Dell EMC Ready Architecture for Red Hat Ceph Storage 3.2

Performance Optimized Block Storage Architecture Guide

Dell EMC Service Provider Solutions

Contents

List of Figures

List of Tables

Trademarks

Notes, Cautions, and Warnings

A **Note** indicates important information that helps you make better use of your system.

A **Caution** indicates potential damage to hardware or loss of data if instructions are not followed.

A **Warning** indicates a potential for property damage, personal injury, or death.

This document is for informational purposes only and may contain typographical errors and technical inaccuracies. The content is provided as is, without express or implied warranties of any kind.

Chapter 1

Introduction

Topics:

Dell EMC has several different Ready Architectures for Red Hat Ceph Storage 3.2 that are designed and optimized to fulfill different objectives. There are architectures for:

- Cost-optimized and balanced block storage with a blend of SSD and NVMe storage to address both cost and performance considerations
- Performance-optimized block storage with all NVMe storage
- Performance- and capacity-optimized object storage, with a blend of HDD and Intel® Optane® storage to provide high-capacity, excellent performance, and cost-effective storage options

This document covers the **Dell EMC Ready Architecture for Red Hat Ceph Storage 3.2 for Performance Optimized Block Storage**.

This chapter gives insight into the key takeaways of deploying the Ready Architecture. It also introduces the readers to the Dell PowerEdge R740xd storage server, as well as the Dell EMC PowerSwitch S5248 switch.

Introduction

Unstructured data has demanding storage requirements across the access, management, maintenance, and particularly the scalability dimensions. To address these requirements, Red Hat Ceph Storage provides native object-based data storage and enables support for object, block, and file storage. Some of the properties are shown in the diagram below.

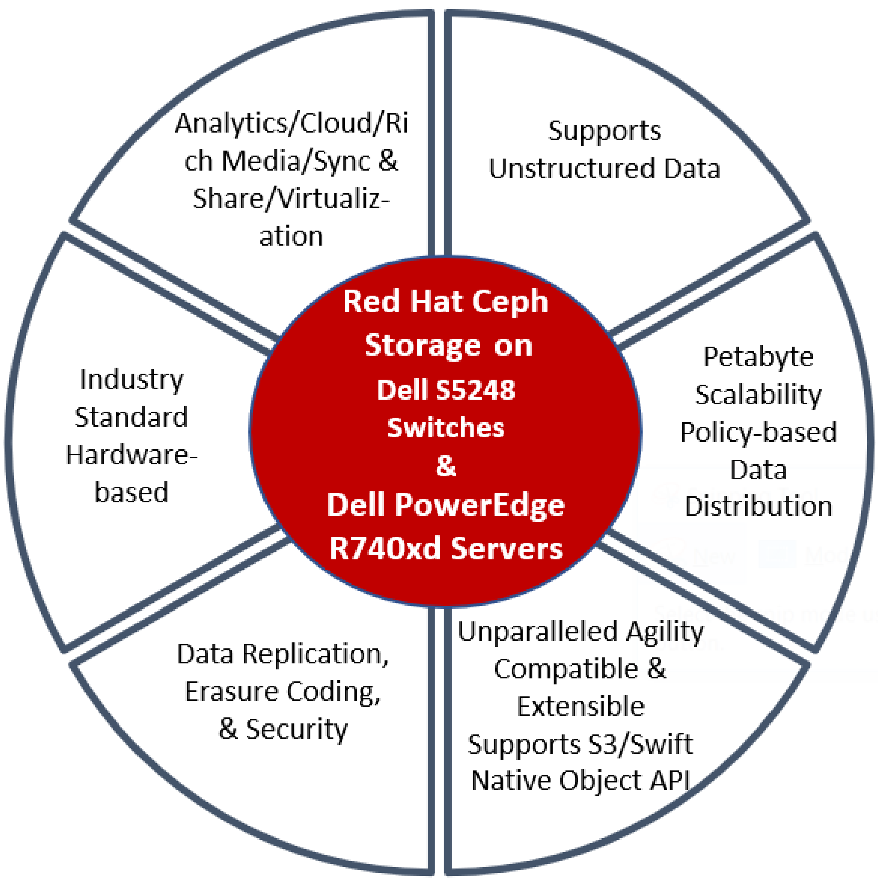

Figure 1: Key takeaways of deploying Red Hat Ceph Storage on Dell EMC PowerEdge R740xd servers

The Red Hat Ceph Storage environment makes use of industry standard servers that form Ceph nodes for scalability, fault-tolerance, and performance. Data protection methods play a vital role in deciding the total cost of ownership (TCO) of a solution. Ceph allows the user to set different data protection methods on different storage pools.

Dell PowerEdge R740xd

The PowerEdge R740xd delivers a perfect balance between storage scalability and performance. The 2U two-socket platform is ideal for Software-defined storage (SDS), service providers or as Virtual desktop infrastructure (VDI).

The scalable system architecture behind the R740xd with up to 24 NVMe drives creates the ideal balance between scalability and performance. The R740xd versatility is highlighted with the ability to mix any drive type to create the optimum configuration of NVMe, SSD and HDD for either performance, capacity or both.

The Dell PowerEdge R740xd offers advantages that include the ability to drive peak performance by:

- Maximizing storage performance with up to 24 NVMe drives and ensures application performance scales to meet demands.
- Freeing up storage space using internal M.2 SSDs optimized for boot.
- Accelerates workloads with up to three double-width 300W GPUs, up to six single-width 150W GPUs or up to four FPGAs.

Dell EMC PowerSwitch S5248F-ON

The S5248 comprises Dell EMC's latest disaggregated hardware and software data center networking solutions, providing state-of-the-art, high-density 25/100GbE ports and a broad range of functionality to meet the growing demands of today's data center environment. It is an ideal choice for organizations looking to enter the software-defined data center era with a choice of networking technologies designed to maximize flexibility.

For applications such as software-defined storage (SDS) requiring the highest bandwidth, the multi-functional 25/100GbE switch is very well suited. This switch can provide high-density Top of Rack (ToR) server aggregation in high-performance data center environments at the desired fabric speed. Some of the features are:

- 1U high-density ToR switch with up to 48 ports of 25GbE, four 100GbE and two 200GbE ports
- Multi-rate 100GbE ports support 100/50/40/25/10GbE
- Line-rate performance via non-blocking switch fabric up to 2.0Tbps
- L2 multipath support via Virtual Link Trunking (VLT) and Routed VLT

Chapter 2

Overview of Red Hat Ceph Storage

Topics:

This chapter introduces the Red Hat software defined storage (SDS) solution Red Hat Ceph Storage (RHCS). It explains the Ceph terminology like pools, placement groups and CRUSH rulesets. Furthermore, it provides details on how to select various components of the solution, including storage access methods and storage protection methods. Finally, it also introduces the new storage backend BlueStore and highlights its features.

Overview of Red Hat Ceph Storage

A Ceph storage cluster is built from a number of Ceph nodes for scalability, fault-tolerance, and performance. Each node is based on industry-standard hardware and uses intelligent Ceph daemons that communicate with each other to:

- Store and retrieve data
- Replicate data
- Monitor and report on cluster health
- Redistribute data dynamically (remap and backfill)
- Ensure data integrity (scrubbing)
- Detect and recover from faults and failures

A few advantages of Red Hat Ceph Storage are:

- Recognized industry leadership in open source software support services and online support
- Only stable, production-ready code, vs. a mix of interim, experimental code
- Consistent quality; packaging available through Red Hat Satellite
- Well-defined, infrequent, hardened, curated, committed 3-year lifespan with strict policies
- Timely, tested patches with clearly-defined, documented, and supported migration path
- Backed by Red Hat Product Security
- Red Hat Certification and Quality Assurance Programs
- Red Hat Knowledgebase (articles, tech briefs, videos, documentation), and Automated Services

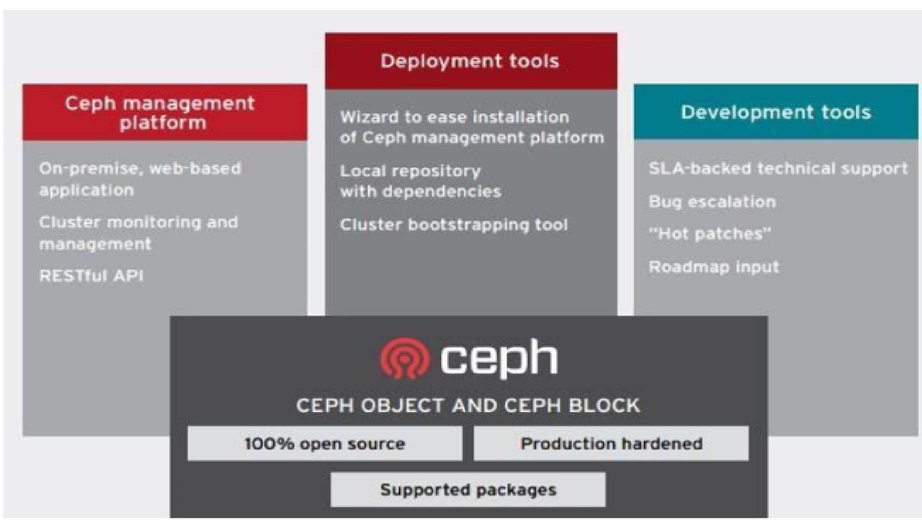

Figure 2: Red Hat Ceph Storage

Red Hat Ceph Storage significantly lowers the cost of storing enterprise data and helps organizations manage exponential data growth. The software is a robust, petabyte-scale storage platform for those deploying public or private clouds. As a modern storage system for cloud deployments, Red Hat Ceph Storage offers mature interfaces for enterprise block and object storage, making it well suited for active archive, rich media, and cloud infrastructure workloads like OpenStack. Delivered in a unified self-healing and self-managing platform with no single point of failure, Red Hat Ceph Storage handles data management so businesses can focus on improving application availability. Some of the properties include:

- Scaling to petabytes
- No single point of failure in the cluster
- Lower capital expenses (CapEx) by running on industry standard server hardware
- Lower operational expenses (OpEx) by self-managing and self-healing

Table 1: Ceph cluster design considerations on page 15 provides a matrix of different Ceph cluster design factors, optimized by workload category. Please see *https://access.redhat.com/documentation/en-us/red_hat_ceph_storage/3/html/configuration_guide/* for more information.

Table 1: Ceph cluster design considerations

Optimization criteria	Potential attributes	Example uses
Capacity-optimized	• Lowest cost per TB • Lowest BTU per TB • Lowest watt per TB • Meets minimum fault domain recommendation (single server is less than or equal to 25% of the cluster)	• Typically object storage • Erasure coding common for maximizing usable capacity • Object archive • Video, audio, and image object archive repositories
Throughput-optimized	• Lowest cost per given unit of throughput • Highest throughput • Highest throughput per Watt • Meets minimum fault domain recommendation (single server is less than or equal to 10% of the cluster)	• Block or object storage • 3x replication • Active performance storage for video, audio, and images • Streaming media

Introduction to Ceph storage pools

For a Ceph client, the storage cluster is very simple. When a Ceph client reads or writes data, it connects to a logical storage pool in the Ceph cluster.

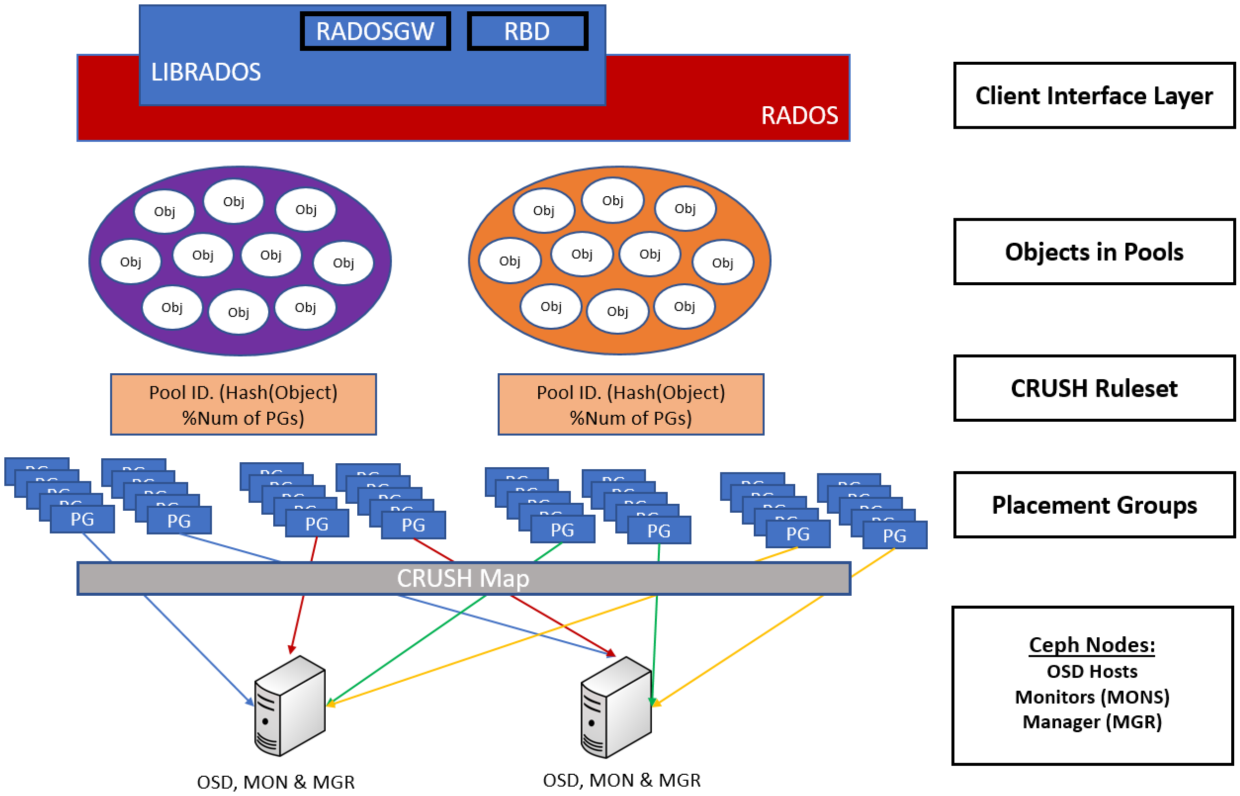

Figure 3: Ceph storage pools

Pools

A Ceph storage cluster stores data objects in logical, dynamic partitions called pools. Pools can be created for particular data types, such as for block devices, object gateways, or simply to separate user groups. The Ceph pool configuration dictates the number of object replicas and the number of placement groups (PGs) in the pool. Ceph storage pools can be either replicated or erasure-coded, as appropriate for the application and cost model. Also, pools can “take root” at any position in the CRUSH hierarchy (see below), allowing placement on groups of servers with differing performance characteristics, encouraging storage to be optimized for different workloads.

Placement groups

Ceph maps objects to Placement Groups (PGs). PGs are shards or fragments of a logical object pool that are composed of a group of Ceph OSD daemons that are in a peering relationship. Placement groups provide a way to create replication or erasure coding groups of coarser granularity than on a per-object basis. A larger number of placement groups (for example, 200/OSD or more) leads to better balancing.

CRUSH rulesets

CRUSH is an algorithm that provides controlled, scalable, and decentralized placement of replicated or erasure-coded data within Ceph and determines how to store and retrieve data by computing data storage locations. CRUSH empowers Ceph clients to communicate with OSDs directly, rather than through a centralized server or broker. By determining a method of storing and retrieving data by algorithm, Ceph avoids a single point of failure, a performance bottleneck, and a physical limit to scalability.

Ceph Monitors (MONs)

Before Ceph clients can read or write data, they must contact a Ceph MON to obtain the current cluster map. A Ceph storage cluster can operate with a single monitor, but this introduces a single point of failure. For added reliability and fault tolerance, Ceph supports an odd number of monitors in a quorum (typically three or five for small to mid-sized clusters). The consensus among various monitor instances ensures consistent knowledge about the state of the cluster.

Ceph OSD daemons

In a Ceph cluster, Ceph OSD daemons store data and handle data replication, recovery, backfilling, and rebalancing. They also provide some cluster state information to Ceph monitors by checking other Ceph OSD daemons with a heartbeat mechanism. A Ceph storage cluster configured to keep three replicas of every object requires a minimum of three Ceph OSD daemons, two of which need to be operational to successfully process write requests.

Ceph dashboard

Ceph Manager has the ability to record many Ceph metrics including the throughput, latency, disk usage, cluster health, and others. Ceph dashboard is a WebUI which can be used to monitor a Ceph cluster. It is powered by the Ceph Manager and provides a detailed visualization of Ceph metrics and cluster status. It's very easy to set up and is available out of the box when Ceph is deployed. It is ideal for monitoring a Ceph cluster with minimum setup effort.

The dashboard currently provides the following features to monitor various aspects of a Ceph cluster:

- Username/password protection
- SSL/TLS support
- Overall cluster health
- Cluster logs
- Hosts
- Performance counters
- Monitors
- Configuration Reference
- Pools
- OSDs
- iSCSI, an Internet Protocol (IP) based storage networking standard for linking data storage facilities
- RADOS Block Devices (RBD) and RBD mirroring
- Ceph Filesystem (CephFS)
- Object Gateway

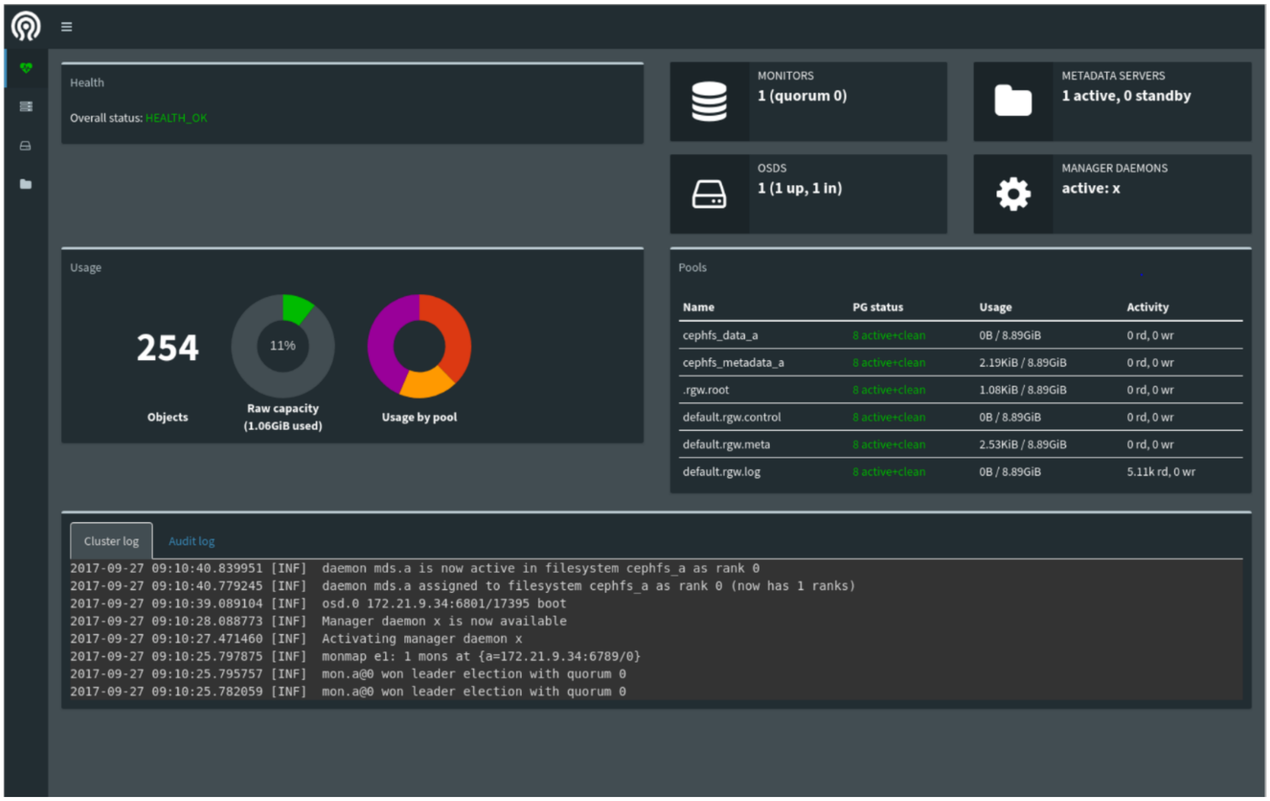

Figure 4: Ceph dashboard

Selecting storage access method

Choosing a storage access method is an important design consideration. As discussed, all data in Ceph is stored in pools, regardless of data type. The data itself is stored in the form of objects using the Reliable Autonomic Distributed Object Store (RADOS) layer which:

- Avoids a single point of failure
- Provides data consistency and reliability
- Enables data replication and migration
- Offers automatic fault-detection and recovery

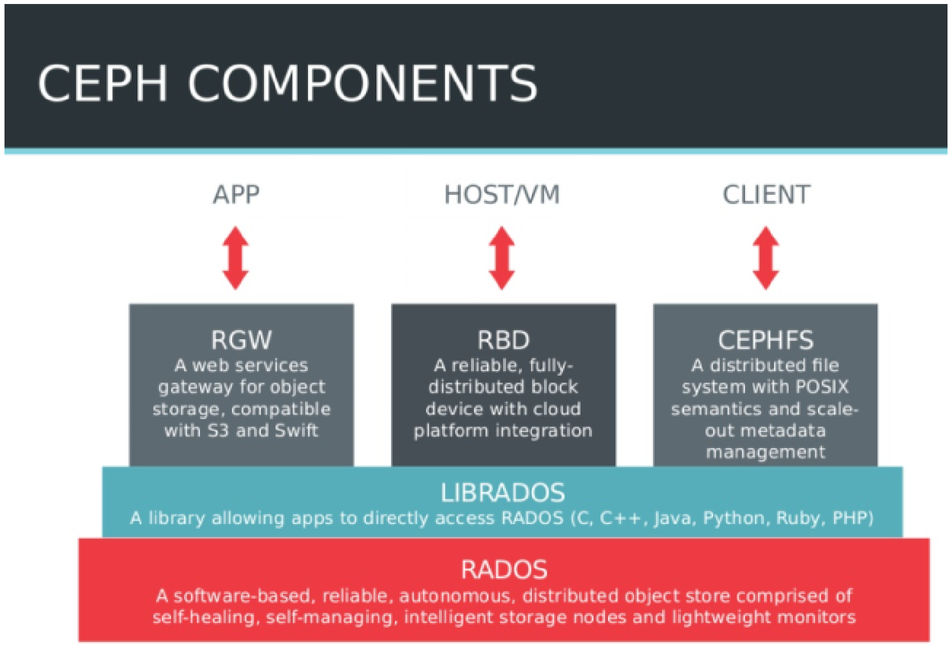

Figure 5: RADOS layer in the Ceph architecture

Writing and reading data in a Ceph storage cluster is accomplished using the Ceph client architecture. Ceph clients differ from competitive offerings in how they present data storage interfaces. A range of access methods are supported, including:

- **RADOSGW** Object storage gateway service with S3 compatible and OpenStack Swift compatible RESTful interfaces
- **LIBRADOS** Provides direct access to RADOS with libraries for most programming languages, including C, C++, Java, Python, Ruby, and PHP
- **RBD** Offers a Ceph block storage device that mounts like a physical storage drive for use by both physical and virtual systems (with a Linux® kernel driver, KVM/QEMU storage backend, or userspace libraries)
- **CephFS** The Ceph Filesystem (CephFS) is a POSIX-compliant filesystem that uses LIBRADOS to store data in the Ceph cluster, which is the same backend used by RADOSGW and RBD.

The storage access method and data protection method (discussed later) are interrelated. For example, Ceph block storage is currently only supported on replicated pools, while Ceph object storage is allowed on either erasure-coded or replicated pools.

Selecting storage protection method

As a design decision, choosing the data protection method can affect the solution's total cost of ownership (TCO) more than any other factor. This is because the chosen data protection method strongly affects the amount of raw storage capacity that must be purchased to yield the desired amount of usable storage capacity. Applications have diverse needs for performance and availability. As a result, Ceph provides data protection at the storage pool level.

Replicated Storage Pools

Replication makes full copies of stored objects and is ideal for quick recovery. In a replicated storage pool, Ceph configuration defaults to a replication factor of three, involving a primary OSD and two secondary OSDs. If two of the three OSDs in a placement group become unavailable, data may be read, but write operations are suspended until at least two OSDs are operational.

Note: Red Hat supports 2x replication on SSD storage devices because of significantly lower failure rates.

Erasure-coded storage pools

Erasure coding provides a single copy of data plus parity, and it is useful for archive storage and cost-effective durability and availability. With erasure coding, storage pool objects are divided into chunks using the n=k+m notation, where k is the number of data chunks that are created, m is the number of coding chunks that will be created to provide data protection, and n is the total number of chunks placed by CRUSH after the erasure coding process. So for instance, n disks are needed to store k disks worth of data with data protection and fault tolerance of m disks.

Ceph block storage is typically configured with 3x replicated pools and is currently not supported directly on erasure-coded pools. Ceph object storage is supported on either replicated or erasure-coded pools. Depending on the performance needs and read/write mix of an object storage workload, an erasure-coded pool can provide an extremely cost effective solution while still meeting performance requirements.

See the Ceph documentation at http://docs.ceph.com/docs/master/architecture/ for more information.

BlueStore

BlueStore is a new backend for the OSD daemons that was introduced in the 'Luminous' release of Ceph. Compared to the traditionally used FileStore backend, BlueStore allows for storing objects directly on raw block devices, bypassing the file system layer. This new backend improves the performance of the cluster by removing the double-write penalty inherent in FileStore.

Table 2: BlueStore/FileStore comparison

OSD backend	Data storage	Internal metadata	Journaling
FileStore	Files within XFS file system	XFS metadata	Journal
BlueStore	Raw volume (no file system)	RocksDB	Write-Ahead Log (WAL)

BlueStore provides the following features and benefits:

- Direct management of storage devices
- Metadata management with RocksDB
- Full data and metadata checksumming
- Inline compression
- Efficient copy-on-write
- No large double-writes
- Multi-device support

Selecting a hardware configuration

As a design decision, choosing the appropriate hardware configuration can have significant effects on the solution’s performance. We provide a performance optimized architecture and it is presented in the next chapter.

Chapter 3

Architecture components

Topics:

This chapter introduces the starter 4-node, 50GbE cluster with containerized Ceph daemons and discusses the rationale for the design. The choices of hardware and software components, along with deployment topology are presented with an explanation of how they support the architectural objectives.

Note: Please contact your Dell EMC representative for sizing guidance beyond this starter kit.

Architecture overview

To handle the most demanding performance requirements for software-defined storage (SDS), we designed an architecture that delivers exceptional performance for Ceph block storage. We make use of Intel® NVMe drives, Intel® Xeon® Platinum CPUs, and x2 Intel® XXV710 based 50GbE networking to achieve very high performance.

The architecture presented in this chapter was designed to meet the following objectives:

- Performance optimized
- Cost savings where possible
- High availability
- Leverage Ceph 3.2 improvements
- Easy to administer

Traditionally, a Ceph cluster consists of any number of storage nodes (for OSD daemons), and three additional nodes to host MON daemons. While the MON daemons are critical for functionality, they have a very small resource footprint. Red Hat Ceph Storage (RHCS) 3 introduced the ability to run Ceph daemons as containerized services. With this, the colocation of MON and OSD daemons on the same server is a supported configuration. This eliminates the need for additional dedicated MON nodes and provides us with a significant reduction in cost.

Since the architecture was designed for high performance block storage with high availability, the components were carefully selected and designed to provide an architecture that is performance-optimized. This, along with the fact that RHCS 3.2 has a much-improved storage backend BlueStore, among other enhancements, allows us to get the most performance from the hardware.

R740xd storage node

We chose the Dell EMC PowerEdge R740xd as it provides the best balance of PCIe slot availability, capacity for internal drives, and performance for Ceph storage nodes. The R740xd server has chassis options to support 3.5" drives and another to support 2.5" drives. The 2.5" drive chassis was selected for this architecture because it provides the most flexibility. As an example, NVMe devices are not supported in drive bays on the 3.5" drive chassis. On the other hand, the 2.5" drive chassis supports a mix of HDD, SSD, and even NVMe when ordered with an NVMe option.

Within the R740xd 2.5" drive chassis, there are two backplane options for U.2 NVMe drives: active (switched) and passive. The active backplane supports up to 24 NVMe U.2 drives in the front drive bays. Since each NVMe drive uses four PCIe lanes (x4), this configuration would require a total of 96 PCIe lanes just for the NVMe drives. Since we're using dual socket servers, this would require each CPU to be able to handle more than 48 lanes each. The Xeon® Skylake CPUs support 48 lanes each. Additional lanes are also needed for PCIe slots on risers and for other functions. In order to overcome this shortage of PCIe lanes, the active backplane multiplexes PCIe lanes across all 24 drives. This multiplexing (switching) has some overhead and introduces latency for all drive operations.

The other backplane option is passive. It does not multiplex any PCIe lanes and does not have any switching logic. Consequently, it does not have any overhead. This backplane allows x16 slots to be extended (using a PCIe bridge card) to four (x4) NVMe U.2 drives. Up to three such x16 slots can be extended, to achieve a total of 12 NVMe drives (four drives per bridge card).

In this architecture we make use of the second backplane option: passive. This allows us to use up to 12 NVMe drives with direct linkage to PCIe slots (no performance hit), and still have the flexibility to use SSD/HDD drives in the unused front drive bays.

Storage devices

Given the objectives of this architecture to provide a performance-optimized configuration, it was decided that all-NVMe devices would best meet the objectives. The Intel® P4610 (see note) was chosen as the NVMe storage device as it is engineered for mixed use workloads.

Note: Our performance testing was conducted with P4600 because the P4610 was not orderable at the time the servers were acquired. Please use P4610 instead of P4600.

Note: A natural question to ask is 'Why not use Optane for Ceph metadata?', since devices like the Intel Optane® P4800X would be a great fit. The reason is because this device was not qualified (orderable) for PowerEdge servers at the time that the equipment was acquired.

The 2.5" drive R740xd chassis provides 24 drive bays on the front of the chassis. Since NVMe devices are typically provisioned in increments of 4 due to the 4:1 nature of the PCIe bridge cards (passive backplane), it was decided to make use of eight NVMe devices. We tested with up to 12 devices but performance degraded due to overassignment of CPU resources. The use of eight drive bays for NVMe devices leaves 16 drive bays available for SSD and HDD drives (up to 4 of these bays can be NVMe).

Table 3: R740xd storage devices

Drive usage	Drive description	Quantity	Drive capacity
Ceph data (OSD and metadata)	Intel® P4610 (see note) NVMe (mixed use)	8	1.6 TB

Note: Our performance testing was conducted with P4600 because the P4610 was not orderable at the time the servers were acquired. Please use P4610 instead of P4600.

Networking

As stated previously, this architecture is based on 25GbE networking components. In accordance with standard Ceph recommendations, two separate networks are used: one for OSD replication, and another for Ceph clients. Standard VLAN tagging is used for traffic isolation. The design includes two Dell S5248 switches for the purpose of high availability. Additionally, two Intel® XXV710 25GbE NICs are installed on each storage node. Each network link is made using dual bonded connections with each switch handling half of the bond. Similarly, each NIC handles half of a bond. In accordance with common Ceph tuning suggestions, an MTU size of 9000 (jumbo frames) is used throughout the Ceph networks.

Aside from the 50GbE Ceph networks, a separate 1GbE network is established for cluster administration and metrics collection. Additionally, a separate 1GbE network is established for iDRAC access.

Table 4: Storage node networking

Network	NIC	Switch	Description
Ceph replication	Intel® XXV710 (dual ports)	Dell S5248F-ON	50GbE (dual bonded 25GbE)
Ceph client			
Provisioning, metrics, iDRAC	i350 QP 1GbE NDC, iDRAC embedded	Dell S3048-ON	1GbE

CPU and memory sizing

As noted above, the architecture is designed with 16 OSDs per node. Current best practices suggest 16 GB of base memory for the OS, with a minimum of 2 GB per OSD. Additionally, it is suggested that a minimum of 1 GB be allocated for each additional Ceph daemon.

Table 5: Sizing memory requirements

Component	Min. RAM per instance (GB)	Recommended RAM per instance (GB)	Instances	Total Min. RAM (GB)	Total Recommended RAM (GB)
Operating system	16	16	1	16	16
Ceph OSD	2	8	16	32	128
Ceph MON	1	1	1	1	1
Ceph MGR	1	1	1	1	1
Total	-	-	-	50	146

The table above illustrates that 50 GB is the minimum memory requirement, with 146 GB as the recommended memory configuration for each storage node. The best performance for memory access in Dell PowerEdge servers is obtained by having all slots in the first memory bank of each CPU populated equally. The R740xd contains a total of 24 memory (DIMM) slots split equally among 2 CPU sockets. The CPU provides six memory channels and the first bank of six slots plug directly into the six CPU memory channels. Since server memory is typically installed in increments of 16 or 32 GB, high performance memory access is achieved by populating each CPU's first memory bank with six 16 GB DIMMs for a total of 192 GB.

Note: Populating all six DIMM slots of each CPU's first memory bank (12 total for both CPUs) provides optimum memory performance.

Current best practices call for approximately five physical CPU cores per NVMe device. Internal lab testing showed that three physical cores per OSD was the appropriate sizing metric with two OSDs per NVMe device. Since there are eight NVMe drives per server with two OSDs per NVMe, approximately 48 CPU cores are needed. Additionally, CPU cores must be available for servicing the operating system and the other Ceph daemons (MON and MGR).

Note: We found that three physical cores per OSD, with two OSDs per device, was the appropriate sizing factor.

Table 6: Sizing CPU physical core requirements

Component	Cores per instance	Instances	Total cores
Operating system	2	1	2
Ceph OSD	3	16 (2 per NVMe)	48
Ceph MON	1	1	1
Ceph MGR	1	1	1
Total	-	-	52

As shown in the above table, the storage nodes require a total of approximately 52 physical CPU cores each. The R740xd is a dual-socket system, allowing the total requirements to be satisfied by two CPUs. Although the total CPU requirements could theoretically be met with 26 core CPUs such as the Xeon®

Platinum 8170, it provides no headroom for other critical, necessary functions such as OSD scrubbing, OSD backfill operations, Meltdown/Spectre/ZombieLoad patches, Ceph authentication, and CRC data checks. In order to achieve this headroom, we chose the 28 core Xeon® Platinum 8176 CPU, giving a total of 56 cores per server and leaving four physical CPU cores to service these additional functions. This CPU is the highest number of cores in the Xeon® Skylake family.

Network switches

Our architecture is based on 50GbE (dual bonded 25GbE) networks for core Ceph functionality. Additionally, we establish a 1GbE network for cluster administration, Ceph monitoring, and iDRAC access.

Table 7: Dell network switches

Dell switch configuration		
Dell EMC PowerSwitch S5248F-ON	Cumulus Linux	50GbE Ceph client and replication networks
Dell EMC PowerSwitch S3048-ON	OS9, 48x 1GbE, 4x SFP+ 1GbE	1GbE cluster admin, metrics/ monitoring, iDRAC network

We chose the Dell EMC PowerSwitch S5248F-ON switch as it's the latest and most advanced Dell EMC switch with 25GbE ports and has enough ports to support a full rack of servers. Each S5248F-ON switch contains 48 ports, giving a total of 96 ports for the pair. Each storage node has four 25GbE links (two for each network) with two link connections per switch. This configuration allows the pair of S5248F-ON switches to support up to 24 storage nodes. A standard full-height rack can hold up to 20 storage nodes. Thus, the pair of S5248F-ON switches can handle a full-rack of storage nodes.

Note: Multi-tier networking is required to handle more than 20 storage nodes. Please contact Dell EMC Professional Services for assistance with this more advanced configuration.

Storage node Ceph NICs

As mentioned earlier, the architecture contains a pair of S5248F-ON switches that are used for both Ceph networks. Additionally, each storage node has a pair of Intel XXV710 25GbE dual port NICs.

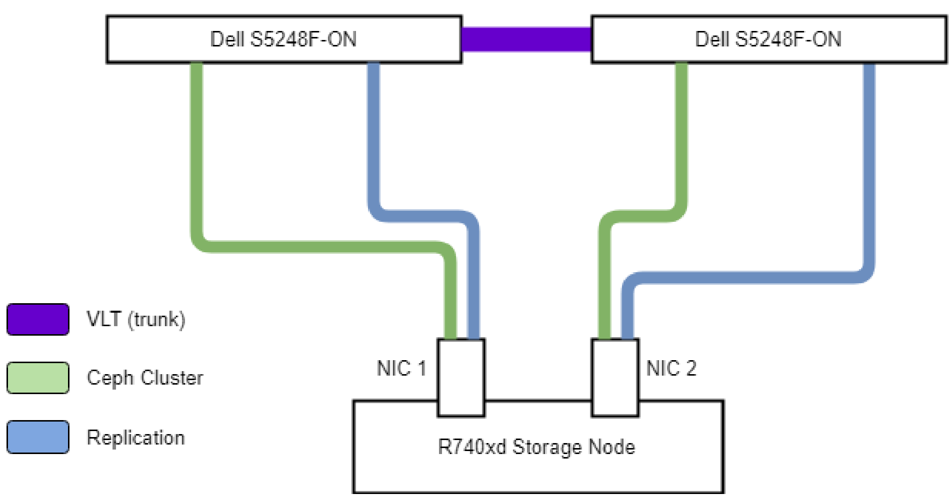

Figure 6: Storage node Ceph networking

The figure above shows how these components are used to integrate the storage nodes into the Ceph networks. As shown in the figure, each network is spread across both switches and both NICs on each storage node. This design provides high availability and can withstand the failure of a NIC, cable, or switch. Additionally, LAG bonds are established for each pair of NIC ports for their respective networks.

Storage node PCIe/NUMA considerations

In order to get the best possible performance, it's necessary to configure devices within the chassis so as to spread the processing load across both CPUs. It's important to separate the two XXV710 NICs and the two PCIe bridge cards. The following figure illustrates how best to balance the devices across the CPUs.

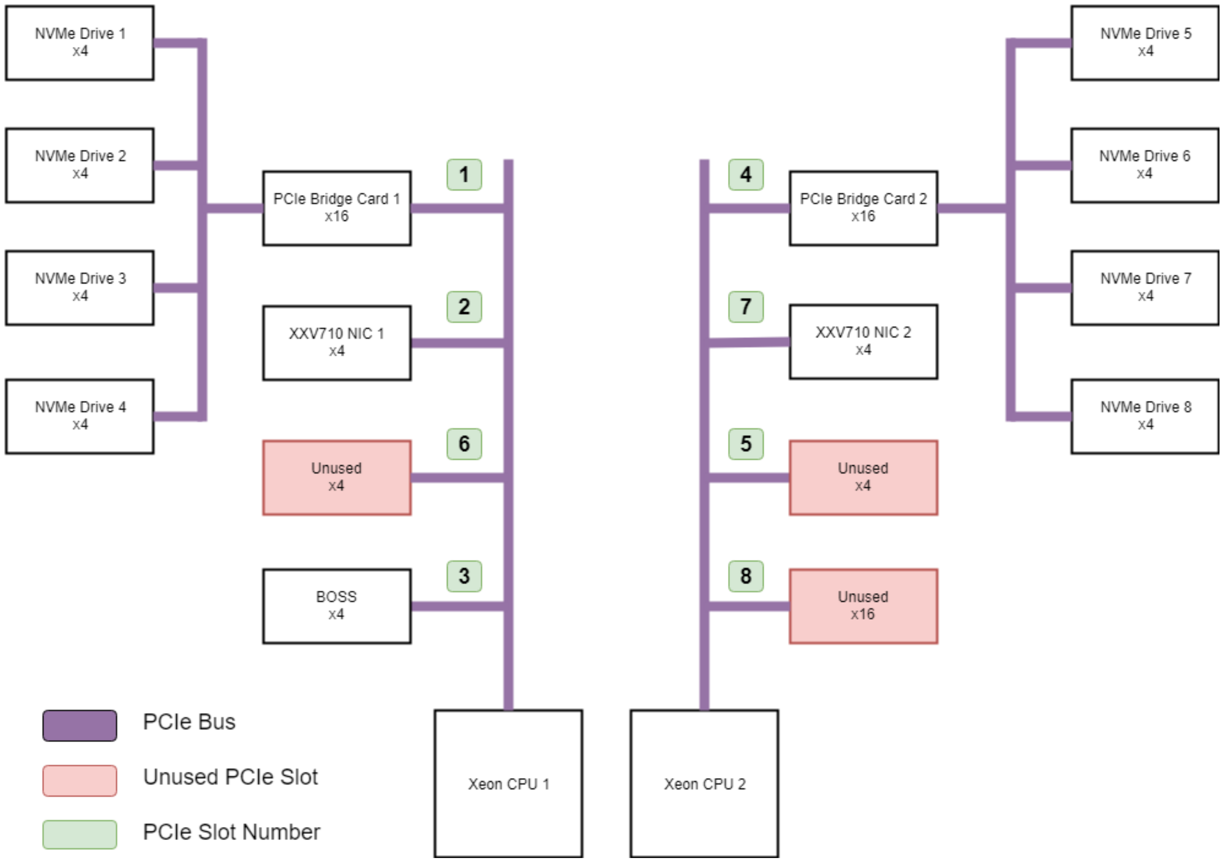

Figure 7: Storage node PCIe slot assignments

The storage nodes used in this architecture make use of Riser Config 6. Each specific riser configuration will have its own set of CPU assignments for PCIe slots. Consulting the specific system diagram is necessary to know these CPU assignments.

Storage node hardware configuration

Table 8: Storage node hardware configuration

Component	Details
Platform	Dell EMC PowerEdge R740xd
CPU	2x Intel® Xeon® Platinum 8176 2.1 GHz
Cores per CPU	28
Memory	192 GB (12x 16GB RDIMM, 2666MT/s)

Component	Details
50GbE (dual bonded 25GbE) network	2x Intel® XXV710 dual port 25GbE SFP28
1GbE network	i350 quad port 1GbE, rNDC
NVMe data storage	8x Intel® P4610 (see note) 1.6TB mixed use
OS storage	BOSS (2x M.2 Sticks 240G RAID 1)

Note: Our performance testing was conducted with P4600 because the P4610 was not orderable at the time the servers were acquired. Please use P4610 instead of P4600.

R640 admin node

Aside from the R740xd nodes, a single R640 is included in the architecture to provide two functions: cluster administration and Ceph monitoring. This node is referred to as the 'admin node'. The admin node has network connectivity as shown in the following table. Since this node is only connected to the 50GbE client network, it only needs a single Intel® XXV710 NIC for provisioning and metrics collection network.

Table 9: Admin node networking

Network	NIC	Switch	Description
Ceph client	Intel® XXV710 (dual ports)	Dell S5248F-ON	50GbE (dual bonded 25GbE)
Provisioning, metrics, iDRAC	i350 QP 1GbE NDC, iDRAC embedded	Dell S3048-ON	1GbE

Note: The monitoring/metrics use the 1GbE network to isolate any extra traffic from the client network used by load generators.

Table 10: Admin node hardware configuration

Component	Details
Platform	Dell EMC PowerEdge R640
CPU	2x Intel® Xeon® Gold 6126 2.6 GHz
Memory	192 GB (12x 16GB RDIMM 2666MT/s)
50GbE (dual bonded 25GbE) network	1x Intel® XXV710/2P
1GbE network	i350 QP 1GbE NDC
Storage devices	8x 10K SAS 600 GB HDD
RAID controller	PERC H740P

Number of nodes

Traditionally, a tiny production Ceph cluster required a minimum of seven nodes, three for Ceph MON and at least four for Ceph OSD. The recent ability to deploy colocated, containerized Ceph daemons has significantly reduced these minimum hardware requirements. By deploying Ceph daemons colocated and containerized, one can eliminate the need for three physical servers.

Our architecture makes use of five physical nodes, four of them for running Ceph storage and one for administrative purposes. We refer to the one with administrative duties as the 'admin node'. The admin node provides the following important functions:

- Collection and analysis of Ceph and server metrics (Ceph dashboard)
- ceph-ansible deployment
- Administration of all Ceph storage nodes (ssh and iDRAC)

Beyond the admin node, the architecture consists of four storage nodes. We consider four storage nodes to be the absolute minimum to be used in production, with five storage nodes being a more pragmatic minimum. The five nodes (four storage + one admin) in our architecture is a starting point for new deployments. The number of storage nodes should be based on your capacity and performance requirements. This architecture is flexible and can scale to multiple racks.

Note: Please contact your Dell EMC representative for sizing guidance beyond this starter kit.

Rack component view

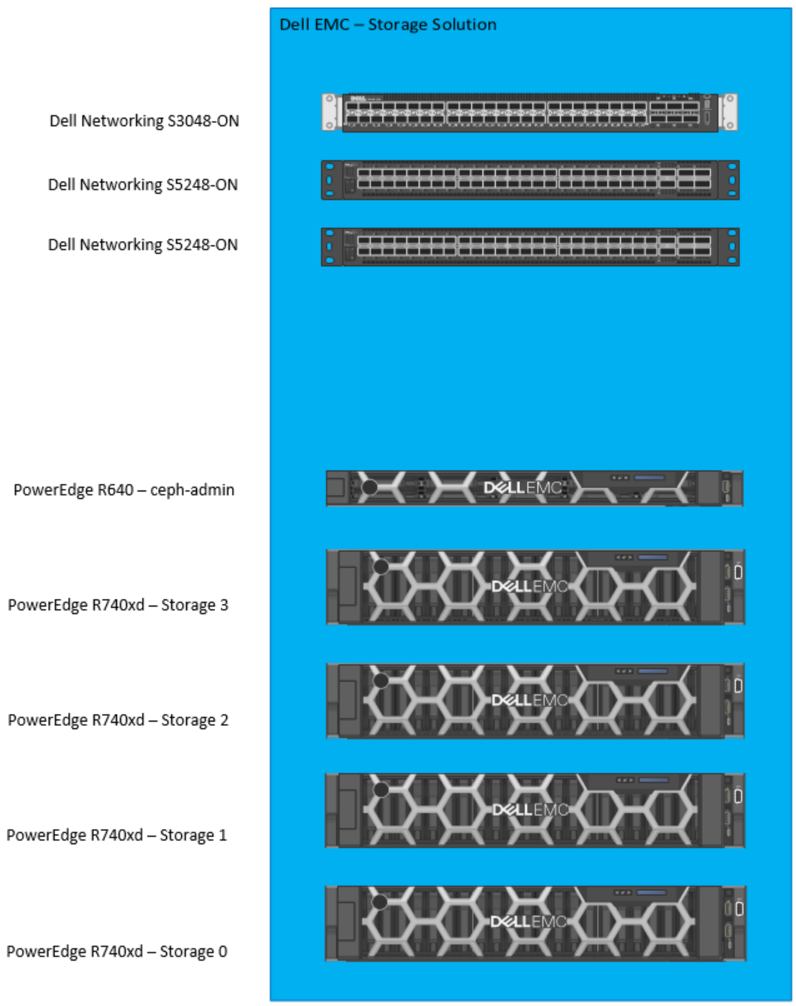

Figure 8: The 4-Node Ceph cluster and admin node based on Dell PowerEdge R740xd and R640 servers

Note: We recommend that heavier storage nodes be located at the bottom of the rack.

Note: We use four storage nodes as a starting point. You should consider five or more for production.

The R740xd is a 2U server, while the R640 and the switches are each 1U. Taken as a whole, the cluster of four storage nodes and one admin node requires 9U for servers and 3U for switches.

It is worth noting that if the MON daemons were not containerized and colocated (with OSD), the rack space requirements would be increased by 3U (e.g., 3 R640). This deployment topology provides significant cost savings and noticeable rack space savings.

Software

Table 11: Architecture software components/configuration

Component	Details
Operating system	Red Hat Enterprise Linux (RHEL) 7.6
Ceph	Red Hat Ceph Storage (RHCS) 3.2
OSD backend	BlueStore
OSDs per NVMe	2
Placement groups (single pool only)	8192
CPU logical cores per OSD container	6
Ceph storage protection	2x replication (see note)
Ceph daemon deployment	Containerized and colocated

Note: Red Hat supports 2x replication on SSD storage devices because of significantly lower failure rates.

Architecture summary

The architecture presented in this chapter was designed to meet specific objectives. The following table summarizes how the objectives are met.

Table 12: Architecture objectives

Objective	How met
Performance optimized	• Intel® Xeon® Platinum 8176 (28C) CPUs • 50GbE (dual bonded 25GbE) networking • Intel® P4610 (see note) NVMe devices • Separate replication network • Jumbo frames enabled
Cost savings where possible	• Colocated daemons (reduce server count) • 25GbE networking components (sweet spot for cost/performance)

Objective	How met
High availability	• Redundant 25GbE switches • Redundant 25GbE NICs • Hot-plug storage devices (including NVMe) • Redundant power supplies
Leverage Ceph 3.2 improvements	• BlueStore OSD backend • Containerized daemons
Easy to administer	• iDRAC9 remote server admin • Separate, integrated Ceph admin node • Ceph dashboard (Grafana based)

Note: Our performance testing was conducted with P4600 because the P4610 was not orderable at the time the servers were acquired. Please use P4610 instead of P4600.

Chapter 4

Test setup

Topics:

This chapter highlights the procedure used to setup the storage cluster along with other components. It includes physical setup, configuration of servers, and deployment of RHEL and RHCS.

Physical setup

The equipment was installed as shown below. When installing the 25GbE NICs in the servers, care needs to be taken to ensure the cards are on separate NUMA nodes. This ensures that the traffic is handled by different CPUs for individual NICs and the traffic is spread across the CPUs.

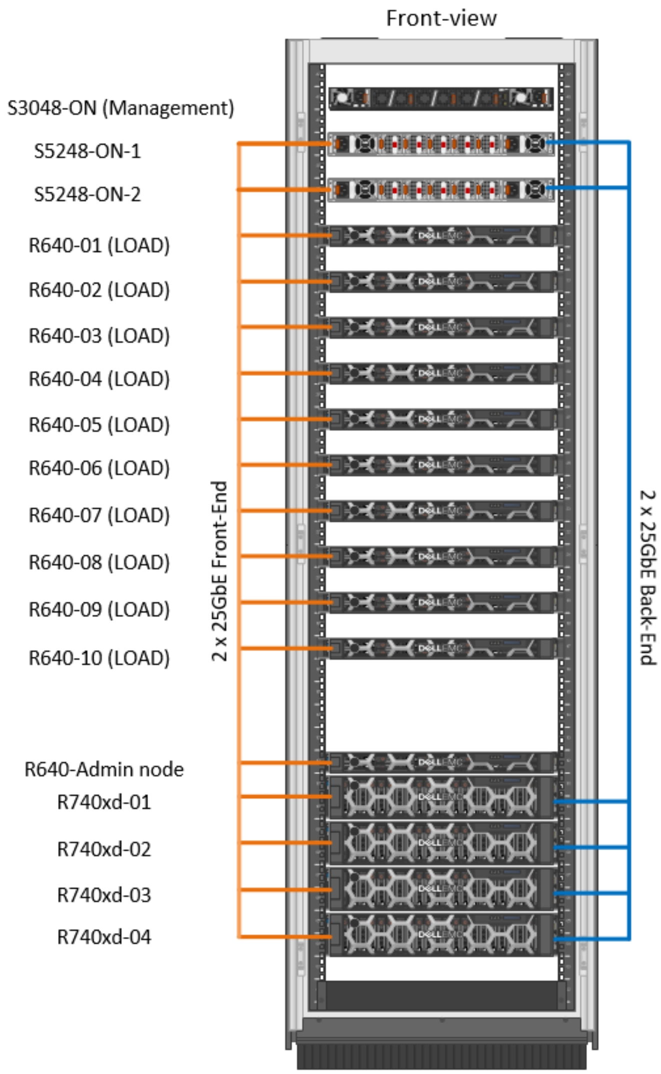

Figure 9: Ceph cluster with R640 servers as load generators

Note: We recommend that heavier storage nodes be located at the bottom of the rack.

Note: We use four storage nodes as a starting point. You should consider five or more for production.

Each Ceph storage node has four 25GbE links going into two leaf switches. These links are created keeping the high availability and bandwidth aggregation architecture in context. One set of the 25GbE links (as an 802.3ad LAG, or bond in terms of RHEL) is connected to the frontend (ceph-client/public API) network. The other set of links is connected to the backend (ceph-storage) network. The load generator servers have 2 x 25GbE link connected to the frontend network. A separate 1GbE management network is used for administrative access to all nodes through SSH.

Note: The following bonding options were used: mode=802.3ad miimon=100 xmit_hash_policy=layer3+4 lacp_rate=1

While the overall physical setup, server types, and number of systems remain unchanged, the configuration of the OSD node's storage subsystems was altered. Throughout the benchmark tests, different I/O subsystem configurations are used to determine the best performing configuration for a specific usage scenario.

Table 13: Software components in testbed

Software components in testbed	
Ceph	Red Hat Ceph Storage 3.2
Operating system	Red Hat Enterprise Linux 7.6
Tools	Ceph Benchmarking Tool (CBT) and FIO 3.1
Server monitoring	Prometheus (node exporter) and Grafana

Table 14: Ceph configuration used in all benchmarks

Configuration	Details
Replication factor	2
Number OSDs	64 (2 per NVMe)

Configuring Dell PowerEdge servers

The Dell PowerEdge R740xd and Dell PowerEdge R640 servers are configured using the iDRAC and the racadm configuration utility. The iDRAC configuration is deployed on the admin node and used to reset the server configuration, including the BIOS configuration. This ensures all systems have the same configuration and were set back to known states between configuration changes. With the `racadm` command, the configuration can be retrieved from and stored to an NFS share, which is provided by the admin node.

Deploying Red Hat Enterprise Linux

To deploy RHEL, the recommended approach is through a coordinated and centralized installation server. This not only reduces the deployment time significantly but also improves the consistency in configurations (for example network configs) by avoiding the manual error-prone setup on individual servers. In our setup, we deploy RHEL 7.6 on all the nodes. This includes the administration node, which handles RHEL as well as RHCS installation, test log aggregation, monitoring, and other management related tasks. This node will be referenced as admin node through the remainder of this document.

Table 15: Required services

Service	Notes
NTP	Time synchronization is very important for all Ceph nodes
DNS	Not strictly required for Ceph, but needed for proper RHEL functioning

Deploying Red Hat Ceph Storage

In production environments, Red Hat Ceph Storage can be deployed with an easy-to-use Ansible playbook, ceph-ansible.

Ceph-ansible is an end-to-end automated installation routine for Ceph clusters based on the Ansible automation framework. Predefined Ansible host groups exist to denote certain servers according to their function in the Ceph cluster, namely OSD nodes, Monitor nodes and Manager nodes. Tied to the predefined host groups are predefined Ansible roles. The Ansible roles are a way to organize Ansible playbooks according to the standard Ansible templating framework, which in turn, are modeled closely to roles that a server can have in a Ceph cluster.

Note: We recommend running ceph-ansible from the admin node. It provides adequate network isolation and ease of management.

The Ceph daemons are colocated as containerized services when deployed. Specifically, out of the four nodes in the architecture, since MONs are to be deployed in an odd number (to maintain consensus on cluster state), they're deployed on three out of four nodes, whereas OSD daemons are deployed on all four nodes. This is shown in the figure below. However, these daemons are isolated on the OS level through use of Linux containers. Since three MONs are enough to maintain cluster state, additional storage nodes (with OSD only) can be added to expand the cluster by simply plugging the additional hardware and deploying OSDs on the nodes.

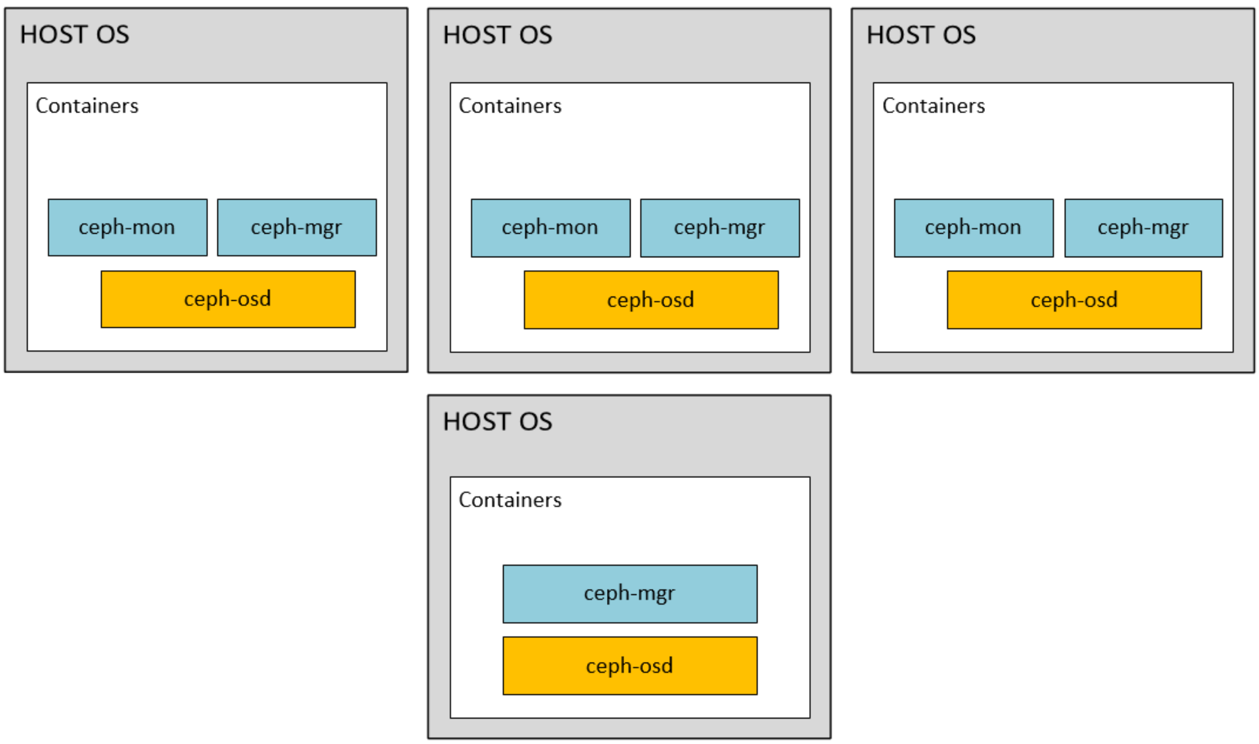

Figure 10: Colocated containerized Ceph block storage

Metrics collection

In order to pinpoint bottlenecks encountered during testing, it's critical to have a variety of server metrics captured during test execution. We made use of Prometheus for monitoring our servers. We installed the standard 'node exporter' on each node and configured our Prometheus server to pull the server metrics every 10 seconds.

The 'node exporter' captures all of the standard server metrics that are used for analysis. Metrics are captured for network, CPU, memory, and storage devices. Our Prometheus server was integrated with Grafana to provide a rich and powerful monitoring tool. This infrastructure allowed to us seamlessly capture all relevant metrics during all of our test runs.

Test and production environments compared

Table 16: Differences between performance testing and production environments

Area of difference	Performance testing	Production
Number storage nodes	4	Higher
Ceph authentication	none (disabled)	cephx (enabled)
Ceph scrubbing	disabled	enabled
Ceph data CRC checks	disabled	enabled
Vulnerability patches	disabled	enabled

Chapter 5

Test methodology

Topics:

This chapter details the testing methodology used throughout the experimentation. It includes the tools and workloads used and the rationale for their choice. It highlights the iterative tuning process which was adopted as performance engineering methodology. Finally, it provides the configurations used which are recommended for use and summarizes the testing approach.

Overview

The methodology used in the testing process was designed in a way that allows us to view the cluster the behavior under various conditions and workloads. We perform random as well as sequential I/O of read and write operations. The block sizes used are 4KB for random and 4MB for sequential workloads. We also vary the ratio of reads and writes in the full workload. For instance, we have one workload comprising 70% read and 30% write operations. This gives us a rich insight into the cluster behavior, and allows extrapolation and speculation for numerous production environment workloads.

For random read workloads, we increase the number of clients and measure IOPS and latency. We fix the queue depth to 32 which gives us a trade off between IOPS and latency (smaller queue depth means smaller latency values, but also less IOPS). Since increasing the number of clients doesn't increase IOPS with write workloads, for random write, we vary queue depth and observe IOPS and latency. This behavior is specific to flash devices and was observed in our baseline hardware testing. Therefore, our methodology was built around varying queue depth.

For sequential workloads, however, we don't measure IOPS and measure throughput because these workloads are not evaluated on per I/O basis. For sequential read workloads, we measure throughput as we vary number of clients. In the case of sequential write workloads, we vary queue depth and observe throughput values. This is again in accordance with behavior of flash devices.

The tests are divided into three stages. Initially, there's a baseline test run, which involves deployment and testing with default values. Next, we have an iterative tuning process (discussed later, *see "Iterative tuning"*). Finally, the last run includes all the necessary tunings done to ensure the best Ceph block performance on the cluster.

To automate this procedure, the Ceph Benchmarking Tool (CBT) is used. It allows one to define a suite of tests that can then be run sequentially. This suite of tests helps in maintenance of metrics and cluster logs.

Workload generation

All of our workloads were generated using the FIO utility (via CBT as discussed later). FIO is able to generate a large number of workload types and through various I/O engines. Our tests made use of the 'librbd' engine.

In all our random workloads we increased loading from 10 up to 100 clients. Each client had its own RBD image for I/O. The different RBD volume sizes we used for each test are given in the table below. Similarly, in our sequential workloads we increased loading from 10 up to 50 clients. This is because 50 clients are enough to hit the bottleneck of the Ceph cluster.

Table 17: RBD Image Sizes Used

Workload	RBD Volume Size	Max Clients	Total Usable/Raw Storage
4KB Random Read	200GB	100	20TB/40TB
4KB Random Write	50GB	50	2.5TB/5TB
4KB Random Mixed	100GB	100	10TB/20TB
4MB Sequential Read	300GB	50	15TB/30TB
4MB Sequential Write	300GB	50	15TB/30TB

All tests made use of 90 seconds of ramp time. It's important to verify that there is sufficient data populated and that read operations are not being satisfied from cache. Across all four storage nodes, we have an aggregate server memory storage of 768GB. This amounts to 15% of provisioned usable storage and 5% of the provisioned raw storage. We configure OSDs to use up to a maximum of 8GB each. Across all 16

OSDs on a given storage node, that represents 128GB of memory. Taken across all four storage nodes, that amounts to 512GB of memory. This aggregate OSD memory amounts to 10% of provisioned usable storage and 3% of provisioned raw storage.

Going beyond our calculated memory percentages of provisioned storage, we also paid close attention to SSD utilization metrics and throughput to confirm that data was being read from the storage devices and not cache.

Ceph Benchmarking Tool

CBT is an open-source benchmarking suite written in Python that takes a modular approach to Ceph benchmarking. The utility is able to use different benchmark drivers for examining various layers of the Ceph storage stack, including RADOS, RADOS Block Device (RBD), and KVM. In this paper, storage performance on the core layer RBD is examined. For this, CBT uses the librbdfio benchmark, which runs FIO with librbd backend. Librbd is a separate library that needs to be installed prior to test execution.

The utility is installed on the admin node. From there, it communicates with various servers in different capacities as follows:

Head Node : A system that has administrative access to the Ceph cluster for the purpose of creating pools and RBD images, changing the configuration or even re-deploying the entire cluster as part of a benchmark run. This is the same as our admin node.

Clients These are the systems which have access to the Ceph cluster and from which CBT will generate load on the cluster using locally installed tools such as FIO or 'rados bench'.

OSDs/MONs : CBT triggers performance metrics collection with various tools on the nodes during the benchmark run and transfers their telemetry back to the head node after each execution.

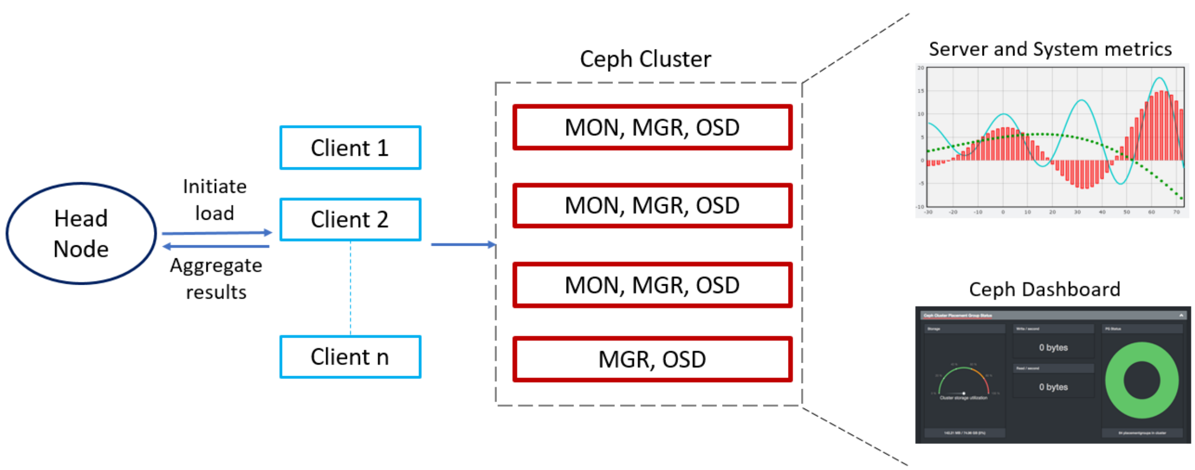

Figure 11: Test methodology components

The “configuration file” option of CBT YAML file was used to orchestrate most of the benchmarks. CBT provides flexibility to run benchmarks over multiple cluster configurations by specifying custom ceph.conf files. In this benchmark, CBT was mainly used to execute the benchmarks. The cluster deployment and configuration was provided by ceph-ansible.

Iterative tuning

A baseline configuration was established before commencing the baseline benchmarks. This baseline configuration largely consisted of default values, but with a few deliberate changes. For example, we knew ahead of time that our tests would be conducted with Ceph authentication (cephx) turned off. Consequently, we turned it off from the very beginning. We selected a starting value of 2048 placement groups (PGs) for our baseline configuration. This value was not calculated and no attempt was made to start with optimum value. We simply picked a value that seemed reasonable and conservative for our environment with the expectation that we would progress to the optimum value as part of our iterative tuning. Once our baseline configuration was established, we ran the predefined workloads on our cluster with increasing load from our load generators.

Once a baseline set of metrics were obtained, a series of iterative tuning efforts were made in an effort to find the optimal performance. The tunings were performed with a workload of 4KB random reads generated by 40 clients, each using a 50GB RBD volume.

As part of our iterative tuning, we increased the size of our placement groups (PGs) and re-ran our workloads. Several iterations of this experiment provided the optimum value of 8192 for our cluster. We cross-checked the value that we found through experimentation with the formula provided for calculating PGs. Reassuringly, the values matched. Even though the calculated value from the formula did give the optimum value for our cluster, we still believe that it's good practice to experiment with "nearby" values (starting below what the formula produced).

The formula to compute the number of PGs is given as:

$$\frac{\text{(Target PGs per OSD) * (OSD \#) * (\%Data)}}{\text{(Size)}}$$

Figure 12: Placement group formula

where the following parameters are used in this architecture:

- Target PGs per OSD = 200
- OSD # = 64
- %Data as ratio = 1.0
- Size (# replicas) = 3

The calculated value is 4266 which we round up to 8192. The CRUSH algorithm performs well if the PGs are given as a power of 2.

⚠ **CAUTION:** Some tuning options have been set to obtain maximum performance and to enable like-comparison with other published whitepapers. These tuning options are not recommended for production use.

Table 18: Tuning with caution

Area	Config. option	Performance	Production
Ceph	auth_client_required	none	cephx
	ms_crc_data	False	True
	scrubbing	disabled	enabled
OS	Meltdown/spectre/ ZombieLoad patches	disabled	enabled

Testing approach

A lower testing duration to reach steady-state is optimal. This is generally only a few minutes.

BlueStore is the new storage backend in RHCS 3. It contains many improvements over the legacy backend FileStore and is recommended for new deployments.

The number of iterations of each test should be more than one to ensure consistency between successive values. It is set to two which allows for faster test cycles and doesn't compromise the accuracy/reliability of our results.

CPU cores per OSD container need to be set according to the available number of cores in a storage node. We have 28-core Intel® Xeon® Platinum 8176 CPUs in a dual-socket configuration providing a total of 56 physical cores and 112 logical cores (threads) with hyperthreading. Therefore, we set six logical cores per OSD container, giving a total of 96 (6x16) logical cores dedicated to OSD containers. This is a very critical parameter as well. The remaining logical cores are allocated as: MON (1), MGR (1), and OS (unspecified/remainder).

A summary of these test parameters is given in the following table for quick reference.

Table 19: Standard test parameters

Parameter	Values
Test time	10 minutes
Iterations	2

The cluster was deployed with containerized RHCS 3.2. Ten load generator servers were used to generate traffic across the block storage cluster. The number of client processes was equally divided among the load generator servers. Each client was used as a separate process to write to an RBD image on the cluster. Linux caches were cleared on storage nodes before running each test. Additionally, BlueStore caches were cleared between test runs by restarting the OSDs.

Finally, two very important functional components of Ceph were disabled during all tests. These components relate to data integrity checks and security.

Scrubbing is a critical function of Ceph used to verify the integrity of data stored on disk. However, scrubbing operations are resource intensive and can interfere with performance. We disable scrubbing to prevent adverse performance effects and to enable study in a more controlled and predictable manner. Scrubbing should not be disabled in production as it provides a critical data integrity function.

CAUTION: Although we disable scrubbing during controlled performance tests, scrubbing must not be disabled in production! It provides a critical data integrity function.

Cephx is an important security function that provides Ceph client authentication. This feature is enabled by default and is recommended for all production use. We disable it in our tests since this feature is commonly disabled in other Ceph benchmarks. This makes our results more comparable with other published studies.

CAUTION: We disabled Cephx for performance studies, but this feature should not be disabled in production! It provides a critical data security function.

Chapter 6

Hardware baseline testing

Topics:

- *Baseline testing overview*
- *CPU baseline testing*
- *Network baseline testing*
- *Storage device baseline testing*

This chapter presents our hardware baseline tests performed on various components such as CPU, network, and storage devices.

Baseline testing overview

Before attempting benchmark scenarios that utilize higher-layer Ceph protocols, it is recommended to establish a known performance baseline of all relevant subsystems. We perform hardware baseline tests for CPU, network, and storage devices.

CPU baseline testing

CPU testing has been performed with Intel® LINPACK benchmark running suitable problem sizes given each server's CPU resources.

Table 20: CPU baseline

Server type	**Storage node:** **PowerEdge R740xd 2x Intel® Xeon® Platinum 8176 CPU @ 2.10GHz**	**Load generator:** **PowerEdge R640 2x Intel® Xeon® Gold 6126 CPU @ 2.60GHz**
LINPACK results	437 GFlops (problem size = 30000)(cores = 88)	448 GFlops (problem size = 30000)(cores = 48)

Network baseline testing

Network performance measurements have been taken by running point-to-point connection tests following a fully-meshed approach; that is, each server's connection has been tested towards each available endpoint of the other servers. The tests were run one by one and thus do not include measuring the switch backplane's combined throughput, although the physical line rate is 50000 MBit/s for each individual link. An MTU value of 9000 was used throughout all tests.

Note: The iPerf network performance utility was used to establish networking baselines.

Table 21: Network baseline

Server type	**PowerEdge R740xd Intel® XXV710**	**PowerEdge R640 Intel® XXV710**
PowerEdge R740xd Intel® XXV710	44.39 GBit/s	44.23 GBit/s
PowerEdge R640 Intel® XXV710	46.10 GBit/s	46.43 GBit/s

Storage device baseline testing

The drive performance measurements were taken with FIO using libaio backend. The test profile used a 4KB block size, four threads/drive for randread and one thread/drive for randwrite, a queue depth of 32 and direct I/O (no page cache). Each test had a runtime of 5 minutes. The objective was to determine the practical bounds of metrics, which were then to serve as points of comparison in the performance tuning process.

Note, however that performance of four drives operating simultaneously is not the same as four times the performance of a single device for random read operations. This is an important practical implication which becomes a part of performance tuning and goes against theoretical calculations.

Table 22: NVMe storage device baseline

# Drives	Operation type	Avg. IOPS	Avg. latency
1	Random read	804k	18.9us
4		2.6M	198us
1	Random write	150k	212us
4		500k	213us

Chapter 7

Benchmark test results

Topics:

This chapter provides the benchmark results along with the bottleneck analysis. The bottleneck analysis is conducted using hardware usage metrics that were gathered throughout the testing process. Finally, it summarizes the key takeaways from the performance analysis work.

Bottleneck analysis

In previous chapters, we discussed *(see "Metrics collection")* our Prometheus and Grafana server monitoring infrastructure. This infrastructure captured all relevant metrics related to CPU, memory, OS, networking, and storage devices. We were then able to analyze these various server metrics from the same time period when specific tests were run.

This analysis enabled us to identify bottlenecks that were hit within various benchmark tests. The most important metrics for this analysis included:

- CPU utilization
- Network throughput
- Memory utilization
- Storage device utilization
- Storage device throughput

All the metrics presented for analysis are from the Ceph storage nodes. This includes the CPU utilization, network utilization, and drive utilization numbers.

4KB random read

The number of clients was increased from 10 to 100 clients. The volume per client was set to 200GB (the RBD image size) and ramp time was set to 90 seconds to allow the cluster to reach a steady-state before measuring metrics. Random I/O reads were tested with a constant queue depth of 32.

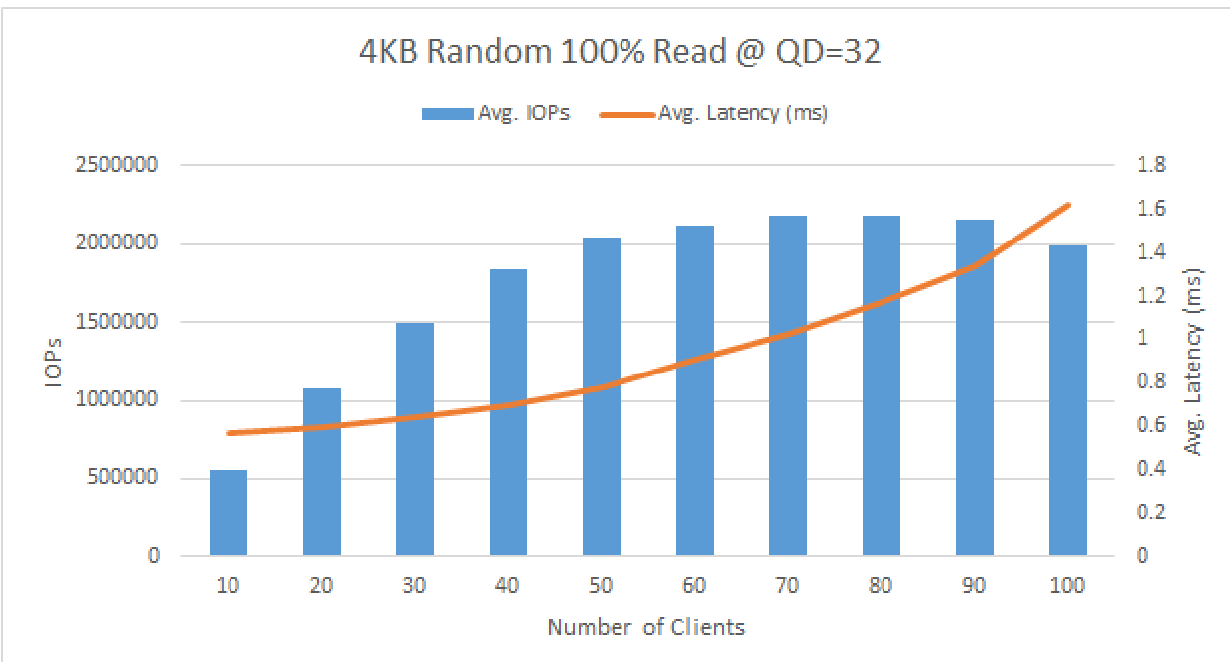

Figure 13: 4KB random read

The disk utilization is depicted along with drive read throughput below. It's clear the drives have reached 100% utilization and this causes the IOPS to level-off. The read throughput, however, is far from it's saturation point. That is expected, since this is a 4KB block size.

The initial smaller peak represents the interval of volume population prior to testing, and is not important for analysis. The higher peak represents the stress testing interval. Moreover, the the multiple sets peaks represent the two iterations of tests, this is to show consistency of numbers across multiple tests. The same stands true for all the storage node metrics presented in this chapter.

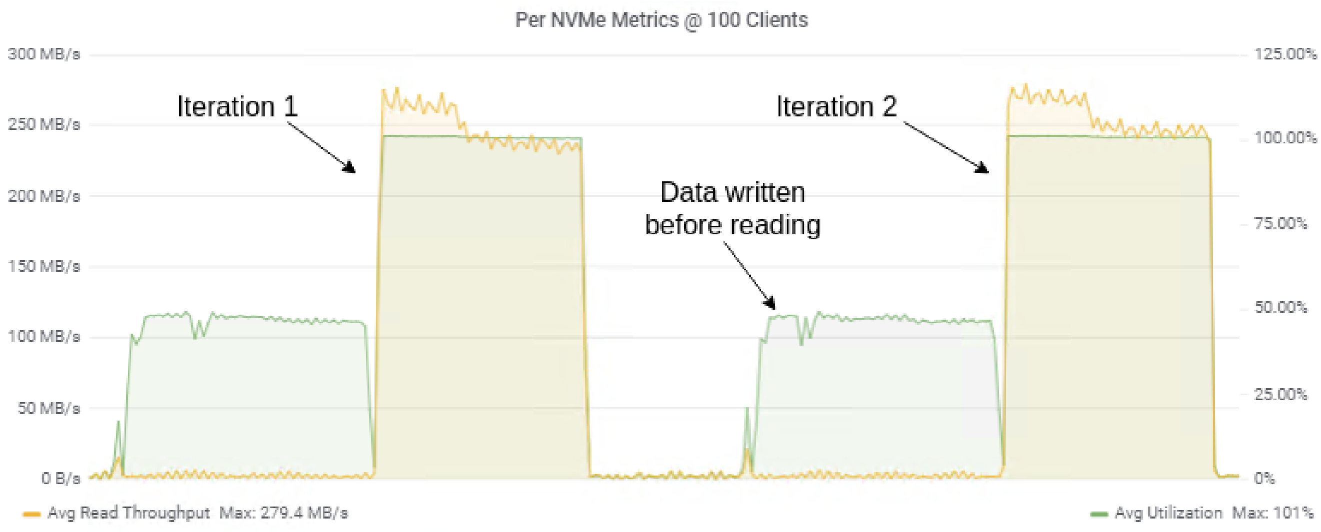

Figure 14: 4KB random read NVMe usage

Also, notice below that the CPU utilization approached 100%. This means that if the drives are upgraded, the CPU might also need an upgrade, or it will become the new bottleneck.

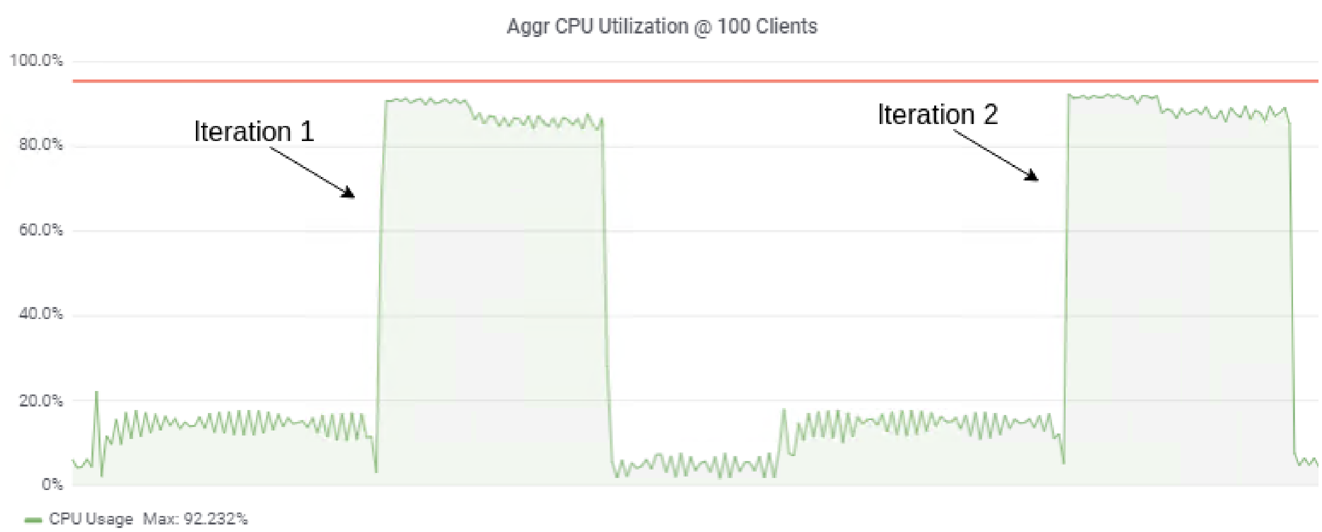

Figure 15: 4KB random read CPU usage

4KB random write

In the 100% write scenario, the queue depth was varied from 1 to 256 as shown in the graph for 50 clients. The volume per client was set to 50 GB and ramp time was set to 90 seconds. The reason for such a high queue depth is to be able to see the bottleneck clearly.

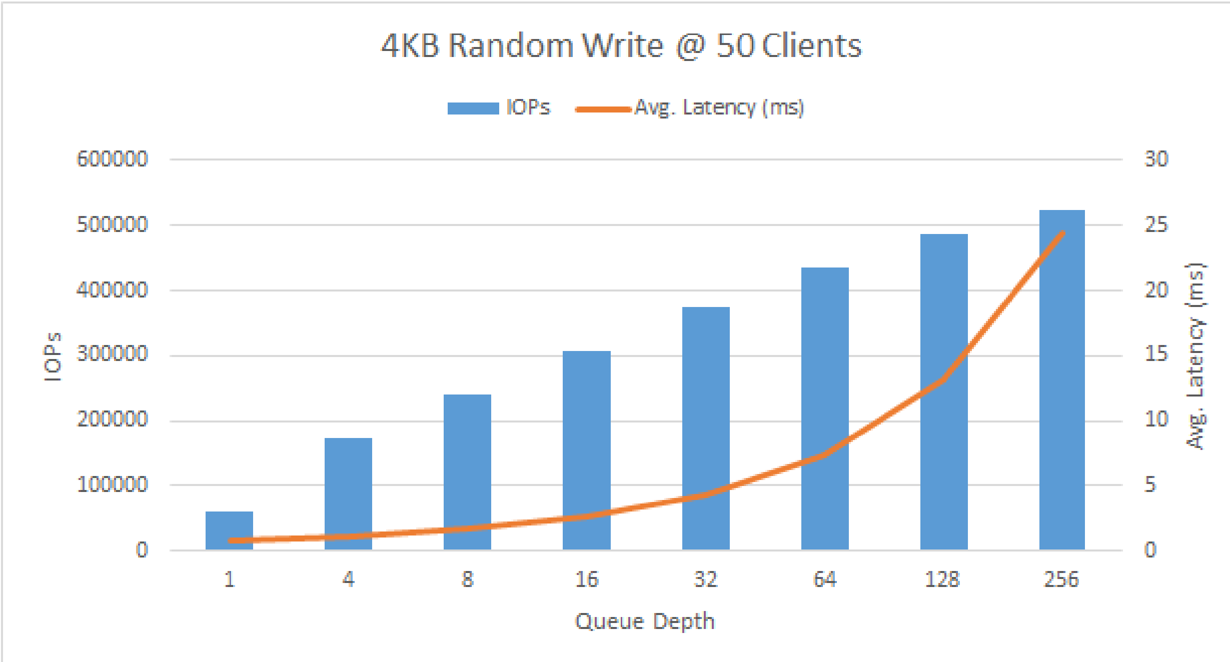

Figure 16: 4KB random write

Generally, we would work with a queue depth value up to 64. But this point didn't have enough stress on the cluster to allow us to clearly see the bottleneck. As visible below, even at a queue depth of 256, neither throughput nor I/O time are saturated on the drives.

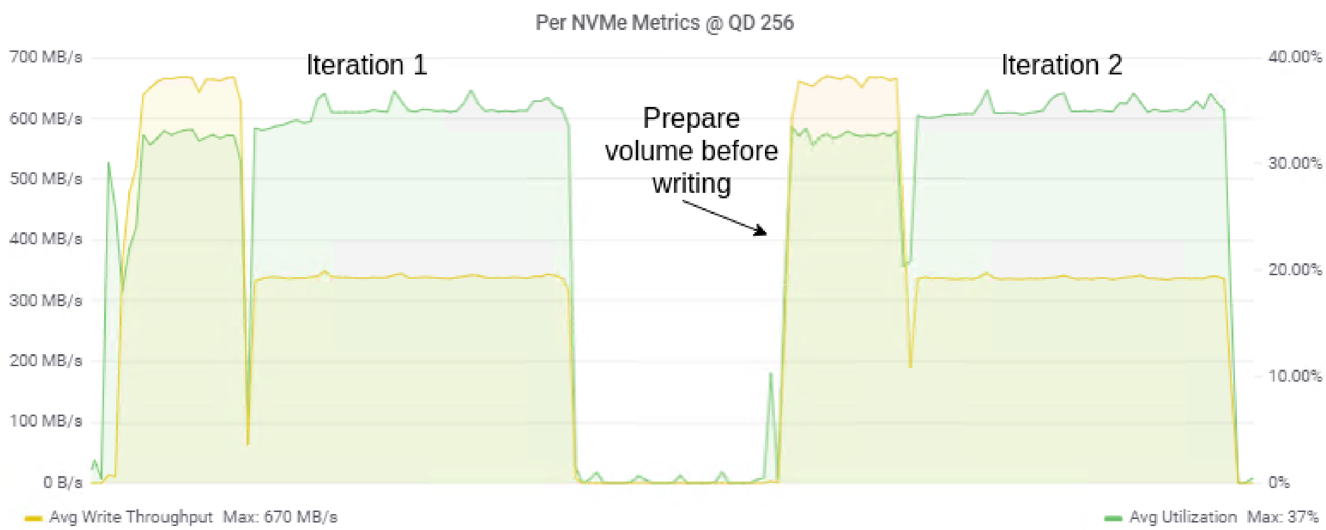

Figure 17: 4KB random write NVMe usage

However, notice below how increasing the queue depth allowed us to reach a point where we can generate enough write workload to determine that the CPU bottleneck will be hit before the drive bottleneck.

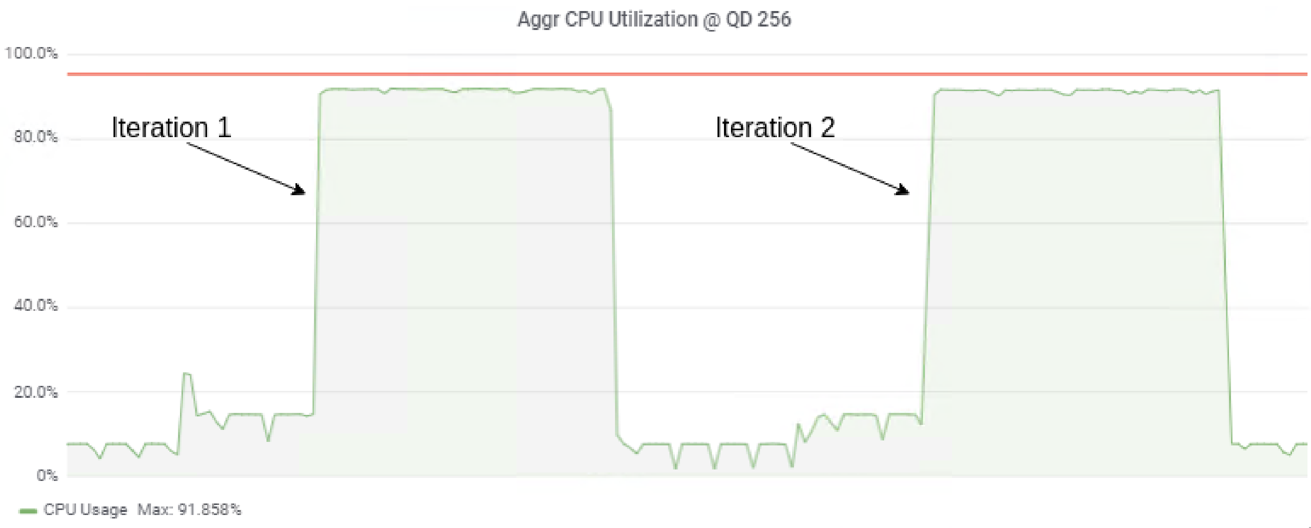

Figure 18: 4KB random write CPU usage

4KB random mixed

In the 70% read/30% write scenario, the number of clients was increased from 10 to 100 clients. The volume per client was set to 100GB and ramp time was set to 90 seconds. Tests used a constant queue depth of 32 for random I/O reads. Use smaller queue depths for write workloads as mentioned previously.

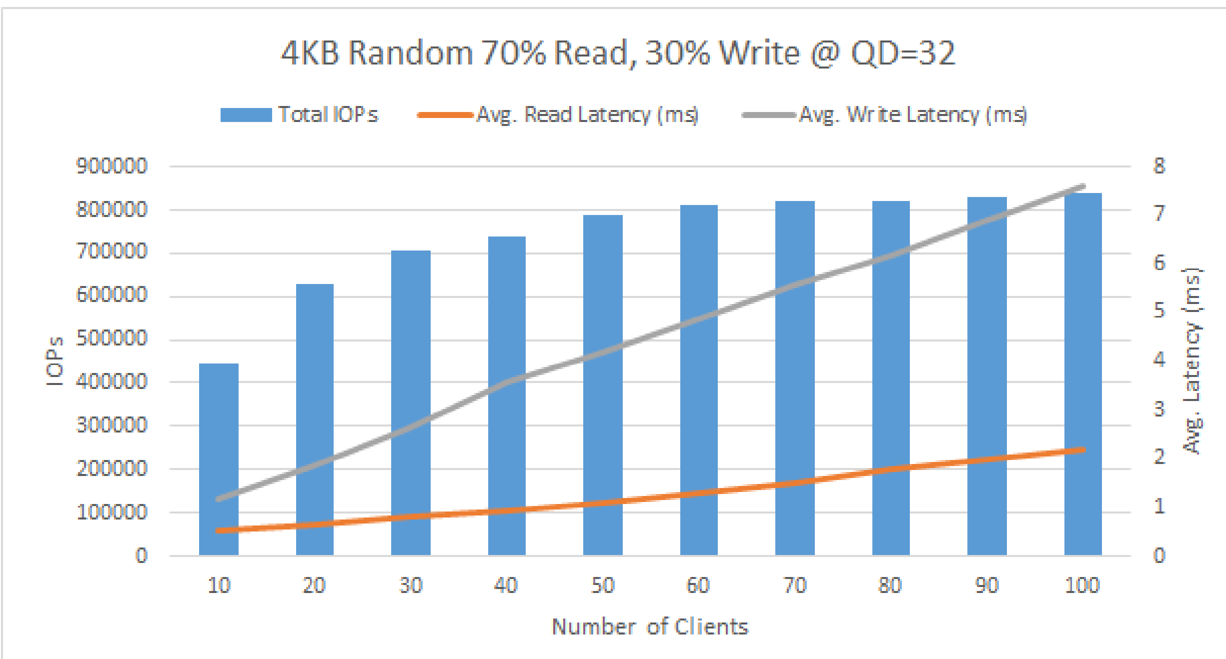

Figure 19: 4KB random mixed

Once again, we notice due to a large component of read operations in the workload, the bottleneck has been disk utilization. The disk throughput for both read and write workloads is minimal.

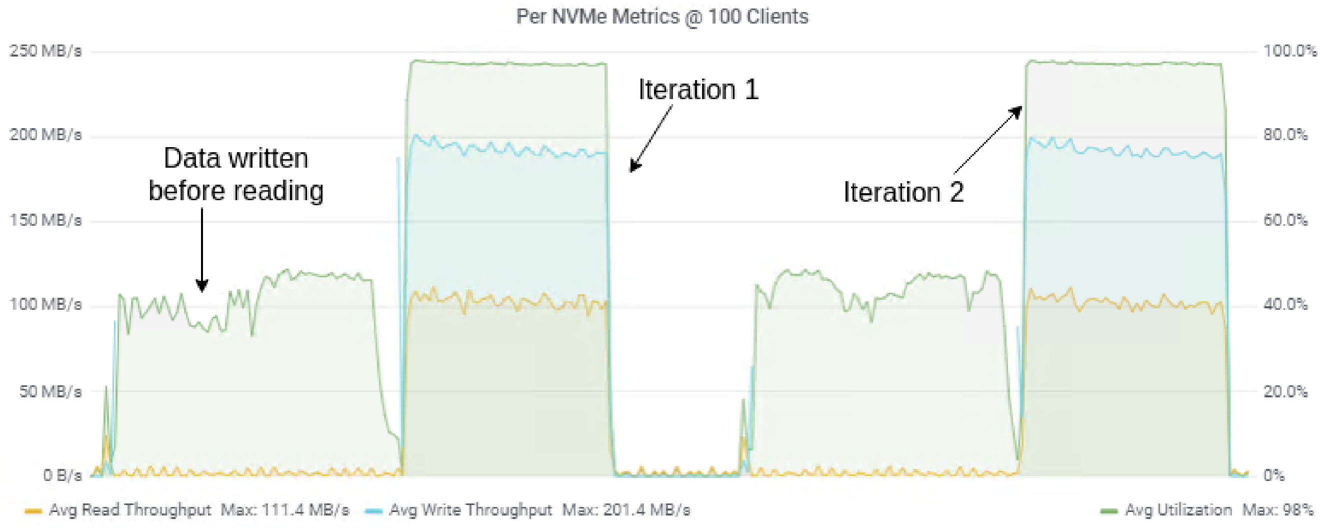

Figure 20: 4KB mixed NVMe usage

Interestingly, in this case, we also get very close to CPU saturation. Therefore, just like in the case for 4KB random read workloads, in this case as well, upgrading would be required for both CPUs and drives.

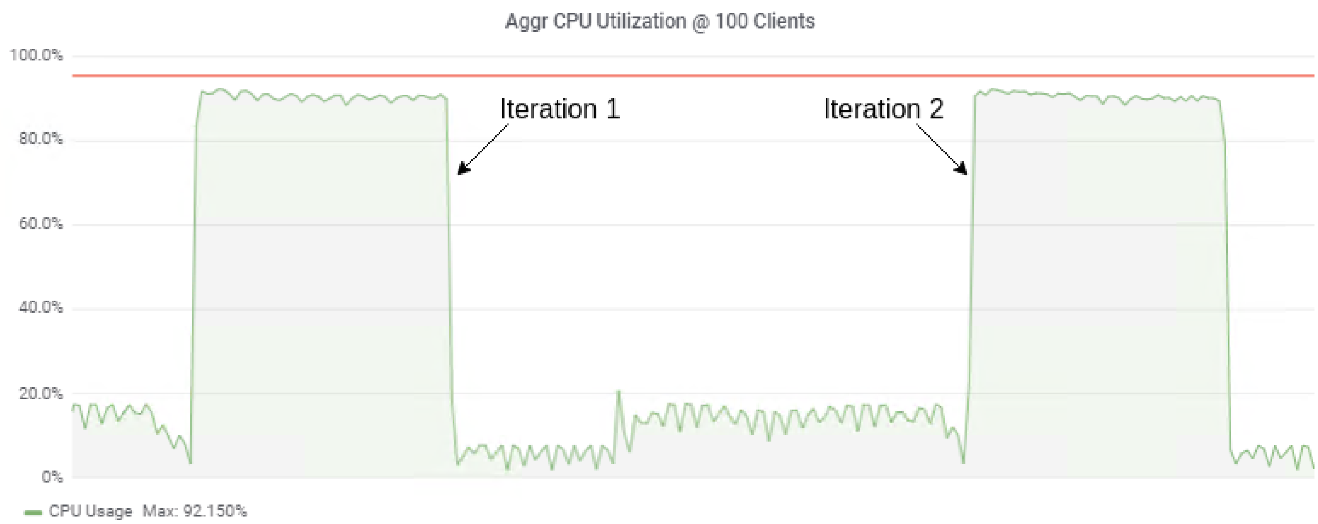

Figure 21: 4KB mixed CPU usage

4MB sequential read

In the 100% read scenario, the queue depth was set to 32, while the volume per client was set to 300GB and a ramp time of 30s. Since we now have a much larger block size of 4MB, a larger client volume is chosen to ensure a longer duration for a test run. This gives more metric data points and improves the accuracy of results.

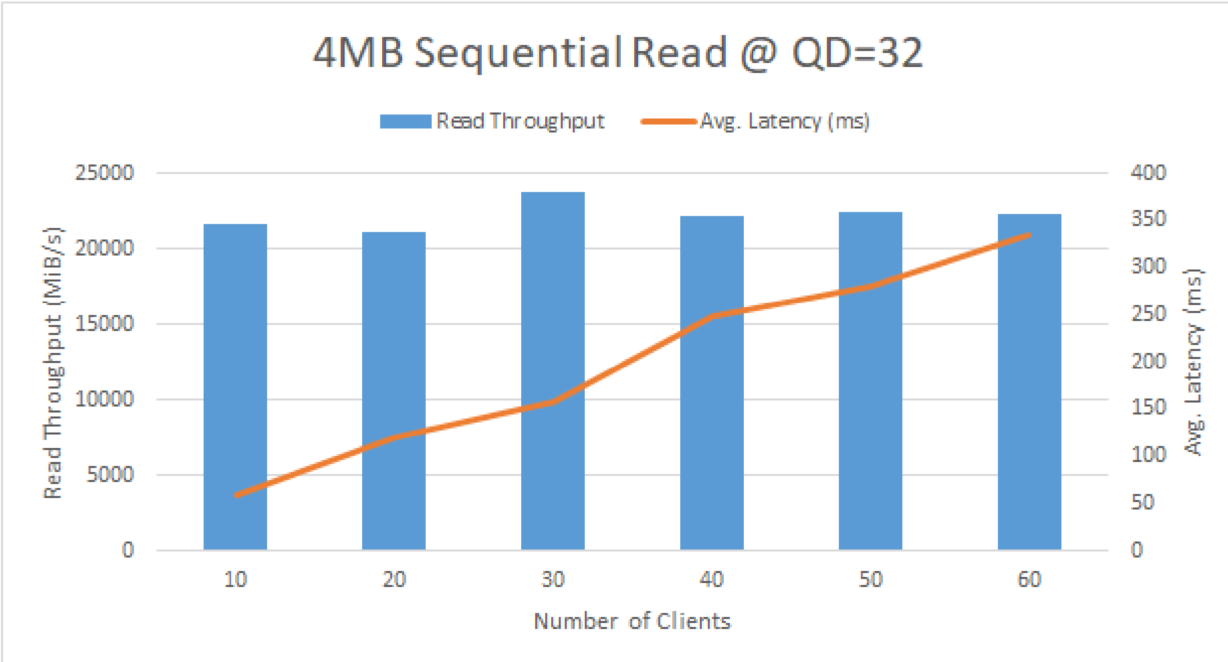

Figure 22: 4MB sequential read

The drive throughput (as show below) is well within it's limits. The I/O time utilization is also barely hitting 50%, therefore, the bottleneck is not in the drives.

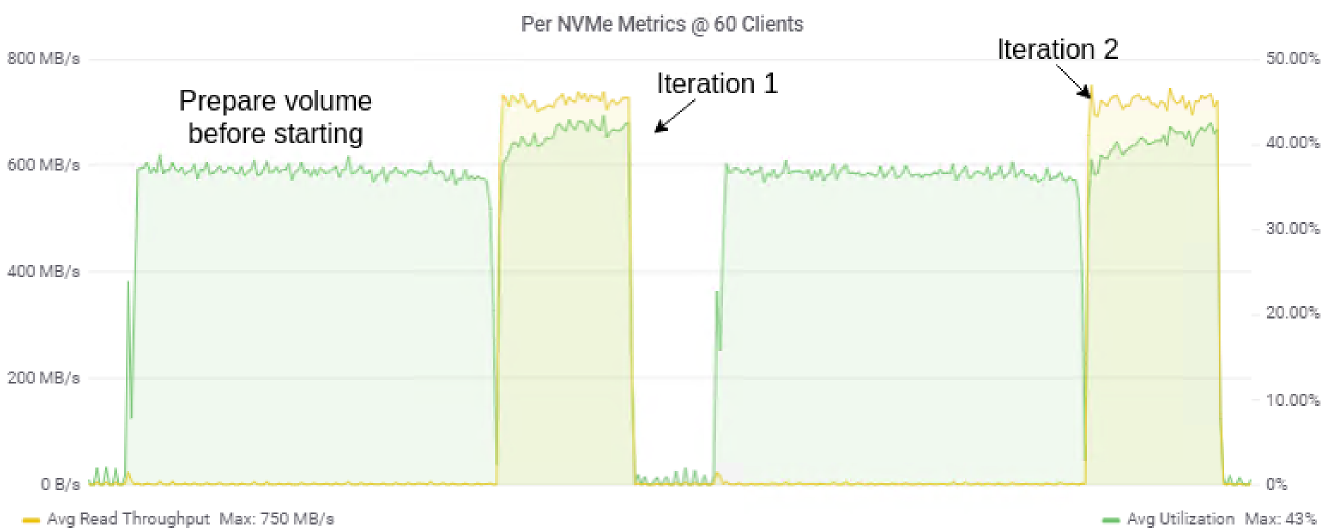

Figure 23: 4MB sequential read NVMe usage

However, notice the network traffic reaching it's saturation point (96% of maximum). This is a clear indication of the network limiting the cluster performance in terms of cluster throughput.

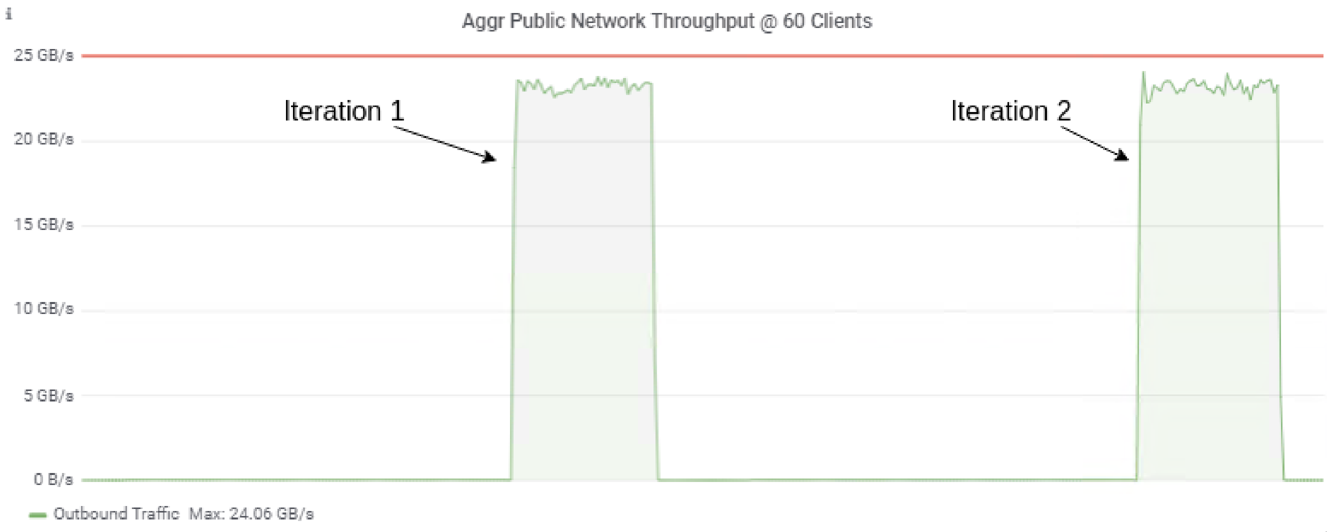

Figure 24: 4MB sequential read retwork usage

4MB sequential write

In 100% write scenario, again the queue depth is increased from 1 to 32. The clients were set to a constant of 50, having volume per client set to 300 GB and ramp time of 30s. Contrary to the case of 4MB write workload, there was no need to go beyond a queue depth of 32 since we had already identified the bottleneck.

Figure 25: 4MB sequential write

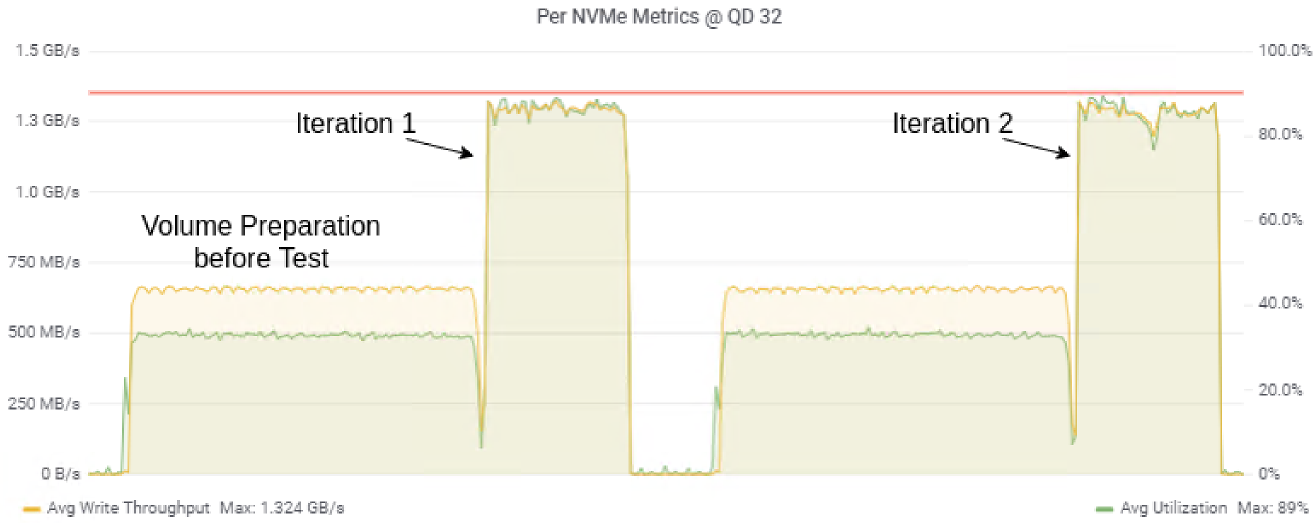

Figure 26: 4MB sequential write NVMe usage

The network throughput is very high (~84% of maximum). This isn't a bottleneck itself. However, this indicates that if we are to eliminate the bottleneck by upgrading the drives, we might want to consider upgrading the network as well.

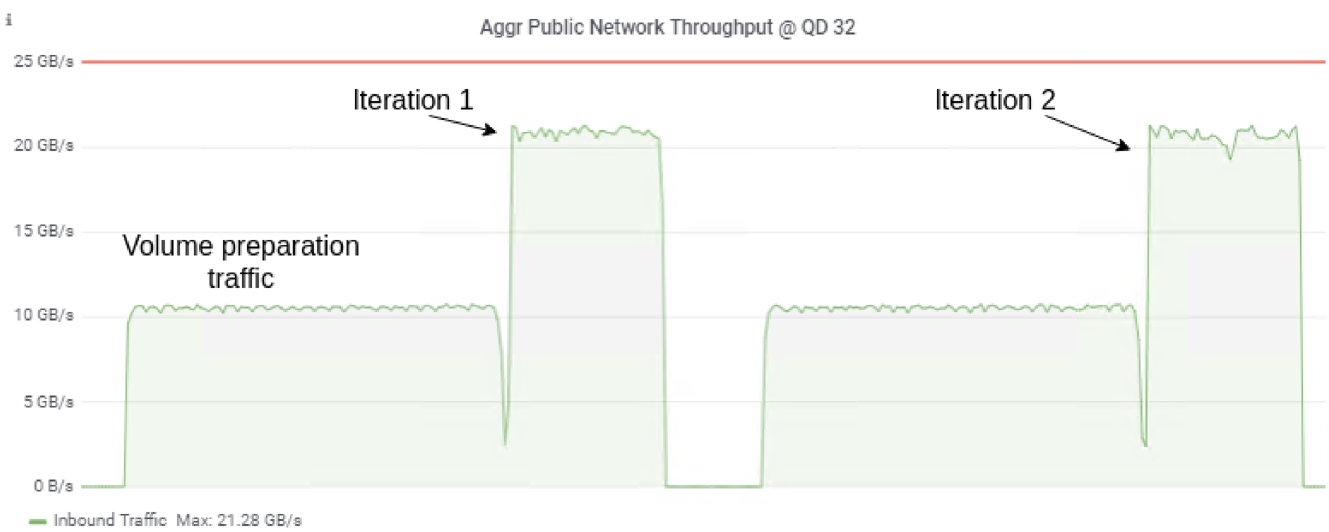

Figure 27: 4MB sequential write network usage

Analysis summary

The testing methodology exercised in the benchmarking process uses controlled, synthetic workloads that can be very different from those in production. However, it has been designed carefully to ensure that the cluster is independently analyzed from different perspectives. This allows for better insight into its properties. We can then use these independent properties to speculate onto cluster performance under custom workloads.

For instance, when we discuss 100% read and 100% write workloads, we can see how the cluster behaves given that all operations are of a single type. Therefore, when we perform a 70% read 30% write test, we can use the hypotheses from the previous (more independent) workloads, to explain the results of a mixed workload. This not only makes them testing methodology more robust and versatile, but also enables designers to predict the cluster performance with a customized workload, which is more representative of a production environment.

The random read workload tests produced outstanding performance. The bottleneck for 4KB random read workload was the NVMe devices. For sequential read workloads, the bottleneck shifted to the network.

One important thing worth highlighting is the relationship between drive utilization and drive bandwidth. Drive utilization is the percentage of time the drive was actively servicing requests. In the case of read workloads, we generally observe a very high drive utilization as well as increased CPU usage (which is a consequence of that). This is because read operations are quicker, and a drive can actively serve a large amount of read operations causing the utilization to go high.

In contrast, the drive bandwidth corresponds to how much data the drive can process at a given time. This is generally a matter of concern in operations that take longer to complete; the write operations. Also, since the drive isn't actively servicing writes, but rather accumulating them (as permitted by bandwidth), there's a lower utilization of drive compared with read operations, and much higher bandwidth usage. This is why we tend to look at drive utilization for read workloads, and drive bandwidth usage for write workloads.

Table 23: Workload bottlenecks

Workload	Bottleneck	Suggested remedy
4KB random read	Intel® P4600 NVMe drives (Intel® Xeon® Platinum 8176 just behind)	Faster NVMe drives (may shift to CPU) or additional storage nodes
4KB random write	Intel® P4600 NVMe drives (Intel® Xeon® Platinum 8176 just behind)	Faster NVMe drives (may shift to CPU) or additional storage nodes
4KB random mixed	Intel® P4600 NVMe drives (Intel® Xeon® Platinum 8176 just behind)	Faster NVMe drives (may shift to CPU) or additional storage nodes
4MB sequential read	Ceph client 50GbE network	Consider 100GbE network
4MB sequential write	Intel® P4600 NVMe drives (Ceph client 50GbE network just behind)	Faster NVMe drives (may shift to network) or additional storage nodes

Table 24: Bottleneck exposure by component

Area	Component	Bottleneck exposure	Comments
Ceph storage system	Intel® P4610 NVMe devices	Very likely	• Disks were our bottlenecks in random read/write operations • System is well-balanced, can easily shift to CPU • P4610 improved over Intel® P4600 (tested), may shift to CPU

Area	Component	Bottleneck exposure	Comments
Storage node motherboard	Intel® Xeon® Platinum 8176 CPU	Likely	• System is well-balanced • 4 extra cores may be enough to prevent bottleneck • May be bottleneck for random read/write workloads of small blocks • Would be bottleneck with additional NVMe drives for random read/write operations
	Memory	None	• Memory configuration has extra 24% capacity over total recommended • No memory pressure observed in any tests
Network	Ceph client 50GbE network	Very likely	• This was bottleneck in sequential read/write operations • Solved with 100GbE
	Ceph cluster (replication) 50GbE network	Vulnerable during failure scenarios	• 2x replication relieves pressure • Likely bottleneck in case of node failure • Mitigated with 100GbE

Chapter 8

Conclusions

Topics:

This chapter presents the key takeaways of the study. We first present differences between devices that were used in performance test as compared to the newer devices specified in the architecture. It summarizes the results achieved and also reiterates the objectives of the work.

Intel® P4610 NVMe guidance

Our architecture specifies the use of Intel® P4610 NVMe drives. However, our benchmark testing was performed with Intel® P4600 NVMe drives. This was because P4610 drives were not available for order at the time our servers were acquired. The following table provides published performance ratings for both drives.

Table 25: Performance ratings of Intel® P4600 and Intel® P4610

Metric	Intel® P4600 U.2	Intel® P4610 U.2	Change
Sequential read (up to)	3200 MB/s	3200 MB/s	none
Sequential write (up to)	1325 MB/s	2080 MB/s	+57%
Random read	559,550 IOPS	643,000 IOPS	+15%
Random write	176,500 IOPS	199,000 IOPS	+13%
Latency read	85 microseconds	77 microseconds	-10%
Latency write	15 microseconds	18 microseconds	+20%

As shown in the table above, the P4610 delivers significantly higher throughput for sequential write workloads and more modest improvements for random write workloads. Additionally, the P4610 delivers appreciably better throughput over the P4600 for random read workloads.

Note: The use of Intel® P4610 NVMe drives should provide consistently better performance than what was measured in our benchmarks with the P4600.

Conclusions

The Performance Optimized Ready Architecture presented in this document is well suited for use cases where performance is the critical design factor. With the high resource density of Dell R740xd servers, the colocation of Ceph services is a very attractive choice for RA design. The combination of RHCS 3.2, RHEL 7.6, and 50GbE (dual bonded 25GbE) networking provides a solid foundation for a performance-optimized Ceph cluster. Additionally, the use of containerized Ceph daemons worked well in this study.

Even with only four storage nodes, we were able to achieve the following performance results:

Table 26: Performance highlights

Workload	Result
4KB random read	2.18 million IOPS, 1.2 ms avg. latency
4KB random write	435,959 IOPS, 7.3 ms avg. latency
4KB random mixed 70/30	836,666 IOPS, 3.8 ms avg. latency
4MB sequential read	23,740 MiB/s
4MB sequential write	20,225 MiB/s

The testing methodology adopted for benchmarking of the architecture has been developed rigorously. This is to ensure that it is generic in nature and is easy to exercise in customized environments. Also, the choice of workloads is based on well-known community standards. This also assists the process of performance comparison with other architectures.

The workload specifications can also affect the choice of other components of the architecture. For instance, with workloads that are comprised primarily of sequential write I/O with a large block size (say 4MB), the network bandwidth is the primary suspect in performance degradation. But for other, less bandwidth hungry workloads, the presented 50GbE (dual bonded 25GbE) network design should be appropriate.

The objective of the work was to provide a detailed insight into the capabilities of the state-of-the-art hardware as well as RHCS 3.2 software, provide a robust and generic methodology of testing, and finally, point out critical design parameters. Most importantly, we present an architecture that is well suited for very high performance.

Appendix A

References

Topics:

Note: If you need additional services or implementation help, please call your Dell EMC sales representative.

Bill of Materials (BOM)

Table 27: Bill of Materials (BOM) - R740xd storage nodes

Component	Configuration
Server model	PowerEdge R740xd Server
BIOS	Performance Optimized
Remote admin access	iDRAC9 Enterprise
Motherboard risers	Riser Config 6, 5 x8, 3 x16 PCIe slots
Chassis	Chassis up to 24 x 2.5” Hard Drives including a max of 12 NVMe Drives, 2CPU Configuration
CPU	2x Intel® Xeon® Platinum 8176 2.1G,28C/56T,10.4GT/s, 30M Cache,Turbo,HT (140W) DDR4-2666
RAM	192GB (12x 16GB RDIMM), 2666MT/s, Dual Rank
Data drives	8x Intel® P4610 (see note) 1.6TB NVMe Mix Use U.2 2.5in Hot-plug Drive
1GbE NIC	I350 QP 1Gb Ethernet, Network Daughter Card
25GbE NICs	2x Intel® XXV710 Dual Port 25GbE SFP28 PCIe Adapter, Full Height
System storage (OS)	BOSS controller card + with 2 M.2 Sticks 240G (RAID 1),FH

Note: Our performance testing was conducted with P4600 because the P4610 was not orderable at the time the servers were acquired. Please use P4610 instead of P4600.

Table 28: Bill of Materials (BOM) - R640 admin node

Component	Configuration
Server	PowerEdge R640 Server
Remote admin access	iDRAC9 Enterprise
Storage drives	8x 600GB 10K RPM SAS 12Gbps 512n 2.5in Hot-plug Hard Drive
Chassis	2.5” Chassis with up to 8 Hard Drives and 3PCIe slots
BIOS	Performance Optimized
RAM	192GB (12x 16GB RDIMM), 2666MT/s, Dual Rank
Disk controller	PERC H740P RAID Controller, 8GB NV Cache, Minicard
Motherboard risers	Riser Config 2, 3 x16 LP
CPU	2x Intel® Xeon® Gold 6126 2.6G,12C/24T,10.4GT/s, 19.25M Cache,Turbo,HT (125W) DDR4-2666
25GbE NICs	1x Intel® XXV710 Dual Port 25GbE SFP28 PCIe Adapter, Low Profile

Component	Configuration
1GbE NIC	I350 QP 1Gb Ethernet, Network Daughter Card

Tested BIOS and firmware

CAUTION: Ensure that the firmware on all servers and switches are up to date. Otherwise, unexpected results may occur.

Table 29: Tested server BIOS and firmware versions

Product	Version
BIOS	1.4.9
iDRAC with Lifecycle controller	3.21.23.22
Intel® XXV710 NIC	18.5.17
PERC H740P (R640)	05.3.3-1512
BOSS-S1 (R740xd)	2.3.13.1084

Table 30: Tested switch firmware versions

Product	Version
S3048-ON firmware	Dell OS 9.9(0.0)
S5248F-ON firmware	Cumulus 3.7.1

Configuration details

Configuration details for storage cluster

Linux network bonding

```
mode=802.3ad miimon=100 xmit_hash_policy=layer3+4 lacp_rate=1
```

all.yml

```
fetch_directory: /root/ceph-ansible-keys
cluster: ceph
mon_group_name: mons
osd_group_name: osds
mgr_group_name: mgrs
configure_firewall: False
redhat_package_dependencies:
  - python-pycurl
  - python-setuptools
ntp_service_enabled: true
ceph_repository_type: iso
ceph_origin: repository
ceph_repository: rhcs
ceph_rhcs_iso_path: /root/rhceph-3.2-rhel-7-x86_64.iso
cephx: false
rbd_cache: "false"
```

```
rbd_cache_writethrough_until_flush: "false"

monitor_address_block: 192.168.170.0/24
ip_version: ipv4
osd_memory_target: 8589934592
public_network: 192.168.170.0/24
cluster_network: 192.168.180.0/24

osd_objectstore: bluestore

os_tuning_params:
  - { name: kernel.pid_max, value: 4194303 }
  - { name: fs.file-max, value: 26234859 }
  - { name: vm.zone_reclaim_mode, value: 0 }
  - { name: vm.swappiness, value: 1 }
  - { name: vm.min_free_kbytes, value: 1000000 }
  - { name: net.core.rmem_max, value: 268435456 }
  - { name: net.core.wmem_max, value: 268435456 }
  - { name: net.ipv4.tcp_rmem, value: 4096 87380 134217728 }
  - { name: net.ipv4.tcp_wmem, value: 4096 65536 134217728 }

ceph_tcmalloc_max_total_thread_cache: 134217728
ceph_docker_image: rhceph/rhceph-3-rhel7
containerized_deployment: true
ceph_docker_registry: registry.access.redhat.com
ceph_conf_overrides:
  global:
    mutex_perf_counter : True
    throttler_perf_counter : False
    auth_cluster_required: none
    auth_service_required: none
    auth_client_required: none
    auth supported: none
    osd objectstore: bluestore
    cephx require signatures: False
    cephx sign messages: False
    mon_allow_pool_delete: True
    mon_max_pg_per_osd: 800
    mon pg warn max per osd: 800
    ms_crc_header: True
    ms_crc_data: False
    ms type: async
    perf: True
    rocksdb_perf: True
    osd_pool_default_size: 2
    debug asok: 0/0
    debug auth: 0/0
    debug bluefs: 0/0
    debug bluestore: 0/0
    debug buffer: 0/0
    debug client: 0/0
    debug context: 0/0
    debug crush: 0/0
    debug filer: 0/0
    debug filestore: 0/0
    debug finisher: 0/0
    debug hadoop: 0/0
    debug heartbeatmap: 0/0
    debug journal: 0/0
    debug journaler: 0/0
    debug lockdep: 0/0
    debug log: 0
    debug mon: 0/0
    debug monc: 0/0
```

```
    debug ms: 0/0
    debug objclass: 0/0
    debug objectcacher: 0/0
    debug objecter: 0/0
    debug optracker: 0/0
    debug osd: 0/0
    debug paxos: 0/0
    debug perfcounter: 0/0
    debug rados: 0/0
    debug rbd: 0/0
    debug rgw: 0/0
    debug rocksdb: 0/0
    debug throttle: 0/0
    debug timer: 0/0
    debug tp: 0/0
    debug zs: 0/0
  mon:
    mon_max_pool_pg_num: 166496
    mon_osd_max_split_count: 10000
  client:
    rbd_cache: false
    rbd_cache_writethrough_until_flush: false
  osd:
    osd_min_pg_log_entries: 10
    osd_max_pg_log_entries: 10
    osd_pg_log_dups_tracked: 10
    osd_pg_log_trim_min: 10
    bluestore_block_db_size: 15360000000
    bluestore_block_wal_size: 15360000000
    bluestore_csum_type: none
    bluestore_cache_kv_max: 200G
    bluestore_cache_kv_ratio: 0.2
    bluestore_cache_meta_ratio: 0.8
    bluestore_cache_size_ssd: 18G
    bluestore_extent_map_shard_min_size: 50
    bluestore_extent_map_shard_max_size: 200
    bluestore_extent_map_shard_target_size: 100
    disable_transparent_hugepage: true
    journal_queue_max_ops : 8092
    journal_queue_max_bytes : 1048576000
    ms_dispatch_throttle_bytes : 1048576000
    objecter_inflight_ops : 10240
    objecter_inflight_op_bytes : 1048576000
    journal_max_write_entries : 5000
    journal_max_write_bytes : 1048576000
    osd_enable_op_tracker: false
    osd_op_num_threads_per_shard: 2
```


osd.yml


```
---
dummy:
osd_scenario: non-collocated
devices:
  - /dev/sdb
  - /dev/sdc
  - /dev/sdd
  - /dev/sde
  - /dev/sdf
  - /dev/sdg
  - /dev/sdh
  - /dev/sdi
  - /dev/sdj
```

```
  - /dev/sdk
  - /dev/sdl
  - /dev/sdm
  - /dev/sdn
  - /dev/sdo
  - /dev/sdp
  - /dev/sdq

dedicated_devices:
  - /dev/nvme0n1
  - /dev/nvme0n1
  - /dev/nvme0n1
  - /dev/nvme0n1
  - /dev/nvme1n1
  - /dev/nvme1n1
  - /dev/nvme1n1
  - /dev/nvme1n1
  - /dev/nvme2n1
  - /dev/nvme2n1
  - /dev/nvme2n1
  - /dev/nvme2n1
  - /dev/nvme3n1
  - /dev/nvme3n1
  - /dev/nvme3n1
  - /dev/nvme3n1
ceph_osd_docker_cpu_limit: 4
```

mgrs.yml

```
ceph_mgr_docker_cpu_limit: 1
```

mon.yml

```
ceph_mon_docker_cpu_limit: 1
```

OS tunings

```
Modified system control in /etc/sysctl.conf in all storage nodes

# Controls IP packet forwarding
   net.ipv4.ip_forward = 0

# Controls source route verification
   net.ipv4.conf.default.rp_filter = 1

# Do not accept source routing
   net.ipv4.conf.default.accept_source_route = 0

# Controls the System Request debugging functionality of the kernel
   kernel.sysrq = 0

# Controls whether core dumps will append the PID to the core filename.
# Useful for debugging multi-threaded applications.
   kernel.core_uses_pid = 1

# disable TIME_WAIT.. wait .
   net.ipv4.tcp_tw_recycle = 1
   net.ipv4.tcp_tw_reuse = 1

# Controls the use of TCP syncookies
   net.ipv4.tcp_syncookies = 0

# double amount of allowed conntrack
```

```
   net.netfilter.nf_conntrack_max = 2621440
   net.netfilter.nf_conntrack_tcp_timeout_established = 1800

# Disable netfilter on bridges.
   net.bridge.bridge-nf-call-ip6tables = 0
   net.bridge.bridge-nf-call-iptables = 0
   net.bridge.bridge-nf-call-arptables = 0

# Controls the maximum size of a message, in bytes
   kernel.msgmnb = 65536

# Controls the default maximum size of a message queue
   kernel.msgmax = 65536

# Controls the maximum shared segment size, in bytes
   kernel.shmmax = 68719476736

# Controls the maximum number of shared memory segments, in pages
  kernel.shmall = 4294967296
  fs.file-max = 6553600
  net.ipv4.ip_local_port_range = 1024 65000
  net.ipv4.tcp_fin_timeout = 20
  net.ipv4.tcp_max_syn_backlog = 819200
  net.ipv4.tcp_keepalive_time = 20
  kernel.msgmni = 2878
  kernel.sem = 256 32000 100 142
  kernel.shmmni = 4096
  net.core.rmem_default = 1048576
  net.core.rmem_max = 1048576
  net.core.wmem_default = 1048576
  net.core.wmem_max = 1048576
  net.core.somaxconn = 40000
  net.core.netdev_max_backlog = 300000
  net.ipv4.tcp_max_tw_buckets = 10000
```

Spectre/Meltdown security patches disabled on storage nodes and load generators

Note: We recommend keeping these security patches in place for production!

```
  echo 0 > /sys/kernel/debug/x86/pti_enabled
  echo 0 > /sys/kernel/debug/x86/retp_enabled
  echo 0 > /sys/kernel/debug/x86/ibrs_enabled
```

Benchmark details

Dropping caches

```
For BlueStore, you need to restart every OSD container to clear OSD cache

Set the noout flag on the cluster
# ceph osd set noout

The following script can be modified as needed

for server in hstor0 hstor1 hstor2 hstor3; do
ssh root@${server} "for osd in {b..q}; do docker container restart ceph-osd-
${server}-sd${osd}; done"
done

Clear the noout flag when done
```

```
# ceph osd unset noout
```

To learn more

For more information on Dell EMC Service Provider Solutions, visit *https://www.dellemc.com/en-us/service-providers/index.htm*

Glossary

Ansible

Ansible is an open source software utility used to automate the configuration of servers.

API

Application Programming Interface is a specification that defines how software components can interact.

BlueStore

BlueStore is a new OSD storage backend that does not use a filesystem. Instead, it uses raw volumes and provides for more efficient storage access.

BMC/iDRAC Enterprise

Baseboard Management Controller. An on-board microcontroller that monitors the system for critical events by communicating with various sensors on the system board, and sends alerts and log events when certain parameters exceed their preset thresholds.

BOSS

The Boot Optimized Storage Solution (BOSS) enables customers to segregate operating system and data on Directly Attached Storage (DAS). This is helpful in the Hyper-Converged Infrastructure (HCI) and Software-Defined Storage (SDS) arenas, to separate operating system drives from data drives, and implement hardware RAID mirroring (RAID1) for OS drives.

Bucket data

In the context of RADOS Gateway, this is the storage pool where object data is stored.

Bucket index

In the context of RADOS Gateway, this is the storage pool that houses metadata for object buckets.

CBT

Ceph Benchmarking Tool

Cluster

A set of servers that can be attached to multiple distribution switches.

COSBench

An open source tool for benchmarking object storage systems.

CRC

Cyclic redundancy check. This is a mechanism used to detect errors in data transmission.

CRUSH

Controlled Replication Under Scalable Hashing. This is the name given to the algorithm used by Ceph to maintain the placement of data objects within the cluster.

Daemon

Daemon is a long-running Linux process that provides a service.

DIMM

Dual In-line Memory Module

FileStore

FileStore is the original OSD storage backend that makes use of the XFS filesystem.

FIO

Flexible IO Tester (synthetic load generation utility)

Grafana

Grafana is open-source software that provides flexible dashboards for metrics analysis and visualization.

iPerf

iPerf is an open-source tool that is widely used for network performance measurement.

JBOD

Just a Bunch of Disks

LAG

Link Aggregation Group

LINPACK

LINPACK is a collection of benchmarks used to measure a system's floating point performance.

MON

MON is shorthand for the Ceph Monitor daemon. This daemon's primary responsibility is to provide a consistent CRUSH map for the cluster.

MTU

Maximum Transmission Unit

NFS

The Network File System (NFS) is a distributed filesystem that allows a computer user to access, manipulate, and store files on a remote computer, as though they resided on a local file directory.

NIC

Network Interface Card

Node

One of the servers in the cluster

NUMA

Non-Uniform Memory Access

NVMe

Non-Volatile Memory Express is a high-speed storage protocol that uses PCIe bus

OSD

Object Storage Daemon is a daemon that runs on a Ceph storage node and is responsible for managing all storage to and from a single storage device (or a partition within a device).

PG

Placement Group is a storage space used internally by Ceph to store objects.

Prometheus

Prometheus is an open-source software that provides metrics collection into a time-series database for subsequent analysis.

RACADM

Remote Access Controller ADMinistration is a CLI utility that operates in multiple modes (local, SSH, remote desktop) to provide an interface that can perform inventory, configuration, update as well as health status check on Dell PowerEdge servers.

RADOS

RADOS is an acronym for Reliable Autonomic Distributed Object Store and is the central distributed storage mechanism within Ceph.

RADOSGW

RADOS Gateway provides S3 and Swift API compatibility for the Ceph cluster. Sometimes also written as RGW.

RBD

RADOS Block Device is a block device made available in Ceph environment using RADOS.

RGW

RADOS Gateway provides S3 and Swift API compatibility for Ceph cluster. Sometimes also written as RADOSGW.

RocksDB

RocksDB is an open source key-value database that is used internally by BlueStore backend to manage metadata.

S3

The public API provided by Amazon's S3 Object Storage Service.

SDS

Software-defined storage (SDS) is an approach to computer data storage in which software is used to manage policy-based provisioning and management of data storage, independent of the underlying hardware.

Storage Node

A server that stores data within a clustered storage system.

Swift

The public API provided by OpenStack Swift object storage project.

U

U used in the definition of the size of the server, example 1U or 2U. A "U" is a unit of measure equal to 1.75 inches in height. This is also often referred to as a rack unit.

WAL

WAL is an acronym for the write-ahead log. The write-ahead log is the journaling mechanism used by the BlueStore backend.