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

Solving Information Technology's **Complexity**

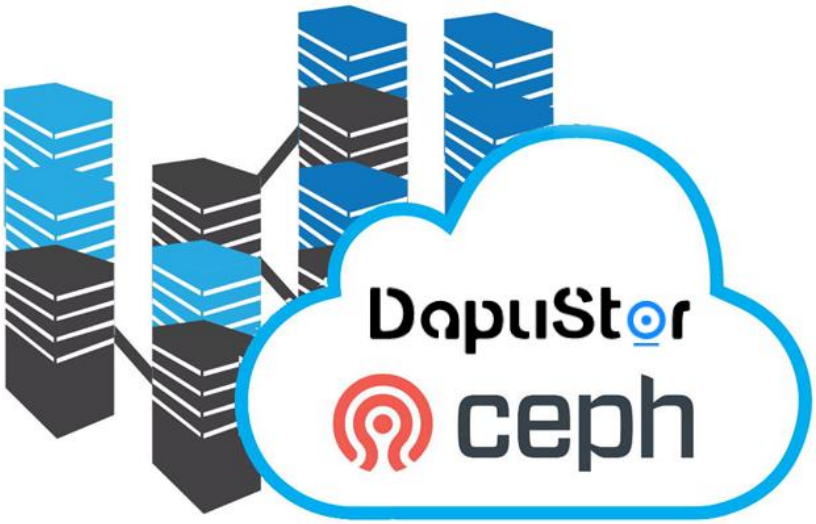
HYPERSCALERS

ZSDS
High Performance Software Defined Storage Solution

Based on Ceph using High Performance NVMe Storage by DapuStor

Reference Guide

HYPERSCALERS



Tuesday, 15 August 2023

1 TABLE OF CONTENTS

Table of Figures

INTRODUCTION

The ZSDS high performance and low latency Storage Solution by Hyperscalers [1] and Dapustor (using Roealsen5 eSSD) was co-developed to fill a need experienced by many organisations for easy to consume, low cost yet blazingly fast NVMe based storage delivered in the context of highly-available block, file & object Ceph storage services.

The following figure outlines key synergies achieved through the ZSDS architecture:

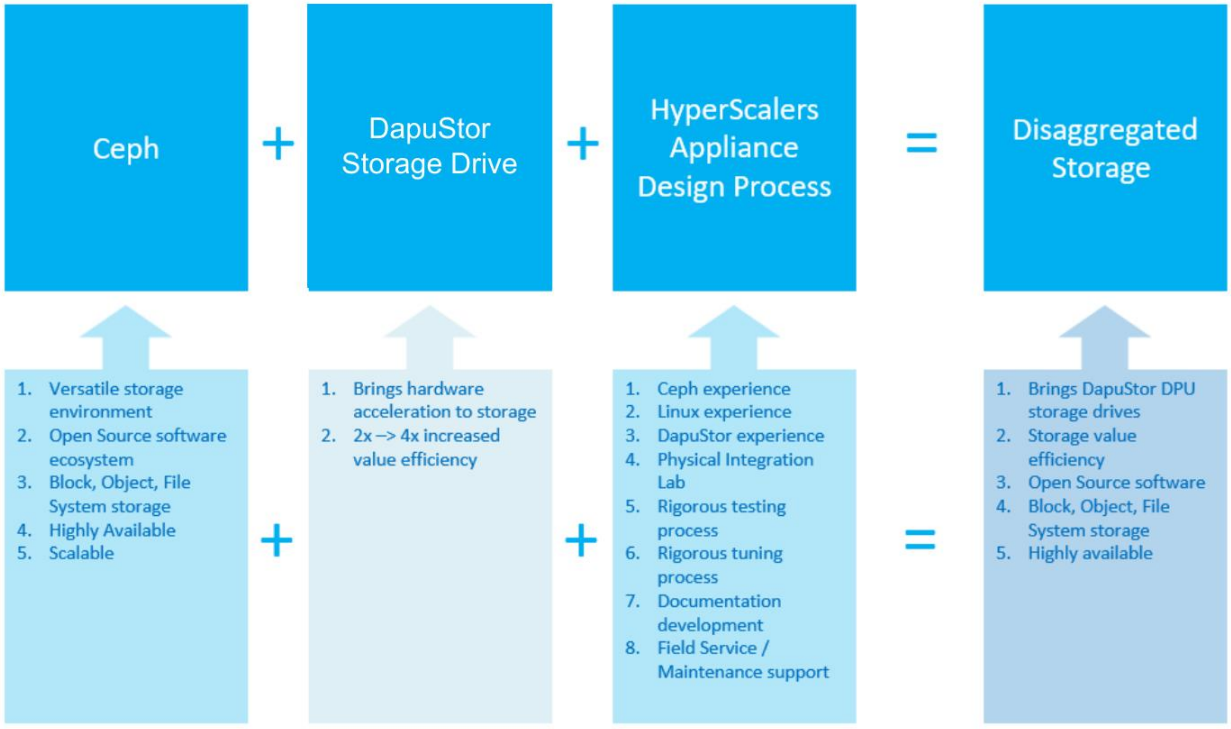


Figure 1 Values of the ZSDS Software Defined Storage Solution

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.

DapuStor Corporation (DapuStor) is a leading expert in premium enterprise solid-state drives (SSD), SOC, and edge computing related products. DapuStor has comprehensive capabilities in storage system ASIC controller chip design, fabrication and manufacturing.

The most recent 3D enterprise TLC NAND from KIOXIA is used in the DapuStor R5 Series [2] design and construction. Industry-leading SSDs with high speed, outstanding reliability, low latency, and excellent power efficiency result from such a unique combination, offering optimised TCO to enterprise IT and cloud facilities. For essential data storage situations across a variety of industries, including enterprise IT, logistics, the internet, banking, intelligent manufacturing, and AI, the DapuStor R5 series is the perfect solution.

In comparison to the Haishen3 series, the DapuStor R5 series PCIe Gen4 SSD delivers a 100% increase in bandwidth and IOPS performance. The Roealsen5 series' latency and QoS have greatly increased in mixed read-write scenarios as a result of several IO route optimizations made by the new DP600 controller.

Ceph (16.2.9/ Pacific) [3] 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 technology from DapuStor [2] in QuantaGrid D53X-1U (S5X) servers from Hyperscalers [4].

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 (17.2.4) cluster with DapuStor R5 in Ubuntu 20.04.

The purpose of this document is to create a Ceph (16.2.9) cluster with DapuStor R5 with SSDs 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 that we use in conjunction with an Appliance Optimizer Utility 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.

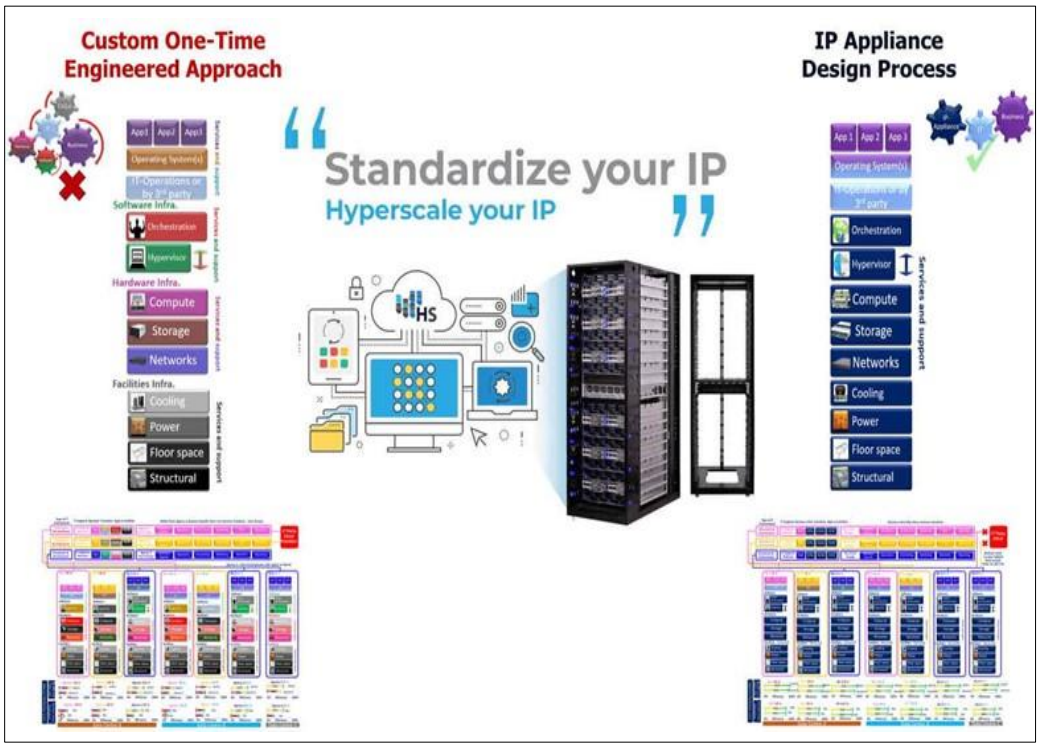


Figure 2 Digital IP-Appliance Design Process

IMPORTANT CONSIDERATIONS

This appliance documentation is qualified and valid only for this hardware (13) and software (14) 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 (13)

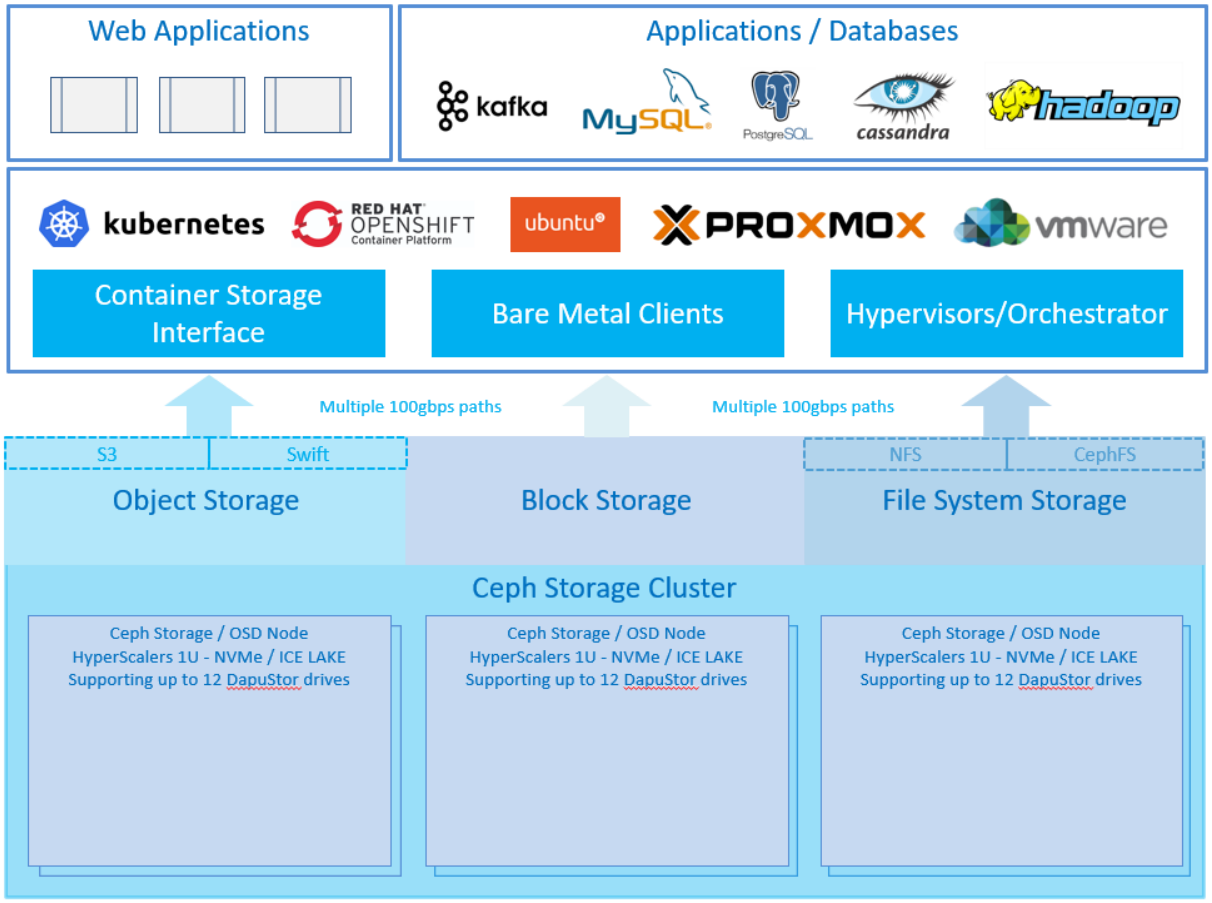


Figure 3 ZSDS storage solution generic solution

BASE PRODUCT DEPLOYMENT

Ceph (16.2.9/ Pacific) [3] 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 R5100 drives (7 TiB) from DapuStor [2] in QuantaGrid D53X-1U (S5X) server from Hyperscalers [4].

INTRODUCTION TO R5100

The newest 112L 3D NAND Flash from KIOXIA is installed in the DapuStor R5 Series, achieving an incredibly high level of power efficiency. Through cutting-edge machine learning techniques that anticipate the NAND workload in complicated settings to avoid systemic failures, it decreases NAND Retry at the system level. The heterogeneous computing interface and an integrated processor are features of the DapuStor DP600 controller for PCIe 4.0 SSDs. It provides higher performance while running Linux, easily transfers software and algorithms, and boosts the effectiveness of the system for database, artificial intelligence, and big data applications.

- **Form Factor** - U.2 15mm Drive
- **Flash Capacity** Up to 15.36 TB KIOXIA 3D NAND, 112 layer, 2 plane eTLC
- **Interface** - PCIe 4.0 x4, NVMe 1.4a
- **Read Bandwidth (128KB) MB/s:** 7400
- **Write Bandwidth (128KB) MB/**s: 5700
- **Random Read (4KB) KIOPS**: 1750
- **Random Write (4KB) KIOPS:** 280
- **4K Random Latency(Typ.)R/W µs:** 65/9
- **4K Sequential Latency (Typ.) R/W µs**: 8/9
- **Power:** Typical: ≤ 20.5 W, Idle: ≤ 7 W: 1 DWPD
- **UBER**:1 sector per 10^17 bits read
- **MTBF**:2 million hours
- **Lifetime:**5 yrs

Key advantages of DapuStor Roealsen5 eSSD drives include:

1. **End-to-end data protection, through Variant Sector Size (VSS) + Protection Information**

VSS is a technology that allows SSDs to support multiple sector sizes, which can help to improve performance and reduce the amount of wasted space on the drive. By supporting

multiple sector sizes, SSDs can be optimized for different types of workloads and provide better performance for specific applications.

Protection Information (PI) is a feature that adds extra data to each sector to detect and correct errors that may occur during data transmission or storage. This helps to ensure data integrity and prevent data loss or corruption.

DapuStor SSDs use a combination of Variant Sector Size and Protection Information to provide end-to-end data protection. This ensures that data is protected from the moment it is written to the SSD, throughout the data transfer process, and during storage on the SSD. Additionally, DapuStor SSDs feature power loss protection, which uses capacitors to ensure that data is not lost in the event of a power failure.

2. **Device heath and capacitor health detection**

SMART (Self-Monitoring, Analysis, and Reporting Technology) - All DapuStor SSDs include an integrated SMART monitoring system. It is used to track various metrics related to the drive's health and performance, such as the number of write cycles, the temperature of the drive, and the amount of available spare blocks. When a drive's SMART data indicates that certain thresholds have been exceeded or errors have been detected, it can be a sign that the drive is experiencing issues or is at risk of failure.

Health Monitoring Software - DapuStor provides health monitoring software that can be used to track the status of SSDs. These tools often provide more detailed information than SMART and can alert users to potential issues before they become critical.

Capacitor Health Detection - The Roealsen5 Series have capacitors that provide power during data writes in the event of a sudden power loss. Capacitor health detection is used to ensure that these components are functioning properly and can provide the necessary power when needed. This is typically done through monitoring the voltage and capacitance of the capacitors and comparing it to expected values.

3. **Six levels of power consumption/optimisation**

DapuStor Roealsen5 series comes with a power management feature that allows users to adjust power level utilisation based on their specific workload requirements, thereby reducing overall power consumption. The Roealsen5 series supports six different power modes, ranging from the lowest power consumption level (P0) to the highest power consumption level (P5).

The power levels adjustment feature allows users to customize the power consumption of their SSD according to their specific needs, such as read or write-intensive workloads. For example, users can set the SSD to a lower power consumption level during periods of low activity, reducing overall power consumption and extending the life of the device.

This feature is especially beneficial for data center and enterprise customers who need to optimize their power consumption while maintaining high-performance levels. By adjusting the power levels of their SSDs, users can reduce their energy bills and lower their carbon footprint, while still achieving high levels of performance and reliability.

4. **Multiple-namespace support (up to 32 namespaces)**

Multiple-namespace support is a feature that allows a single physical SSD to be divided into multiple logical namespaces, each with its own independent storage space and access controls. This can be useful in a variety of scenarios, such as virtualization, multi-tenancy, and data isolation.

DapuStor SSDs offer support for up to 32 namespaces, which is a significant number compared to other SSDs on the market. This means that users can create up to 32 independent namespaces on a single DapuStor SSD, each with its own unique identifier and access controls. This provides a high level of flexibility and customization, allowing users to tailor their storage environment to their specific needs.

In addition to multiple-namespace support, DapuStor SSDs also offer other advanced features such as power-loss protection, wear-leveling, and encryption, making them a popular choice for enterprise and data center environments.

5. **Weighted round robin command arbitration and high priority command support**

Weighted round robin command arbitration is a technique used to manage the ordering of commands that are submitted to an SSD. With this technique, each command is assigned a weight or priority, and the SSD processes commands in a round-robin fashion, taking into account their assigned weight. This ensures that higher priority commands are processed more quickly than lower priority ones.

DapuStor SSDs do support weighted round robin command arbitration, which means that users can assign priorities to their commands and ensure that high-priority commands are processed with a higher priority than lower-priority ones. This can be particularly useful in scenarios where certain commands need to be processed urgently, such as in a real-time application or a database server.

Overall, the support for weighted round robin command arbitration is another example of the advanced features that DapuStor SSDs offer, making them a popular choice for high-performance computing environments.

6. **Enhanced secure erase**

DapuStor SSDs offer enhanced secure erase features, which are designed to clear existing data safely and protect data security and privacy before drives are reused.

The secure erase feature in DapuStor SSDs is a block erase/block overwrite operation that can be initiated by the user or via a software command. When initiated, the secure erase process overwrites all user data on the drive with a predefined pattern, rendering the data unrecoverable. This is done by overwriting the entire user addressable space on the SSD, including any hidden or reserved areas.

The enhanced secure erase feature in DapuStor SSDs provides a high level of data security, which is particularly important for enterprise and data center environments where sensitive data may be stored. By securely erasing data before a drive is reused, DapuStor SSDs help to prevent the possibility of data leakage or other security breaches.

Overall, the enhanced secure erase feature is another example of the advanced features that DapuStor SSDs offer, making them a popular choice for users who require high-performance storage solutions with robust data security and privacy features.

7. **SR-IOV support (virtual environment support).**

DapuStor SSDs come with SR-IOV (Single Root Input/Output Virtualization) support, which can provide high performance and low latency in virtual compute environments.

SR-IOV is a technology that allows a single physical device, such as an SSD, to appear as multiple virtual devices to virtual machines. This can provide several benefits in virtualized environments, including improved performance, reduced latency, and better resource utilization.

By supporting SR-IOV, DapuStor SSDs can help to reduce the overhead associated with virtualization, allowing virtual machines to access the SSD directly and achieve high levels of performance and low latency. This can be particularly useful in scenarios where high-performance storage is required, such as in database servers or high-performance computing environments.

Overall, the SR-IOV support in DapuStor SSDs is another example of the advanced features that these SSDs offer, making them a popular choice for users who require high-performance and low-latency storage solutions in virtualized environments.

Figure 4 DapuStor eSSD Storage Drive, and high performance ASIC

TERMINOLOGIES OF A CEPH CLUSTER

There are three services that form the backbone of the cluster [5]:

- **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 elements 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 [6].

HARDWARE CONFIGURATION

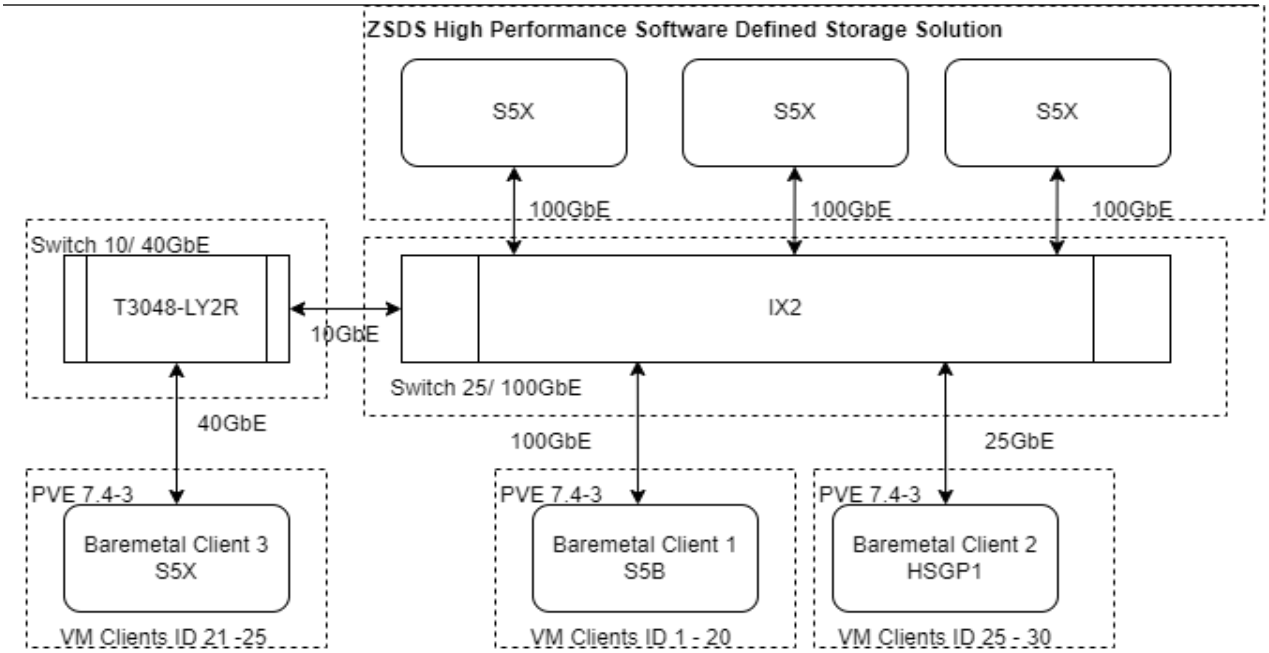


Figure 5 Hardware Architecture

Server	Number of nodes	CPU	RAM	NIC Mezz	Storage card	PCIe NIC	Object Storage Drives	OS
QuantaGrid D53X-1U (S5X) [4]	3	Intel Xeon 4316 x 2	32/2933 MHz x8 units	ConnectX 10/25G (active)	Null	ConnectX 100 G (active)	DapuStor R5100 (7 TiB) x 6.	Ubuntu 20.04 (5.15.0-58-generic)

DEPLOYMENT

Hardware Deployment

There are a few hardware requirements for Ceph that need to be considered while deploying Ceph/ DapuStor appliance.

CEPH REQUIREMENTS

The following guidelines should be referenced when selecting hardware for a Ceph Pacific (16.2.9) 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
	Network	1x 1GbE+ NICs

Installation of Software components

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

- Installation of operating system
- Installing the prerequisites for Ceph Pacific (16.2.9)
- Installation of Ceph

By the end of this document, we'll have implemented the following key services/ daemons to build our ZSD appliance

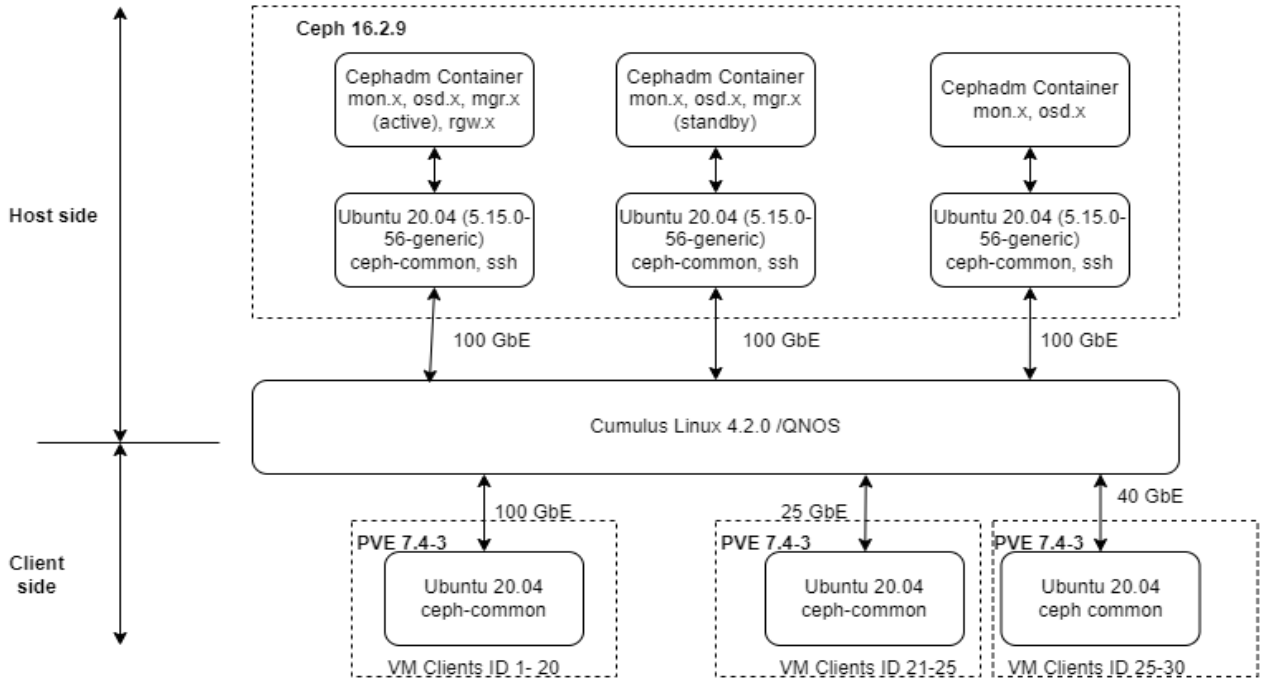


Figure 6 Key services/ daemons in Ceph

Installation of operating system

We will begin by installing all the nodes (minimum of 3) with Ubuntu 20.04 [7].

Preparing servers

While the server restarts after installation of the operating system, access the BIOS on each node and set the CPU to performance mode at Socket Configuration -> Pwr and Perf Profile -> High Performance. (Might change depending on the hardware manufacturer)

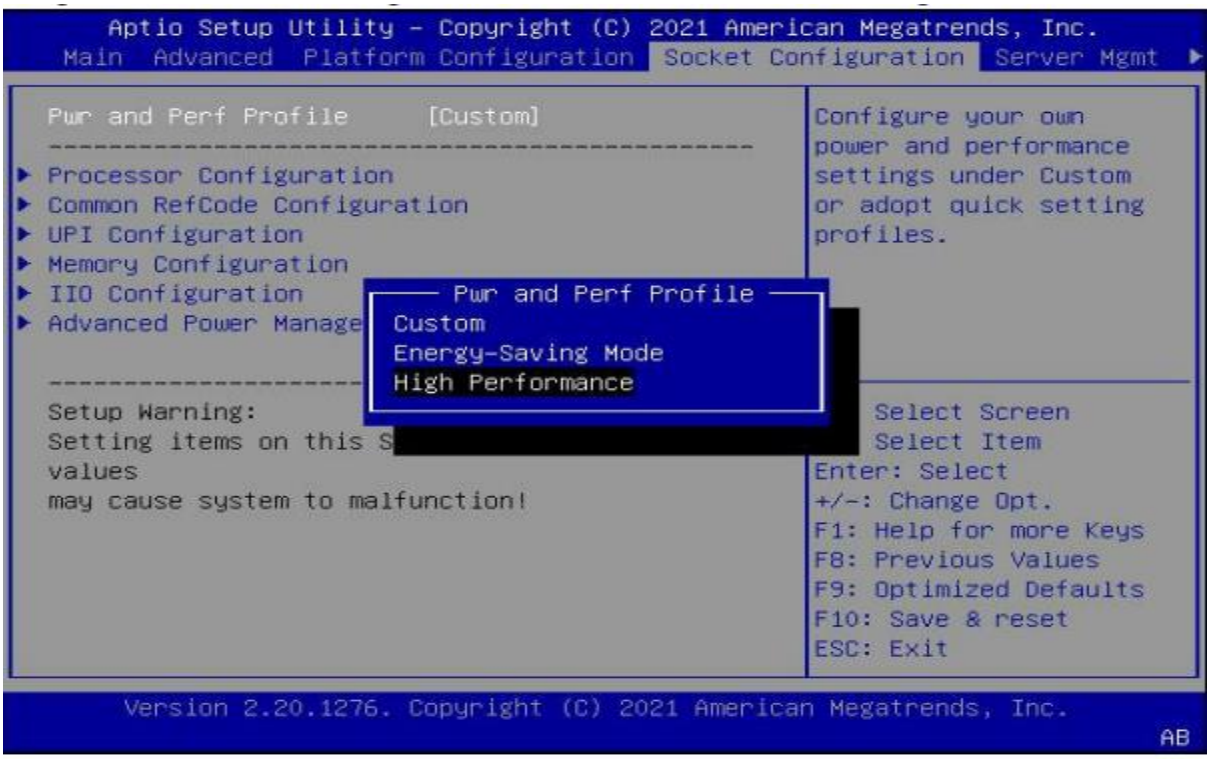


Figure 7 BIOS CPU Performance profile

Installation of prerequisites for Ceph

In this version of ZSDS we'll be using 3 nodes (QuantaGrid D53X-1U) in a cluster configuration to create the overall 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 not recommended as the disk's Input/Output speed might limit the object storage drive daemons' performance.

Before you deploy Ceph, firewall settings (or other corresponding resources) must be configured to allow traffic on 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 software packages must be installed within the operating system environment (Ubuntu 20.04) of every node in the Ceph storage cluster [8]:

- Python 3
- Systemd
- Podman or Docker for running containers [9]
- 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 access the clients and nodes as a root user. Ceph also relies on seamless secure shell connection for autonomous cluster management, so we must create a private-public key pair and place it on every host that is to be included in the cluster to enable password-less access between them [10]. In this deployment method (cephadm orchestrator), the first node of the cluster is considered to be the 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 [8]:

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 [6]:

- 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 of our appliance is shown:

```
root@cephnvme-QuantaGrid-D53X-1U-1S5X2000079:~# ceph -s
  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 2d)
    mgr:         cephnvme-QuantaGrid-D53X-1U-1S5X2000079.iytztt(active, since 2d), standbys:
cephnvmetwo-QuantaGrid-D53X-1U-1S5X2000079.vqxxwd
    mds:         1/1 daemons up, 1 standby
    osd:         18 osds: 18 up (since 20h), 18 in (since 20h)
    rbd-mirror:  1 daemon active (1 hosts)
    rgw:         1 daemon active (1 hosts, 1 zones)
    tcmu-runner: 1 portal active (1 hosts)

  data:
    volumes: 1/1 healthy
    pools:   14 pools, 401 pgs
    objects: 419.02k objects, 1.6 TiB
    usage:   2.9 TiB used, 123 TiB / 126 TiB avail
    pgs:     401 active+clean

  io:
    client:   237 MiB/s rd, 5.9 MiB/s wr, 2.10k op/s rd, 1.73k op/s wr

#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]
```

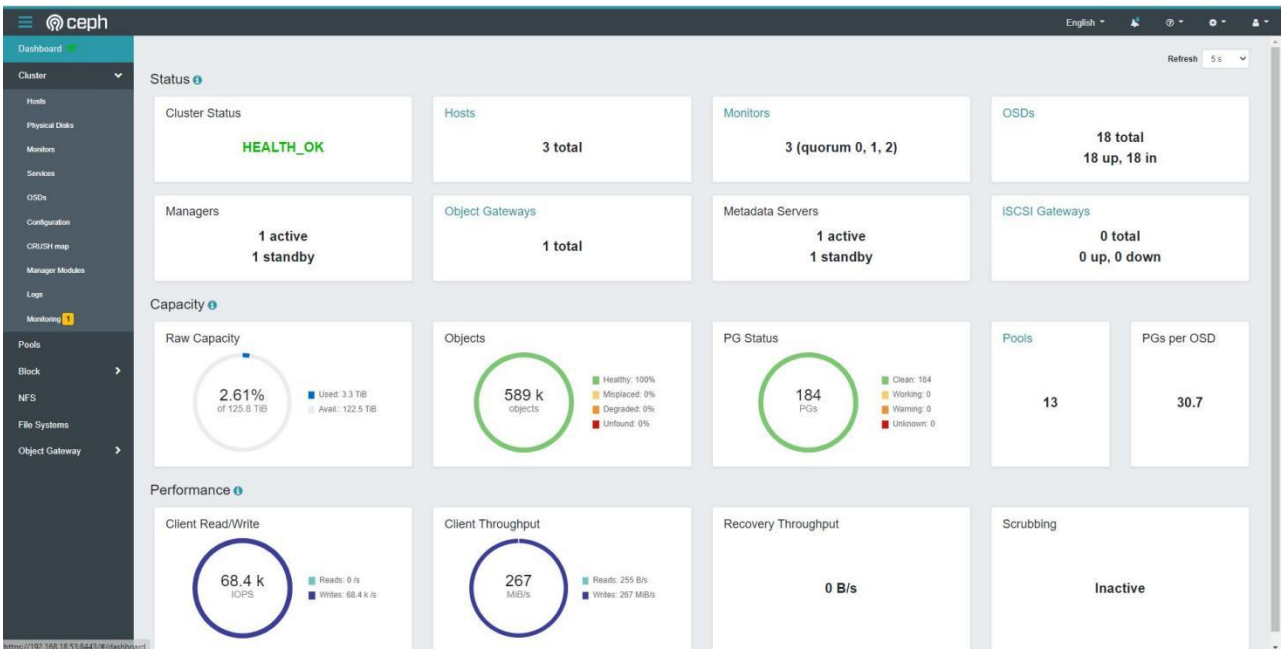


Figure 8 Ceph Dashboard

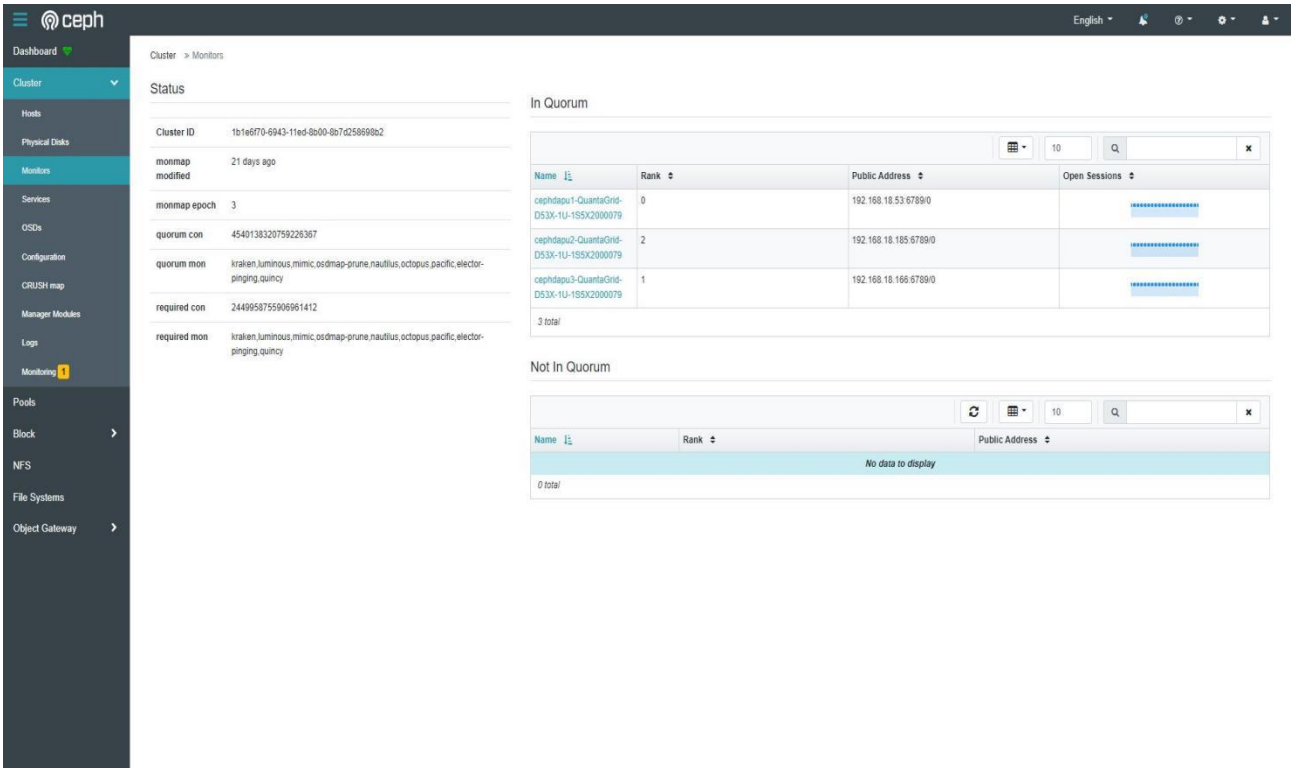


Figure 9 Monitors in Ceph cluster

CONFIGURE THE APPLIANCE

Ceph (16.2.9) offers three types of storage to its users. These are: 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
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" (错误!未找到引用源。).

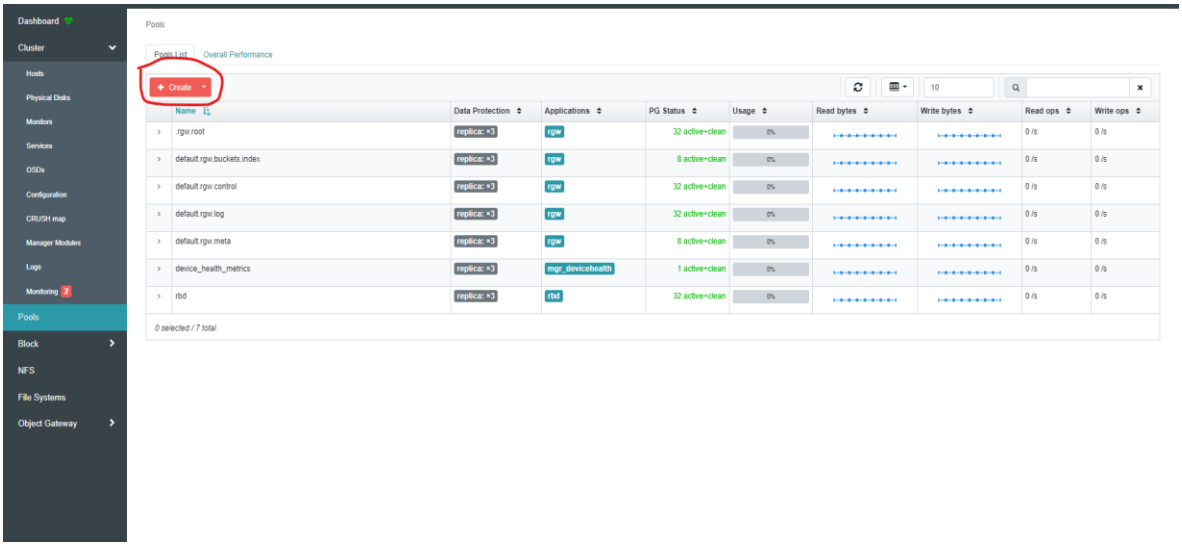


Figure 10 Interface to view / create pools

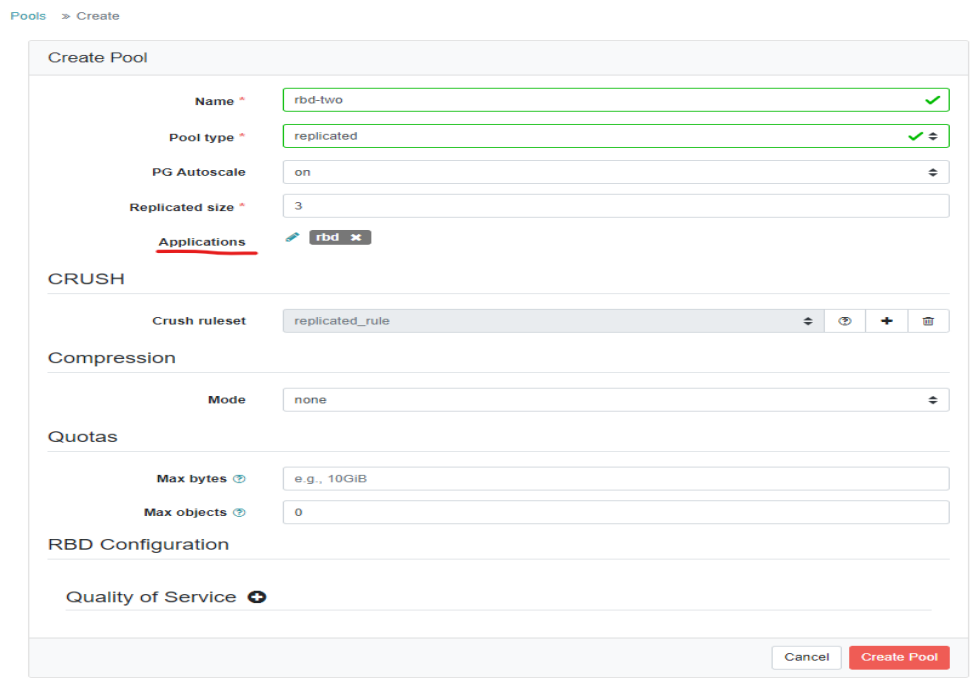


Figure 11 "Create Pool" form

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
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"

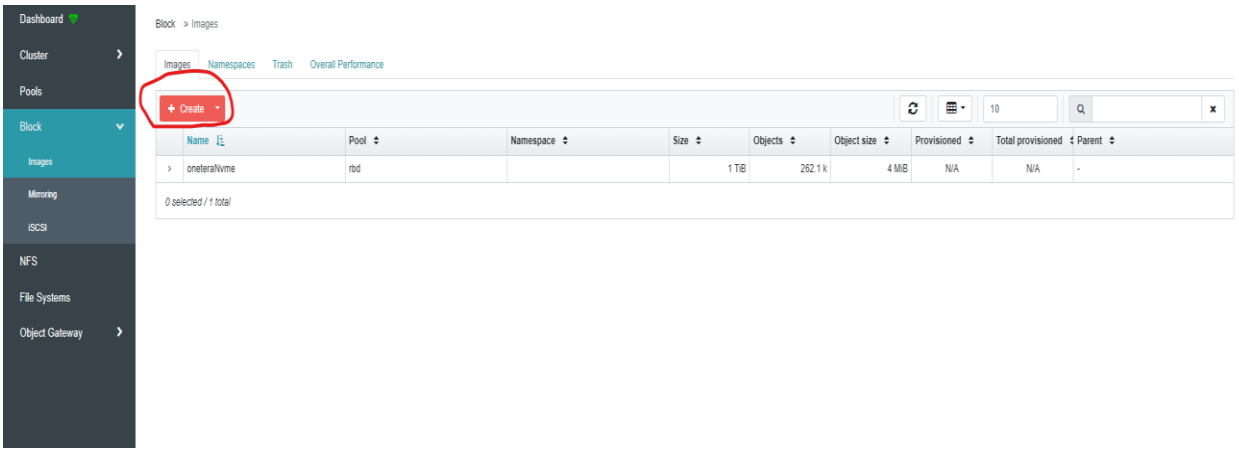


Figure 12 Interface to view/ create block image

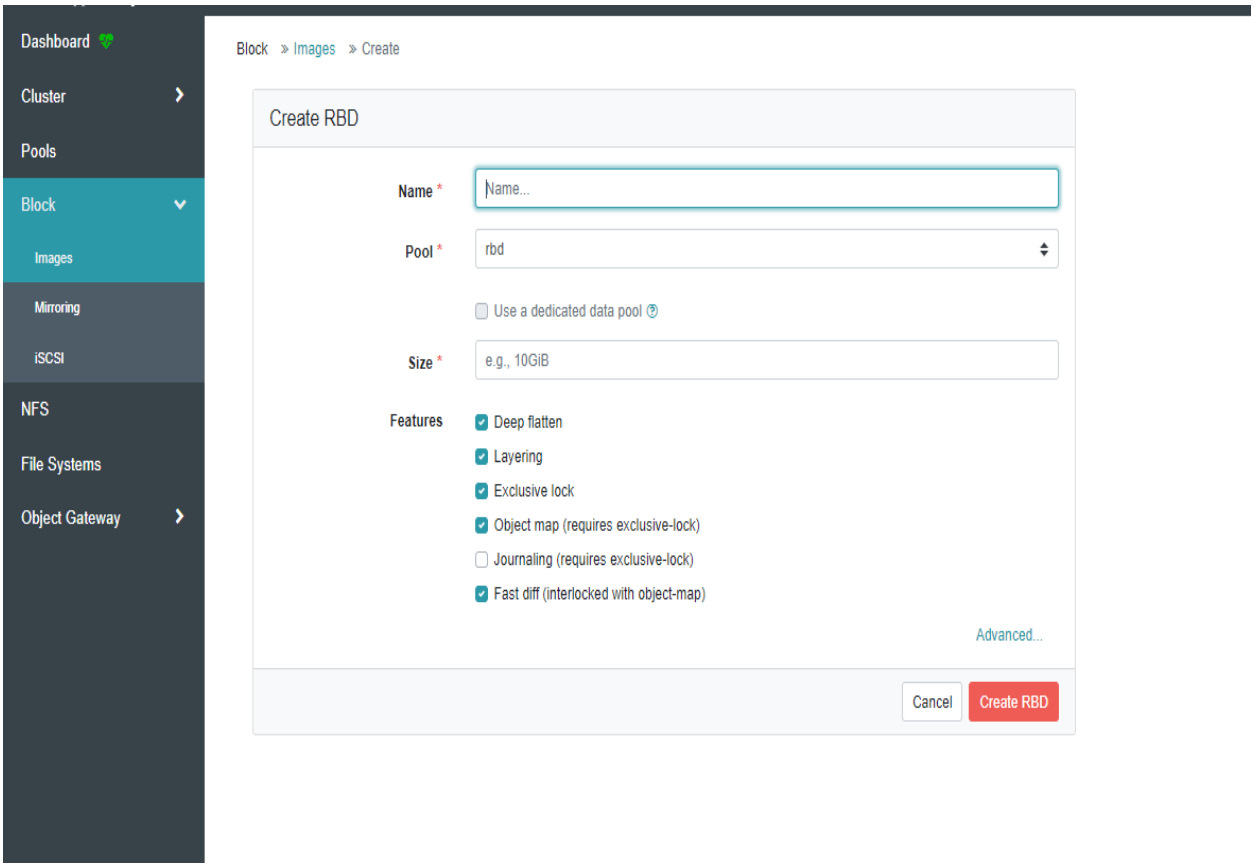


Figure 13 Create block image

In order to map the image of the block device to a client, we need to execute the following commands in the client's terminal. Please note that `ceph.conf` and `ceph.client.admin.keyring` are needed 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 [11] with an SSL certificate, access one of the monitor nodes via ssh and performing the following actions:

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) [12] [13] of object gateway.
6. Upon rebalancing, In the ceph dashboard select `Cluster -> Services -> rgw.<gateway-name> -> Edit`
7. In the pop-up window 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.

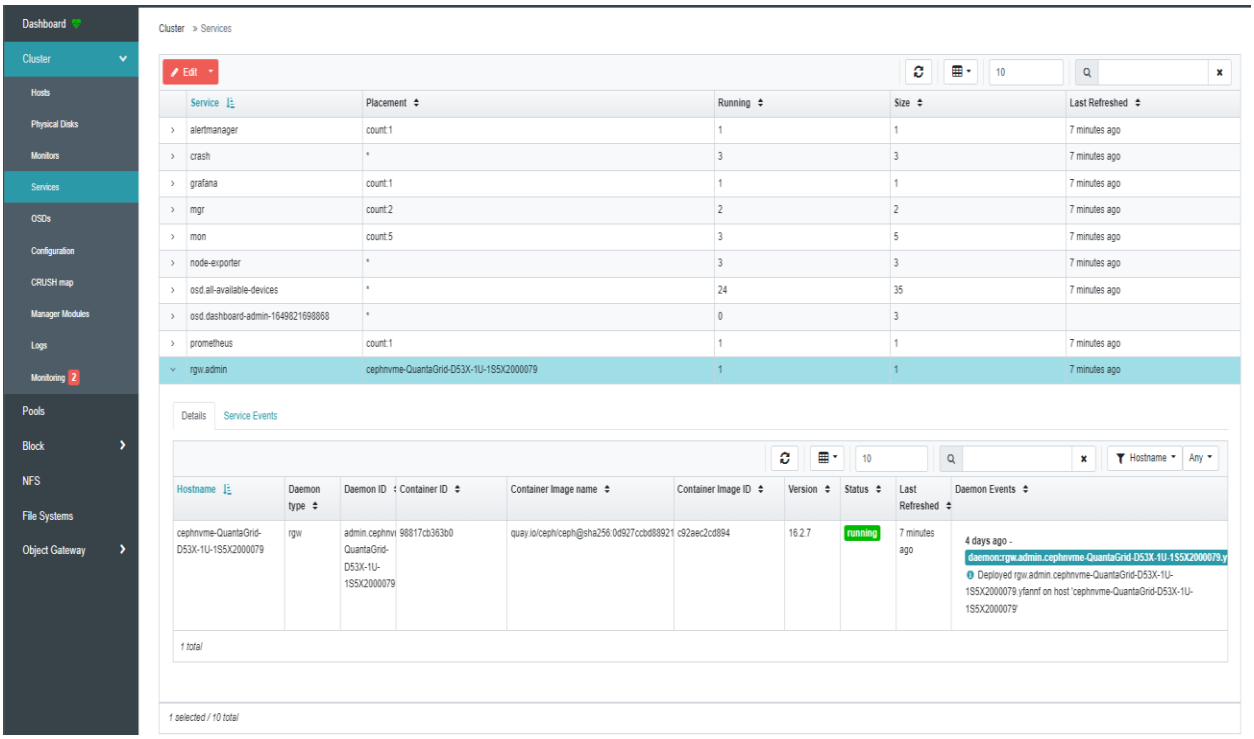


Figure 14 Edit rgw. <service-name>

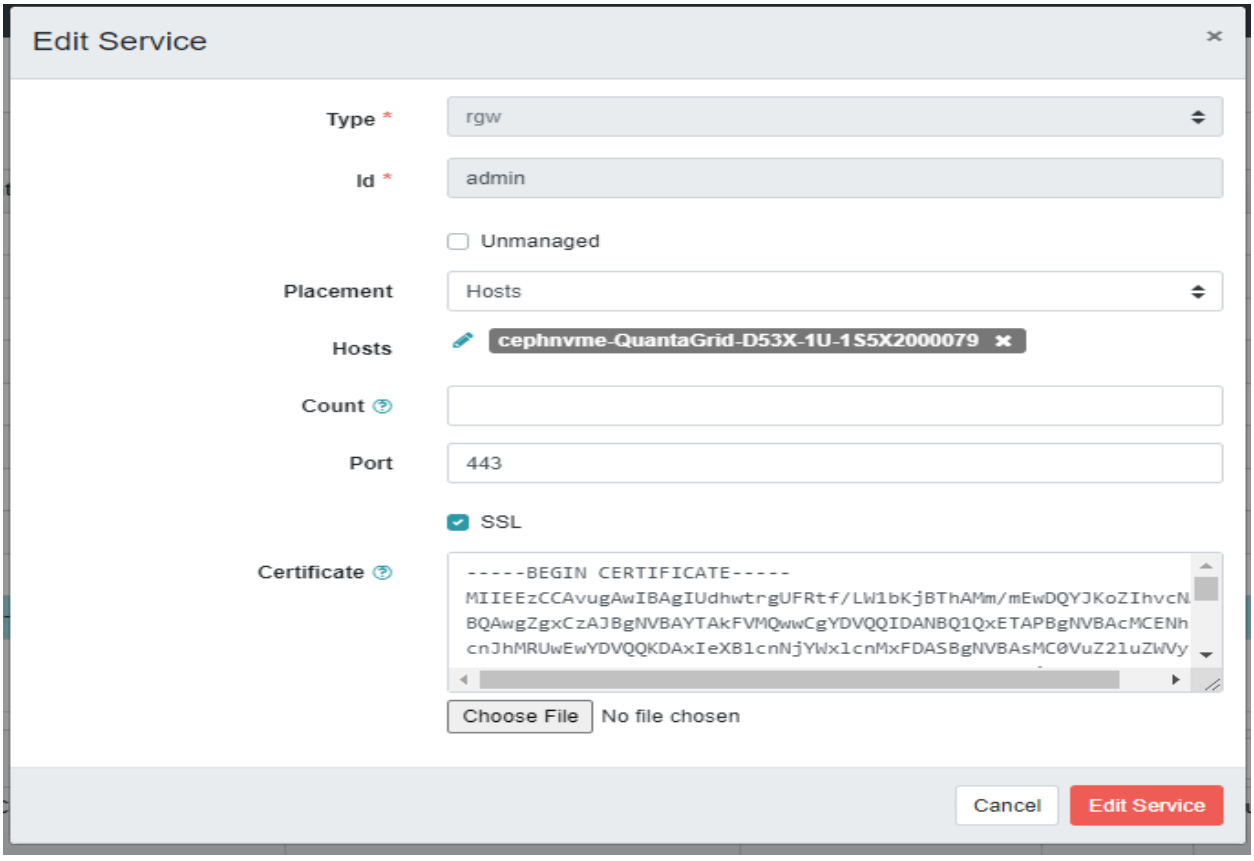


Figure 15 RGW Service edit pop-up.

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 a bucket for the object gateway:

1. In the dashboard navigate to `Object gateway -> Buckets -> Create` (**错误! 未找到引用源。**)
2. In the Create bucket panel, configure bucket name, owner and placement target to create the bucket.

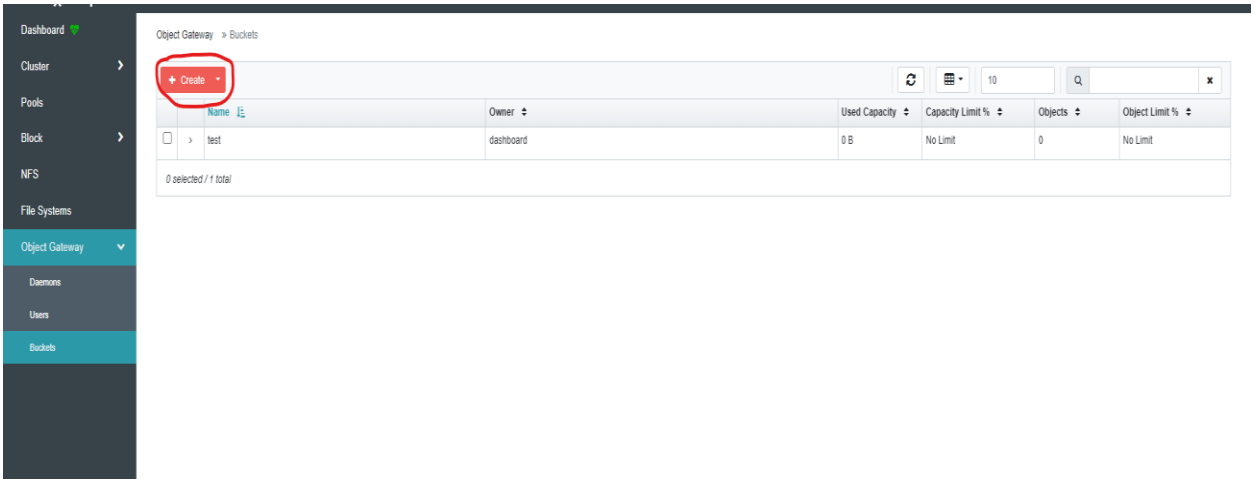


Figure 16 Interface to view/ create buckets.

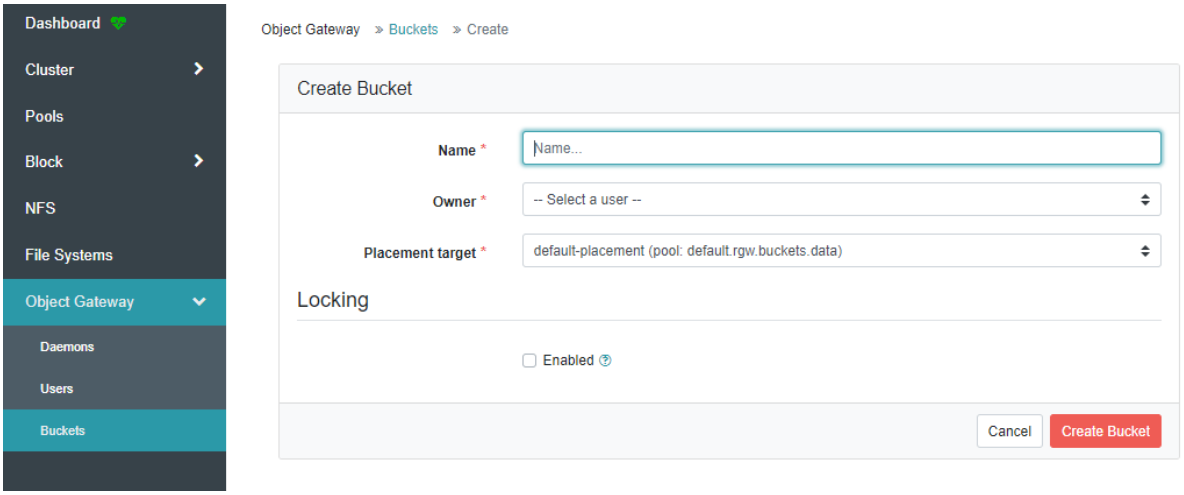


Figure 17 Create buckets.

To access a specific bucket:

1. Execute `radosgw-admin user info --uid=<user-name>`, in any of the monitor nodes
2. Note the access and secret key to the user that the bucket belongs to.
3. In postman, configure the ip `https://<placement-host-name-ip>:443/<bucket-name>`, secret key private key and type of bucket as S3.

.

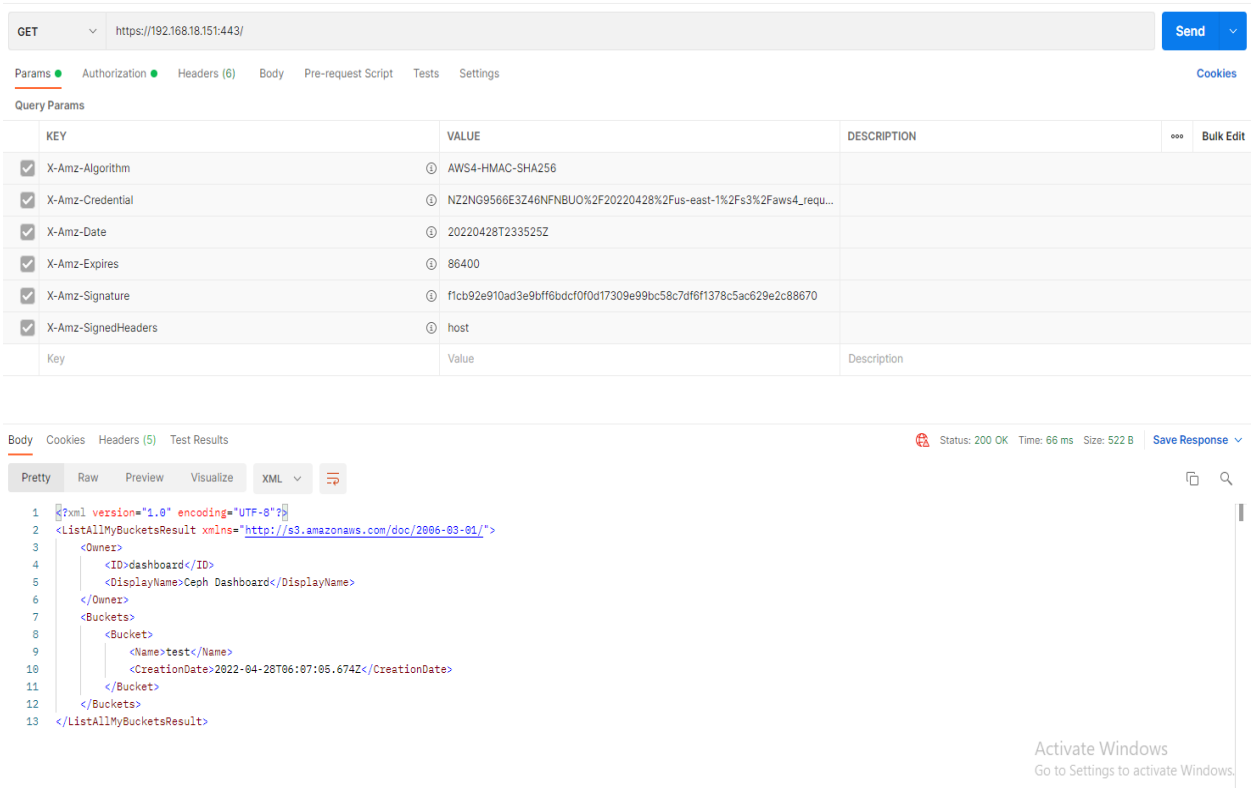


Figure 18 Generic object gateway response with access and secret keys

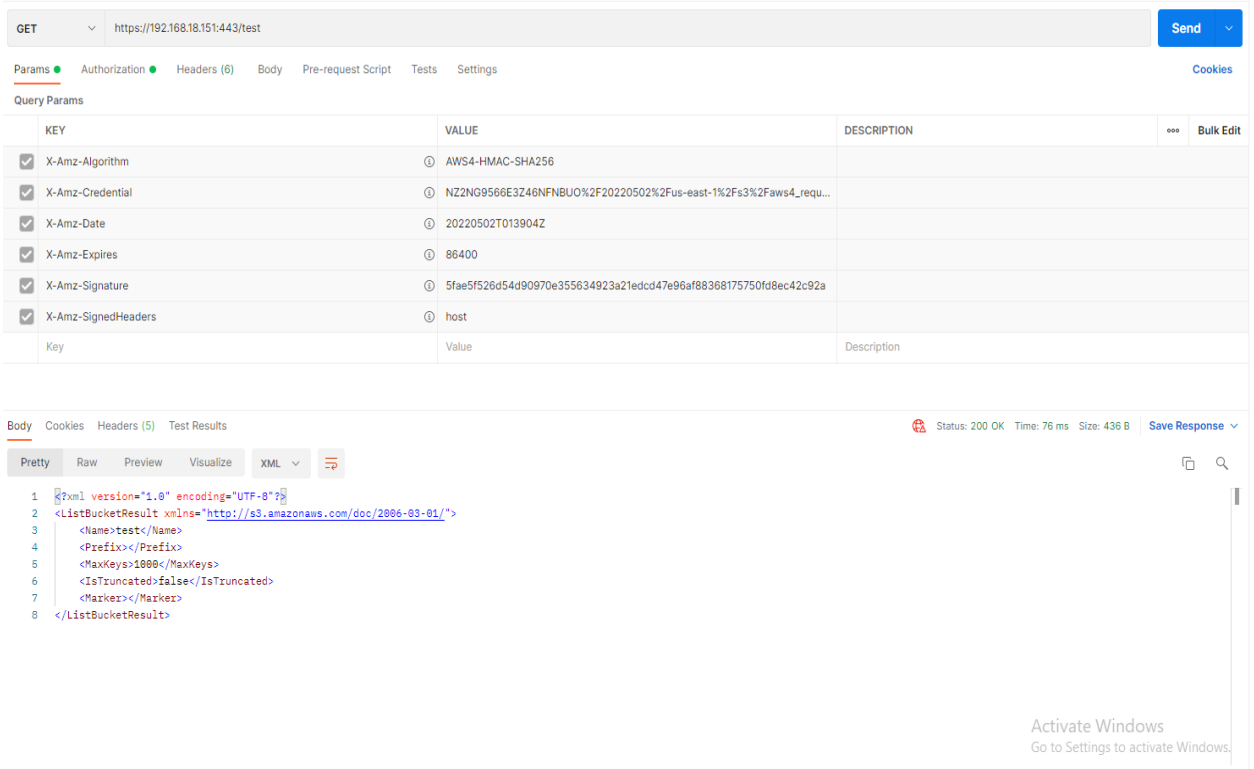


Figure 19 Accessing a specific bucket in the object gateway.

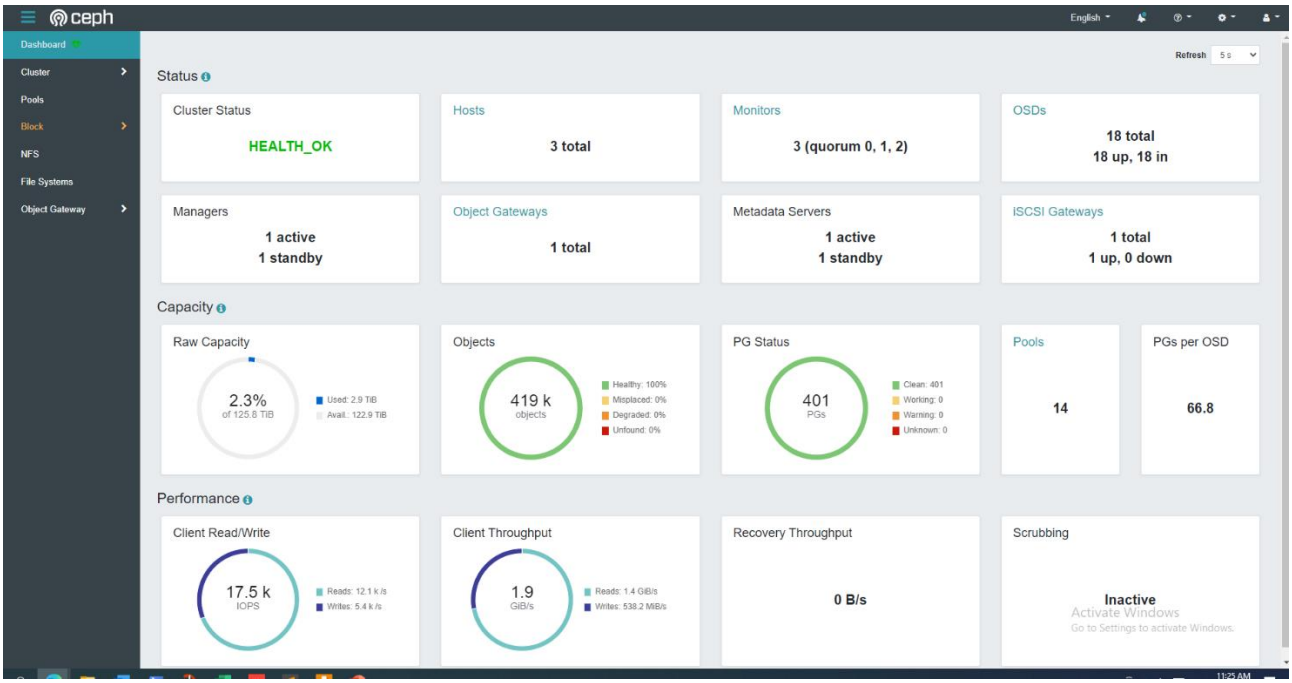


Figure 20 Ceph Dasboard after object gateway deployment

UPDATING THE APPLIANCE

Adding a host

To add a host to the cluster, execute the following commands (lines 1-10) 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
```

TESTING THE APPLIANCE

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

Testing block device with one client node with 100Gb/s network

While testing the block device sequential and random read write performances, we used the FIO tool [14] to perform these tests by modifying the commands from [15] and [16]. The client was mapped with the block device image by following the instructions at 23.

Sequential read

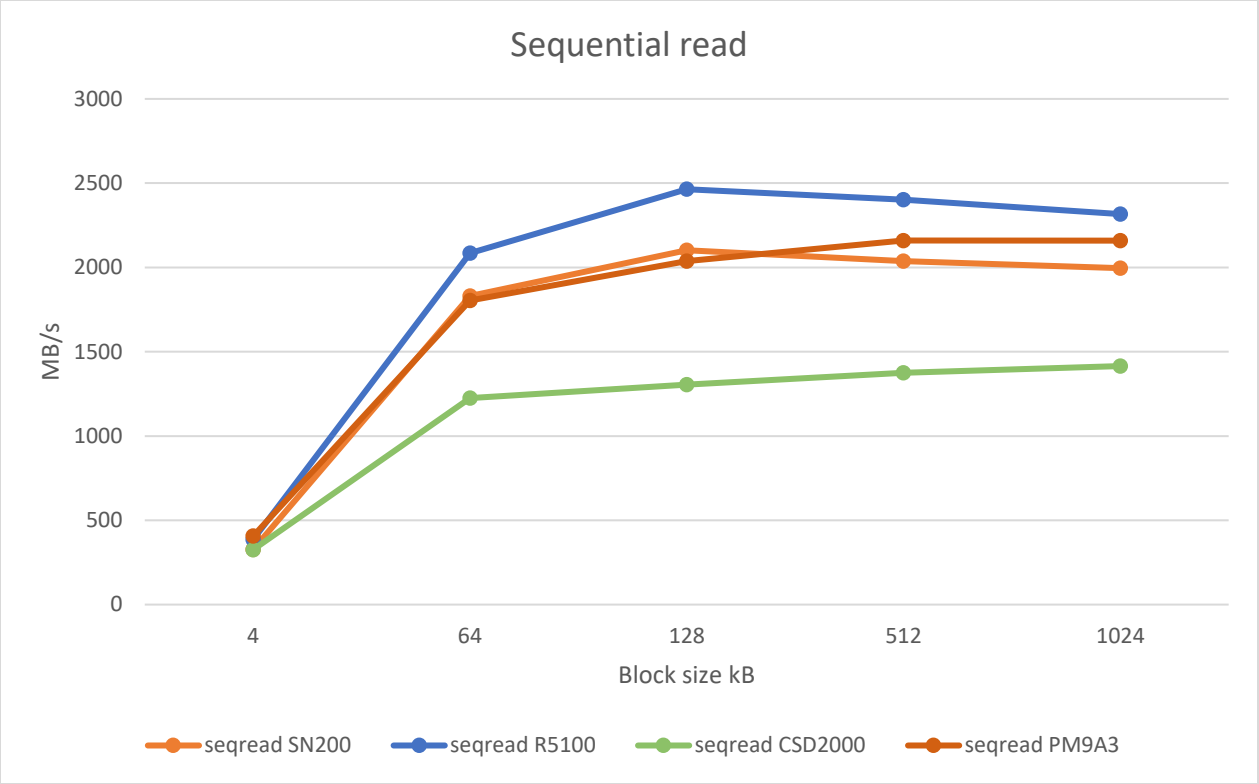
```
fio --name=io-test --ioengine=libaio --iodepth=64 --rw=read --bs=<block-size> --numjobs=32 --
direct=1 --time_based --runtime=120 --group_reporting --filename /dev/rbd0
```

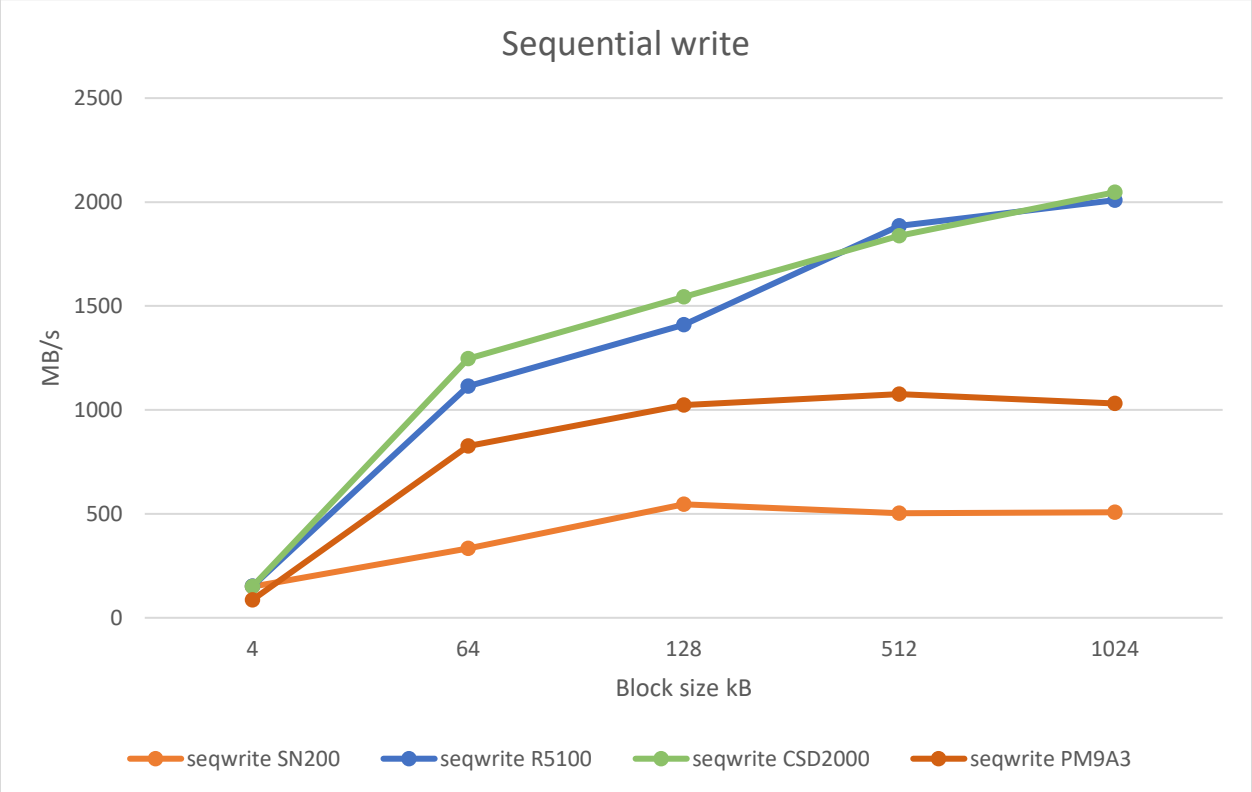
Sequential write

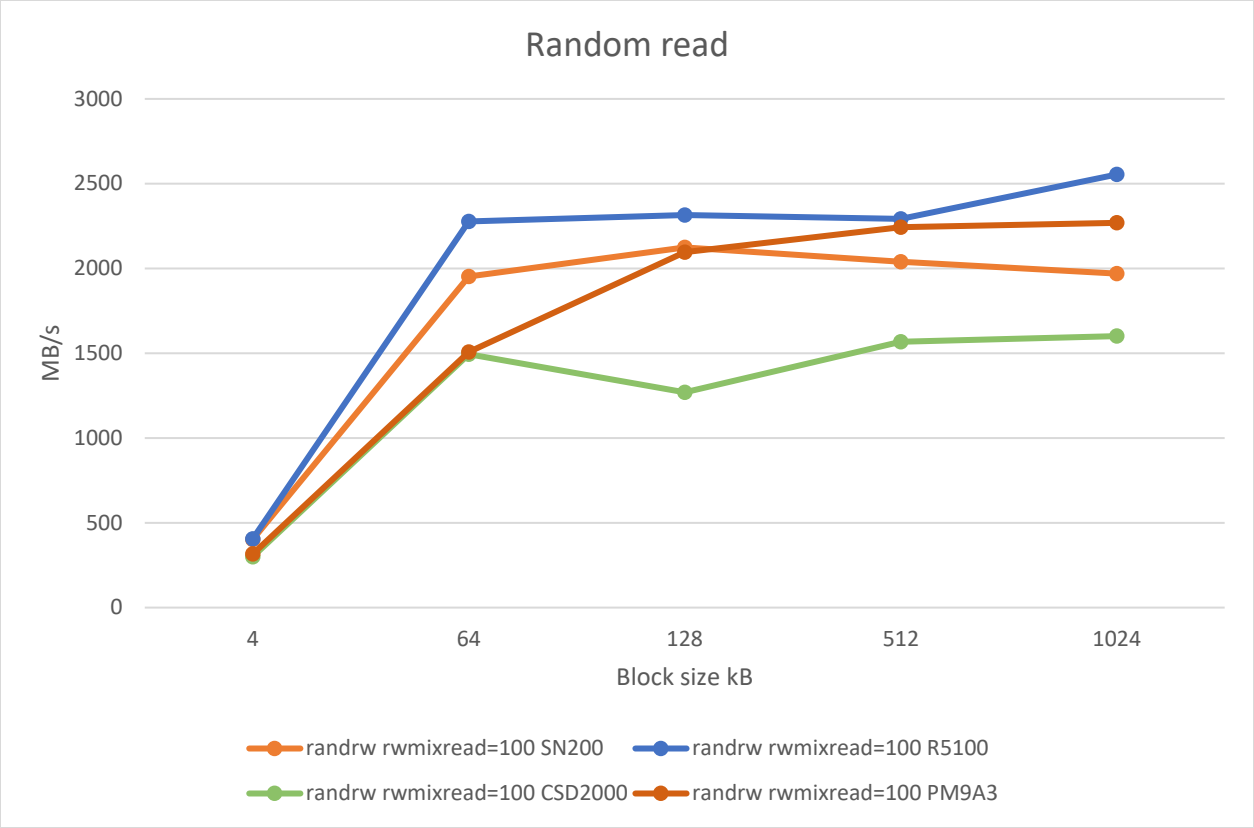
```
fio --name=io-test --ioengine=libaio --iodepth=64 --rw=write --bs=<block-size> --numjobs=32 --
direct=1 --time_based --runtime=120 --group_reporting --filename /dev/rbd0
```

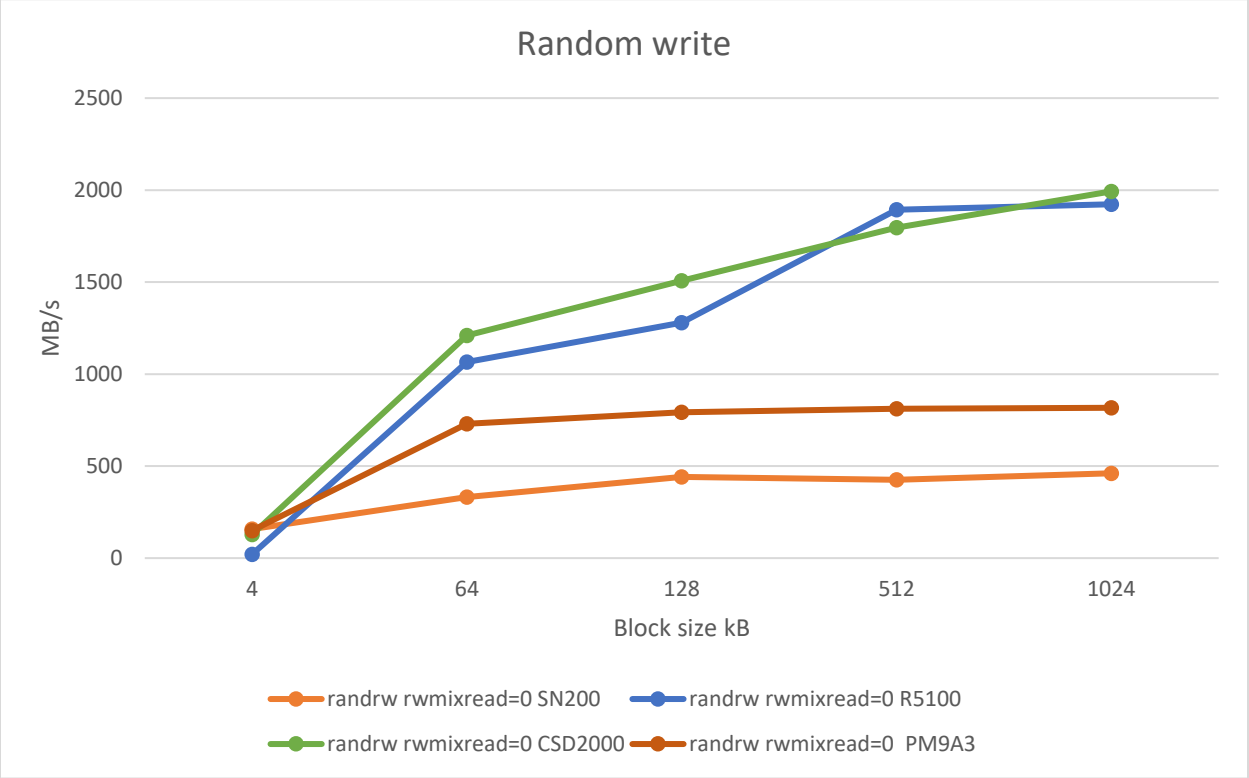
Random read-write

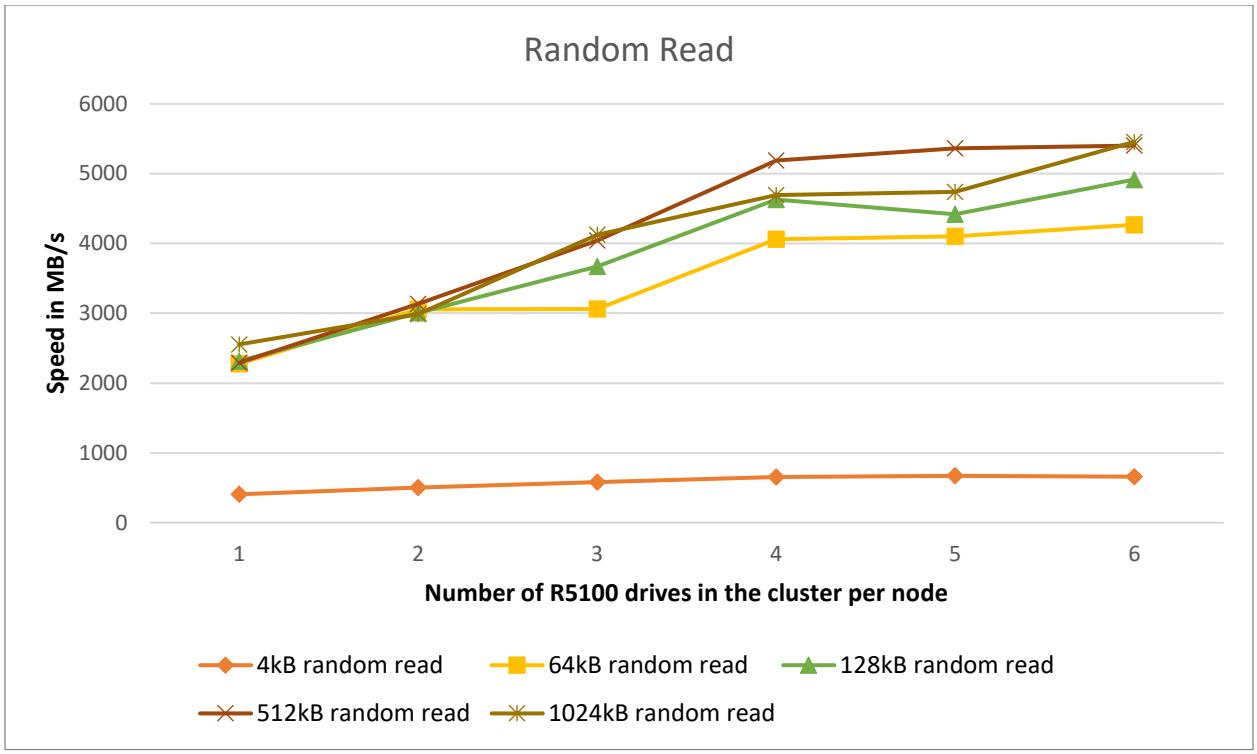
```
fio --name=io-test --ioengine=libaio --iodepth=64 --rw=randrw --rwmixread=<percentage-of-read> --
bs=<block-size> --numjobs=32 --direct=1 --time_based --runtime=120 --group_reporting --filename
/dev/rbd0
```


Sequential read
3000
2500
2000
1500
1000
500
0
MB/s
4
64
128
512
1024
Block size kB
seqread SN200
seqread R5100
seqread CSD2000
seqread PM9A3

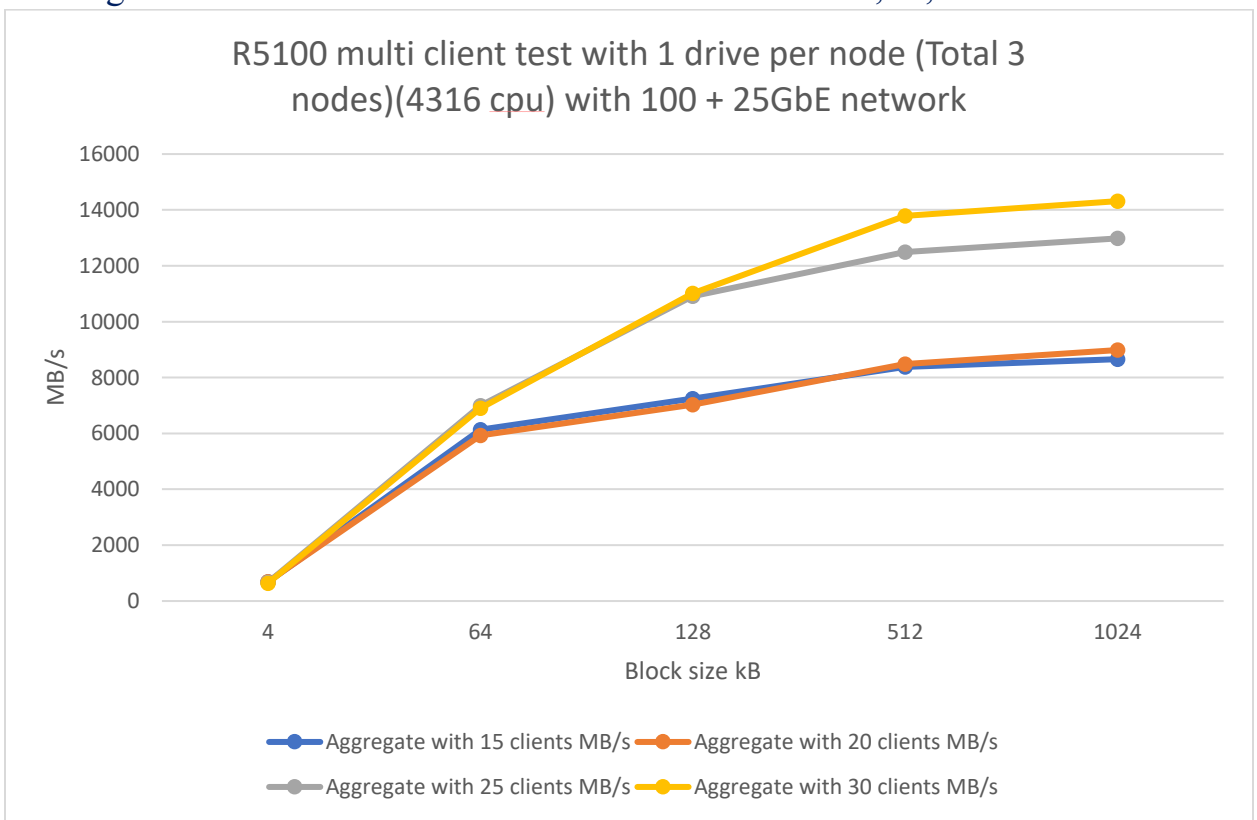

Sequential write
2500
2000
1500
1000
500
0
MB/s
4
64
128
512
1024
Block size kB
seqwrite SN200
seqwrite R5100
seqwrite CSD2000
seqwrite PM9A3


Random read
3000
2500
2000
1500
1000
500
0
MB/s
4
64
128
512
1024
Block size kB
randrw rwmixread=100 SN200
randrw rwmixread=100 R5100
randrw rwmixread=100 CSD2000
randrw rwmixread=100 PM9A3


Random write
2500
2000
1500
1000
500
0
MB/s
4
64
128
512
1024
Block size kB
randrw rwmixread=0 SN200
randrw rwmixread=0 R5100
randrw rwmixread=0 CSD2000
randrw rwmixread=0 PM9A3



Testing block device with three client nodes with 100,40,25GbE network



Test results with Direct parameter in FIO

These tests are done with aggregated 100Gb/s network links in Ceph cluster with following FIO command.

fio --name=io-test --ioengine=libaio --iodepth=64 --rw=<test-name > --bs=<block-size> --numjobs=32 --time_based --runtime=120 --direct=1 --group_reporting --filename /dev/rbd0

The results of Ceph with 6 x R5100 7TiB per node on **10 GbE client baremetal** are tabulated below with varying test results. By using SSDs R5100 dapustor in our Ceph cluster, we were able to achieve consistent sequential read, sequential write and random read speeds with **1168 MB/s, 1146 MB/s, 1174 MB/s** respectively.

Table 1 Ceph with 6 x R5100 per node 7T at 100 GbE client

block size	Seqread (MB/s)	seqwrite (MB/s)	randrw rwmixread=0 (MB/s)	randrw rwmixread=25 (MB/s)	randrw rwmixread=50 (MB/s)	randrw rwmixread=75 (MB/s)	randrw rwmixread=100 (MB/s)
4kB	872	1061	152	44.8 | 135	121 | 121	221 | 73.7	348
64kB	1153	1137	1026	349 | 1047	838 | 839	1036| 346	1139
128kB	1161	1141	1087	355 | 1066	934 | 934	1123 | 374	1171
512kB	**1168**	1145	1110	363 | 1090	984 | 985	1166 | 391	1173
1024kB	1159	**1146**	1123	364 | 1097	968 | 973	1158 | 387	**1174**

The results of Ceph with 1 x R5100 7TiB per node on **100 GbE client baremetal** are tabulated below with varying test results. By using SSDs R5100 dapustor in our Ceph cluster, we were able to achieve consistent sequential read, sequential write and random read speeds with **2464 MB/s, 2009 MB/s, 2554 MB/s** respectively.

Table 2 Ceph with 1 x R5100 per node 7T at 100 GbE client

block size	Seqread (MB/s)	seqwrite (MB/s)	randrw rwmixread=0 (MB/s)	randrw rwmixread=25 (MB/s)	randrw rwmixread=50 (MB/s)	randrw rwmixread=75 (MB/s)	randrw rwmixread=100 (MB/s)
4kB	386	152	20.8	56.1 | 168	161 | 161	269| 89.7	406
64kB	2085	1114	1066	344 | 1032	946 | 947	1585 | 529	2277
128kB	**2464**	1409	1279	432 | 1296	1145 | 1146	1847 | 616	2316
512kB	2402	1886	1894	615 | 1843	1448 | 1450	2013 | 672	2292

1024 kB	2316	**2009**	1923	610 | 1834	1447 | 1452	1977 | 660	**2554**

The results of Ceph with 3 x R5100 7TiB per node on **100 GbE client baremetal** are tabulated below with varying test results. By using SSDs R5100 dapustor in our Ceph cluster, we were able to achieve consistent sequential read, sequential write and random read speeds with **3277 MB/s, 4261 MB/s, 3133 MB/s** respectively.

Table 3 Ceph with 3 x R5100 per node 7T at 100 GbE client

block size	Seqread (MB/s)	seqwrite (MB/s)	randrw rwmixread=0 (MB/s)	randrw rwmixread=25 (MB/s)	randrw rwmixread=50 (MB/s)	randrw rwmixread=75 (MB/s)	randrw rwmixread=100 nm (MB/s)
4kB	488	224	245	67.2 | 202	157 | 157	292 | 97.5	504
64kB	2610	2507	2600	700 | 2100	1311| 1313	2119| 708	3059
128kB	3007	3129	3003	824| 2473	1704 | 1706	2411 | 805	3005
512kB	**3277**	3748	3638	1149 | 3446	2015 | 2018	2613 | 873	**3133**
1024 kB	3062	**4261**	4332	1275 | 3824	1959| 1963	2613 | 873	2987

The results of Ceph with 6 x R5100 7TiB per node on 100 GbE client baremetal are tabulated below with varying test results. By using SSDs R5100 dapustor in our Ceph cluster, we were able to achieve consistent sequential read, sequential write and random read speeds with **5343 MB/s, 6235 MB/s, 5455 MB/s** respectively.

Table 7 Ceph with 6 x R5100 per node 7T at 100 GbE client

block size	Seqread (MB/s)	seqwrite (MB/s)	randrw rwmixread=0 (MB/s)	randrw rwmixread=25 (MB/s)	randrw rwmixread=50 (MB/s)	randrw rwmixread=75 (MB/s)	randrw rwmixread=100 (MB/s)
4kB	557	171	136	46.0 | 141	123 | 123	266 | 88.8	661
64kB	4194	3226	3246	1016 | 3049	1923 | 1925	3317 | 1107	4268
128kB	4830	4276	3990	894 | 2681	2236 |2239	4175 | 1394	4916
512kB	5253	6009	5879	1862 | 5585	3645| 3647	4994| 1666	5403
1024 kB	**5343**	**6235**	6108	1920| 5755	4168| 4171	4807 | 1605	**5455**

In later parts of testing, we striped 5 drives to one LVM per node and attached to the cluster achieving speed up to 12.5GB/s with two 100GbE, one 25 GbE client testing using the following command. (Results are in addendum)

```
fio --name=io-test --ioengine=rbd --iodepth=128 --rw=read --bs=128k --numjobs=32 --time_based --
runtime=120 --direct=1 --group_reporting --pool rbd --rbdname <rbd-image-name>
```

Testing block device with client nodes

To coherently run tests on three machines, we used FIO to perform tests on the clients. In these tests, all the client machines were running Ubuntu 20.04. Out of the three clients, one client was connected using 100Gb/s, one client was connected using a 25 Gb/s network interface and one client was connected using 40 Gb/s network interface. All the clients were mapped with the different block device (`rbd0`) by following the instructions at 23.

Using link aggregation (combining two or more network interfaces to function as a single link), we were able to aggregate links on client systems.

To aggregate links [17], please execute the following commands in every relevant host/client:

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
```

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*). While erasure coding has been available for object storage for some time now, support for block storage is still under active development and is 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 ZSDS. 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 ZSDS, minimum *k* is 2 for all available plugins. An overview of how data is stored in erasure coded pool and a replicated pool able to tolerate 3 concurrent object storage drive failure is shown below (Figure).

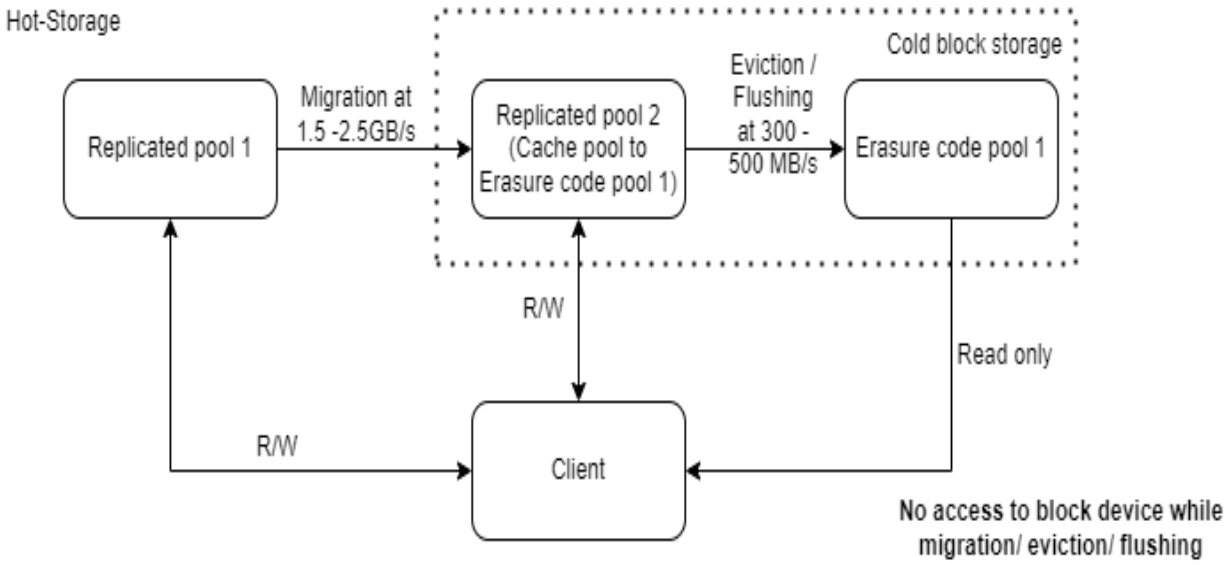


Figure 27 Cold storage overview in ZSDS

To tolerate 3 concurrent failures

Data size = 11.2 TiB

Data size does not represent drive raw/usable size

Data/ Parity/ Replica chunks represent physical drive

Erasure Coded Pool

k= 2; m =3

5.6 TiB
5.6 TiB
Data
5.6 TiB
5.6 TiB
5.6 TiB
Parity

28 TiB stored to save 11.2TiB

Replicated pool

replica = 3

11.2 TiB
Data
11.2 TiB
11.2 TiB
11.2 TiB
Replica

44.8 TiB stored to save 11.2 TiB

Figure 28 Erasure coded pool vs Replicated pool

To create erasure coded block pools with cache tiering:

```
# Erasure Coding
ceph osd pool create <erasure-coded-pool-name> PG_NUM PGP_NUM erasure default
# PG numbers depend on values of k and m values related to erasure coding
ceph osd pool create <cache-pool-name> PG_NUM PGP_NUM replicated
ceph osd tier add <erasure-coded-pool-name> <cache-pool-name> --force-nonempty
ceph osd tier cache-mode testpool writeback
ceph osd pool set <cache-pool-name> hit_set_type bloom
ceph osd tier set-overlay <erasure-coded-pool-name> <cache-pool-name>
# to set up auto-eviction
ceph osd pool set {cachepool} target_max_objects {#objects}
ceph osd pool set {cachepool} target_max_bytes {#bytes}

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 [18]. The test was conducted on a 100 Gbps client with warp client accessing the host. The command used to test the object gateway was as follows:

```
warp mixed --host 192.168.18.115:443 --access-key=NZ2NG9566E3Z46NFNBUO --secret-key=<secret-key> --
autoterm --tls --insecure
```

```
root@cephosd4-QuantaPlex-T41S-2U:/home/cephosd4/Downloads# warp mixed --host 192.168.18.151:80 --access-key=NZ2NG9566E3Z46NFNBUO
Throughput 105.5 objects/s within 7.500000% for 12.467s. Assuming stability. Terminating benchmark.
warp: Benchmark data written to "warp-mixed-2022-05-24[150016]-dI3q.csv.zst"
Mixed operations.
Operation: DELETE, 10%, Concurrency: 20, Ran 44s.
 * Throughput: 33.52 obj/s

Operation: GET, 45%, Concurrency: 20, Ran 44s.
 * Throughput: 1505.56 MiB/s, 150.56 obj/s

Operation: PUT, 15%, Concurrency: 20, Ran 44s.
 * Throughput: 501.48 MiB/s, 50.15 obj/s

Operation: STAT, 30%, Concurrency: 20, Ran 44s.
 * Throughput: 100.21 obj/s

Cluster Total: 2004.18 MiB/s, 333.99 obj/s over 45s.
warp: Cleanup Done.root@cephosd4-QuantaPlex-T41S-2U:/home/cephosd4/Downloads#
```

Figure 29 Speed test - object storage

IMPROVEMENTS AND BUGS

Improvements

librbd

Ceph provides librbd as an alternative method to make use of block devices. librbd is a user-mode library running at the client to communicate to Ceph storage cluster. It is different from the

default method in this document which uses rbd command to create a kernel module at the client to implement block devices.

In a series of random read tests, during which combinations of io depth and block size are enumerated, it is observed that librbd offers higher throughput metrics comparing to kernel module method. The test environment, fio test commands and results can be found in the following tables. The pool's name and image's name in Ceph are rbd and speed-test-image, respectively. And the image is mapped to client as a block device named /dev/rbd0.

Test environment

All the server and client nodes are connected to a network of 100Gb bandwidth.

Server	Number of nodes	CPU	RAM	NIC Mezz	Storage card	PCIe NIC	Object Storage Drives	OS
QuantaGrid D53X-1U (S5X)	3	Intel Xeon 4316 x 2	32/2933 MHz x8 units	N/A	Null	Connect X 100 G (active)	DapuStor R5100 (7 TiB) x 6.	CentOS 8.5.2111

Test with kernel module

./fio --name=io-test --ioengine=libaio --iodepth=<io-depth> --rw=randread --bs=<block-size> --numjobs=16 --time_based --direct=1 --runtime=120 --group_reporting --filename /dev/rbd0

All values of test results are in MB/s

Iodepth / Block size	4 kB	64kB	128kB	512kB	1024kB	2048kB
4	452	3292	3601	3978	4013	4023
8	557	3142	3612	4088	4143	4095

16	568	3167	3618	4127	4140	4121
32	573	3256	3774	4158	4174	4071

Test with librbd

./fio --name=io-test --ioengine=rbd --iodepth=<io-depth> --rw=randread --bs=<block-size> --numjobs=32 --time_based --runtime=300 --direct=1 --group_reporting --clientname=admin --pool=rbd --rbdname=speed-test-image

All values of test results are in MB/s

Iodepth / Block size	4 kB	64kB	128kB	512kB	1024kB	2048kB
4	568	5702	6648	5773	4820	4642
8	574	5652	6340	5385	4706	4561
16	563	5694	6151	5231	4686	4532
32	569	5732	5880	5152	4669	4534

In the above test with librbd, it is observed the cpu utilization reached to almost 100% after the block size is larger than 512K. Therefore, the test results at larger block size may be further improved if a higher performance client node is adopted.

Others

These are the areas of potential testing improvement that were identified across the scope of the appliance:

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

Bugs

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

ADDENDUM

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
```

Network bandwidth testing

Network bandwidth between two nodes consisting of Mellanox ConnectX NICs can be tested using the following commands [19]:

```
22. sudo apt install iperf # on both client and server
23. iperf -s -P 4 #on server
24. iperf -c server-ip -P 4 # on client
```

Commands cheat sheet

The following is a complete set of example commands for appliance installation, and also some additional helpful / frequently used commands for initial appliance configuration:

```
1.  # Assumes a fresh install of Ubuntu OS without updates while installation
2.  # Executed in an elevated termial
3.
4.  #Ceph Pre-requisites Install
5.
6.  apt install ntp
7.  apt install net-tools
8.  apt-get install     ca-certificates     gnupg     lsb-release
```

```
9.  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
10. apt-get install docker-ce docker-ce-cli
11. apt-get update
12. apt-get install docker-ce docker-ce-cli containerd.io
13. apt install openssh-server
14. nano /etc/ssh/sshd_config
15. # Edit the ssh config with PermitRootLogin yes
16. passwd # set/change root password for ssh access
17. ssh-keygen # Generates public-private key pair
18. nano /etc/hosts
19. # Add the hosts and their corrsponding ip address. Ensure hostname matches the actual
    hostname.
20. ssh-copy-id <host-name>
21. #This allows passwordless ssh access
22. apt install lvm2
23.
24. #Ceph Installation
25.
26. #Navigate to any location of interest where you want the "cephadm" file to be placed
27. curl --silent --remote-name --location https://github.com/ceph/ceph/raw/<release-
    name>/src/cephadm/cephadm
28. chmod +x cephadm
29. # For help and available options use "./cephadm --help"
30. ./cephadm add-repo --release <release-name>
31. ./cephadm install
32. cephadm bootstrap --mon-ip <monitor-ip>
33. # 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
34. cephadm install ceph-common # helps in accessing cluster details outside the "cephadm"
    container
35. ssh-copy-id -f -i /etc/ceph/ceph.pub <host-name>
36. ./cephadm prepare-host <host-name>
37. # checks the host for necessary pre-requisites
38. ceph orch host add <host-name>
39. # adds node to the cluster
40.
41. cephadm shell # To access the container shell
42.
43. # To remove a host from the cluster
44.
45. systemctl stop <ceph-osd-service>
46. ceph osd out osd.x
47. ceph osd down osd.x
48. ceph osd rm osd.x
49. ceph osd crush rm osd.x
50. ceph auth del osd.x
51. ceph osd destroy x --yes-i-really-mean-it
52. ceph orch host drain <host-name>
53. # Deactivates monitor and manager services, removes them and updates the monmap of the
    cluster
54. ceph orch host rm <host-name>
55.
56. # To stop adding available OSD to the cluster
57. # 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
58. ceph orch apply osd --all-available-devices unmanaged = true # Stops adding OSD
    automatically into the cluster in any given node
59.
60. # To remove an OSD from the cluster
61.
62. systemctl stop <ceph-osd-service>
63. ceph osd out osd.x
64. 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

# Failed daemons managed by cephadm
ceph orch daemon rm <daemon-name> --force

# 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 <image-name> --size <pool-size> --image-feature layering -m mon-ip -k
/path/to/ceph.client.admin.keyring -c /path/to/ceph.conf
rbd map <image-name> --name client.admin -m monitor-ip -k /path/to/ceph.client.admin.keyring
-c /path/to/ceph.conf

#rbd-mirror (ONLY JOURNALED IMAGES)
rbd --cluster ceph mirror pool peer add rbdNvme client.rbd-mirror@hot --conf
/etc/ceph/hot.conf --keyring /etc/ceph/hot.client.admin.keyring --remote-mon-host
192.168.18.151,192.168.18.178,192.168.18.180 --remote-key-file /etc/ceph/hot.client.rbd-
mirror.keyring
rbd-nbd map oneteraNvme --name client.admin -m 192.168.18.151 -k
/home/vncserver/ceph.client.admin.keyring

mkfs.ext4 -m0 /dev/rbdX
# Automap block devices on boot . Ensure ceph.conf file to drives that are to mapped are
present at /etc/ceph/ceph.conf
nano /etc/ceph/rbdmap
pool-name/image-name       name=client.admin,keyring=/path/to/ceph.client.admin.keyring
systemctl enable rbdmap

#To map /unmap devices
rbdmap map
rbdmap unmap
# To create cephFS
ceph fs volume create <cephfs-volume-name>

# Network File System
#Prerequisites
# NFS Service/ cephFS services should be up and running
  #Add the client IP to the mount volume
  mount -t nfs -o nfsvers=4.1,proto=tcp <ip-of-nfs-service-running-node>:/<psuedo-path-name>
/path/to/mount/location

  # To remove an existing pool

  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

  # To restore "device_health_metrics" in case of removal of all OSDs

  ceph tell mon.\* injectargs '--mon-allow-pool-delete=true'
  ceph osd pool delete device_health_metrics device_health_metrics --yes-i-really-mean-it
  ceph osd pool delete device_health_metrics device_health_metrics --yes-i-really-really-
mean-it
  ceph device scrape-health-metrics

  # 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
```

```
121.
122.  # To deploy object gateway with ssl
123.
124.  ssh <one-of-monitor-nodes>
125.  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
126.  # Navigate to any desired location
127.  touch nvmeServer.pem
128.  cat /etc/ssl/certs/ceph-rgw.crt >> /home/cephnvme/nvmeServer.pem
129.  cat /etc/ssl/private/ceph-rgw-cert.key >> /home/cephnvme/nvmeServer.pem # concatenate key
    and certificate files
130.  cat nvmeServer.pem # verify that key and certificate files are concatenated
131.  ceph orch apply rgw admin --realm=default --zone=default --placement=<host-name>
132.  # In Ceph Dashboard Cluster -> Services -> rgw.admin -> Edit
133.  # Change port to 443 ; Tick the SSL  box ; Attach the nvmeServer.pem file
134.  ceph dashboard set-rgw-api-ssl-verify False
135.  curl -k https://<placement-host-name-ip>:443 # verify "anonymous" response from the ip
136.  # Verify similar response from the browser
137.  radosgw-admin user info --uid=dashboard # To view the details of the user
138.
139.  #To list active ceph related services
140.
141.   ceph orch ls
142.
143.  # Erasure Coding
144.  ceph osd pool create <erasure-coded-pool-name> PG_NUM PGP_NUM erasure default
145.  # PG numbers depend on values of k and m values related to erasure coding
146.  ceph osd pool create <cache-pool-name> PG_NUM PGP_NUM replicated
147.  ceph osd tier add <erasure-coded-pool-name> <cache-pool-name> --force-nonempty
148.  ceph osd tier cache-mode testpool writeback
149.  ceph osd pool set <cache-pool-name> hit_set_type bloom
150.  ceph osd tier set-overlay <erasure-coded-pool-name> <cache-pool-name>
151.  # to set up auto-eviction
152.  ceph osd pool set {cachepool} target_max_objects {#objects}
153.  ceph osd pool set {cachepool} target_max_bytes {#bytes}
154.
155.  rados -p <cache-pool-name> cache-flush-evict-all # To free up cache
156.
157.
158.  # Purging a cluster
159.  # THIS WILL DESTROY ALL DATA IN THE CLUSTER
160.
161.  ceph mgr module disable cephadm
162.  ceph fsid # note down fsid
163.  #executed in every host
164.  cephadm rm-cluster --force --zap-osds --fsid <fsid>
165.  fio --filename=/mnt/cephNvmeNFS/testcomp50ratio --rw=write --rwmixread=0 --bs=128k  --
    buffer_compress_percentage=50 --ioengine=libaio --iodepth=256 --numjobs=16 --time_based --
    group_reporting --name=iops-test-job --eta-newline=1 --size=1TB
166.
167.
168.  mount -t nfs -o nfsvers=4.1,proto=tcp 192.168.18.180:/nfsTest /mnt/cephNvmeNFS/
169.  fio --filename=/mnt/cephNvmeNFS/testcomp80ratio --rw=write --rwmixread=0 --bs=128k  --
    buffer_compress_percentage=80 --ioengine=libaio --iodepth=256 --numjobs=16 --time_based --
    group_reporting --name=iops-test-job --eta-newline=1 --size=1TB
170.
171.  # Multisite setup
172.
173.  #primary zone config
174.
175.  radosgw-admin realm create --rgw-realm={realm-name} [--default]
176.  radosgw-admin zonegroup create --rgw-zonegroup={name} --endpoints={url} --rgw-
    realm={realm-name} --master --default
177.  radosgw-admin zone create --rgw-zonegroup={zone-group-name} --rgw-zone={primary-zone-name}
    --master --default --endpoints={http://ip-site-1:80}
```

```
#Delete default zone, zone group and their respective pools
radosgw-admin zonegroup delete --rgw-zonegroup=default --rgw-zone=default
radosgw-admin period update --commit
radosgw-admin zone delete --rgw-zone=default
radosgw-admin period update --commit
radosgw-admin zonegroup delete --rgw-zonegroup=default
radosgw-admin period update --commit
ceph osd pool rm default.rgw.control default.rgw.control --yes-i-really-really-mean-it
ceph osd pool rm default.rgw.data.root default.rgw.data.root --yes-i-really-really-mean-it
ceph osd pool rm default.rgw.gc default.rgw.gc --yes-i-really-really-mean-it
ceph osd pool rm default.rgw.log default.rgw.log --yes-i-really-really-mean-it
ceph osd pool rm default.rgw.users.uid default.rgw.users.uid --yes-i-really-really-mean-it
#user creation
radosgw-admin user create --uid="{user-name}" --display-name="{Display Name}" --system
radosgw-admin zone modify --rgw-zone={zone-name} --access-key={access-key} --
secret={secret}
radosgw-admin period update --commit

#secondary zone config

radosgw-admin realm pull --url={url-to-master-zone-gateway} --access-key={access-key} --
secret={secret}
radosgw-admin realm default --rgw-realm={realm-name}
radosgw-admin zone create --rgw-zonegroup={zone-group-name} --rgw-zone={secondary-zone-
name} --access-key={system-key} --secret={secret} --endpoints=http://{fqdn}:80

# Remove earlier default zones and pools
radosgw-admin zone delete --rgw-zone=default
ceph osd pool rm default.rgw.control default.rgw.control --yes-i-really-really-mean-it
ceph osd pool rm default.rgw.data.root default.rgw.data.root --yes-i-really-really-mean-it
ceph osd pool rm default.rgw.gc default.rgw.gc --yes-i-really-really-mean-it
ceph osd pool rm default.rgw.log default.rgw.log --yes-i-really-really-mean-it
ceph osd pool rm default.rgw.users.uid default.rgw.users.uid --yes-i-really-really-mean-it
radosgw-admin period update --commit

# To check sync status
radosgw-admin sync status
```

```
MULTI CLIENT TESTING WITH DIFFERENT BLOCK IMAGES WITH 15 STRIPED DRIVES IN THE CLUSTER (3 way
replication)


TOTAL = 2.9 + 5.2 + 4.3 GBPS


root@ubuntu-Standard-PC-i440FX-PIIX-1996:~# PDSH_RCMD_TYPE=ssh pdsh -w root@192.168.18.75 fio --
name=io-test --ioengine=rbd --iodepth=64 --rw=read --bs=128k --numjobs=32 --time_based --runtime=120
--direct=1 --group_reporting --pool rbd --rbdname speed-test-image-1 & PDSH_RCMD_TYPE=ssh pdsh -w
root@192.168.18.193 fio --name=io-test --ioengine=rbd --iodepth=64 --rw=read --bs=128k --numjobs=32
--time_based --runtime=120 --direct=1 --group_reporting --pool rbd --rbdname speed-test-image-2 &
PDSH_RCMD_TYPE=ssh pdsh -w root@192.168.18.201 fio --name=io-test --ioengine=rbd --iodepth=64 --
rw=read --bs=128k --numjobs=32 --time_based --runtime=120 --direct=1 --group_reporting --pool rbd --
rbdname speed-test-image-3
[1] 1946
[2] 1947
192.168.18.193: io-test: (g=0): rw=read, bs=(R) 128KiB-128KiB, (W) 128KiB-128KiB, (T) 128KiB-128KiB,
ioengine=rbd, iodepth=64
192.168.18.193: ...
192.168.18.193: fio-3.25
192.168.18.193: Starting 32 processes
192.168.18.201: io-test: (g=0): rw=read, bs=(R) 128KiB-128KiB, (W) 128KiB-128KiB, (T) 128KiB-128KiB,
ioengine=rbd, iodepth=64
```

```
192.168.18.201: ...
192.168.18.201: fio-3.25
192.168.18.201: Starting 32 processes
192.168.18.75: io-test: (g=0): rw=read, bs=(R) 128KiB-128KiB, (W) 128KiB-128KiB, (T) 128KiB-128KiB,
ioengine=rbd, iodepth=64
192.168.18.75: ...
192.168.18.75: fio-3.25
192.168.18.75: Starting 32 processes
192.168.18.201:
192.168.18.201: io-test: (groupid=0, jobs=32): err= 0: pid=9566: Thu Apr 20 09:54:33 2023
192.168.18.201:   read: IOPS=22.3k, BW=2794MiB/s (2929MB/s)(328GiB/120281msec)
192.168.18.201:     slat (nsec): min=621, max=2563.1k, avg=4558.75, stdev=5371.33
192.168.18.201:     clat (usec): min=239, max=2146.8k, avg=91526.75, stdev=98483.39
192.168.18.201:      lat (usec): min=243, max=2146.8k, avg=91531.31, stdev=98483.74
192.168.18.201:     clat percentiles (usec):
192.168.18.201:      |  1.00th=[   578],  5.00th=[   963], 10.00th=[  1467], 20.00th=[  3163],
192.168.18.201:      | 30.00th=[  7504], 40.00th=[ 25560], 50.00th=[ 60556], 60.00th=[ 96994],
192.168.18.201:      | 70.00th=[135267], 80.00th=[177210], 90.00th=[233833], 95.00th=[278922],
192.168.18.201:      | 99.00th=[367002], 99.50th=[404751], 99.90th=[526386], 99.95th=[608175],
192.168.18.201:      | 99.99th=[968885]
192.168.18.201:    bw (  MiB/s): min=  643, max= 5707, per=100.00%, avg=2799.50, stdev=28.10,
samples=7662
192.168.18.201:    iops        : min= 5144, max=45660, avg=22395.99, stdev=224.82, samples=7662
192.168.18.201:   lat (usec)   : 250=0.01%, 500=0.36%, 750=2.34%, 1000=2.70%
192.168.18.201:   lat (msec)   : 2=8.57%, 4=9.02%, 10=9.52%, 20=5.44%, 50=9.18%
192.168.18.201:   lat (msec)   : 100=13.68%, 250=31.31%, 500=7.74%, 750=0.11%, 1000=0.01%
192.168.18.201:   lat (msec)   : 2000=0.01%, >=2000=0.01%
192.168.18.201:   cpu          : usr=0.49%, sys=0.35%, ctx=2684453, majf=2, minf=5020
192.168.18.201:   IO depths    : 1=0.1%, 2=0.1%, 4=0.1%, 8=0.1%, 16=0.1%, 32=0.1%, >=64=99.9%
192.168.18.201:      submit    : 0=0.0%, 4=100.0%, 8=0.0%, 16=0.0%, 32=0.0%, 64=0.0%, >=64=0.0%
192.168.18.201:      complete  : 0=0.0%, 4=100.0%, 8=0.0%, 16=0.0%, 32=0.0%, 64=0.1%, >=64=0.0%
192.168.18.201:      issued rwts: total=2688086,0,0,0 short=0,0,0,0 dropped=0,0,0,0
192.168.18.201:      latency   : target=0, window=0, percentile=100.00%, depth=64
192.168.18.201:
192.168.18.201: Run status group 0 (all jobs):
192.168.18.201:    READ: bw=2794MiB/s (2929MB/s), 2794MiB/s-2794MiB/s (2929MB/s-2929MB/s), io=328GiB
(352GB), run=120281-120281msec
192.168.18.201:
192.168.18.201: Disk stats (read/write):
192.168.18.201:     dm-1: ios=0/590, merge=0/0, ticks=0/4, in_queue=4, util=0.27%, aggrios=48/397,
aggrmerge=0/193, aggrticks=23/106, aggrin_queue=132, aggrutil=0.38%
192.168.18.201:   sdb: ios=48/397, merge=0/193, ticks=23/106, in_queue=132, util=0.38%
192.168.18.193:
192.168.18.193: io-test: (groupid=0, jobs=32): err= 0: pid=361024: Thu Apr 20 09:54:33 2023
192.168.18.193:   read: IOPS=40.2k, BW=5028MiB/s (5273MB/s)(591GiB/120344msec)
192.168.18.193:     slat (nsec): min=214, max=2417.3k, avg=2261.04, stdev=3614.29
192.168.18.193:     clat (usec): min=151, max=4487.2k, avg=50861.54, stdev=184494.16
192.168.18.193:      lat (usec): min=152, max=4487.2k, avg=50863.81, stdev=184494.20
192.168.18.193:     clat percentiles (usec):
192.168.18.193:      |  1.00th=[    449],  5.00th=[    783], 10.00th=[   1156],
192.168.18.193:      | 20.00th=[   1991], 30.00th=[   3097], 40.00th=[   4686],
192.168.18.193:      | 50.00th=[   7242], 60.00th=[  11469], 70.00th=[  19006],
192.168.18.193:      | 80.00th=[  32375], 90.00th=[  65799], 95.00th=[ 202376],
192.168.18.193:      | 99.00th=[1002439], 99.50th=[1350566], 99.90th=[2055209],
192.168.18.193:      | 99.95th=[2365588], 99.99th=[3204449]
192.168.18.193:    bw (  MiB/s): min=    8, max=22097, per=100.00%, avg=5079.58, stdev=168.94,
samples=7619
192.168.18.193:    iops        : min=   64, max=176776, avg=40636.59, stdev=1351.55, samples=7619
192.168.18.193:   lat (usec)   : 250=0.02%, 500=1.56%, 750=3.00%, 1000=3.45%
192.168.18.193:   lat (msec)   : 2=12.01%, 4=16.11%, 10=21.00%, 20=13.96%, 50=15.72%
192.168.18.193:   lat (msec)   : 100=5.97%, 250=2.71%, 500=1.77%, 750=1.06%, 1000=0.66%
192.168.18.193:   lat (msec)   : 2000=0.90%, >=2000=0.11%
192.168.18.193:   cpu          : usr=0.50%, sys=0.40%, ctx=4779926, majf=1, minf=4022
192.168.18.193:   IO depths    : 1=0.1%, 2=0.1%, 4=0.1%, 8=0.1%, 16=0.1%, 32=0.1%, >=64=100.0%
192.168.18.193:      submit    : 0=0.0%, 4=100.0%, 8=0.0%, 16=0.0%, 32=0.0%, 64=0.0%, >=64=0.0%
```

```
192.168.18.193:      complete  : 0=0.0%, 4=100.0%, 8=0.0%, 16=0.0%, 32=0.0%, 64=0.1%, >=64=0.0%
192.168.18.193:      issued rwts: total=4841080,0,0,0 short=0,0,0,0 dropped=0,0,0,0
192.168.18.193:      latency   : target=0, window=0, percentile=100.00%, depth=64
192.168.18.193:
192.168.18.193: Run status group 0 (all jobs):
192.168.18.193:    READ: bw=5028MiB/s (5273MB/s), 5028MiB/s-5028MiB/s (5273MB/s-5273MB/s), io=591GiB
(635GB), run=120344-120344msec
192.168.18.193:
192.168.18.193: Disk stats (read/write):
192.168.18.193:     dm-1: ios=0/286, merge=0/0, ticks=0/0, in_queue=0, util=0.12%, aggrios=52/192,
aggrmerge=0/94, aggrticks=9/15, aggrin_queue=24, aggrutil=0.23%
192.168.18.193:   nvme2n1: ios=52/192, merge=0/94, ticks=9/15, in_queue=24, util=0.23%
root@ubuntu-Standard-PC-i440FX-PIIX-1996:~# 192.168.18.75:
192.168.18.75: io-test: (groupid=0, jobs=32): err= 0: pid=327422: Thu Apr 20 09:54:34 2023
192.168.18.75:   read: IOPS=33.2k, BW=4154MiB/s (4356MB/s)(487GiB/120163msec)
192.168.18.75:     slat (nsec): min=415, max=9287.8k, avg=5353.23, stdev=14660.53
192.168.18.75:     clat (usec): min=228, max=4289.3k, avg=61555.40, stdev=207503.79
192.168.18.75:      lat (usec): min=232, max=4289.3k, avg=61560.75, stdev=207503.85
192.168.18.75:     clat percentiles (usec):
192.168.18.75:      |  1.00th=[    570],  5.00th=[    914], 10.00th=[   1287],
192.168.18.75:      | 20.00th=[   2180], 30.00th=[   3392], 40.00th=[   5211],
192.168.18.75:      | 50.00th=[   8160], 60.00th=[  13435], 70.00th=[  22676],
192.168.18.75:      | 80.00th=[  39060], 90.00th=[  85459], 95.00th=[ 299893],
192.168.18.75:      | 99.00th=[1149240], 99.50th=[1451230], 99.90th=[2197816],
192.168.18.75:      | 99.95th=[2499806], 99.99th=[3070231]
192.168.18.75:    bw (  MiB/s): min=    8, max=19629, per=100.00%, avg=4194.67, stdev=143.10,
samples=7562
192.168.18.75:    iops        : min=   64, max=157035, avg=33557.15, stdev=1144.79, samples=7562
192.168.18.75:   lat (usec)   : 250=0.01%, 500=0.40%, 750=2.53%, 1000=3.22%
192.168.18.75:   lat (msec)   : 2=12.02%, 4=15.78%, 10=20.26%, 20=13.29%, 50=16.43%
192.168.18.75:   lat (msec)   : 100=7.13%, 250=3.43%, 500=2.09%, 750=1.24%, 1000=0.83%
192.168.18.75:   lat (msec)   : 2000=1.21%, >=2000=0.16%
192.168.18.75:   cpu          : usr=1.00%, sys=0.68%, ctx=3874741, majf=0, minf=389
192.168.18.75:   IO depths    : 1=0.1%, 2=0.1%, 4=0.1%, 8=0.1%, 16=0.1%, 32=0.1%, >=64=99.9%
192.168.18.75:      submit    : 0=0.0%, 4=100.0%, 8=0.0%, 16=0.0%, 32=0.0%, 64=0.0%, >=64=0.0%
192.168.18.75:      complete  : 0=0.0%, 4=100.0%, 8=0.0%, 16=0.0%, 32=0.0%, 64=0.1%, >=64=0.0%
192.168.18.75:      issued rwts: total=3993166,0,0,0 short=0,0,0,0 dropped=0,0,0,0
192.168.18.75:      latency   : target=0, window=0, percentile=100.00%, depth=64
192.168.18.75:
192.168.18.75: Run status group 0 (all jobs):
192.168.18.75:    READ: bw=4154MiB/s (4356MB/s), 4154MiB/s-4154MiB/s (4356MB/s-4356MB/s), io=487GiB
(523GB), run=120163-120163msec
192.168.18.75:
192.168.18.75: Disk stats (read/write):
192.168.18.75:     dm-1: ios=0/1758, merge=0/0, ticks=0/68, in_queue=68, util=0.25%,
aggrios=343/2276, aggrmerge=184/340, aggrticks=83/314, aggrin_queue=400, aggrutil=0.70%
192.168.18.75:   sda: ios=343/2276, merge=184/340, ticks=83/314, in_queue=400, util=0.70%
```

REFERENCES

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

[2] DapuStor, "R5100," [Online]. Available: https://en.dapustor.com/product/1.html. [Accessed 2022].

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

[4] 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].

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

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

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

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

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

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

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

[12] 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].

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

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

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

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

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

[18] MinIO, "S3 Warp," [Online]. Available: https://github.com/minio/warp. [Accessed 2022].

[19] NVIDIA Corporation, "How To Install iperf and Test Mellanox Adapters Performance," [Online]. Available: https://enterprise-support.nvidia.com/s/article/howto-install-iperf-and-test-mellanox-adapters-performance. [Accessed 2023].

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

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

[22] Ceph, "Multisite," [Online]. Available: https://docs.ceph.com/en/quincy/radosgw/multisite/. [Accessed 2022].