Technical Report

Ceph on E-Series

Reference Architecture for Ceph Clusters Using NetApp E-Series

Mike Bostock and Tom Schmitz, NetApp
July 2017 | TR-4549

Abstract

Ceph is an open-source storage project that is increasing in popularity and adoption as organizations build new platforms. This technical report describes how to build a Ceph cluster by using a tested E-Series reference architecture. The report also describes the performance benchmarking methodologies used, along with test results.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1 Introduction

Ceph is an open-source storage project that is increasing in popularity and adoption as organizations build new platforms. Ceph is commonly found in OpenStack environments, but it is also growing as a standalone solution. A primary driver of Ceph adoption is the potential to build a scalable and flexible storage platform by using only commodity hardware. However, Ceph can be complex to manage and can pose challenges, particularly at scale. NetApp has developed and tested an E-Series reference architecture for Ceph that greatly improves stability and manageability while allowing customers to meet their cost targets.

Although Ceph is commonly deployed using "just-a-bunch-of-disks" (JBOD) or server-based storage, test results with NetApp® E-Series show that enterprise storage should be strongly considered. This is particularly true for large clusters supporting demanding workloads, such as a Cinder back end for Nova compute environments. To get the most out of a Ceph deployment, enterprises need to clearly define production requirements and build an architecture that performs well in all situations.

E-Series enterprise storage building blocks are streamlined for workloads that require high performance. The reliability and price-to-performance ratio of the E-Series make it a robust foundation for Ceph.

This report first covers important architectural elements of Ceph and then discusses how to build a Ceph cluster by using E-Series modular building blocks. The report also includes results of performance testing comparing a JBOD configuration to the E-Series Ceph reference configuration.

2 Ceph Overview

Ceph is an open-source project that enables enterprises to build a distributed storage system. The platform offers a unified block, object, file, and API library for data access. Ceph is particularly popular in the OpenStack community, where it has been the most prevalent back end for OpenStack Cinder block storage in semiannual user surveys. Because it is software-defined storage, users have the flexibility to choose from a wide variety of hardware to serve as the foundation for Ceph.

Ceph software is freely available from the Ceph Storage Community and also commercially available from Red Hat as Red Hat Ceph Storage.

2.1 Ceph Architecture

At its core, Ceph consists of a distributed object layer called *reliable autonomic distributed object store* (RADOS). Clients, however, consume Ceph through different access points that provide block, object, file, and API access (Table 1). In OpenStack environments, Ceph is commonly used to serve Cinder block storage to Nova virtual instances. The E-Series reference architecture is designed to be well suited for these OpenStack cloud environments, which are often demanding environments for a Ceph cluster.

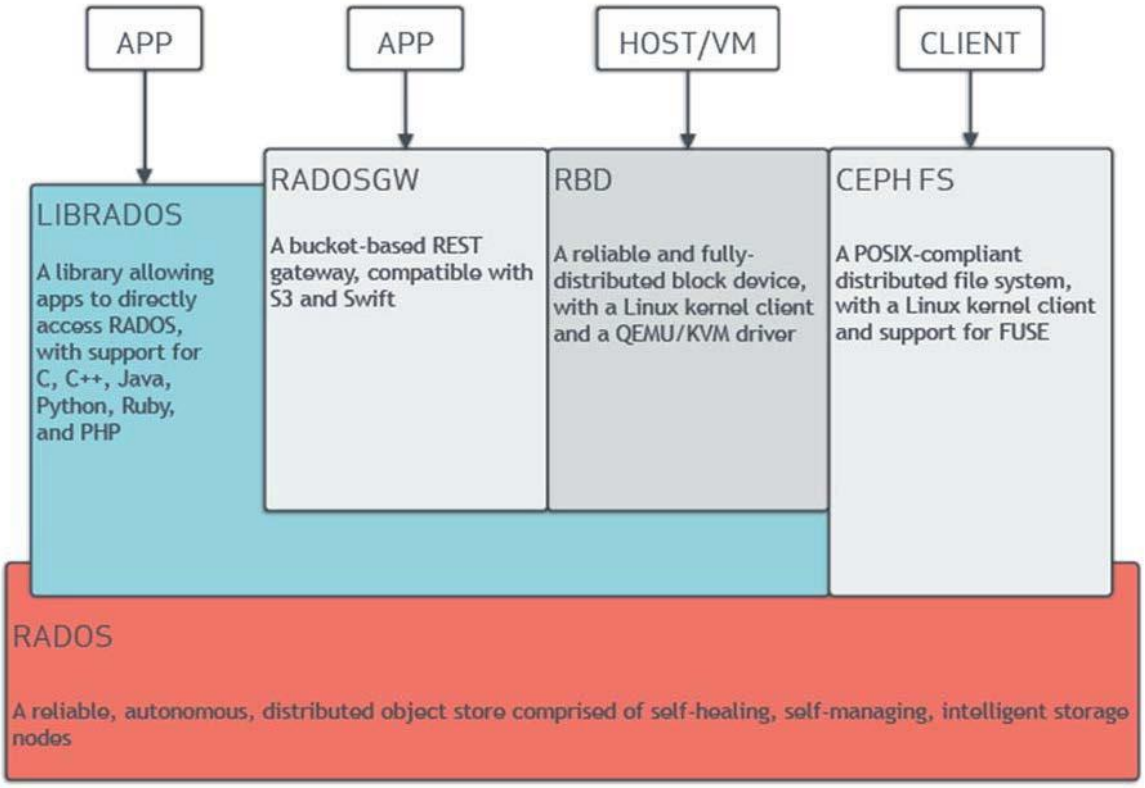
Table 1) Ceph client access methods.

Access Point	Type	Description
RADOS Block Device (RBD)	Block	Thinly provisioned block storage that is automatically distributed across the Ceph cluster.
RADOS Gateway (RADOSGW)	Object	RESTful object interface that is compatible with S3 and Swift protocols.
Ceph Filesystem (CephFS)*	File	POSIX-compliant file system for a Ceph cluster. CIFS and NFS are among the available protocols.
Librados	API	Library that enables direct access to a Ceph storage cluster.

* CephFS in Red Hat Ceph 2 is Tech Preview, it should be used with caution in production environments.

Figure 1 is a common depiction of Ceph's architecture from the Ceph Storage Community.

Figure 1) High-level Ceph architecture.



2.2 Ceph Cluster Components

As software-defined storage, Ceph offers flexibility when it comes to hardware. At the most basic level, Ceph clusters are constructed using servers, network switches, and external storage.

A Ceph cluster is generally constructed using three types of servers:

- **Ceph monitors.** Maintain maps of the cluster state.
- **Ceph object storage device (OSD) servers.** Store data; handle data replication, recovery, backfilling, and rebalancing. They also provide some monitoring information to monitors.
- **Ceph clients.** Servers that access the Ceph cluster through block, object, file, or API.

Note: OSD servers can use local storage or external storage. If Ceph Filesystem (CephFS) is used, a fourth server type is required:

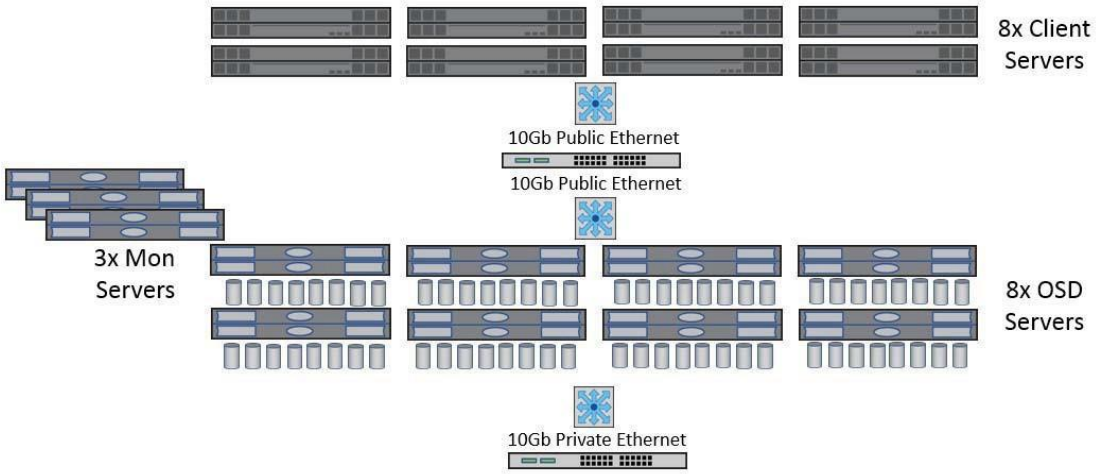
- **Ceph metadata server (MDS).** Stores metadata on behalf of CephFS.

In addition to server components, a Ceph cluster typically has two dedicated Ethernet networks:

- **Public network.** Allows the Ceph clients to send read and write user data to the OSD servers.
- **Cluster or private network.** Allows each OSD server to communicate with the others, sending heartbeats, status, and cluster recovery operations. The cluster network also handles the data replication; it is important to plan for adequate bandwidth to keep performance optimal.

Putting the compute and networking pieces together leads to the architecture shown in Figure 2.

Figure 2) Example of a small Ceph cluster.



All but the largest Ceph clusters require just three monitor servers. The number and ratio of Ceph clients and OSD servers vary, depending on the cluster's capacity and performance requirements.

2.3 Object Storage Devices and Journals

From a storage perspective, there are two types of devices that support a Ceph cluster:

- **Object storage device (OSD).** A physical or logical storage unit. Ceph distributes data among OSDs by using the Controlled Replication Under Scalable Hashing (CRUSH) algorithm to compute object storage locations
- **Journal.** Typically, a partition on an SSD. Each OSD must have a journal, which can reside on the OSD itself. Ceph uses a journal for two reasons: speed, because the journal allows Ceph to commit small writes quickly; and consistency, because the journal guarantees automatic compound operations.

Note: Journals don't add to the capacity of the Ceph cluster. However, they do have a major influence on the overall performance of the cluster. For performance reasons, a common rule of thumb is to have 1 SSD for journals for every 5 HDDs when using server-based storage.

For the E-Series reference configuration, each OSD is a RAID 5 volume built with HDDs, and each journal is partitioned on a RAID 10 volume built using SSDs. The E-Series reference architecture uses only 1 SSD per 12.5 HDDs. Section 5 covers the OSD and journal configuration in detail.

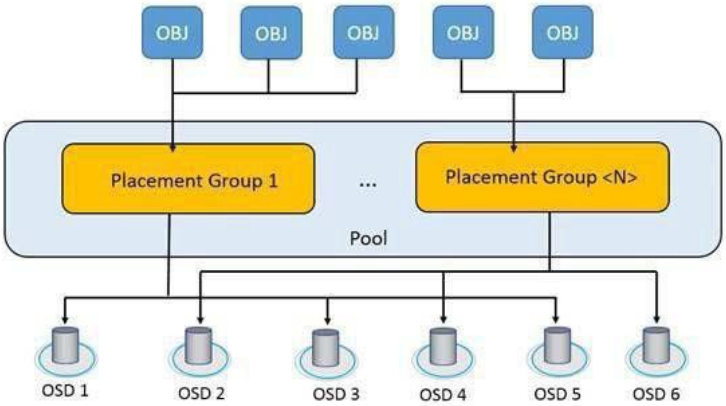
2.4 Ceph Pools and Placement Groups

The Ceph cluster is made up of pools and placement groups that determine where data is stored as shown in Figure 3.

- **Pools.** A pool is a logical group that contains placement groups. Pools are configured at creation time with attributes such as the number of data replicas within the pool.
- **Placement groups (PGs).** PGs are created within the pool to contain all the data objects. Many placement groups are created within the pool to spread the data across multiple OSDs. Below is the Ceph algorithm for the number of PGs per pool. Ideally, the total PGs need to be a power of 2, and Ceph modifies the range (min/max) to account for the number of OSDs. If the number of PGs is below or above the range, Ceph presents a health error. For the JBOD, we used 4,096 PGs (the minimum number allowed by Ceph for 160 OSDs), and 2,048 for the RBOD (the minimum allowed by Ceph for 40 OSDs). The JBOD used three replicas, whereas the RBOD used two.

$$\text{Total PGs} = \frac{(\text{OSDs} * 100)}{\text{Replicas}}$$

Figure 3) Relationship of pools and placement groups.



2.5 CRUSH Algorithm for Data Placement

CRUSH stands for Controlled Replication Under Scalable Hashing. This algorithm controls how data is stored and retrieved within the cluster. CRUSH creates and maintains a complete map of the cluster, called the *CRUSH map.* This map holds the information about where PGs are stored and which OSDs make up the placement groups. Data is then stored in the PGs, depending on modifiable configuration rules contained in the CRUSH map. The map contains information about how replication is performed, along with how many copies of the data to create.

For details about CRUSH, read this paper: http://ceph.com/wp-content/uploads/2016/08/weil-crush-sc06.pdf

2.6 Ceph Data Protection

Ceph is designed to run on commodity hardware, so it has two built-in capabilities for protecting data:

- **Replicas.** Replicas create identical duplicates of all data written. If the replica configuration is set to 3, then three copies of the data are stored within the cluster. Replicas provide a robust data protection environment, but they can decrease usable capacity.
- **Erasure coding.** Erasure coding creates a single copy of the data and uses protection similar to RAID with parity to store data and use for recovery. Erasure coding can increase available storage space, but it can require increased resources, such as compute power, and it can potentially lower performance with increased write latencies and extended recovery times.

Because Ceph is often deployed on internal server storage or JBOD without redundancy, the default replica setting is three copies. This means that every client write request must be written to three different locations within a Ceph cluster before a completion is sent back to the client that issued the write. With three replicas, each object is replicated such that any single failure leaves two replicas intact. Read requests are served from the primary OSD that holds the data.

The number of replicas can easily be changed. With E-Series, the replica count can be reduced from three to two while using the back-end RAID engine to provide the necessary additional protection. Section 5 covers how to decrease the number of replicas.

Note: Only the replica protection option is currently supported when using RBD or CephFS. E-Series testing focused on optimizing RBD block performance, so only replicas were used during testing.

3 E-Series Enterprise Building Blocks

The NetApp E-Series is a robust platform for delivering exceptional performance and reliability to mission-critical workloads. The streamlined E-Series offers low latency and high bandwidth, making it well suited to handle diverse workloads. The Ceph reference architecture builds on the E-Series strong heritage as a building block for large distributed solutions, including Lustre, Hadoop, General Parallel File System (GPFS), and the StorNext File System. In all of these environments, sophisticated software combined with the rock-solid and fast E-Series leads to better overall solutions. With Ceph, the E-Series continues its tradition as an optimal building block that improves cluster stability and scalability.

3.1 E-Series Hardware Overview

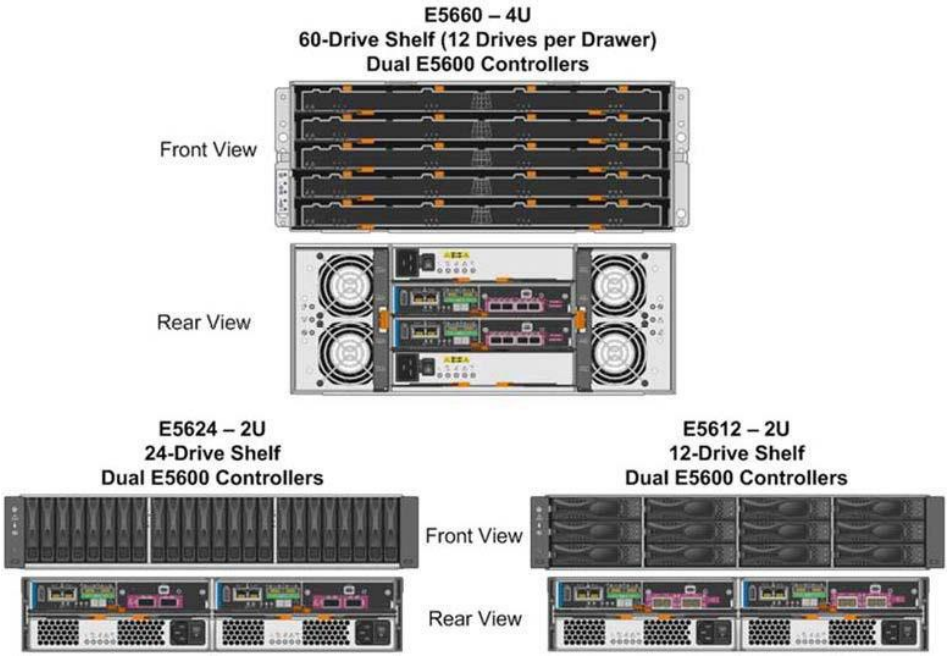
As shown in Table 2, the E5600 is available in three shelf options that support both HDDs and SSDs to meet a wide range of performance and application requirements.

Table 2) E5600 controller shelf and drive shelf models.

Controller Shelf Model	Drive Shelf Model	Number of Drives	Type of Drives
E5660	DE6600	60	2.5" and 3.5" SAS drives (HDDs and SSDs)
E5624	DE5600	24	2.5" SAS drives (HDDs and SSDs)
E5612	DE1600	12	3.5" SAS drives (HDDs only)

All three shelf options include dual controller modules, dual power supplies, and dual fan units for redundancy. The 12-drive and 24-drive shelves have integrated power and fan modules. The shelves are sized to hold 60 drives, 24 drives, or 12 drives, as shown in Figure 4.

Figure 4) E5600 controller-drive shelf options.



Each E5600 controller shelf includes two controllers, with each controller providing two Ethernet management ports for out-of-band management. The system also supports in-band management access and has two 6Gbps wide-port SAS drive expansion ports for redundant drive expansion paths. The E5600 controllers do not include built-in host ports but must be ordered with one of the following host interface cards (HICs) installed in each controller:

- 4-port 12Gb SASHIC.
- 2-port 56Gb InfiniBand (IB) HIC. This HIC runs the iSCSI Extensions for RDMA (iSER) protocols shipped, but it can be converted to SCSI RDMA Protocol (SRP) before initial use by applying a software feature pack in the field at no additional cost.
- 4-port optical HIC, which can be factory-configured as either 16Gb Fibre Channel or 10Gb iSCSI. A software feature pack can be applied in the field to change the host protocol of this HIC:
  - From FC to iSCSI
  - From iSCSI to FC
  - From either FC or iSCSI to FC-iSCSI split mode
  - From FC-iSCSI split mode back to FC or iSCSI

Note: Both controllers in an E5600 array must be configured identically. In FC-iSCSI split mode, ports 1 and 2 operate as iSCSI, and ports 3 and 4 operate as FC.

E5600 controllers are available with two memory options: a standard 12GB DIMM, which can be used with all protocols, and a 48GB DIMM, which can be used with iSCSI and FC only. Controller memory upgrades from 12GB to 48GB are not available.

3.2 SANtricity Overview

E-Series systems are managed by the NetApp SANtricity® Storage Manager application. Simple to download and install, SANtricity Storage Manager provides an intuitive, wizard-led GUI as well as full support for a CLI. SANtricity Storage Manager can be installed on Linux, Windows, or Solaris for out-of-band management of the storage array.

To create volume groups on the array, the first step when configuring SANtricity Storage Manager is to assign a RAID level. This assignment is then applied to the disks selected to form the volume group. The E5600 storage systems support RAID levels 0, 1, 3, 5, 6, and 10 or dynamic disk pools.

To simplify storage provisioning, SANtricity offers an automatic configuration feature. The configuration wizard analyzes the available disk capacity on the array. It then selects disks that maximize array performance and fault tolerance while meeting capacity requirements, hot spares, and any other criteria specified in the wizard.

Dynamic Capabilities

From a management perspective, SANtricity offers a number of capabilities to ease the burden of storage management, including the following:

- New volumes can be created and are immediately available for use by connected servers.
- New RAID sets (volume groups) or dynamic disk pools can be created any time from unused disk devices.
- Volumes, volume groups, and disk pools can all be expanded online as necessary to meet any new requirements for capacity or performance.
- Dynamic RAID migration allows the RAID level of a particular volume group to be modified online, for example from RAID 10 to RAID 5, if new requirements require a change.
- Flexible cache block and segment sizes enable optimized performance tuning based on a particular workload. Both items can also be modified online.
- There is built-in performance monitoring of all major storage components, including controllers, volumes, volume groups, pools, and individual disk drives.

- Automated remote connection to the NetApp AutoSupport® function enables “phone home” capabilities and automated parts dispatching in case of component failures.
- Path failover and load balancing (if applicable) are provided between the host and the redundant storage controllers in the E5600.

SANtricity Web Services Proxy

The NetApp SANtricity Web Services Proxy provides web APIs to configure, manage, and monitor E-Series and EF-Series storage arrays. The proxy gives access to a collection of REST (REpresentational State Transfer) style interfaces to access services defined for storage arrays. The proxy can be installed on Windows or Linux systems.

3.3 E-Series Data Protection Features

E-Series has a reputation for reliability and availability. Many of the data protection features found in E--Series systems can be beneficial in a Ceph environment. This section highlights key E-Series protection features that can improve Ceph reliability and availability.

Drive Encryption (Full Disk Encryption)

Full Disk Encryption services provide comprehensive security for data at rest without affecting storage system performance or ease of use. The native key management system saves the expense and complexity of an external key manager. Drive-based AES-256 encryption that is compliant with Federal Information Process Standards (FIPS) 140-2 level 2 using validated drives provides data security in case of drive theft, routine defective drive servicing, or repurposing of drives.

Data Assurance (T10 PI)

The Data Assurance feature provides controller-to-drive data integrity protection through the SCSI direct-access block device protection information model. This model protects user data by appending protection information to each block of user data. The protection model is sometimes referred to as Data Integrity Field protection or T10 PI. This model ensures that an I/O has completed without any bad blocks written to or read from disk. It protects against displacement errors, data corruption resulting from hardware or software errors, bit flips, and silent drive errors, such as when the drive delivers the wrong data on a read request or writes to the wrong location.

Proactive Drive Health Monitor

Proactive Drive Health Monitoring examines every completed drive I/O and tracks the rate of error and exception conditions returned by the drives. It also tracks drive performance degradation, which is often associated with unreported internal drive issues. Using predictive failure analysis technology, when any error rate or degraded performance threshold is exceeded—indicating that a drive is showing signs of impending failure—SANtricity software issues a critical alert message and takes corrective action to protect the data.

Data Evacuator

With Data Evacuator, nonresponsive drives are automatically power-cycled to see if the fault condition can be cleared. If the condition cannot be cleared, the drive is flagged as failed. For predictive failure events, the evacuator feature removes data from the affected drive in an effort to move the data before the drive actually fails. If the drive fails, rebuild picks up where the evacuator was disrupted, thus reducing the rebuild time.

Background Media Scan

Media Scan is a background process that is performed by the controllers to provide error detection on the drive media. The main purpose of the feature is to detect and repair media errors on disk drives that are

infrequently read by user applications and where data loss might occur if other drives in the volume group fail. A secondary purpose is to detect redundancy errors such as data/parity mismatches. A background media scan can find media errors before they disrupt normal drive reads and writes.

Unreadable Sector Management

This feature provides a controller-based mechanism for handling unreadable sectors detected both during normal I/O operation of the controller and during long-lived operations, such as reconstructions. The feature is transparent to the user and requires no special configuration.

Hot Spare Support

The system supports global hot spares that can be automatically used by the controller to reconstruct the data of the failed drive if enough redundancy information is available. The controller selects the best match for the hot spare based on several factors, including capacity and speed.

Online Drive Firmware Upgrade

This feature upgrades one drive at a time and tracks writes to the affected drives during the upgrade window. The feature should be used only during very low-write I/O periods. Parallel drive firmware upgrades are supported offline to more quickly upgrade multiple drives during a maintenance window.

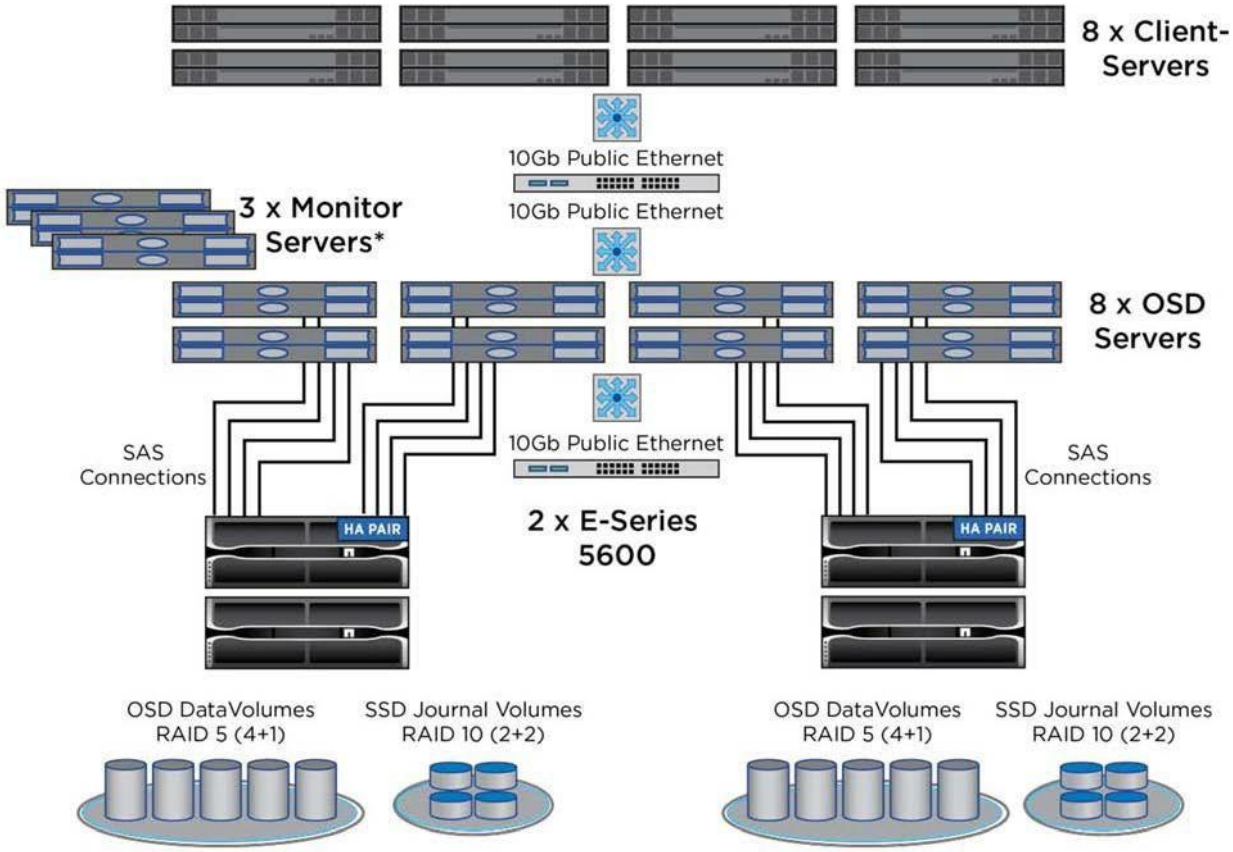
SSD Wear Life Monitoring and Reporting

If an SSD supports wear life reporting, users are provided this information in the GUI so that they can monitor how much of the useful life of an SSD remains. For SSDs that support wear life monitoring, the percentage of spare blocks remaining in solid-state media is monitored by controller firmware at approximately one-hour intervals. Think of this as a fuel gauge for SSDs.

4 Ceph Reference Cluster with E-Series

The reference architecture shown in Figure 5 focuses on building a platform to provide predictable performance primarily for the block use case. A common use case of RBD is to act as the Cinder back end in an OpenStack environment. The block RBD use case is one of the more demanding uses for Ceph.

Figure 5) E-Series Ceph reference architecture overview.



* For testing, the monitor servers coexisted with 3 of the OSD servers.

The following sections describe in detail the specific hardware and software used to build this Ceph cluster on E-Series hardware.

4.1 E-Series Hardware

The core E-Series building block for a Ceph cluster is a single E5660 with SAS host ports. Given 8 SAS host ports, the E5660 supports 4 dual-connected Ceph OSD servers. Multiple units of this building block can be deployed as needed to satisfy capacity or performance requirements. The reference architecture deploys two of the base building blocks.

Table 3) E-Series Ceph building blocks.

Component	Quantity	Description
E5600 Controller Pair with SAS Host Ports	1	• High-bandwidth/low-latency RAID engine • SAS host ports for high performance, low cost, and simplicity
DE6600	2	• 60-drive, 4U dense enclosure
3TB NL-SAS	102	• 100 drives to build RAID 5 (4+1) for OSDs • 2 hot spares
400GB SSDs	9	• 8 SSDs to build RAID 10 (2+2) for journals • 1 hot spare

Note: The OSD volumes were built using 3TB drives. This capacity was chosen because the drives were readily available for testing. Larger NL-SAS drives can be used for greater capacity.

This configuration does leave empty slots. In Ceph, OSD servers can have different numbers of volumes, which allows more of these slots to be used. However, for test purposes each OSD server had an equal number of volumes.

Because Ceph best practices dictate that no single OSD server should contain more than approximately 10% of the overall capacity for a cluster, we constructed our E-Series Ceph reference architecture cluster using two of the E-Series building blocks.

Table 4) E-Series systems used for reference architecture testing.

Component	Quantity	Description
E5600 Controller Pair with SAS Host Ports	2	• High-bandwidth/low-latency RAID engine • SAS host ports for high performance, low cost, and simplicity
DE6600	4	• 60-drive, 4U dense enclosure
3TB NL-SAS	204	• 200 drives to build RAID 5 (4+1) for OSDs • 4 hot spares
400GB SSDs	18	• 16 SSDs to build RAID 10 (2+2) for journals • 2 hot spares

4.2 Server Hardware

Table 5) Ceph servers used for reference architecture testing.

Component	Quantity	Description
Monitor Servers	3	• Processors: 32 core • Memory: 64GB • Networking: 10Gbit NIC
Client Servers	8	• Processors: 32 core • Memory: 128GB • Networking: 10Gbit NIC
OSD Servers	8	• Processors: 32 core • Memory: 64GB • Networking: 2 10Gigabit NICs • SAS Connectivity: Dual-port 12Gbit SAS HBA

Note: For test purposes, monitor services ran on three of the OSD servers.

4.3 Network Hardware

Table 6) Networking switches used for reference architecture testing.

Component	Quantity	Description
Public Network Switch	1	• 24 ports • 10Gbit • Jumbo frames enabled
Cluster Network Switches	1	• 24 ports • 10Gbit • Jumbo frames enabled

Note: In a production environment, NetApp recommends redundant switches.

4.4 Software Versions

Table 7) Software versions used for reference architecture testing.

Component	Version
Operating System	• Red Hat Enterprise Linux Server release 7.3
Ceph	• Red Hat Ceph Storage 2—based on Jewel Release Version 10.2.2-38
SANtricity	• 11.30

5 Building a Ceph Cluster with E-Series

5.1 E-Series Configuration

E-Series Volume Group and Volume Configuration

As mentioned in section 2, when constructing a Ceph cluster there are two fundamental storage requirements: OSDs and journals. Ceph does not impose any restrictions on the media type, so clusters can range from all-spinning media to all-flash. The E-Series cluster is intended to balance performance, capacity, and cost, so we selected NL-SAS drives for OSDs and SSDs for journals.

OSD Volumes

For the OSDs, the reference cluster uses RAID 5 (4+1) volume groups. Each volume group is consumed by a single volume. Larger volume groups can be used, but volume groups that are too large could be detrimental to Ceph. For example, a single large volume eliminates the parallelism of multiple OSD processes working together on a single OSD server. Providing too many OSDs per server can also be detrimental. For instance, if a Ceph OSD server becomes very thick with volumes, a server failure results in a tremendous amount of data copying on the cluster network.

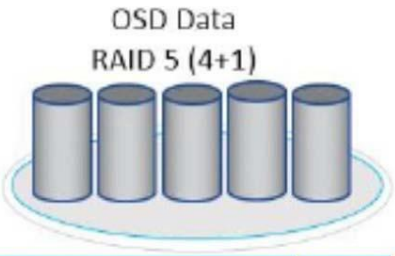
For this reference architecture, five RAID 5 volumes per OSD server were used. This configuration results in 25 NL-SAS drives dedicated per OSD server, which is similar to the common 20-OSD JBOD configuration.

Volume Group Best Practice
Each E-Series volume group contains only a single volume. This approach minimizes contention within the volume. The goal of this approach is to make each RAID 5 volume behave much like an individual HDD in the JBOD case, but with better performance, scalability, and redundancy.

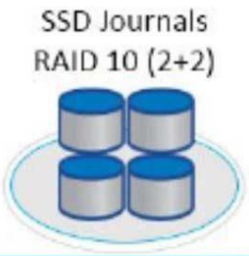
Journal Volumes

For journals, the reference cluster uses RAID 10 (2+2) volume groups. RAID 10 was chosen to optimize write performance and redundancy. Also, the journal volumes do not require significant capacity. For each OSD server, a 100GB volume is created. This volume is then divided into five Linux partitions to support five OSD volumes.

Figure 6) OSD and journal volume details.



Parameter	Value
Volume Group Size	10.894 TB
Volume Size	10.894 TB
Segment Size	256 KB
SSD Cache	Disabled
Read Cache	Enabled
Write Cache	Enabled
Write Cache Without Batteries	Disabled
Write Cache Mirroring	Enabled
Flush Write Cache After (in seconds)	10
Dynamic Cache Read Prefetch	Disabled



Parameter	Value
Volume Group Size	10.894 TB
Volume Size	10.894 TB
Segment Size	256 KB
SSD Cache	Disabled
Read Cache	Enabled
Write Cache	Enabled
Write Cache Without Batteries	Disabled
Write Cache Mirroring	Enabled
Flush Write Cache After (in seconds)	10
Dynamic Cache Read Prefetch	Disabled

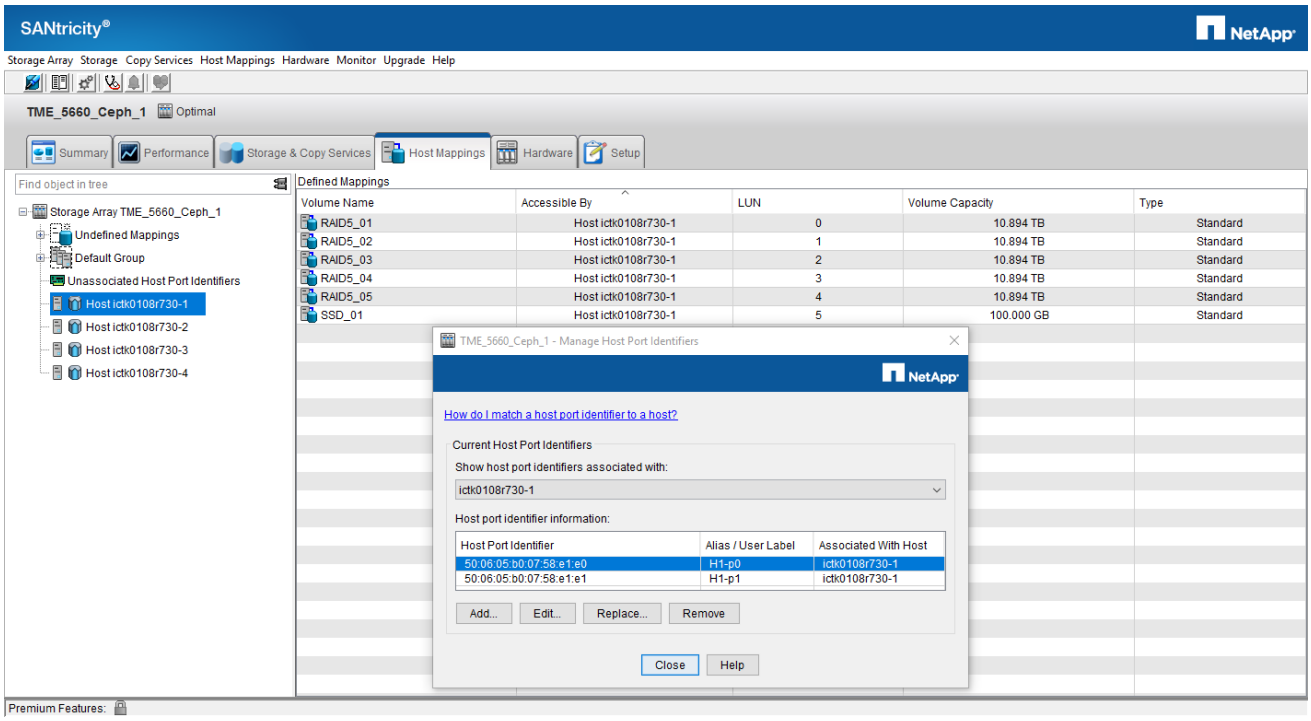
E-Series Hot Spares

NetApp recommends hot spares so that OSD and journal volumes begin rebuilding instantly in case of a media failure. For the reference configuration, NetApp recommends a total of two NL-SAS hot spares for each E5600. NetApp also recommends a single SSD hot spare for each E5600.

E-Series Host Mappings

Once the volumes are created, they must be mapped by using the storage partitioning feature in SANtricity. A host is defined for each OSD server by using World Wide Identifiers of the SAS HBA ports. In the tested configuration, each OSD server has five 10.894TB RAID 5 volumes and one 100GB RAID 10 volume assigned to it. Two SAS ports are connected to each server. For multipathing support, a host type of ALUA was selected, and user friendly names disabled in the multipath.conf file.

Figure 7) SANtricity host mappings for Ceph cluster.



5.2 LinuxConfiguration

All servers in the Ceph cluster were tested with Red Hat Enterprise Linux 7.3 (RHEL 7.3) with default kernel settings.

Configuring Network Jumbo Frames

Jumbo frames should be set on both the public and private networks to increase performance. In RHEL 7.3 they are configured by adding an entry of MTU=9000 to the interface file in the /etc/sysconfig/network-scripts directory and restarting the interface. To validate that jumbo frames have been set, use the `ip link show` command:

```
[root@ictk0103r720-4 ~]# ip link show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 9000 qdisc mq state UP mode DEFAULT qlen 1000
    link/ether b0:83:fe:d5:ae:62 brd ff:ff:ff:ff:ff:ff
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 9000 qdisc mq state UP mode DEFAULT qlen 1000
    link/ether b0:83:fe:d5:ae:64 brd ff:ff:ff:ff:ff:ff
```

No other network parameters were altered in the test configuration.

Network Testing

Network configuration plays a critical role in the overall performance of the cluster. Ceph recommends at least two networks: a public network (client to OSD) and a private network (OSD to OSD). Depending on the size of the cluster, multiple networks might be required.

Network validation was performed using `iperf`. Each network connection between the clients to each OSD server and connections between individual OSD servers were tested for maximum bandwidth performance.

`iperf` is a client-to-server tool. Figure 8 shows an example of `iperf` running between one Ceph client and one OSD server in order to test the bandwidth of the public network.

Figure 8) Network testing using iperf.

```
###iperf as seen from client node1
[root@ictk0103r720-1 ~]# iperf3 -c 192.168.124.57
Connecting to host 192.168.124.57, port 5201
[  4] local 192.168.124.11 port 43276 connected to 192.168.124.57 port 5201
[ ID] Interval           Transfer     Bandwidth       Retr  Cwnd
[  4]   0.00-1.00   sec  1.15 GBytes  9.92 Gbits/sec    3    856 KBytes
[  4]   1.00-2.00   sec  1.15 GBytes  9.90 Gbits/sec    0    874 KBytes
[  4]   2.00-3.00   sec  1.15 GBytes  9.90 Gbits/sec    0    874 KBytes
[  4]   3.00-4.00   sec  1.15 GBytes  9.90 Gbits/sec    0    891 KBytes
[  4]   4.00-5.00   sec  1.15 GBytes  9.90 Gbits/sec    0    900 KBytes
[  4]   5.00-6.00   sec  1.15 GBytes  9.90 Gbits/sec    0    918 KBytes
[  4]   6.00-7.00   sec  1.15 GBytes  9.90 Gbits/sec    0    918 KBytes
[  4]   7.00-8.00   sec  1.15 GBytes  9.90 Gbits/sec    0    926 KBytes
[  4]   8.00-9.00   sec  1.15 GBytes  9.90 Gbits/sec    0   1.11 MBytes
[  4]   9.00-10.00  sec  1.15 GBytes  9.90 Gbits/sec    0   1.45 MBytes
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bandwidth       Retr
[  4]   0.00-10.00  sec  11.5 GBytes  9.90 Gbits/sec    3             sender
[  4]   0.00-10.00  sec  11.5 GBytes  9.90 Gbits/sec                  receiver
```

```
###iperf as seen from RBOD osd-node-7
[root@ictk0108r730-7 Desktop]# iperf3 -s
-----------------------------------------------------------
Server listening on 5201
-----------------------------------------------------------
Accepted connection from 192.168.124.11, port 43274
[  5] local 192.168.124.57 port 5201 connected to 192.168.124.11 port 43276
[ ID] Interval           Transfer     Bandwidth
[  5]   0.00-1.00   sec  1.11 GBytes  9.53 Gbits/sec
[  5]   1.00-2.00   sec  1.15 GBytes  9.90 Gbits/sec
[  5]   2.00-3.00   sec  1.15 GBytes  9.90 Gbits/sec
[  5]   3.00-4.00   sec  1.15 GBytes  9.90 Gbits/sec
[  5]   4.00-5.00   sec  1.15 GBytes  9.90 Gbits/sec
[  5]   5.00-6.00   sec  1.15 GBytes  9.90 Gbits/sec
[  5]   6.00-7.00   sec  1.15 GBytes  9.90 Gbits/sec
[  5]   7.00-8.00   sec  1.15 GBytes  9.90 Gbits/sec
[  5]   8.00-9.00   sec  1.15 GBytes  9.90 Gbits/sec
[  5]   9.00-10.00  sec  1.15 GBytes  9.90 Gbits/sec
[  5]  10.00-10.04  sec  43.9 MBytes  9.88 Gbits/sec
- - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bandwidth
[  5]   0.00-10.04  sec  0.00 Bytes  0.00 bits/sec                  sender
[  5]   0.00-10.04  sec  11.5 GBytes  9.86 Gbits/sec                  receiver
-----------------------------------------------------------
```

5.3 Ceph Configuration

After creating the E-Series volumes and establishing a solid network, Ceph can be installed on the cluster of servers. The installation and configuration of Ceph require numerous steps, so a complete description of the process is outside the scope of this document. The E-Series cluster was installed by using the instructions in the Red Hat Ceph Storage 2 Installation Guide.

When deployed with E-Series, Ceph can be customized by using the following modifications:

- Using multipath devices with Ceph
- Creating XFS file systems tuned for E-Series
- Modifying Ceph CRUSH hierarchy for E-Series
- Changing default replicas from three to two

These changes are detailed in the following sections.

Multipath Devices with Ceph

By default, the deployment tools with Red Hat Ceph Storage 2 do not recognize multipath devices. With E-Series, NetApp recommends using a dual-controller, dual-path model using multipath, which can be easily accomplished by manually configuring the devices. The following example assumes that native Linux multipath service, Device Mapper Multipath (DM-Multipath), is already installed and running, and that the volumes have been mapped with SANtricity to the OSD server.

Example of Configuring Multipath Ceph OSD with E-Series

For this example, the OSD spans an entire RAID 5 E-Series volume. The journal is a separate E-Series RAID 10 volume. The World Wide Name (WWN) of the volume is used as the reference point for adding an OSD through DM-Multipath. When user friendly names are disabled in the multipath.conf file, the WWN can be located by using the `multipath -ll` command and comparing the output to volume information in SANtricity. The LUN number can be used for additional verification.

Step 1: Check multipath –ll output

In this example, the WWN is `360080e5000435a4c0000045d56df15f9` and the LUN number is 3.

```
# multipath -ll
360080e5000435a4c0000045d56df15f9 dm-7 NETAPP ,INF-01-00
size=11T features='4 queue_if_no_path pg_init_retries 50retain_attached_hw_handle' hwhandler='1
rdac' wp=rw
|-+- policy='service-time 0' prio=14 status=active
| `- 8:0:0:3 sde 8:64  active ready running
`-+- policy='service-time 0' prio=9 status=enabled
  `- 8:0:1:3 sdk 8:160 active ready running
```

Step 2: Partition entire volume

Using `fdisk`, `gparted`, or `parted`, create a gpt partition spanning the entire volume for the OSD. Afterward, the output should look similar to the following:

```
# fdisk -l /dev/dm-7

Disk /dev/dm-7: 11978.7 GB, 11978747674624 bytes, 23395991552 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk label type: dos
Disk identifier: 0x00000000


     Device Boot      Start         End      Blocks   Id  System
/dev/dm-7p1               1  4294967295  2147483647+  ee  GPT
```

Step 3: Create an XFS file system

Create an XFS file system on the partition.

```
# mkfs.xfs -f -d su=256k,sw=4 /dev/mapper/360080e5000435a4c0000045d56df15f9P1
```

XFS Best Practice

For optimal data alignment, specific parameters can be used when creating the XFS file system to match the underlying RAID configuration.

- stripe unit (su) matches the volume segment size of 256k
- stripe width (sw) matches the data drives in the RAID 5 (4+1)

Step 4: Create mount points

Add directories for mounting. This example uses `/var/lib/ceph/osd/ceph-$i` as the mount point.

```
mkdir /var/lib/ceph/osd/ceph-$i     <---Where $i is the unique osd number.
Example: mkdir  /var/lib/ceph/osd/ceph-1

sudo mount -o noatime /dev/mapper/360080e5000435a4c0000045d56df15f9P1 /var/lib/ceph/osd/ceph-1
```

Step 5: Add entries to file system table

Add mount point to `/etc/fstab`:

```
echo "/dev/mapper/360080e5000435a4c0000045d56df15f9P1 /var/lib/ceph/osd/ceph-1 xfs
defaults,noatime 0 0" >> /etc/fstab';
```

Step 6: Create partitions on the SSD journal

Use `parted` to create gpt partitions on the SSD volume for the journals. In this example, five 10GB partitions are created, so there is one partition for each of the five LUNs used as OSD data drives. No file system needs to be created on the journal partitions because it is used as a raw partition.

```
sudo parted -s /dev/mapper/360080e5000435a4c0000048156df18e6 mklabel gpt mkpart ceph-journal-1
'0%' 10GB mkpart ceph-journal-2 10GB 20GB mkpart ceph-journal-3 20GB 30GB mkpart ceph-journal-4
30GB 40GB mkpart ceph-journal-5 40GB 50GB
```

Afterward, `fdisk` output should be similar to the following:

```
# fdisk -l /dev/dm-2
WARNING: fdisk GPT support is currently new, and therefore in an experimental phase. Use at
your own discretion.

Disk /dev/dm-2: 107.4 GB, 107374182400 bytes, 209715200 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 4096 bytes
I/O size (minimum/optimal): 4096 bytes / 4096 bytes
Disk label type: gpt

 #          Start            End    Size  Type            Name
 1           2048       20482047    9.8G  Linux filesyste ceph-journal-1
 2       20482048       40962047    9.8G  Linux filesyste ceph-journal-2
 3       40962048       61442047    9.8G  Linux filesyste ceph-journal-3
 4       61442048       81922047    9.8G  Linux filesyste ceph-journal-4
 5       81922048      102402047    9.8G  Linux filesyste ceph-journal-5
```

Step 7: Prepare and activate OSD

The commands inherent to Ceph, Red Hat Ceph Storage 2, and ceph-deploy fail to recognize multipath devices. Therefore, neither Red Hat nor Ceph supports multipathing or provides documentation to use multipath devices. The solution can be found in the appendix section of this document “Sample E-Series Create_OSDs.sh.”

Step 8: Validate

Validate that the OSD is paired with the proper journal by reviewing the logs found in `/var/log/ceph/`. Then list the contents of the mount point to make sure that the appropriate files were created.

```
# The log should contain the following details:
mkjournal created journal on /dev/mapper/360080e5000435a4c0000045d56df15f9p1
filestore(/var/lib/ceph/osd/ceph-1) mkfs done in /var/lib/ceph/osd/ceph-1


# Listing the mount point directory should display the following:
[root@ictk0105r730-4 ceph]# ls -lrt /var/lib/ceph/osd/ceph-1
total 10240044
-rw-r--r--   1 ceph ceph          37 Feb 24 16:54 fsid
-rw-r--r--   1 ceph ceph           4 Feb 24 16:54 store_version
-rw-r--r--   1 ceph ceph          53 Feb 24 16:54 superblock
-rw-r--r--   1 ceph ceph          10 Feb 24 16:54 type
-rw-r--r--   1 ceph ceph          21 Feb 24 16:54 magic
-rw-r--r--   1 ceph ceph           2 Feb 24 16:54 whoami
-rw-r--r--   1 ceph ceph          37 Feb 24 16:54 ceph_fsid
-rw-r--r--   1 ceph ceph           6 Feb 24 16:54 ready
-rw-------   1 ceph ceph          56 Feb 24 16:54 keyring
-rw-r--r--   1 ceph ceph 10485760000 Feb 25 16:08 journal
drwxr-xr-x 164 ceph ceph        4096 Mar 16 19:32 current
```

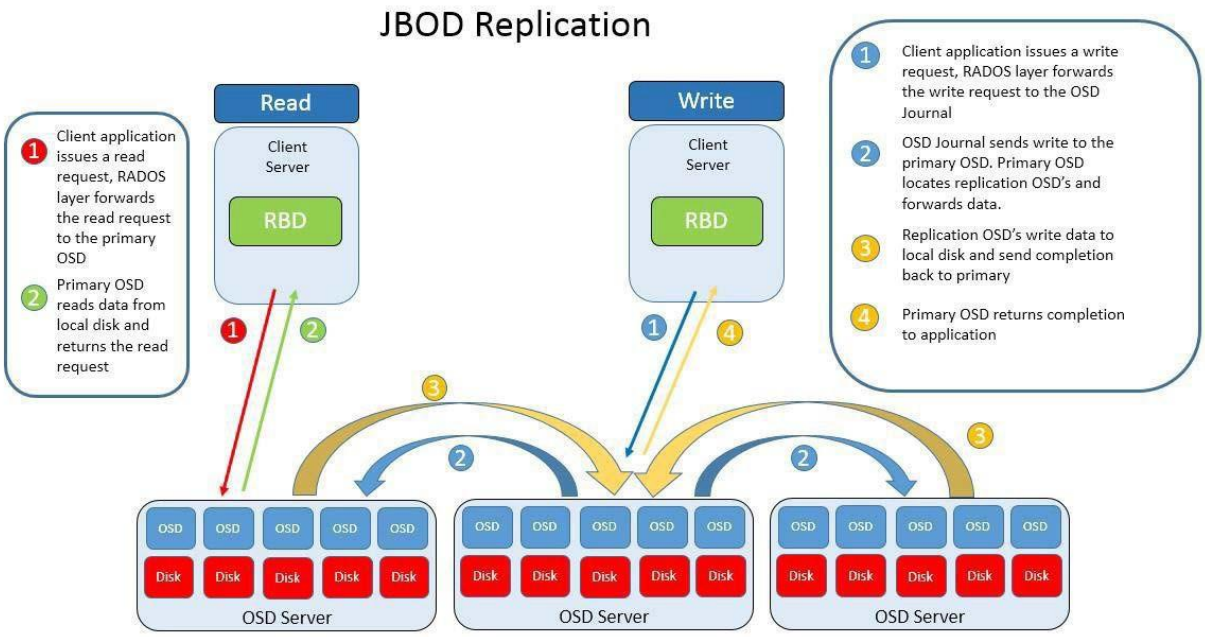
Decreasing Ceph Replicas

By using the RAID engine with E-Series, the default number of Ceph replicas can be decreased from three to two to increase capacity utilization without losing redundancy. Before describing exactly how to reduce the number of replicas, this section covers the default replica behavior that would typically be used when deploying Ceph on JBOD storage.

Ceph Replication for JBOD

Figure 9 depicts the default setting of three replicas for Ceph. Every write is sent from the client to be replicated three times before a completion is sent back to the client that issued the write. Each object is replicated such that any single failure leaves two replicas intact. Any read request will come from the primary OSD holding the data.

Figure 9) Ceph JBOD replication.

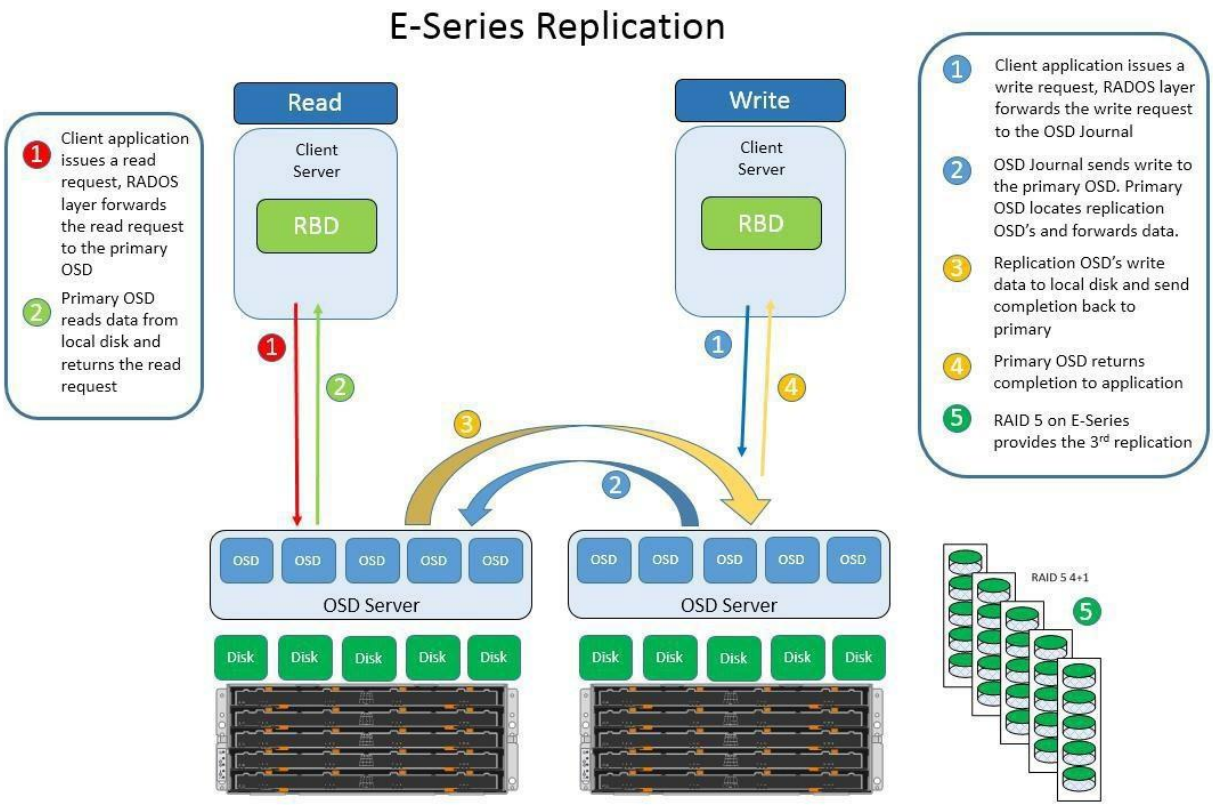


Ceph Replication for E-Series

Using E-Series reduces the number of replicas required from three to two by using the back-end RAID engine as the effective third copy. With two E-Series systems in the configuration, further fault tolerance can be added by making sure that the replication occurs across the two E-Series systems. This is easily configured by modifying the CRUSH map.

Figure 10 shows the flow of data using E-Series arrays.

Figure 10) Ceph E-Series replication.



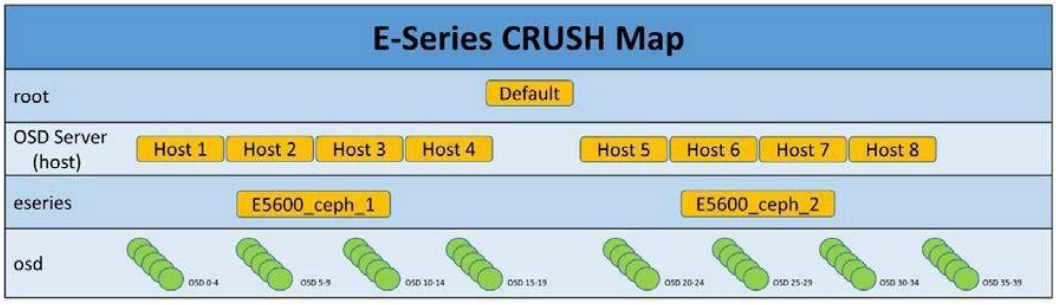
CRUSH Map and Rules for E-Series Replication

The default CRUSH map can be modified to reflect the E-Series Ceph reference cluster described in section 4. The CRUSH map's hierarchy enables users to control how data is replicated based on the attributes of a particular environment. For example, the CRUSH map typically contains types such as OSD, host, rack, row, and data center.

With E-Series, a new type called *eseries* is inserted between host and OSD. The CRUSH map modification is used to specify that data replicas are created on a different E-Series system.

In the reference architecture there are two separate E5600 systems. Each E5600 supports 4 OSD servers (hosts), with 5 OSDs mapped to each server. Combined, all 8 servers create a total of 40 OSDs in the cluster. Figure 11 shows the E-Series CRUSH map hierarchy.

Figure 11) Example CRUSH map hierarchy with E-Series.



Make the following modifications to the CRUSH hierarchy for E-Series:

- **E-Series type.** Modify one of the entries under types to add an eseries layer to the topology. This layer places the replicas across E5600 systems.
- **Buckets for ODS servers.** Add bucket entries for each E5600 that includes the attached OSD servers.
- **Root entries.** Add entries in the root defaults for each of the E5600 systems.
- **Change noleaf entry to eseries.** In the rules section, change the noleaf entry to eseries.

The following is an example of a CRUSH map. A full example of the CRUSH map is in the appendix section “Sample CRUSH Map.”

```
# types
type 0 osd
type 1 host
type 2 eseries              ← Add entry for E-Series arrays
type 3 rack
...
# Add a bucket for the first array
eseries e5600_ceph_1 {
        id -50          # do not change unnecessarily

        # weight 217.600
        alg straw
        hash 0  # klanier
        item ictk0108r730-1 weight 54.400
        item ictk0108r730-2 weight 54.400
        item ictk0108r730-3 weight 54.400
        item ictk0108r730-4 weight 54.400
…

# Add a bucket for the second array
eseries e5600_ceph_2 {
        id -51          # do not change unnecessarily
        # weight 217.600
        alg straw
        hash 0  # klanier
        item ictk0108r730-7 weight 54.400
        item ictk0108r730-8 weight 54.400
        item ictk0108r730-9 weight 54.400
        item ictk0108r730-10 weight 54.400
}
root default {
        id -1           # do not change unnecessarily
        # weight 435.199
        alg straw
        hash 0# klanier
        item e5600_ceph_1 weight 217.600    ← Add entry to the root default for the first array
        item e5600_ceph_2 weight 217.600    ← Add entry to the root default for the second array

}

# rules
rule replicated_ruleset {
        ruleset 0
        type replicated
        min_size 1
        max_size 10
        step take default
        step chooseleaf firstn 0 type eseries      ← Modify entry to eseries
        step emit
}

# end crush map
```

This configuration results in the data being replicated on each E-Series system.

Note: The CRUSH map is a compiled program within Ceph. It can be modified by using CephCLI or by using the crush tool utility to decompile and recompile the map.

For more information about editing the CRUSH map, see section 10, References.

Default Replicas in ceph.conf

The parameter that defines the default number of replicas is set in the /etc/ceph/ceph.conf configuration file. With E-Series, NetApp recommends changing the value of the osd_pool_default_size from three to two. Changing this option ensures that whenever a new pool is created, it automatically uses two replicas rather than the default value of three.

```
osd_pool_default_size = 2
```

Examples of ceph.conf files are in the appendix section.

5.4 Monitoring Ceph Clusters

Throughout testing, it was critical to closely monitor the status and health of the Ceph cluster and its components. Following is a list of helpful common commands to use. These commands can be run from any of the Ceph OSD or monitor nodes.

- `ceph health`
- `ceph -s`
- `ceph -w`
- `ceph osd tree`
- `ceph osd lspools`

For details about these commands, see the Ceph documentation.

6 Comparison of Ceph Clusters Using E-Series and BOD

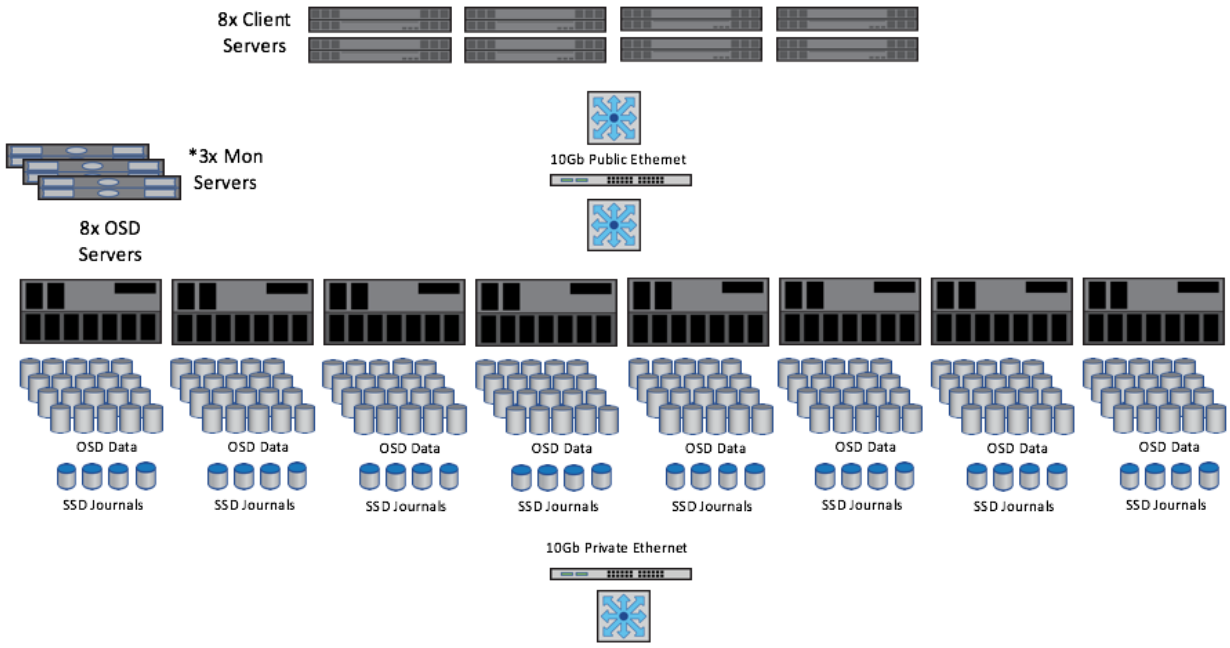
The E-Series–based Ceph cluster was validated against a similarly sized JBOD cluster. This section first describes the JBOD cluster and then draws comparisons to the E-Series cluster.

6.1 Ceph JBOD-Based Cluster

Both the E-Series cluster and the JBOD cluster were built using 8 OSD servers. One of the most common server-based building blocks for Ceph, particularly when used for OpenStack Cinder block storage, is a server with 24 slots. For this reason, we chose to use 24-slot JBODs for comparison. Following the common 5:1 HDD-to-SSD ratio results in 20 NL-SAS drives per server for OSDs and 4 SSDs per server for journals. The E-Series and JBOD cluster configuration used the same 8 client servers. Tests were performed by running with the same test configuration scripts.

Figure 12 shows the Ceph cluster based on JBOD.

Figure 12) Tested Ceph JBOD cluster.



* For testing purposes the Mons co-existed with 3 of the OSD servers.

6.2 Test Methodology

Performance testing was accomplished using the Ceph Benchmarking Tool (CBT) test harness provided with the Ceph installation package on stock distributions of Linux and Ceph. The CBT generates detailed bandwidth and latency output files for each test run.

Performance testing focused on optimizing the cluster for RBD performance. The goal of testing was to place the most demanding block workload possible on the Ceph cluster. The CBT librbdfio benchmark module was used because of its performance characteristics representing a virtual environment, a common area for Ceph.

Each test cycle was executed by calling the cbt.py script using a custom yaml file that contained the various block sizes and I/O types to be tested. See the appendix for an example yaml test file. After each test cycle, the information provided by CBT was collected and tabulated.

All CBT tests were validated by using various host utilities such as dstat, iostat, and top-on OSD servers and monitoring the disk performance by using the SANtricity Performance Monitor GUI.

6.3 Capacity Comparison and OSD Size

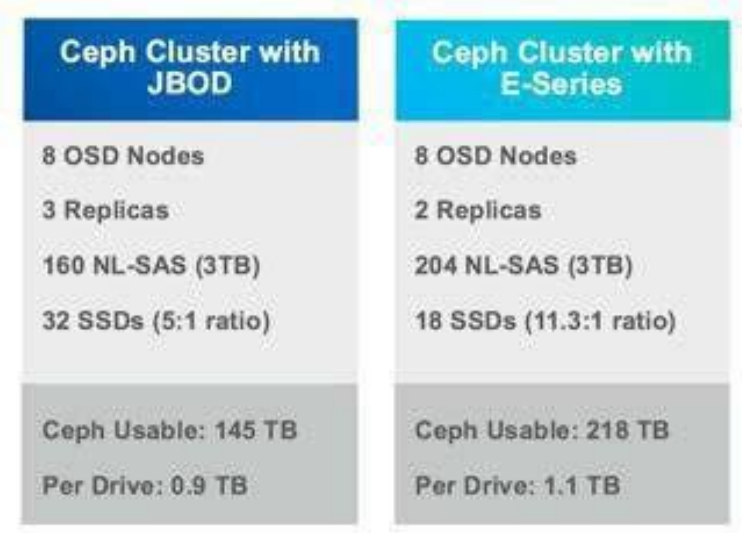
Capacity Increase

In the E-Series reference architecture, each OSD was a RAID 5 (4+1) volume with two replicas. In the JBOD cluster, each OSD was an NL-SAS drive with three replicas. The difference in approaches resulted in a 20% increase in usable capacity per NL-SAS drive.

The tested E-Series cluster also used 50% fewer SSDs for journals. Whereas the JBOD cluster had 1 SSD for every 5 NL-SAS drives, the E-Series cluster had 1 SSD for every 12.5 NL-SAS drives.

Figure 13 shows the capacity differences of comparable E-Series and JBOD Ceph clusters based on these different building blocks. With fewer SSDs and replicas, E-Series provides better hardware utilization.

Figure 13) E-Series and JBOD capacity comparison.



Note: SSDs are not required for journals. If performance is not of primary concern, journals can be collocated on the OSD volume, eliminating the need for SSDs.

Larger OSDs

The approach of using RAID 5 (4+1) volumes naturally makes the OSD size 4 times larger than in a JBOD environment. In addition to the increase in usable capacity, the use of RAID offers a couple of additional benefits. First, there are fewer OSDs to manage, which can help simplify the deployment. Second, the largest Ceph clusters are limited by the OSD count. Reducing the OSD count enables larger Ceph clusters.

6.4 Performance During Media Failure

Every cluster will experience hardware failures at some point, and media is one of the most common sources of failure. Very large clusters are likely to be recovering from some type of media failure the majority of time because of the sheer number of media devices they contain. When designing and building clusters, performance under failure and performance under duress must be taken into consideration.

One significant difference between the architecture of an E-Series Ceph cluster and a JBOD-based Ceph cluster is how media fails. With E-Series, the RAID engine makes use of hot spares so that media failures are transparent to Ceph. With JBOD, a media failure results in a Ceph backfill and recovery operation that moves some of the data within the cluster by using the back-end cluster network.

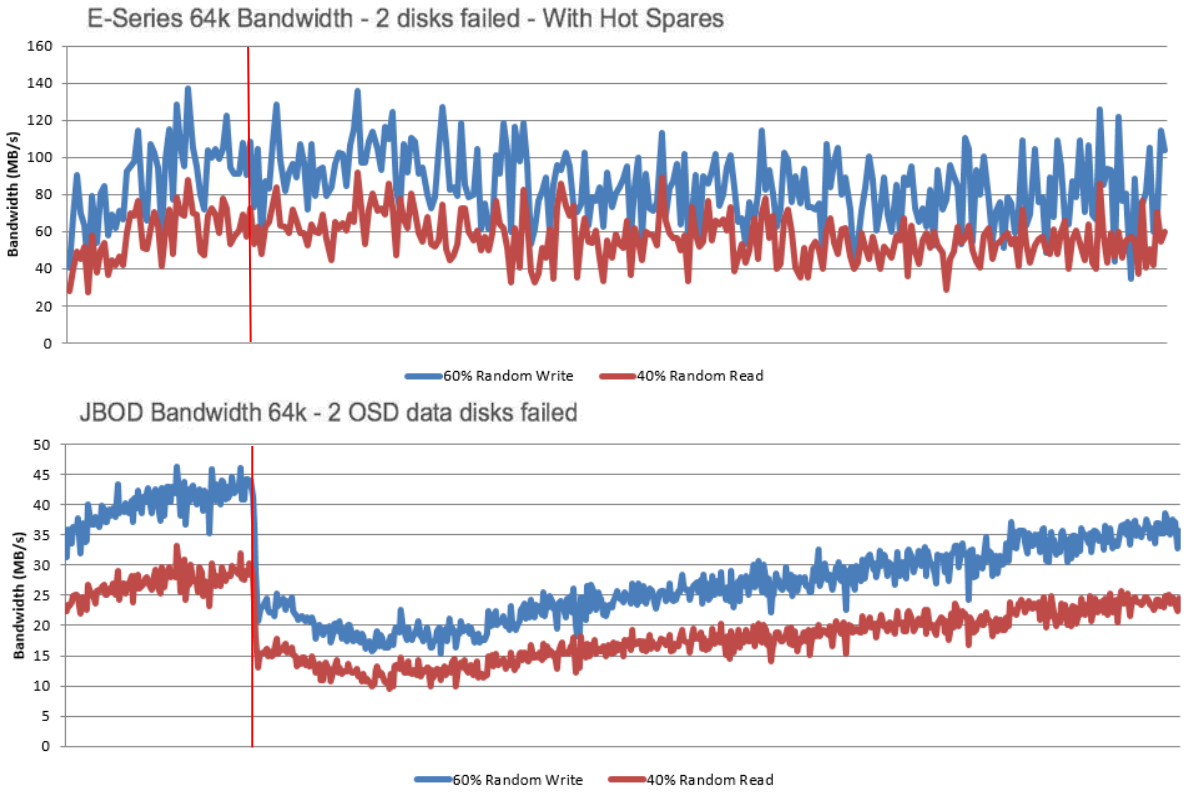
We ran a performance test during media failure to compare the two approaches.

Performance Under Failure Test Results

The test consisted of a 2-hour run with a steady, 60/40% write/read, workload of 64K I/O against a 143TB pool with a single volume populated with 86TB of data. After 20 minutes, two drives were intentionally failed. The test was duplicated between the E-Series Ceph cluster and the JBOD cluster.

Figure 14 illustrates the performance impact that resulted from the two drive failures. The time of drive failure is shown by the red line.

Figure 14) Impact of dual drive failure on Ceph cluster performance.



There was no discernable performance impact on the E-Series. However, the performance with JBOD was decreased by approximately 55% during the backfill and recovery operation.

E-Series also offers benefits from a management standpoint. With E-Series, a media failure is transparent, so the identity of the OSD never needs to change as failed media is replaced. On the other hand, a failed OSD in the JBOD configuration requires a recovery process. A replacement disk drive triggers a maintenance procedure of removing the old OSD and adding a new OSD.

7 Comparison of Ceph Performance with E-Series and JBOD

A full range of block performance tests were run with the E-Series Ceph cluster and the JBOD clusters in an optimal state. The tests were run using the Ceph Benchmarking Tool (CBT) described in section 6.2. To cover common block use cases, block sizes ranging from 4K to 512K were tested. For each block size, multiple 5-minute tests were run with a Ceph volume size of 30GB.

Note: The medium block sizes of 64K to 256K are most applicable when sizing for OpenStack Cinder.

Table 8 shows the suite of tests and block sizes.

Table 8) Ceph CBT test matrix for block testing.

I/O Type	Range of Block Sizes			
Read	4k	64k	256k	512k
Write	4k	64k	256k	512k
Random Read	4k	64k	256k	512k
Random Write	4k	64k	256k	512k

All performance metrics were measured from the cluster level. Notice that cluster performance is not simply the sum of the E-Series building blocks. This is particularly true in the case of writes resulting from write amplification.

Ceph Write Amplification

When planning storage for a Ceph cluster, it is important to be aware of Ceph's write amplification. A single write from the Ceph perspective becomes multiple writes from the storage perspective because of replicas and journals. For example, with the replica count set to two, each Ceph write must write to two journals and two OSDs, so there is a 4-times write amplification. With the default replica value of three, there is a 6-times write amplification. In general, each Ceph write results in an amplification of 2 times the replica value.

Note: Write amplification is one of the reasons that NetApp recommends two E5600 systems for the reference cluster, to avoid saturating the bandwidth of a single system.

7.1 Ceph Block Throughput Test Results with E-Series and JBOD

The results in this section show throughput at increasing block sizes for the E-Series cluster and JBOD cluster for both sequential and random reads and writes. In three of the four cases, the E-Series outperformed the JBOD cluster.

Figure 15) Impact on throughput of increasing block size for sequential reads.

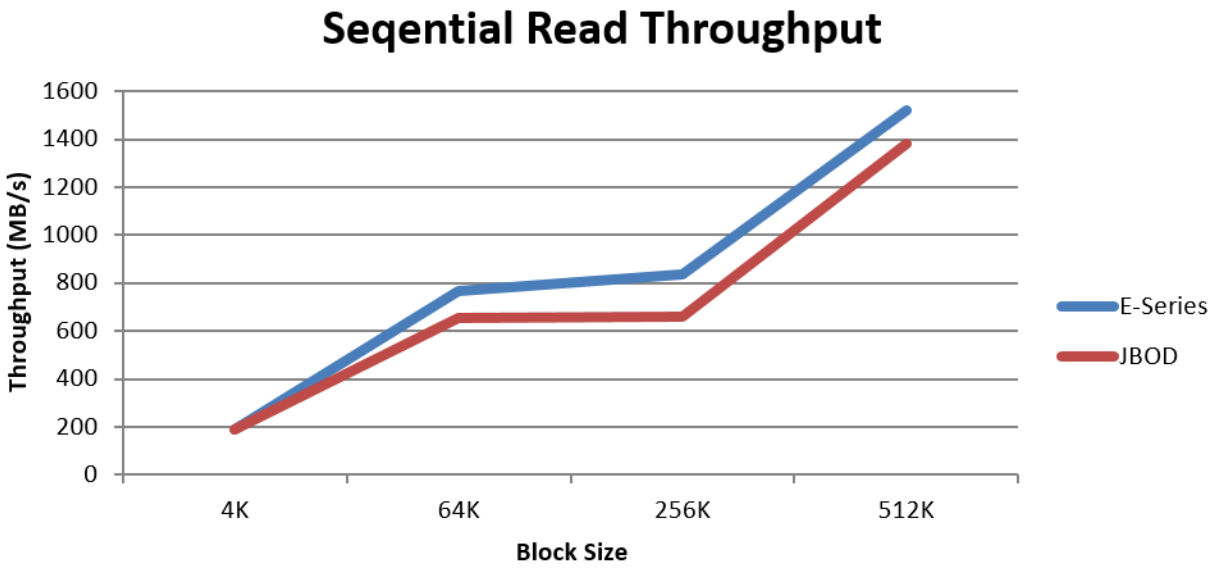


Figure 16) Impact on throughput of increasing block size for random reads.

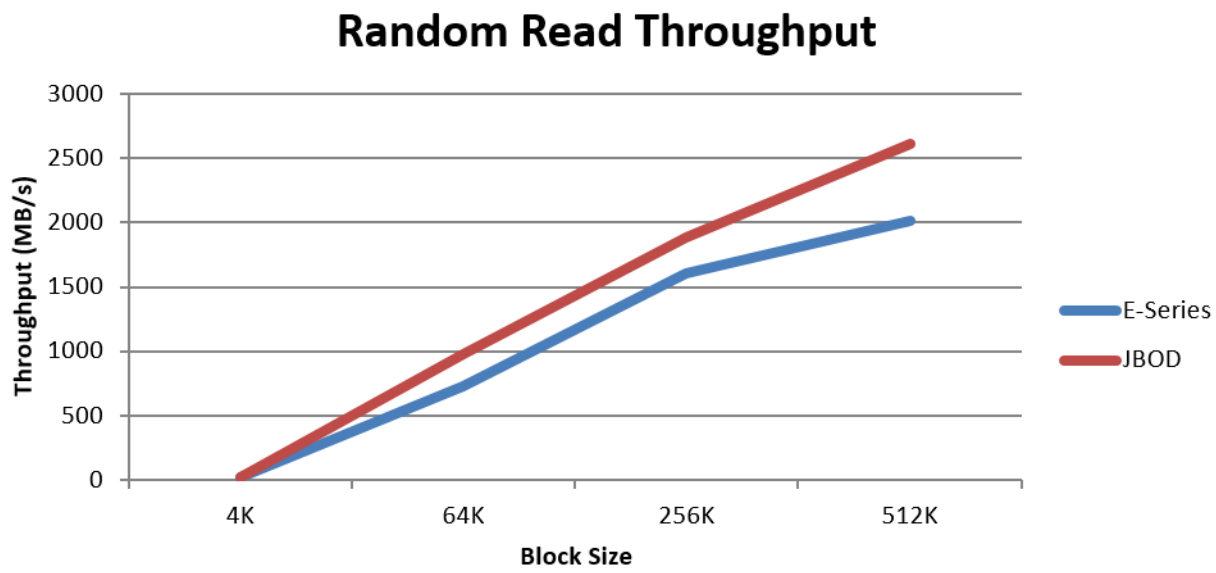


Figure 17) Impact on throughput of increasing block size for sequential writes.

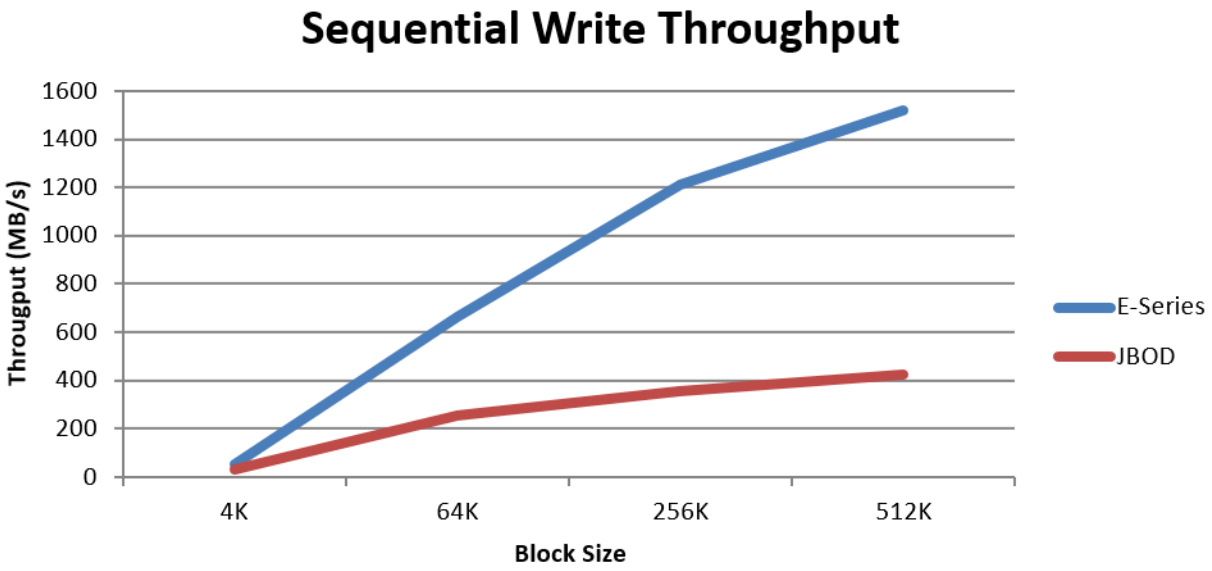
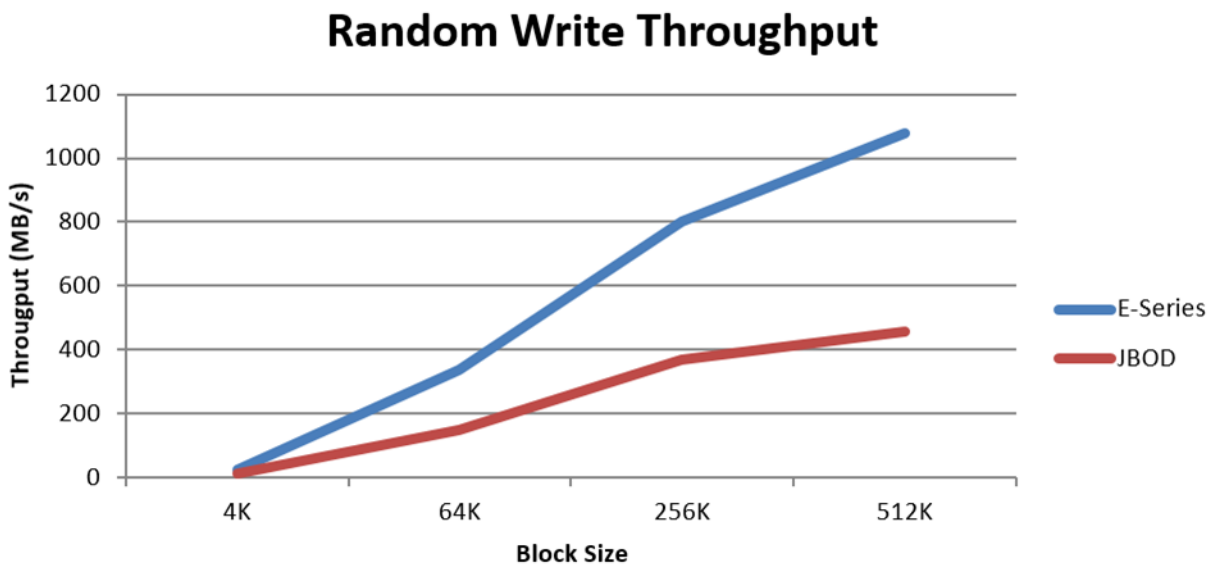


Figure 18) Impact on throughput of increasing block size for random writes.



7.2 Ceph Block IOPS Test Results with E-Series and JBOD

The results in this section show IOPS at increasing block sizes for the E-Series cluster and JBOD cluster for both sequential and random reads and writes. Again, in three of the four cases, the E-Series was on par with or outperformed the JBOD cluster.

Figure 19) Impact on IOPS of increasing block size for sequential reads.

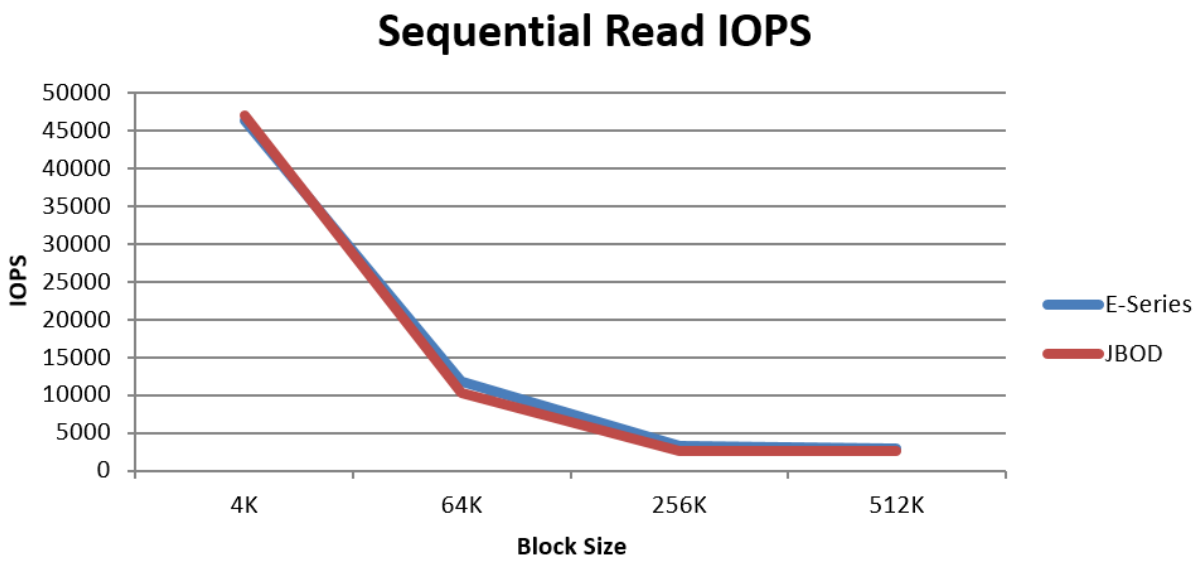


Figure 20) Impact on IOPS of increasing block size for random reads.

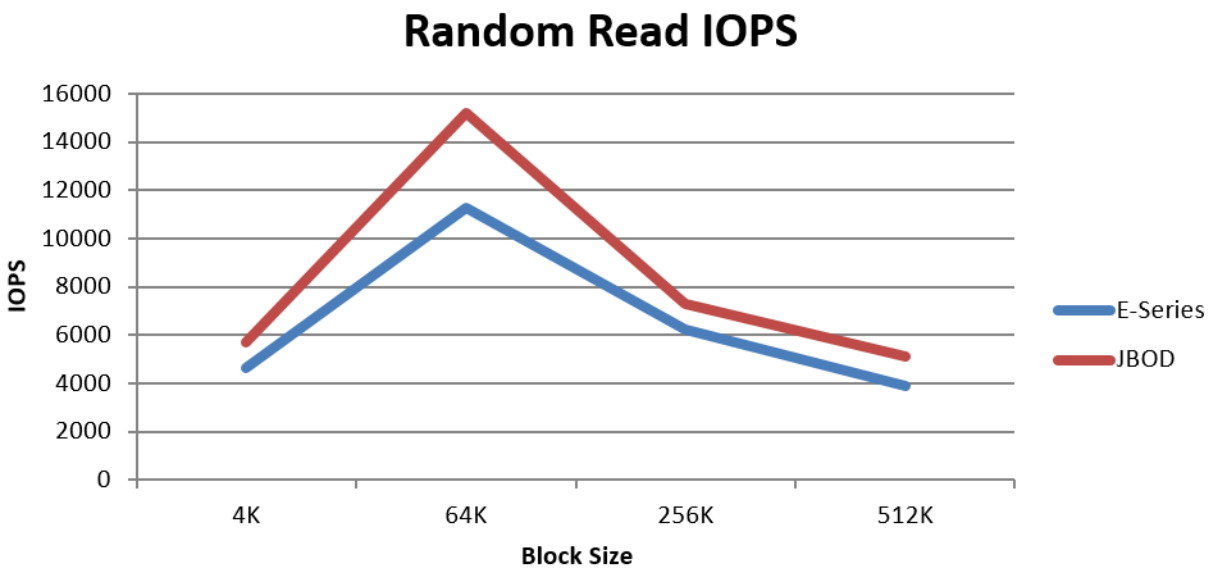


Figure 21) Impact on IOPS of increasing block size for sequential writes.

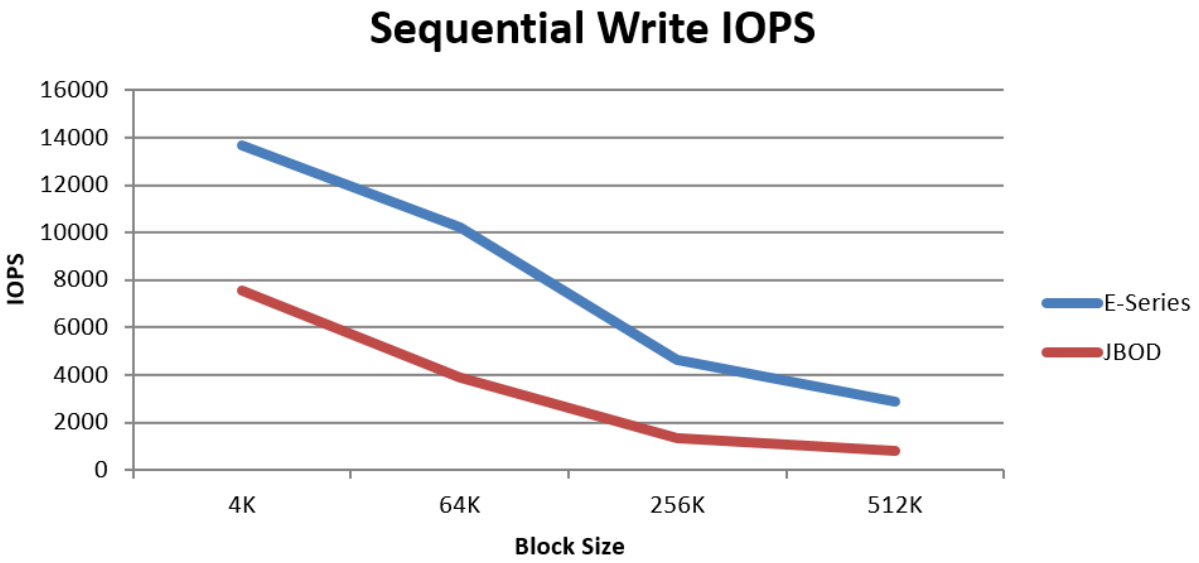
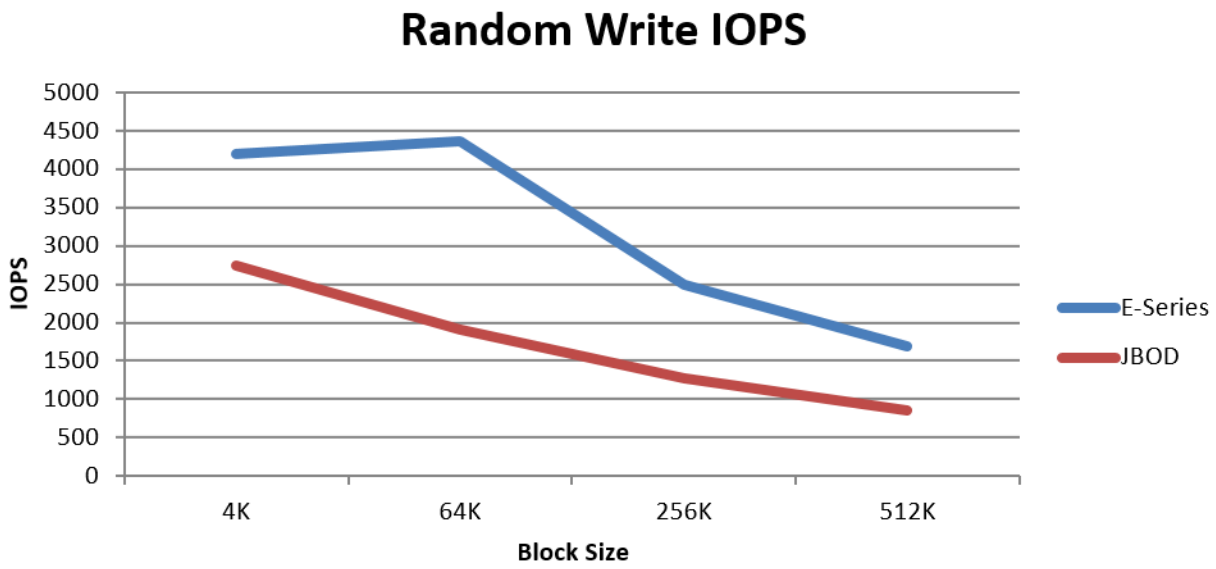


Figure 22) Impact on IOPS of increasing block size for random writes.



7.3 Ceph Block Latency Test Results with E-Seriesand JBOD

The results in this section show latency at increasing block sizes for the E-Series Ceph cluster and the JBOD Ceph cluster for both sequential and random reads and writes. Just as with throughput and IOPS, the E-Series was on par with or outperformed JBOD in three of the four cases.

Figure 23) Impact on latency of increasing block size for sequential reads.

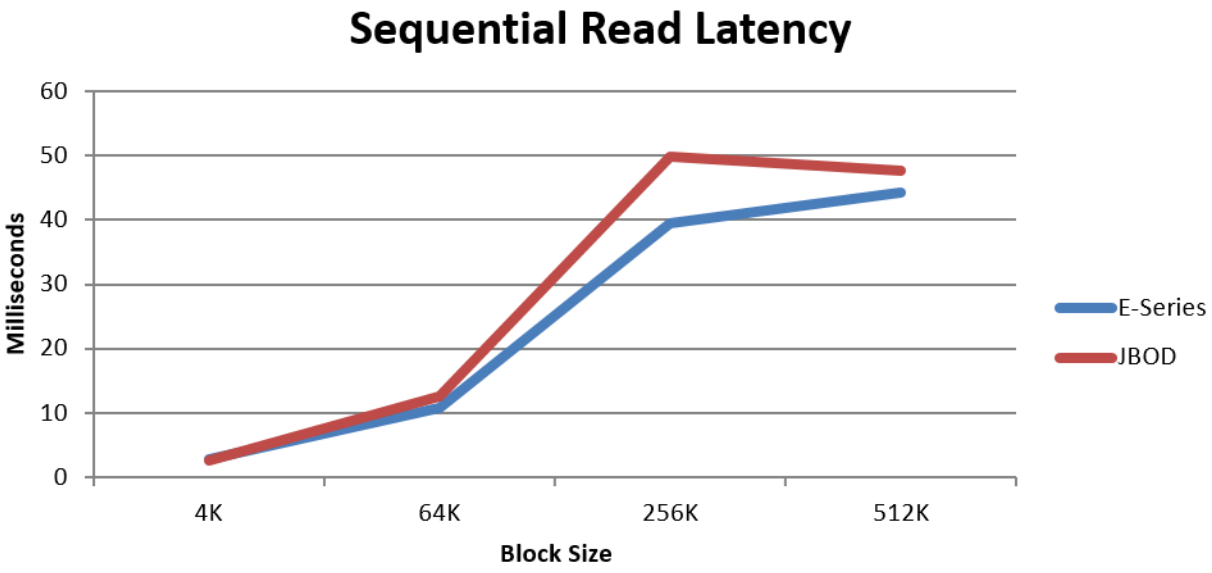


Figure 24) Impact on latency of increasing block size for random reads.

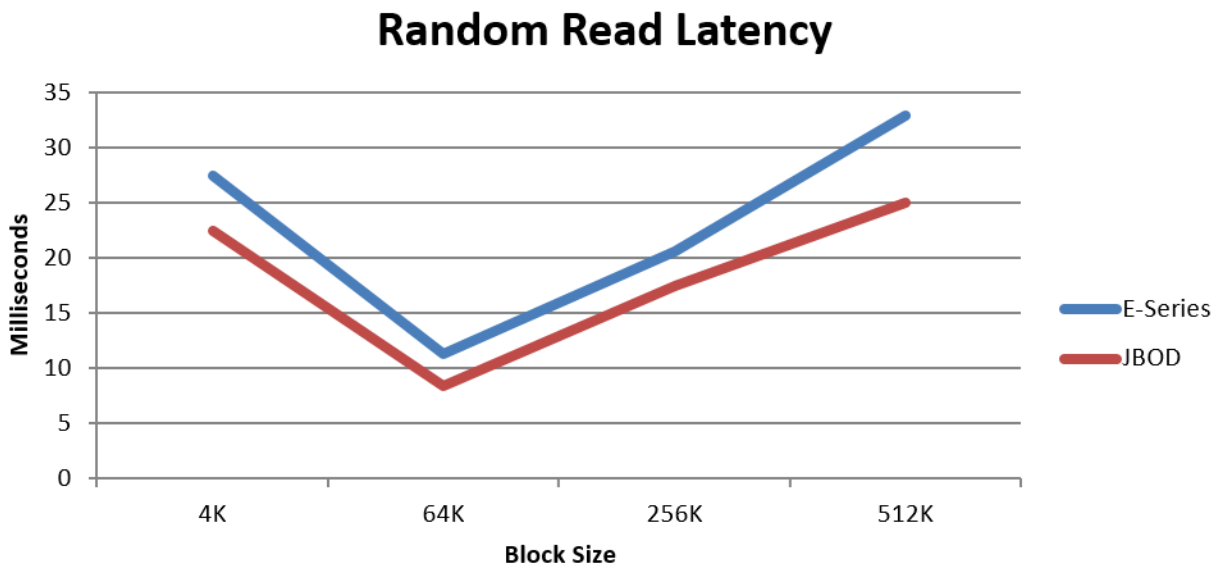


Figure 25) Impact on latency of increasing block size for sequential writes.

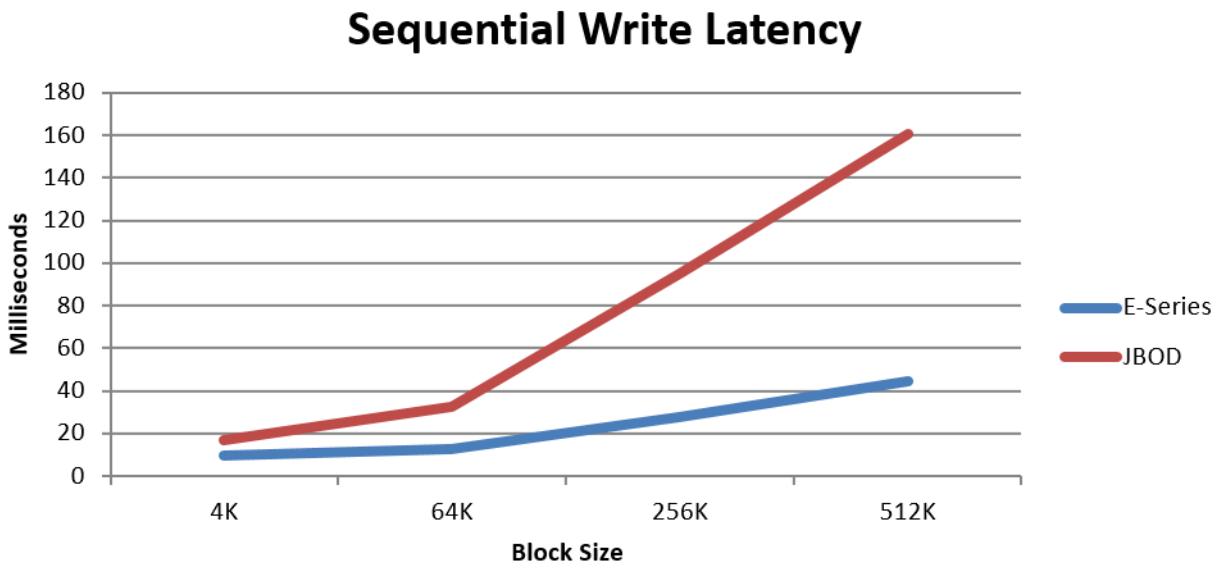
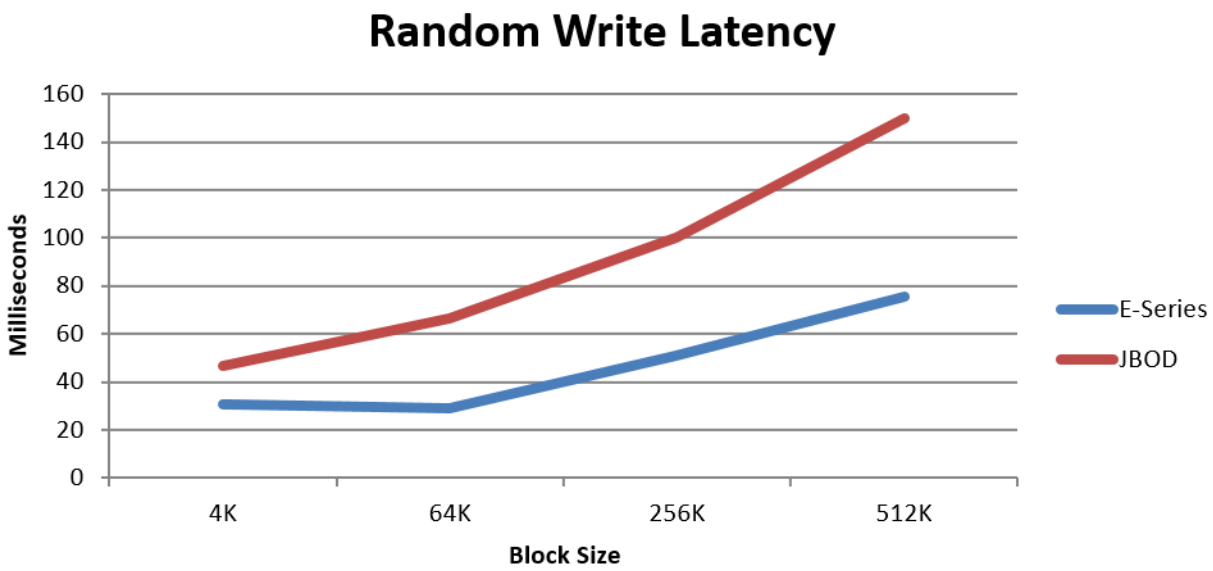


Figure 26) Impact on latency of increasing block size for random writes.



8 Summary

E-Series enterprise storage brings tremendous value to Ceph clusters. Internal storage or JBOD is a common hardware choice when conducting initial Ceph proof of concepts. However, these clusters typically become more complex to manage and tune as the cluster scales. The test results with E-Series demonstrate the benefits of predictable performance during failure as well as increased usable capacity.

The benefits of using E-Series for Ceph include:

- Improved Ceph cluster performance under media failure. There is no noticeable impact with E-Series but up to 50% lower performance with JBOD.
- Transparent drive failures. Recovering from JBOD drive failures requires backfill and recovery.
- Up to 20% increase in usable capacity. Using the E-Series RAID engine decreases the number of Ceph replicas from three to two.
- 99.999% reliability with 4x MTBF over JBOD. E-Series is an enterprise building block with nearly 1,000,000 units deployed.
- RAID-protected SSD journals. With JBOD, a journal failure typically impacts multiple OSD volumes.
- SAS connectivity maintains the fundamental architecture of Ceph.
- Enterprise manageability and support. E-Series offers a call-home feature with automatic case generation.
- SANtricity Storage Manager for simple management, error detection, and recovery.
- REST API and support for orchestration tools.

As Ceph continues to mature and find its way into more production environments, we believe that more people will integrate it with other enterprise technology.

For the latest information about using Ceph with E-Series, visit http://netapp.github.io.

9 Appendixes

Sample E-Series Multipath PrepareOsdPartitions.sh

```
#Author, Kyle Lanier
#this script was executed on osd node-1

#/dev/mapper/360080e5000435a4c0000047456df183d This is the WWN for a mounted RAID-10 SSD LUN
#/dev/mapper/360080e5000435a4c0000043b56df12a7 This is the WWN for a mounded RAID-5 HDD LUN
#/dev/mapper/360080e5000435a4c0000042d56df118d This is the WWN for a mounded RAID-5 HDD LUN
#/dev/mapper/360080e5000435a4c0000043756df126d This is the WWN for a mounded RAID-5 HDD LUN
#/dev/mapper/360080e5000435a4c0000043f56df12e3 This is the WWN for a mounded RAID-5 HDD LUN
#/dev/mapper/360080e5000435a4c0000043356df1230 This is the WWN for a mounded RAID-5 HDD LUN

echo creating the gpt partitions on the RAID-5 HDD LUNs
sudo parted -s /dev/mapper/360080e5000435a4c0000043b56df12a7 mklabel gpt mkpart primary xfs 0% 100%
sudo parted -s /dev/mapper/360080e5000435a4c0000042d56df118d mklabel gpt mkpart primary xfs 0% 100%
sudo parted -s /dev/mapper/360080e5000435a4c0000043756df126d mklabel gpt mkpart primary xfs 0% 100%
sudo parted -s /dev/mapper/360080e5000435a4c0000043f56df12e3 mklabel gpt mkpart primary xfs 0% 100%
sudo parted -s /dev/mapper/360080e5000435a4c0000043356df1230 mklabel gpt mkpart primary xfs 0% 100%

echo creating the gpt partitions on the RAID-10 SSD LUN
sudo parted -s /dev/mapper/360080e5000435a4c0000047456df183d mklabel gpt mkpart ceph-journal-1 '0%' 10GB mkpart ceph-journal-2 10GB 20GB mkpart ceph-
journal-3 20GB 30GB mkpart ceph-journal-4 30GB 40GB mkpart ceph-journal-5 40GB 50GB

echo modifying the partition type code to be used as a ceph journal
sudo sgdisk -t 1:45b0969e-9b03-4f30-b4c6-b4b80ceff106 /dev/mapper/360080e5000435a4c0000047456df183d
sudo sgdisk -t 2:45b0969e-9b03-4f30-b4c6-b4b80ceff106 /dev/mapper/360080e5000435a4c0000047456df183d
sudo sgdisk -t 3:45b0969e-9b03-4f30-b4c6-b4b80ceff106 /dev/mapper/360080e5000435a4c0000047456df183d
sudo sgdisk -t 4:45b0969e-9b03-4f30-b4c6-b4b80ceff106 /dev/mapper/360080e5000435a4c0000047456df183d
sudo sgdisk -t 5:45b0969e-9b03-4f30-b4c6-b4b80ceff106 /dev/mapper/360080e5000435a4c0000047456df183d

echo creating an xfs file system for each RAID-5 HDD LUN gpt partition, optimized for 256k stripe unit with a stripe width of 4
sudo mkfs.xfs -f -d su256k,sw=4 /dev/mapper/360080e5000435a4c0000043b56df12a7p1
sudo mkfs.xfs -f -d su256k,sw=4 /dev/mapper/360080e5000435a4c0000042d56df118d1
sudo mkfs.xfs -f -d su256k,sw=4 /dev/mapper/360080e5000435a4c0000043756df126d1
sudo mkfs.xfs -f -d su256k,sw=4 /dev/mapper/360080e5000435a4c0000043f56df12e3p1
sudo mkfs.xfs -f -d su256k,sw=4 /dev/mapper/360080e5000435a4c0000043356df1230p1
```

Sample E-Series ceph.conf

```
[global]
fsid = 60007a73-964b-4c5f-ba0a-c15ed1d7f985
mon_initial_members = ictk0108r730-1, ictk0108r730-2, ictk0108r730-3 mon_host =
192.168.124.51,192.168.124.52,192.168.124.53
auth_cluster_required = cephx
auth_service_required = cephx
auth_client_required = cephx
filestore_xattr_use_omap = true
public_network = 192.168.124.0/24
cluster_network = 192.168.126.0/24
osd_pool_default_size = 2
osd_pool_default_min_size = 1
osd_pool_default_pg_num = 2048
osd_pool_default_pgp_num = 2048
mon_pg_warn_max_per_osd = 0
osd_journal_size = 10000
mon_clock_drift_allowed = 10
mon_clock_drift_warn_backoff = 30
setuser_match_path = /var/lib/ceph/$type/$cluster-$id
```

Sample JBOD ceph.conf

```
[global]
fsid = 60007a73-964b-4c5f-ba0a-c15ed1d7f985
mon_initial_members = ictk0105r730-1, ictk0105r730-2, ictk0105r730-3
mon_host = 192.168.124.31,192.168.124.32,192.168.124.33
auth_cluster_required = cephx
auth_service_required = cephx
auth_client_required = cephx
filestore_xattr_use_omap = true
public_network = 192.168.124.0/24
cluster_network = 192.168.126.0/24
osd_pool_default_size = 3
osd_pool_default_min_size = 1
osd_pool_default_pg_num = 4096
osd_pool_default_pgp_num = 4096
mon_pg_warn_max_per_osd = 0
osd_journal_size = 10000
mon_clock_drift_allowed = 10
mon_clock_drift_warn_backoff = 30
setuser_match_path = /var/lib/ceph/$type/$cluster-$id
osd_journal = /dev/disk/by-partlabel/ceph-journal-$id
```

Sample E-Series CreateOSDs.sh

```
#Author Kyle Lanier

echo creating osds on this osd node
osdNode=ictk0108r730-1
echo the osd node will be "$osdNode"

echo retrieving ceph configuration file and admin keyring from the admin node
scp ceph@ictk0108r730-5:/etc/ceph/ceph.conf /etc/ceph/
scp ceph@ictk0108r730-5:/etc/ceph/ceph.client.admin.keyring /etc/ceph/

echo adding osd node host name into the crush map
ceph osd crush add-bucket "$osdNode" host

echo moving the newly added osd node to the default placement within the crushmap
ceph osd crush move "$h" root=default

echo beginning loop to create new osds paired with an ssd journal partition on this osd node
for i in 1 2 3 4 5;
do
        osdDataDrive=xxx
        ssdJournalPartition=xxx
        if [ $i == 1 ]
        then
                osdDataDrive=360080e5000435a4c0000043b56df12a7p1
                ssdJournalPartition=/dev/mapper/360080e5000435a4c0000047456df183d1
        fi
        if [ $i == 2 ]
        then
                osdDataDrive=360080e5000435a4c0000042d56df118d1
                ssdJournalPartition=/dev/mapper/360080e5000435a4c0000047456df183d2
        fi
        if [ $i == 3 ]
        then
                osdDataDrive=360080e5000435a4c0000043756df126d1
                ssdJournalPartition=/dev/mapper/360080e5000435a4c0000047456df183d3
        fi
        if [ $i == 4 ]
        then
                osdDataDrive=360080e5000435a4c0000043f56df12e3p1
                ssdJournalPartition=/dev/mapper/360080e5000435a4c0000047456df183d4
        fi
        if [ $i == 5 ]
        then
                osdDataDrive=360080e5000435a4c00000433356df1230p1
                ssdJournalPartition=/dev/mapper/360080e5000435a4c0000047456df183d5
        fi


        echo the drive will be "$osdDataDrive"

        echo creating a unique uuid for the new osd
        ID=$(uuidgen)
        echo "$ID"

        echo sudo ceph osd create "$ID" $i
        ceph osd create "$ID" $i

        echo sudo mkdir /var/lib/ceph/osd/ceph-$i
        sudo mkdir /var/lib/ceph/osd/ceph-$i

        echo sudo mount -o noatime /dev/mapper/"$osdDataDrive" /var/lib/ceph/osd/ceph-$i
        sudo mount -o noatime /dev/mapper/"$osdDataDrive" /var/lib/ceph/osd/ceph-$i

        echo updating the default ssd journal path in line 20 of the ceph.conf to use to a specific journal partition
        echo sed -i "20s,.*,osd_journal = $ssdJournalPartition," /etc/ceph/ceph.conf
        sed -i "20s,.*,osd_journal = $j," /etc/ceph/ceph.conf

        echo creating a new key to be paired with the unique uuid for the new osd
        echo sudo ceph-osd -i $i --mkfs --mkkey --osd-uuid "$ID"
        sudo ceph-osd -i $i --mkfs --mkkey --osd-uuid "$ID"

        echo allowing monitor nodes to access the new osd keyring
        echo sudo ceph auth add osd.$i osd 'allow *' mon 'allow profile osd' -i /var/lib/ceph/osd/ceph-$i/keyring
        sudo ceph auth add osd.$i osd 'allow *' mon 'allow profile osd' -i /var/lib/ceph/osd/ceph-$i/keyring

        echo adding the new osd to the crush map under "$osdNode"
        echo sudo ceph osd crush add osd.$i 1.0 host="$osdNode"
        sudo ceph osd crush add osd.$i 1.0 host="$osdNode"

        echo granting ceph full ownership and permission for the new osd directories
        echo sudo chown -R ceph:ceph /var/lib/ceph/osd
        sudo chown -R ceph:ceph /var/lib/ceph/osd;

        echo sudo chown -R ceph:ceph /var/log/ceph
        sudo chown -R ceph:ceph /var/log/ceph

        echo sudo chown -R ceph:ceph /var/run/ceph
        sudo chown -R ceph:ceph /var/run/ceph

        echo sudo chown -R ceph:ceph /etc/ceph
        sudo chown -R ceph:ceph /etc/ceph

        echo enabling the systemctl master osd target
        echo sudo systemctl enable ceph-osd.target
        sudo systemctl enable ceph-osd.target;

        echo enabling systemctl for the new osd
        echo sudo systemctl enable ceph-osd@$i
        sudo systemctl enable ceph-osd@$i

        echo starting systemctl for the new osd
        echo sudo systemctl start ceph-osd@$i
        sudo systemctl start ceph-osd@$i
done
```

Sample E-Series CBT Optimal Performance librbdfio_optimal_bench.yaml

```
cluster:
  user: 'root'
  head: "ictk0108r730-5"
  clients: ["ictk0103r720-1", "ictk0103r720-2", "ictk0103r720-3", "ictk0103r720-4", "ictk0103r720-5", "ictk0103r720-6", "ictk0103r720-7", "ictk0103r720-8"]
  osds: ["ictk0108r730-4", "ictk0108r730-7", "ictk0108r730-8", "ictk0108r730-9", "ictk0108r730-10", "ictk0108r730-1", "ictk0108r730-2", "ictk0108r730-3"]
  mons: ["ictk0108r730-1", "ictk0108r730-2", "ictk0108r730-3"]
  osds_per_node: 5
  fs: 'xfs'
  mkfs_opts: '-f -i size=2048 -n size=64k'
  mount_opts: '-o inode64,noatime,logbsize=256k'
  conf_file: '/etc/ceph/ceph.conf'
  ceph.conf: '/etc/ceph/ceph.conf'
  iterations: 1
  rebuild_every_test: False
  clusterid: "ceph"
  tmp_dir: "/root/tmp/cbt"
  pool_profiles:
    cbt-librbdfio:
      pg_size: 2048
      pgp_size: 2048
      replication: 2
benchmarks:
  librbdfio:
      use_existing_volumes: False
      time: 300
      vol_size: 30720
      mode: [write, read, randread, randwrite]
      op_size: [4096,65536,262144,524288,1048576,2097152,4194304]
      concurrent_procs: 1
      iodepth: [16]
      osd_ra:  [4096]
      cmd_path: '/usr/bin/fio'
      pool_profile: 'cbt-librbdfio'
      log_avg_msec: 10000
      iopsavgtime: 10000
      bwavgtime: 10000
```

Sample E-Series CBT Performance Under Media Failure librbdfio_bench.yaml

```
cluster:
  user: 'root'
  head: "ictk0108r730-5"
  clients: ["ictk0103r720-1", "ictk0103r720-2", "ictk0103r720-3", "ictk0103r720-4", "ictk0103r720-5", "ictk0103r720-6", "ictk0103r720-7", "ictk0103r720-8"]
  osds: ["ictk0108r730-4", "ictk0108r730-7", "ictk0108r730-8", "ictk0108r730-9", "ictk0108r730-10", "ictk0108r730-1", "ictk0108r730-2", "ictk0108r730-3"]
  mons: ["ictk0108r730-1", "ictk0108r730-2", "ictk0108r730-3"]
  osds_per_node: 5
  fs: 'xfs'
  mkfs_opts: '-f -i size=2048 -n size=64k'
  mount_opts: '-o inode64,noatime,logbsize=256k'
  conf_file: '/etc/ceph/ceph.conf'
  ceph.conf: '/etc/ceph/ceph.conf'
  iterations: 1
  rebuild_every_test: False
  clusterid: "ceph"
  tmp_dir: "/root/tmp/cbt"
  pool_profiles:
    cbt-librbdfio:
      pg_size: 2048
      pgp_size: 2048
      replication: 2
benchmarks:
  librbdfio:
      use_existing_volumes: True
      time: 7200
      vol_size: 2182000
      mode: [randrw]
      rwmixread: [40]
      op_size: [65536]
      concurrent_procs: 1
      iodepth: [16]
      osd_ra:  [4096]
      cmd_path: '/usr/bin/fio'
      pool_profile: 'cbt-librbdfio'
      log_avg_msec: 10000
      iopsavgtime: 10000
      bwavgtime: 10000
```

Sample Crush Map

```
# begin crush map
 tunable choose_local_tries 0
 tunable choose_local_fallback_tries 0
 tunable choose_total_tries 50
 tunable chooseleaf_descend_once 1
 tunable chooseleaf_vary_r 1
 tunable straw_calc_version 1

 # devices
 device 0 device0
 device 1 osd.1
 device 2 osd.2
 device 3 osd.3
 device 4 osd.4
 device 5 osd.5
 device 6 osd.6
 device 7 osd.7
 device 8 osd.8
 device 9 osd.9
 device 10 osd.10
 device 11 osd.11
 device 12 osd.12
 device 13 osd.13
 device 14 osd.14
 device 15 osd.15
 device 16 osd.16
 device 17 osd.17
 device 18 osd.18
 device 19 osd.19
 device 20 osd.20
 device 21 osd.21
 device 22 osd.22
 device 23 osd.23
 device 24 osd.24
 device 25 osd.25
 device 26 osd.26
 device 27 osd.27
 device 28 osd.28
 device 29 osd.29
 device 30 osd.30
 device 31 osd.31
 device 32 osd.32
 device 33 osd.33
 device 34 osd.34
 device 35 osd.35
 device 36 osd.36
 device 37 osd.37
 device 38 osd.38
 device 39 osd.39
 device 40 osd.40

 # types
 type 0 osd
 type 1 host
 type 2 eseries
 type 3 rack
 type 4 row
 type 5 pdu
 type 6 pod
 type 7 room
 type 8 datacenter
 type 9 region
 type 10 root

# buckets
 host ictk0108r730-1 {
          id -7                    # do not change unnecessarily
          # weight 5.000
          alg straw
          hash 0          #kylelanier1
          item osd.26 weight 1.000
          item osd.27 weight 1.000
          item osd.28 weight 1.000
          item osd.29 weight 1.000
          item osd.30 weight1.000
 }
```

```
host ictk0108r730-2 {
        id -8                   # do not change unnecessarily
        # weight 5.000
        alg straw
        hash 0      #kylelanier1
        item osd.31 weight 1.000
        item osd.32 weight 1.000
        item osd.33 weight 1.000
        item osd.34 weight 1.000
        item osd.35 weight1.000
}
host ictk0108r730-3 {
        id -9                   # do not change unnecessarily
        # weight 5.000
        alg straw
        hash 0      #kylelanier1
        item osd.36 weight 1.000
        item osd.37 weight 1.000
        item osd.38 weight 1.000
        item osd.39 weight 1.000
        item osd.40 weight1.000
}
host ictk0108r730-4 {
        id -2                   # do not change unnecessarily
        # weight 5.000
        alg straw
        hash 0      #kylelanier1
        item osd.1 weight 1.000
        item osd.2 weight 1.000
        item osd.3 weight 1.000
        item osd.4 weight 1.000
        item osd.5 weight1.000
}
eseries e5600_ceph_1 {
        id -50                  # do not change unnecessarily
        # weight 217.600
        alg straw
        hash 0      #kylelanier1
        item ictk0108r730-1 weight 54.400
        item ictk0108r730-2 weight 54.400
        item ictk0108r730-3 weight 54.400
        item ictk0108r730-4 weight 54.400
}
host ictk0108r730-7 {
        id -3                   # do not change unnecessarily
        # weight 5.000
        alg straw
        hash 0      #kylelanier1
        item osd.6 weight 1.000
        item osd.7 weight 1.000
        item osd.8 weight 1.000
        item osd.9 weight 1.000
        item osd.10 weight1.000
}
host ictk0108r730-8 {
        id -4                   # do not change unnecessarily
        # weight 5.000
        alg straw
        hash 0      #kylelanier1
        item osd.11 weight 1.000
        item osd.12 weight 1.000
        item osd.13 weight 1.000
        item osd.14 weight 1.000
        item osd.15 weight1.000
}
host ictk0108r730-9 {
        id -5                   # do not change unnecessarily
        # weight 5.000
        alg straw
        hash 0      #kylelanier1
        item osd.16 weight 1.000
        item osd.17 weight 1.000
        item osd.18 weight 1.000
        item osd.19 weight 1.000
        item osd.20 weight1.000
}
```

```
host ictk0108r730-10 {
        id -6                   # do not change unnecessarily
        # weight 5.000
        alg straw
        hash 0      #kylelanier1
        item osd.21 weight 1.000
        item osd.22 weight 1.000
        item osd.23 weight 1.000
        item osd.24 weight 1.000
        item osd.25 weight1.000
}
eseries e5600_ceph_2 {
        id -51                  # do not change unnecessarily
        # weight 168.200
        alg straw
        hash 0      #kylelanier1
        item ictk0108r730-7 weight 54.400
        item ictk0108r730-8 weight 54.400
        item ictk0108r730-9 weight 54.400
        item ictk0108r730-10 weight 5.000

}
root default {
        id -1                   # do not change unnecessarily
        # weight 385.800
        alg straw
        hash 0      #kylelanier1
        item e5600_ceph_1 weight 217.600
        item e5600_ceph_2 weight 168.200

}

# rules
rule replicated_ruleset {
        ruleset 0
        type replicated
        min_size 1
        max_size 10
        step take default
        step chooseleaf firstn 0 type eseries
        step emit
}

# end crush map
```

10 References

The following references were used in this technical report:

Resource	Link
Ceph Website	http://ceph.com
CRUSH: Controlled, Scalable, Decentralized Placement of Replicated Data (white paper)	http://ceph.com/papers/weil-crush-sc06.pdf
Editing the CRUSH Map (Red Hat)	https://access.redhat.com/documentation/en-us/red_hat_ceph_storage/1.2.3/html/storage_strategies/editin
Editing the CRUSH Map (Ceph Community)	http://docs.ceph.com/docs/master/rados/operations/crush-map/
Red Hat Ceph Storage 2 Installation Guide	https://access.redhat.com/webassets/avalon/d/Red_Hat_Ceph_Storage-2-Installation_Guide_for_Red_Hat_Enterprise_Linux-en-US/Red_Hat_Ceph_Storage-2-Installation_Guide_for_Red_Hat_Enterprise_Linux-en-US.pdf
Using Ceph with E-Series	http://netapp.github.io

Refer to the Interoperability Matrix Tool (IMT) on the NetApp Support site to validate that the exact product and feature versions described in this document are supported for your specific environment. The NetApp IMT defines the product components and versions that can be used to construct configurations that are supported by NetApp. Specific results depend on each customer's installation in accordance with published specifications.