	589598	768849	136952	5883082	464251	163865	13998	523	476	141	

				49	522								
ubuntu-QuantaGrid-D52B-1U	**3253.349718**	0	3325	2001	3649	748277	781750	255815	178223	26664	328	163	40
Worker 1	3253.349718	0	3325	2001	3649	748277	781750	255815	178223	26664	328	163	
rbd0	3253.349718	0	3325	2001	3649	748277	781750	255815	178223	26664	328	163	
cephosd4-QuantaPlex-T41S-2U	**6946.743098**	4050	470318	6511	6368	416152	356779	106429	7378	491	493	94	25
Worker 1	6946.743098	4050	470318	6511	6368	416152	356779	106429	7378	491	493	94	
rbd0	6946.743098	4050	470318	6511	6368	416152	356779	106429	7378	491	493	94	
cephclientthree-QuantaPlex-T42SP-2U-LBG-4-21S5SMA0110	**8261.689957**	426	12466	2490	475838	395012	3003	371	545	609	450	56	25
Worker 1	8261.689957	426	12466	2490	475838	395012	3003	371	545	609	450	56	
rbd0	8261.689957	426	12466	2490	475838	395012	3003	371	545	609	450	56	

8 TRADEMARKS AND LICENSING (OPTIONAL)

9 REFERENCES

[1] Hyperscalers, "About HS," [Online]. Available: https://www.hyperscalers.com/about-us-hyperscalers.

[2] Scaleflux, "About-Overview," [Online]. Available: https://www.scaleflux.com/intro/1. [Accessed 2022].

[3] ScaleFlux, "What is Computational Storage," [Online]. Available: https://www.scaleflux.com/. [Accessed 2022].

[4] Ceph, "Ceph Homepage," [Online]. Available: https://ceph.com/en/. [Accessed 2022].

[5] Scaleflux, "CSD 2000," [Online]. Available: https://www.scaleflux.com/product/item/1002. [Accessed 2022].

[6] Hyperscalers, "S5X 2.5'' | D53X-1U," [Online]. Available: https://www.hyperscalers.com/storage/storage-servers/hyperscalers-S5X-D53X-1U-ice-lake-densest-hyperscale-server-nvme-drives-buy. [Accessed 2022].

[7] Ceph, "Ceph Glossary," [Online]. Available: https://docs.ceph.com/en/pacific/glossary/. [Accessed 2022].

[8] Ceph, "DEPLOYING A NEW CEPH CLUSTER," [Online]. Available: https://docs.ceph.com/en/latest/cephadm/install/. [Accessed 2022].

[9] Canonical, "Ubuntu 20.04.4 LTS (Focal Fossa)," [Online]. Available: https://releases.ubuntu.com/20.04.4/. [Accessed 2022].

[10] J. Wang, "FIO-Baseline," Github, [Online]. Available: https://github.com/jinqiangwang/fio-baseline. [Accessed 2022].

[11] Ceph, "Deploying a new Ceph cluster," [Online]. Available: https://docs.ceph.com/en/pacific/cephadm/install/#requirements. [Accessed 2022].

[12] Docker docs, "Get Docker," [Online]. Available: https://docs.docker.com/get-docker/.

[13] Liquid web, "Enable root login via ssh in Ubuntu," [Online]. Available: https://www.liquidweb.com/kb/enable-root-login-via-ssh/.

[14] Ceph, "Ceph Object gateway," [Online]. Available: https://docs.ceph.com/en/pacific/radosgw/index.html. [Accessed 2022].

[15] Ceph, "what's the difference between pg and pgp?," [Online]. Available:

] http://lists.ceph.com/pipermail/ceph-users-ceph.com/2015-May/001610.html. [Accessed 2022].

[16] Ceph Archives, "Ceph PGs per pool calculator," [Online]. Available: https://web.archive.org/web/20210301111112/http://ceph.com/pgcalc/. [Accessed 2022].

[17] J. Axboe, "FIO Documentation," [Online]. Available: https://fio.readthedocs.io/en/latest/fio_doc.html. [Accessed 2022].

[18] Ceph, "Benchmark Ceph Cluster Performance," [Online]. Available: https://tracker.ceph.com/projects/ceph/wiki/Benchmark_Ceph_Cluster_Performance. [Accessed 2022].

[19] Intel Corporation, "Iometer - Downloads," [Online]. Available: http://www.iometer.org/doc/downloads.html. [Accessed 2022].

[20] Canonical, "Ubuntu Bonding," [Online]. Available: https://help.ubuntu.com/community/UbuntuBonding. [Accessed 2022].

[21] Scaleflux , "Extended Capacity User Guide CSD 2000 Series," San Jose, CA, 2021.

[22] Ceph, "Storage cluster quickstart," [Online]. Available: https://docs.ceph.com/en/mimic/start/quick-ceph-deploy/. [Accessed 2022].

[23] Ceph, "Orchestrator CLI," [Online]. Available: https://docs.ceph.com/en/latest/mgr/orchestrator/. [Accessed 2022].

Index