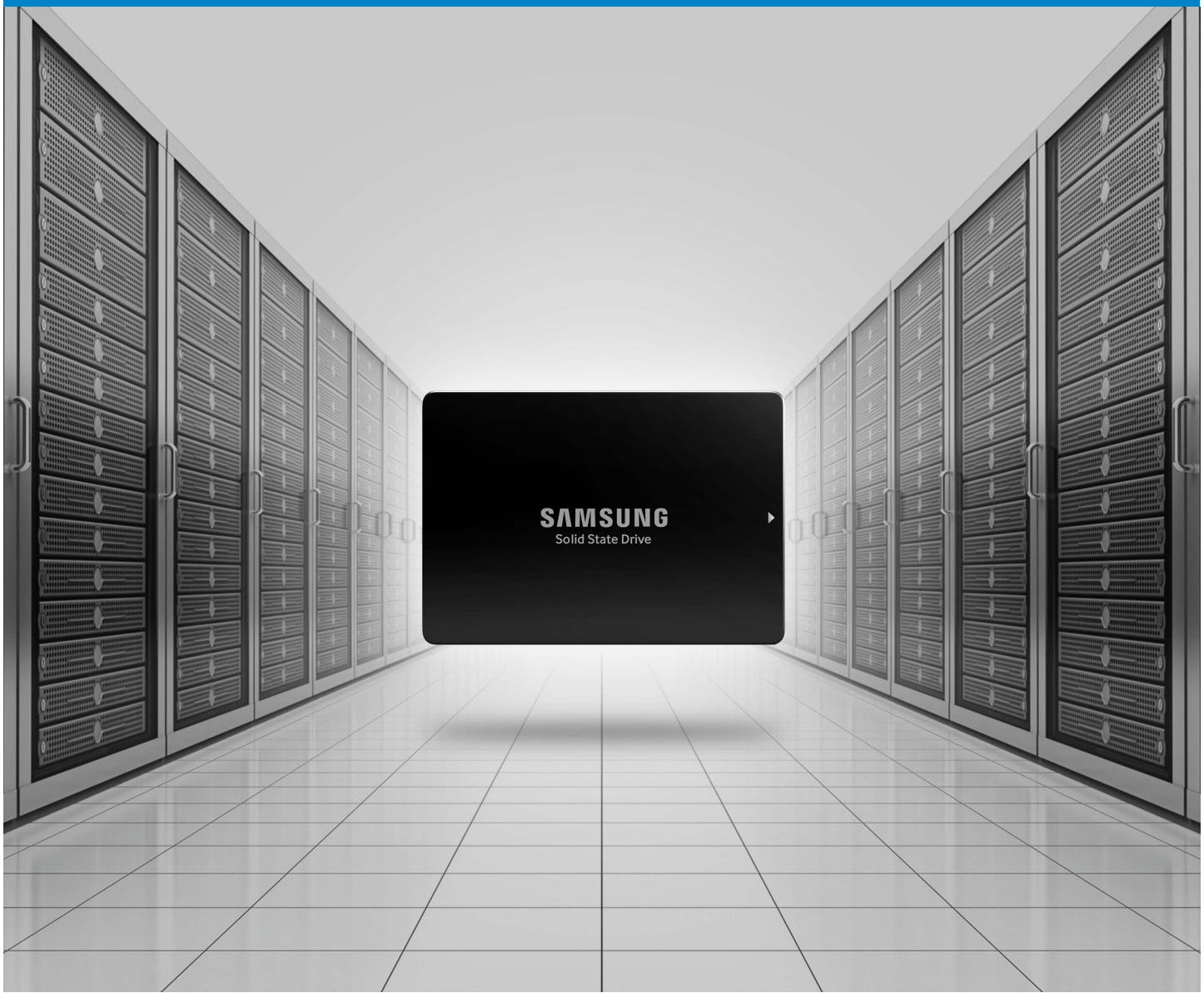
White Paper

High-performance cluster storage for IOPS-intensive workloads

Optimize Ceph cluster performance by combining Red Hat Ceph Storage on Samsung NVMe SSDs



redhat.

SAMSUNG

Table of contents

Table of contents

List of Figures

List of Tables

Red Hat Ceph Storage on Samsung NVMe SSDs

Executive summary

Ceph users frequently request simple, optimized cluster configurations for different workload types. As Ceph takes on high performance intensive workloads, SSDs become a critical component of Ceph clusters. To address the needs of Ceph users to effectively deploy All-Flash Ceph clusters optimized for performance, Samsung Semiconductor Inc. and Red Hat have performed extensive testing to characterize optimized configurations for deploying Red Hat® Ceph Storage on Samsung NVMe SSDs deployed in a Samsung NVMe Reference

Document purpose

This document presents Reference Architecture for deploying a high-performance Red Hat Ceph Storage cluster using Samsung NVMe SSDs and Samsung NVMe Reference Design. This document details hardware and software building blocks used in performance characterization. It covers Ceph cluster and Linux operating system configuration, hardware configuration including Samsung NVMe Reference Design, network and Samsung NVMe SSDs. The test methodologies to characterize Ceph cluster performance used Ceph Benchmarking Tool (CBT)[1] for benchmarking.

The targeted audience for this document is system administrators, solution architects and IT planners.

1. Ceph Benchmarking Tool (CBT) - https://github.com/ceph/cbt

Introduction

Storage infrastructure is undergoing tremendous change, particularly as organizations deploy storage to support big data and private clouds. Two major leading causes for this change are:

- **Adoption of high-performance flash based storage media**

  NVMe has emerged as an industry standard based low-latency storage interface. NVMe SSDs in standard 2.5-inch U.2 drive form factor are widely available in the market today. Samsung has developed a x86 server reference platform with drive bays to support 24 x 2.5-inch NVMe SSDs.

- **Software-defined storage infrastructure based on clustered storage servers**

  Ceph has emerged as a leading solution to deploy cost-effective and manageable storage at scale. 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 infrastructures like OpenStack®.

While NVMe SSDs provide high raw performance and Ceph is extremely flexible, deployments should be carefully designed to deliver high performance while meeting desired fault domain risk tolerance. Organizations need flexible and scalable configurations that scale from a cluster built for IOPS intensive workloads with 100s TBs to a cluster built for throughput intensive workloads with 10s of PBs capacity.

Table 1 provides the criteria used to identify optimal Red Hat Ceph Storage cluster configurations using Samsung NVMe SSDs on Samsung NVMe reference platforms that can take up to 24 x 2.5-inch NVMe SSDs. These criteria are provided as general guidelines for hardware purchase and configuration decisions that can be adjusted to satisfy unique workload blends of different operators. As the workload mix varies from organization to organization, actual hardware configurations chosen will vary.

IOPS optimized	• Lowest cost per IOP • Highest IOPS • Meets minimum fault domain recommendation (single server is less than or equal to 10% of the cluster)	• MySQL®/MariaDB®-based apps on OpenStack • Block storage
Throughput optimized	• Lowest cost per MBps* • Highest MBps • Highest MBps per BTU • Highest MBps per watt • Meets minimum fault domain recommendation (single server is less than or equal to 10% of thecluster) * MBps = Mbytes per sec	• Digital media server workloads • Block or object storage

Table 1: Common workload characteristics

Red Hat Ceph Storage on Samsung NVMe SSDs

The Reference Architecture configurations described in this document are the result of extensive testing by Samsung and Red Hat to evaluate the performance of Red Hat Ceph Storage cluster using Samsung NVMe SSDs within a Samsung NVMe Reference Design. The goals were to provide optimized, repeatable configurations for the IOPS and throughput-optimized criteria listed in Table 1.

- Tests were run on Ceph Hammer (0.94.5) release and Linux kernel 3.10.
- Tests were run on a 3-node Ceph OSD cluster based on a Samsung NVMe reference platform using Samsung PM953 2.5-inch NVMe SSDs.
- TCP/IP stack running on Mellanox® 40 GbE was used for cluster messaging.
- Test results were produced via the CBT framework.
  - librbdfio test suite is used for characterizing random IO performance
  - radosbench test suite is used for characterizing sequential IO performance
- A replication factor of 2 was used for data protection.
- The performance measurements listed in this document are average measurements across multiple clients.
- The test run durations in this study are based on 5 minute test runs.

The resulting optimized configurations can be used as starting points to build a range of cluster sizes, from hundreds of terabytes to multiple petabytes in size. While the configuration listed in this document may work with other software combinations than outlined, further fine-tuning may be required to maximize the performance in such configurations.

Samsung NVMe Reference Design [2]

Samsung NVMe Reference Design is engineered to provide a well-balanced storage server node that includes matching CPUs, networking, storage and PCIe connectivity to deploy large amounts of NVMe SSDs and maximize the performance of software defined storage stacks such as Ceph.

Samsung NVMe Reference Design is a High Performance all-Flash NVMe scale-out storage Server with up to 24 x 2.5-inch hot-pluggable Samsung advanced NVMe SSDs to provide extremely high capacity in a small footprint. It is based on PCIe Gen3 NVMe SSDs to offer the lowest latency in the industry with an optimized data path from the CPU to the SSDs. Each SSD slot provides power and cooling for up to 25 W per SSD to enable the support of current and future generation large capacity SSDs as well as SSDs with different endurance and performance levels. Using the PM953, the max capacity per system is 46 TB. With next generation PM963 SSDs, the max capacity per system is 92 TB, while using the high-endurance PM1725a, the max capacity per system is 153 TB. This is a dual-socket Xeon-based system and EIA compliant 2RU chassis. It also uses 4 x 40 GB/s networking connectivity with RDMA.

The Samsung NVMe Reference Design is available through StackVelocity (a subsidiary of Jabil Systems) as the Greyguard platform.

2. Samsung NVMe Reference Design - http://www.samsung.com/semiconductor/support/tools-utilities/All-Flash-Array-Reference-Design/

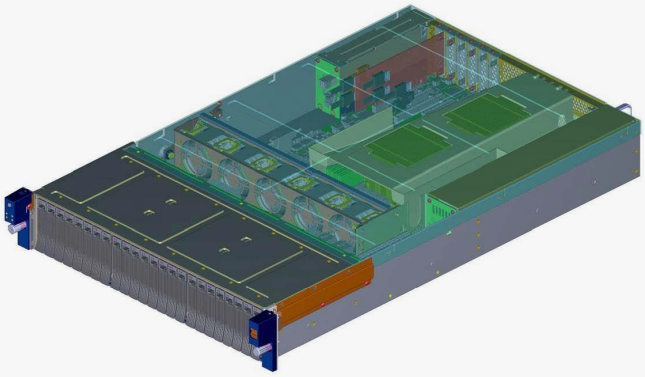


Figure 1: Samsung NVMe Reference Design

Red Hat Ceph Storage on Samsung NVMe SSDs

Ceph distributed architecture overview

A Ceph storage cluster is built from large numbers of Ceph nodes for scalability, fault-tolerance, and performance. Each node is based on commodity 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

To the Ceph client interface that reads and writes data, a Ceph storage cluster looks like a simple pool where data is stored. However, the storage cluster performs many complex operations in a manner that is completely transparent to the client interface. Ceph clients and Ceph object storage daemons (Ceph OSD daemons, or OSDs) both use the CRUSH (controlled replication under scalable hashing) algorithm for storage and retrieval of objects.

For a Ceph client, the storage cluster is very simple. When a Ceph client reads or writes data (referred to as an I/O context), it connects to a logical storage pool in the Ceph cluster. Figure 1 illustrates the overall Ceph architecture, with concepts that are described in the sections that follow.

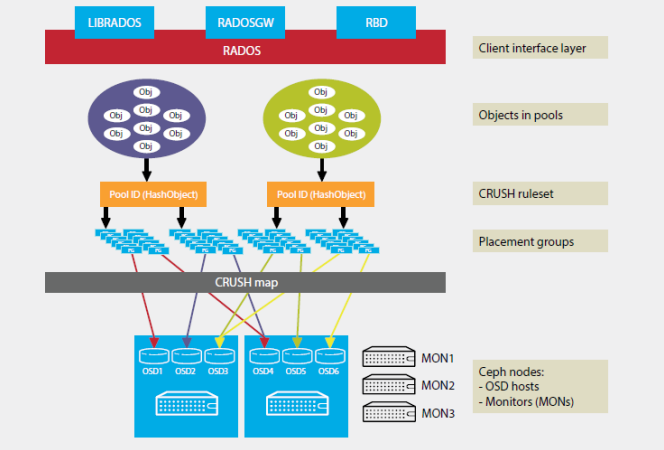


Figure 2: Ceph Architecture

Clients write to Ceph storage pools while the CRUSH ruleset determines how placement groups are distributed across object storage daemons (OSDs).

- **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. Additionally, pools can "take root" at any position in the CRUSH hierarchy, allowing placement on groups of servers with differing performance characteristics—allowing 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 means of creating replication or erasure coding groups of coarser granularity than on a per object basis. A larger number of placement groups (e.g., 200 per OSD or more) leads to better balancing.
- **CRUSH ruleset:** The CRUSH algorithm provides controlled, scalable, and declustered 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 an 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). 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

Red Hat Ceph Storage on Samsung NVMe SSDs

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 OSD daemons roughly correspond to a file system on a physical hard disk drive.

Reference Architecture elements

Each node in the Ceph cluster used in this Reference Architecture has five elements as shown in Figure 3.

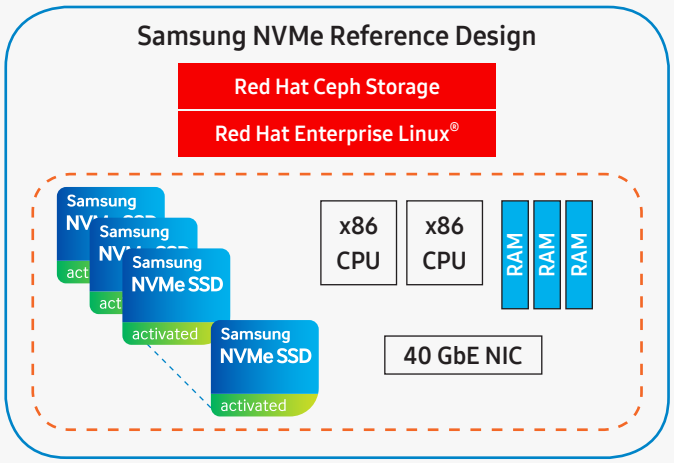


Figure 3: High Performance Ceph Reference Architecture elements

The following sections detail each of the elements in the Reference Architecture.

Red Hat® Ceph Storage

Designed for the cloud, Red Hat® Ceph Storage significantly lowers the cost of storing enterprise data and helps to manage exponential data growth—efficiently and automatically. Delivered in a self-healing, self-managing platform with no single point of failure, Red Hat Ceph Storage handles data management so administrators can focus on improving data availability for business.

The key benefits are:

- **Value**
  Significantly lower the cost of storing data. Lay a foundation for managing its exponential growth at a low cost per gigabyte.
- **Enterprise readiness**
  Integrate tightly with OpenStack® and enjoy advanced block storage capabilities that work just like a traditional block storage device but with hardware flexibility and massive scalability.
- **Longevity**
  Start with block storage and grow into object storage, or vice versa. Integrate with existing storage infrastructure easily.
- **Expert backed**
  Take advantage of the expertise of Ceph's creators and primary sponsors through best-in-class professional services and training.

This Reference Architecture is based on Red Hat Ceph Storage 1.3.

Red Hat Enterprise Linux

With Red Hat® Enterprise Linux®, a platform with unparalleled stability and flexibility, businesses can reallocate infrastructure resources toward meeting the next challenges instead of just maintaining the status quo.

The key benefits are:

- **Freedom through stability**
  Business applications require a tested, proven, predictable platform. Red Hat Enterprise Linux frees IT personnel to deliver meaningful business results by providing exceptional reliability and military-grade security.
- **An ecosystem of solutions and support**
  With a Red Hat Enterprise Linux subscription, administrators are connected to the industry's largest ecosystem of partners, customers, and experts that support and accelerate organizations' success.
- **Confidence through flexibility**
  Red Hat Enterprise Linux gives you the flexibility to tailor your infrastructure for business needs now and in the future. As markets shift and technologies evolve, you'll have the agility, adaptability, and performance to succeed.

This Reference Architecture uses Red Hat Enterprise Linux 7.2 with Linux kernel 3.10.x.

Samsung NVMe SSDs

Samsung Enterprise Solid State Drives (SSDs) are being used increasingly as data storage media in computing, communication, and multimedia devices. Most SSDs use

Red Hat Ceph Storage on Samsung NVMe SSDs

NAND flash memory as the storage media, which is capable of retaining data without an external power supply.

SSDs offer superior reliability compared to traditional hard disk drives (HDDs). Advances in semiconductor flash memory technologies have enabled the development of SSDs that are much larger in capacities compared to HDDs and can be used as direct replacements. SSDs also prove to be highly cost effective in-use due to their much lower power consumption and maintenance costs. As the world leader in semiconductor memory technology, Samsung revolutionized the storage industry by shifting the planar NAND to a vertical structure. Samsung V-NAND technology features a unique design that stacks 48 layers on top of one another instead of trying to decrease the cells' pitch size. Samsung offers a comprehensive range of SSDs for deployment in a wide range of devices across every industry segment. Samsung NVMe Reference Design supports the following Samsung NVMe SSDs:

- **PM953:** PM953 presents outstanding performance with instant responsiveness to the host system, by applying PCIe and NVMe with cost effectiveness. Through V-NAND, it also provides DWPD of 1.3 over 3 years. These drives support power fail support. Standard Linux NVMe drivers work with Samsung PM953 drives.

Table 2 provides summary of PM953 specifications[3].

Model	PM953	**Interface**	PCIe Gen3 x 4
Form factor	2.5 inch	**Capacity**	Up to 1.92 TB
Sequential read (128 KB)	Up to 1,000 MB/s	**Sequential write (128 KB)**	Up to 870 MB/s
Random read IOPS (4 KB)	Up to 240K IOPS	**Random write IOPS (4 KB)**	Up to 19K IOPS
DWPD	1.3 DWPD	**Production status**	Mass production

Table 2: PM953 specification summary

3. Samsung SSD PM953 - http://www.samsung.com/semiconductor/products/flash-storage/enterprise-ssd/MZQLV1T9HCJM?ia=832

- **PM1725:** PM1725 presents the highest levels with unsurpassed random read speeds and an ultra-low latency rate using Samsung's highly innovative 3D vertical-NAND (V-NAND) flash memory and an optimized controller. It also provides DWPD of 5 over 5 years. These drives support power fail support. Standard Linux NVMe drivers work with Samsung PM1725 drives.

Table 3 provides summary of PM1725 specifications .

Model	PM1725	**Interface**	PCIe Gen3 x 4
Form factor	2.5 inch	**Capacity**	Up to 6.4 TB
Sequential read (128 KB)	Up to 3,100 MB/s	**Sequential write (128 KB)**	Up to 2,000 MB/s
Random read IOPS (4 KB)	Up to 750K IOPS	**Random write IOPS (4 KB)**	Up to 120K IOPS
DWPD	5 DWPD	**Production status**	Mass production

Table 3: PM1725 specifications summary

4. Samsung SSD PM1725 -http://www.samsung.com/semiconductor/global/file/insight/2015/11/pm1725-ProdOverview-2015-0.pdf

Red Hat Ceph Storage on Samsung NVMe SSDs

Samsung NVMe Reference Design

This Reference Architecture uses the **Samsung NVMe Reference Design** as OSD nodes in the Ceph cluster. Table 4 provides a summary of the Reference system specifications.

Chassis	
Form factor	2U x 17.2-inch x 30.4-inch
Capacity	20 x 2.5-inch Samsung NVMe SSD slots (hot swap, front) 4 x 2.5-inch Samsung NVMe/SATA/SAS SSD slots (hot swap, front) 2 x 2.5-inch Samsung SAS/SATA SSD slots (hot swap, rear bulkhead)
Temperature management	4 + 1 redundant fans
Power	1,200 W 1 + 1 redundant Gold
Compute and networking	
CPU	2 x Intel® E5-2600v3 up to 145W
Memory	16 x DDR4 2133MHz L/RDIMM
Solid state drive	20 x 2.5-inch Samsung NVMe SSDs w/3D V-NAND (hot swap, front) 4 x 2.5-inch Samsung NVMe/SATA/SAS SSDs (hot swap, front) 2 x 2.5-inch Samsung SAS/SATA SSD slots (hot swap, rear bulkhead)
BMC	AST2400 with dedicated 1 GbE MAC for IPMI 2.0
PCIe expansion	2 x PCIe 3.0 x 16 slots (for host network cards) 1 x PCIe 3.0 x 8 slot (used for optional RAID card)
LAN	4 x 1 GbE, 10 GbE SFP+, 40 GbE QSFP, or FDR Balanced between CPUs for quick data transfers between the network and the drive
Rear IO	2 x USB 3.0 2 x 10 GbE VGA 1 x 1 GbE (IPMI dedicated)
Front IO	Buttons: Pwr, Rst USB 2.0 (x2) LEDs: Pwr, SSD
Enclosure management	I2C Internal Communication

Table 4: All-Flash NVMe Storage System specifications

The key benefits are:

- **Quick Time-to-market:**
  Fully tested and validated platform available from StackVelocity
- **Open Reference Design**
  Standard x-86 based design
  Customizable to support different combinations depending on market requirements

Networking

This Reference Architecture uses Mellanox ConnectX®-4 EN adapters[5] in 2 x 40 GbE configuration. The key features of the adapters are:

- 100 GB/s Ethernet per port
- 1/10/20/25/40/50/56/100 GB/s speeds
- Single and dual-port options available
- Erasure Coding offload
- T10-DIF Signature Handover
- Power8 CAPI support
- CPU offloading of transport operations
- Application offloading
- Mellanox PeerDirect™ communication acceleration
- Hardware offloads for NVGRE and VXLAN encapsulated traffic
- End-to-end QoS and congestion control
- Hardware-based I/O virtualization
- Ethernet encapsulation (EoIB)
- RoHS-R6

This Reference Architecture also uses Mellanox SX1036[6], a 36-port Non-blocking 40/56 GbE Open Ethernet Spine Switch System. The Reference Architecture configures the switch ports in 40 Gb Ethernet mode.

5. Mellanox ConnectX®-4 EN - http://www.mellanox.com/page/products_dyn?product_family=204&mtag=connectx_4_en_card

6. Mellanox SX1036 - http://www.mellanox.com/page/products_dyn?product_family=115&mtag=sx1036

Red Hat Ceph Storage on Samsung NVMe SSDs

Operational planning considerations

This section presents general guidance on operational planning for deploying high performance Ceph Storage clusters using Samsung NVMe SSDs.

- Minimum number of Ceph Storage nodes: 3 (an odd number of monitor nodes is mandatory); it is recommended to have at least 3 storage nodes in a Ceph cluster to become eligible for Red Hat technical support.
- Monitor nodes: 3; it is recommended to configure monitor nodes on separate nodes; while these nodes do not need to have high performance CPUs, they would benefit from high performance SSDs to store monitor map data.
- Capacity of 1 storage server: <= 10% Ceph cluster capacity; this recommendation may be relaxed for less critical pilot projects.
- Replication factor: 2; given the better MTBF and MTTR of flash-based media, many Ceph customers have chosen to run 2x replication in production when deploying OSDs on flash, vs. the 3x replication common with magnetic media deployments.
- Reserved capacity for self-healing: Ceph has self-healing capabilities to automatically recover the data from drive/node failures. Care should be taken to operate the Ceph cluster to leave enough unused capacity to recover from failures. Clusters comprised of >10 nodes are frequently run at 70-75% capacity in production.
- Networking: at least a 10 GbE cluster will be required to leverage the performance benefits of NVMe SSD based Ceph cluster. For throughput-oriented workloads, at least 40 GbE per server is recommended.
- Thermal management: Care should be taken to maintain cooling as specified in the hardware specifications.

Optimized configurations

Despite the flexibility of Ceph, no one cluster or pool configuration fits all applications or situations. Instead, successful configuration of a Ceph cluster requires answering key questions about how the cluster will be used and the applications it will serve.

Identifying target workload I/O profiles

Accommodating the target workload I/O profile is perhaps the most crucial design consideration. As a first approximation, organizations need to understand if they are simply deploying low-cost archive storage or if their storage needs to meet specific performance requirements. For performance-oriented Ceph clusters, IOPS, throughput, and latency requirements must be clearly defined. On the other hand, if the lowest cost per terabyte is the overriding need, Ceph cluster architecture can be designed at dramatically lower costs. For example, Ceph object archives with erasure-coded pools and without dedicated SSD write journals can be dramatically lower in cost than Ceph block devices on 3x-replicated pools with dedicated flash write journals.

For more performance-oriented needs, IOPS and throughput targets are often established. Historically, Ceph has performed very well with high-throughput workloads, and has been widely deployed for these use cases. These use cases are frequently characterized by large-block, asynchronous, sequential I/O (e.g., digital media performance nodes). In contrast, high IOPS workloads are frequently characterized by small-block synchronous random I/O (e.g., 4 KB random I/O). The use of Ceph for high IOPS open source database workloads is emerging (e.g., MySQL, MariaDB, and PostgreSQL)[7]. Moreover, when Ceph is deployed as Cinder block storage for OpenStack virtual machine (VM) instances, it typically serves a mix of IOPs- and throughput-intensive I/O patterns.

Table 1 summarizes generic properties of these two workload categories – IOPS Optimized and Throughput Optimized.

7. MySQL on Ceph - http://www.slideshare.net/Red_Hat_Storage/my-sql-on-ceph

Generic optimizations

This section presents configuration optimizations that are applicable to both IOPS optimized and throughput optimized deployments of a Ceph cluster on Samsung NVMe reference platforms.

System tunables

Ceph IO path traverses through several kernel modules in the Linux stack. Default values of these respective modules will not be the best fit for a Ceph configuration optimized for performance. Section 9.3 lists the various system parameters that are tuned and used in Ceph reference test clusters used for this Reference Architecture.

Red Hat Ceph Storage on Samsung NVMe SSDs

Symmetric configuration

In a high-performance server platform such as a Samsung NVMe reference platform, NVMe SSDs and high-speed NICs compete for common CPU and memory system resources. Maintaining CPU socket affinity across SSDs and NIC ports serving Ceph OSD daemons is critical to achieve higher performance. This is one of the key design features of the Samsung NVMe Reference Design. Figure 4 shows a dual-socket configuration where cores on CPU0 service interrupts from one set of NVMe SSDs and NIC ports, while cores on CPU1 service interrupts from a different set of NVMe SSDs and NIC ports.

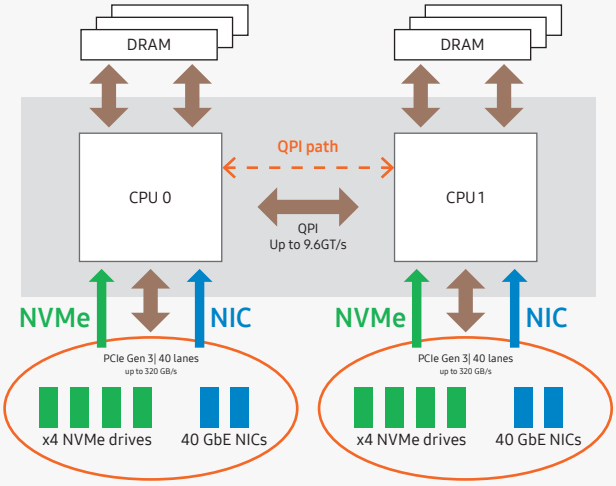


Figure 4: Symmetric configuration in Ceph

This kind of assignment of system resources results in better cache locality and avoids the overhead associated with crossing QPI in Intel® Xeon® CPU based x86 systems. In multiple NIC port test configurations used in this Reference Architecture, Samsung NVMe reference nodes are configured in symmetric modes as shown in Figure 4.

CPU sizing

Ceph provides rich data management features; these features, not listed exhaustively, include dynamic space allocation to objects at the time of object creation rather than volume/image creation, flexibility to dynamically update the cluster configuration, protection across multiple failure domains, periodic checking to maintain data integrity, etc. In order to provide such rich functionality on top of off-the-shelf hardware, OSD daemons in Ceph perform a lot of processing.

A higher CPU core count results in higher performance in Ceph for IO intensive workloads. To reach highest IOPS targets, a ratio of 10 Xeon® cores per NVMe drive is used. For throughput-optimized workloads characterized by large sequential IO patterns, this ratio is relaxed.

NIC configuration

It's a common practice to bond low-speed network interfaces to create a higher speed single network interface, which can then be used to configure the public and cluster networks for ceph-osd daemons. However, bonding configurations may not be required when deploying high performance network interfaces, thereby reducing operational complexity. A Samsung NVMe reference platform supports two PCIe Gen3 x16 slots for external network connectivity; each of these slots is attached to one CPU socket. Each slot allows 4 x 25 GbE OR 2 x 40 GbE OR 1 x 100 GbE NIC card. To deliver high performance, it is recommended both the PCIe slots are populated with high-performance NIC cards based on the cluster interconnect configuration.

NIC ports on these cards can be configured in stand-alone mode without any bonding. This allows maintaining symmetric configuration as shown in Figure 4. To avoid inter-OSD communication within the same node on different NIC ports, ceph.conf needs to be configured as follows.

```
osd_crush_chooseleaf_type = 0
```

In addition, to allow a higher number of simultaneous connections from each OSD, ceph.conf can be configured to have its own TCP port range. An example setting is shown below.

```
ms_bind_port_min = 7000
ms_bind_port_max = 8000
```

Red Hat Ceph Storage on Samsung NVMe SSDs

Journals

In a hybrid Ceph configuration, it is common to use separate SSDs for OSD journals. However, in an all-flash Ceph cluster using Samsung NVMe SSDs on Samsung NVMe reference platforms, separating the journal from the OSD datastore usually does not produce additional benefits. In these all-flash configurations, a Ceph journal is frequently co-located on the same NVMe drive in a different partition from the OSD data. This maintains a simple to use configuration and also limits the scope of the any drive failures in an operational cluster.

IOPS optimized configuration

When deploying higher performance NVMe SSDs to deliver higher random IO performance, the OSD node configuration in the cluster needs to support a higher number of CPU cores. In a single OSD node with 4 NVMe SSDs, as shown in Figure 5, Ceph delivers a 24% performance increase in 100% 4K random read IOPS as the core count on an OSD node is increased from 24 cores to 36 cores.

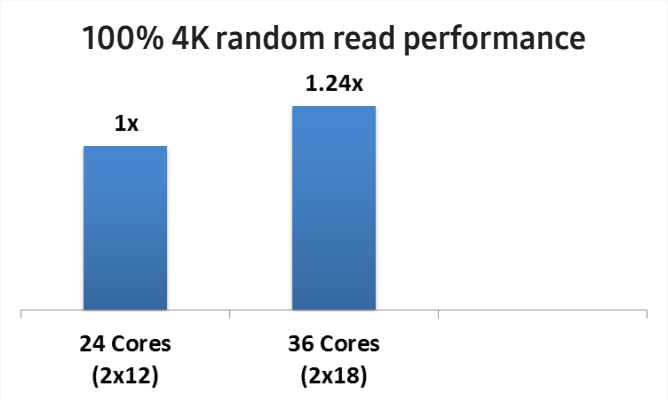


Figure 5: Performance scaling with CPU core count

Samsung NVMe reference platforms offer flexibility in provisioning hardware so as to meet the deployment requirements in terms of random IOPS performance and capacity. Different ceph.conf settings can be used to deploy a Ceph cluster to meet the needs. Table 5 summarizes the random IO performance for 4 KB IOs observed with different SSDs in a 3-node Ceph reference test cluster. Note that all performance is measured from the client perspective. As such, the write performance numbers mask the write amplification required for OSD journal writes as well as 2x replication. Performance from a higher number of SSDs in the cluster is capped by CPU bottlenecks.

# of SSDs per OSD node	# of OSDs per SSD	Effective cluster capacity (replication factor = 2)	100% random read (4 KB)	100% random write (4 KB)
4	4	5.76 TB	579K	70.3K
24	1	34.56 TB	693K	87.8K

Table 5: IOPS performance comparison

To achieve desired IOPS performance levels in a Ceph cluster, there are four key attributes that should be considered:

- Number of OSD nodes – Delivers aggregate IOPS across many nodes
- Number of SSDs per OSD node – Provides raw device IOPS
- Number of CPU cores per SSDs – Drives raw device IOPS.
- Number of OSDs per SSD – In smaller SSD configurations, up to 4 OSDs per SSD will give higher performance; but in larger SSD configurations, multiple OSDs per SSD may have a negative performance impact due to increased context switching costs.

While sizing IOPS optimized Ceph clusters, care should also be taken to the IOPS size that cluster is being optimized for. Table 6 summarizes the random IO performance for 4 KB and 8 KB IOs observed with the same number of SSDs in a 3-node Ceph reference test cluster. The write performance accounts for OSD journal writes as well as 2x replication.

# of SSDs per OSD node	# of OSDs per SSD	IO size	100% random read	100% random write
4	4	4 KB	579K	70.3K
4	4	8 KB	484K	60K
24	1	4 KB	693K	87.8K
24	1	8 KB	529K	82.7K

Table 6: IOPS Performance – 4 KB vs. 8 KB

As shown in Table 6, a node with only 4 SSDs can deliver 80% as many IOPS as a 24 SSD node. As such, to achieve optimal price-performance, workloads requiring highest small, random IO performance should be deployed on Samsung NVMe Reference Design based systems with 4 NVMe drives and dual-socket Xeon® E5-2699v3 processors.

Red Hat Ceph Storage on Samsung NVMe SSDs

Throughput optimized configuration

Samsung NVMe Reference Design based systems offer flexibility in provisioning hardware so as to meet a variety of deployment requirements across IOPS, throughput, and capacity targets.

Samsung NVMe Reference Design uniquely leverages the high throughput capabilities of NVMe SSDs by matching them with high throughput capable NICs, with a balanced NVMe bus design. Samsung NVMe Reference Designs enable Ceph clusters to deliver very high throughput rates with a small server footprint and lower operational costs.

Different ceph.conf settings can be used to deploy a Ceph cluster to meet these throughput needs. Table 7 summarizes the throughput performance for 1 MB IOs observed with different SSDs in 3-node Ceph reference test cluster. Note that all performance is measured from the client perspective. As such, the write performance numbers mask the write amplification required for OSD journal writes as well as 2x replication. Performance from a higher number of SSDs in the cluster is capped by CPU bottlenecks.

# of SSDs per OSD node	# of OSDs per SSD	Effective cluster capacity (replication factor = 2)	100% random read (1 MB)	100% random write (1 MB)
4	4	5.76 TB	11.6 GB/s	2.1 GB/s
24	1	34.56 TB	28.5 GB/s	6.25 GB/s

Table 7: Throughput performance comparison

Unlike for random IOPS, increased number of OSDs per SSD showed minimal impact on the throughput performance.

To achieve desired throughput performance levels in a Ceph cluster, there are three key attributes that should be considered:

- Number of OSD nodes – Delivers aggregate throughput across nodes.
- Number of SSDs per OSD node – Provides raw device throughput.
- Number of NICs per OSD node – Provides network throughput to match the drive throughput.

Benchmark results

This section presents the detailed results gathered on a Ceph reference test cluster.

Reference test configurations

The sizing guidelines are based on the performance benchmarking studies performed on Samsung NVMe Reference Design using NVMe SSDs in a 3-node Ceph cluster as shown in Figure 6.

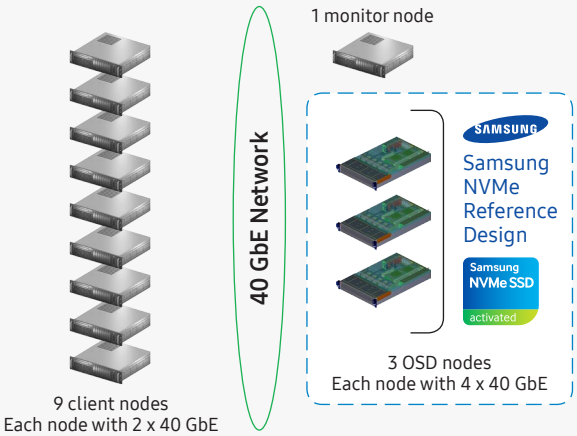


Figure 6: Samsung/Red Hat Ceph Reference test cluster

Table 8 summarizes the system details for a Samsung NVMe reference platform used in the reference test cluster.

Component	Configuration details
2 x Intel® Xeon® CPU E5-2699 v3 @ 2.30GHz	With 18 cores per socket, this will have 36 cores per system. With HT on, OS will list 72 hardware threads.
256 GB	To avoid memory limitations in using high-performance SSDs and NICs.
Up to 24 x PM953	Each drive with 960 GB raw capacity.
Mellanox ConnectX®-4 EN adapters	1x to 4x 40 GbE ports; to avoid any network bottlenecks in delivering high IOPS in a cluster with replication or recovery traffic.

Table 8: OSD node configuration

Red Hat Ceph Storage on Samsung NVMe SSDs

Table 9 summarizes the system details for client nodes used in the reference test cluster.

Component	Configuration details
2x Intel® Xeon® CPU E5-2670 v3 @ 2.30 GHz	With 12 cores per socket, this will have 24 cores per system. With HT on, OS will list 48 hardware threads.
128 GB	To avoid memory limitations in using high-performance NICs.
Mellanox ConnectX®-3 EN adapters	2 x 40 GbE port

Table 9: Client node configuration

This Reference Architecture used Ceph Benchmarking Tool1 (CBT) to characterize the performance of Ceph clusters. CBT is a testing harness written in python that can automate a variety of tasks related to testing the performance of Ceph clusters. CBT records system metrics with collectl. ceph-osd daemons are configured with the same NIC port for both public and cluster networks.

CBT benchmark modules that were used in this Reference Architecture are presented below:

- **radosbench for sequential IO**
  RADOS bench testing uses the rados binary that comes with the ceph-common package. It contains a benchmarking facility that exercises the cluster by way of librados, the low level native object storage API provided by Ceph. Currently, the RADOS bench module creates a pool for each client.
- **librbdfio for random IO**
  The librbdfio benchmark module is the simplest way of testing block storage performance of a Ceph cluster. Recent releases of the flexible IO tester (fio) provide a Ceph RBD ioengine. This allows fio to test block storage performance of RBD volumes without KVM/QEMU configuration, through the user-space librbd libraries. These libraries are the same ones used by the, QEMU backend, so it provides an approximation to KVM/QEMU performance.

The following sections present performance results of the above benchmark runs on 3-node Ceph cluster as shown in Figure 6.

Table 10 lists the debug values that were used when gathering the data, as documented in the default ceph.conf file to make the results closer to actual real life deployments.

```
debug_crush = 1/1
debug_auth = 1/5
debug_finisher = 1/5
debug_heartbeatmap = 1/5
debug_perfcounter = 1/5
debug_rgw = 1/5
debug_asok = 1/5
debug_throttle = 1/1
```

Table 10: Default debug values in test cluster

Table 11 provides the software versions used in the Reference Architecture.

Software	Version
Linux distribution	CentOS 7.2
Linux kernel	3.10.0-327.13.1.el7.x86_64
Ceph	Hammer LTS (0.94.5)

Table 11: Reference Architecture software versions

Sequential IO

Sequential performance was evaluated for sequential IO workloads of large IO sizes; associated CPU, memory and network utilization were also observed. All the tests were run with 2 x replication, and both journal and OSD data are stored in different partitions on the same SSD. Throughput was measured using radosbench under CBT test framework; 128 concurrent ops and 8K PGs per pool were used.

Throughput Scaling with SSDs

For sequential reads of large IO sizes (1 MB), the 3-node Ceph cluster running on a Samsung NVMe reference platform delivers 28.5 GB/s. As shown in Figure 7, the throughput performance scales from 11.6 GB/s to 28.5 GB/s as the SSD count increased in the cluster. In clusters with a lower number of SSDs per node (4 per OSD node), the throughput performance is pretty close to the raw drive performance, whereas with a higher number of SSDs per node (24 per OSD node), CPUs on the OSD nodes saturate before achieving the raw drive performance. However, the increased number of SSDs deliver ~ 2.4 x throughput.

Red Hat Ceph Storage on Samsung NVMe SSDs

For IOPS-optimized workloads, multiple OSDs mapped to a single SSD have shown to improve performance of small block random IO. For throughput-optimized workloads with large block sequential IO, however, as shown in Figure 7, this is not the case. For a given number of SSDs per node, the sequential read performance for large IOs is relatively same as the number of OSDs mapped to each SSD increased from 1 to 4. For throughput intensive workloads, the recommended configuration is a 1:1 mapping between OSD:SSD.

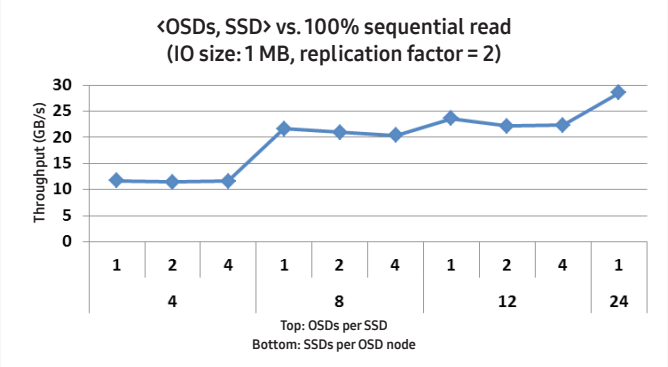


Figure 7: 3-Node Cluster Read Throughput – SSD scaling

For sequential writes of large IO sizes (1 MB), a 3-node Ceph cluster running on a Samsung NVMe reference platform delivers effective throughput of 6.2 GB/s. This write performance is measured from the client perspective, and masks the write cluster write amplification from Ceph OSD journal writes and 2 x replication for data protection. As shown in Figure 8, the throughput performance scales from 2.1 GB/s to 6.2 GB/s as the SSD count increased in the cluster. In clusters with a lower number of SSDs per node (4 per OSD node), the throughput performance is ~ 80% of the raw drive performance, whereas with a higher number of SSDs per node (24 per OSD node), CPUs on the OSD nodes saturate before achieving the raw drive performance. However, the increased number of SSDs deliver ~ 2.9x throughput.

As shown in Figure 8, for a given number of SSDs per node, the sequential write performance for large IOs is relatively same as the number of OSDs mapped to each SSD increased from 1 to 4. In some cases, due to increase in context switches, lower performance is observed when more OSDs are mapped to a single SSD. As discussed above, for throughput intensive workloads, the recommended configuration is the more common 1:1 mapping between OSD:SSD.

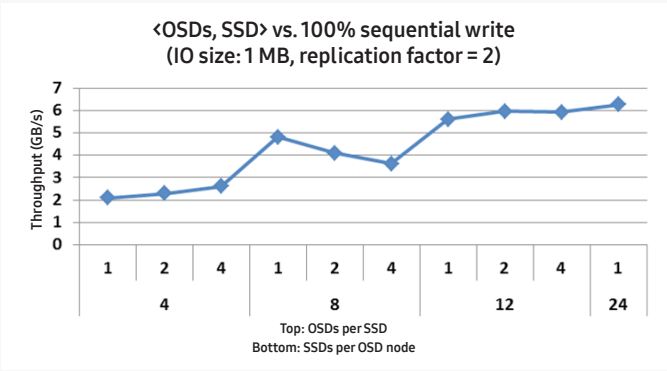


Figure 8: 3-Node Cluster Write Throughput – SSD scaling

Capacity sizing

Samsung PM953 NVMe SSDs of 960 GB capacity are used in the reference test cluster. As the number of SSDs are increased from 4 per OSD node to 24 per OSD node, the effective capacity of the cluster accounting for 2x replication increases from 5.76 TB to 34.56 TB.

Figure 9 shows the increased sequential read for large IOs (1 MB) throughput to scale from 11.6 GB/s to 28.5 GB/s as the cluster capacity increases. The dotted line in the figure shows read throughput in MB/s per TB of effective storage capacity in the cluster. Figure 10 shows the increased sequential write for large IOs (1 MB) throughput to scale from 2.1 GB/s to 6.2 GB/s as the cluster capacity increases; this performance accounts for OSD journal writes and 2 x replication for data protection. The dotted line in the figure shows read throughput in MB/s per TB of effective storage capacity in the cluster.

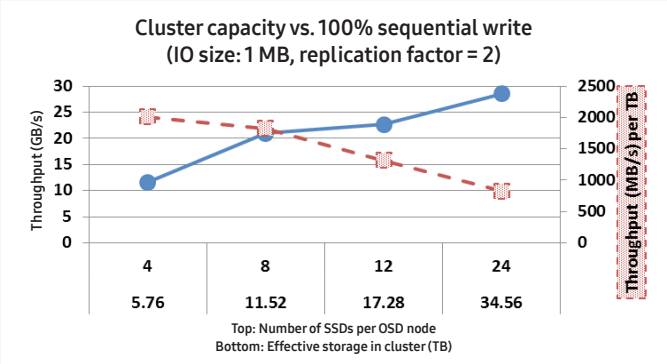


Figure 9: 3-node cluster read throughput – Capacity sizing

Red Hat Ceph Storage on Samsung NVMe SSDs

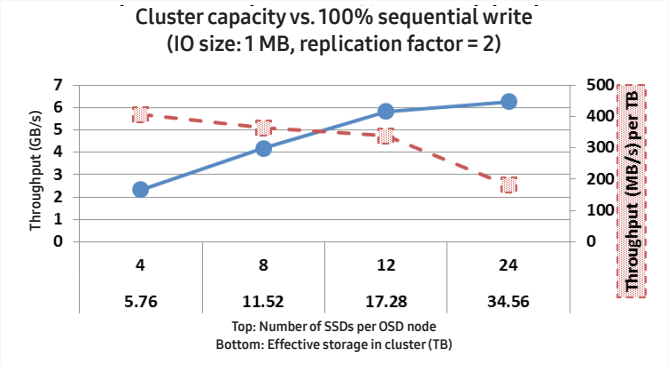


Figure 10: 3-node cluster write throughput – Capacity sizing

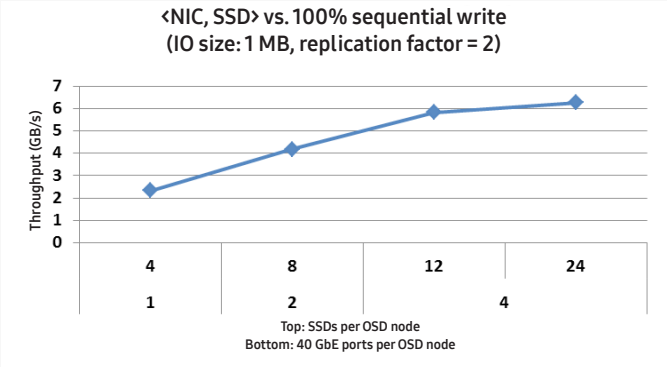


Figure 12: 3-node cluster write throughput – NIC scaling

Throughput Scaling with NICs

The Ceph reference test cluster is built with 40 GbE fabric. Each of the Samsung NVMe reference nodes in the cluster can support up to 4 x 40 GbE ports. The NIC ports are not bonded in the test cluster.

Figure 11 and Figure 12 show the benefits of balancing SSD throughput with a number of 40 GbE NIC ports on each OSD node. As the number of SSDs is increased per OSD, the increased number of 40 GbE NIC ports reflects the throughput gains for both sequential read and write operations of large IOs.

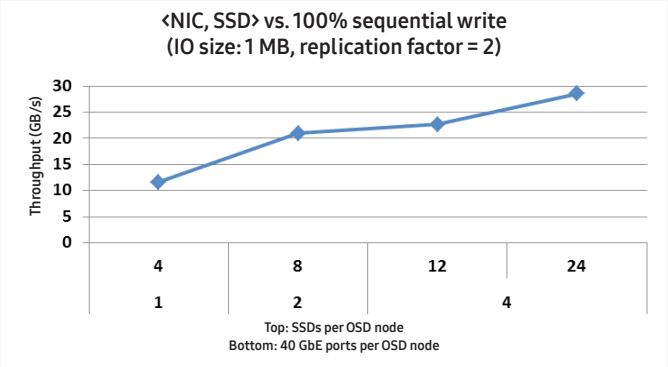


Figure 11: 3-node cluster read throughput – NIC scaling

Random IO

IOPS performance was evaluated for random IO workloads of small IO sizes; associated CPU, memory and network utilization were also observed. All the tests were run with 2 x replication, and both journal and OSD data are stored in separate partitions on the same SSD.

IOPS performance was measured using librbdfio under a CBT test framework; only one OSD pool with 16K PGs was used during the tests. The dataset size was 5.8 TB.

IOPS Scaling with SSDs

For random reads of small IO sizes (4 KB) 3-node Ceph cluster running on a Samsung NVMe reference platform delivers 693K IOPS. As shown in Figure 13, the IOPS performance scales from 394K IOPS to 693K IOPS as the SSD count is increased in the cluster. For random IOPs tests, CPUs on the OSD nodes saturate before achieving the raw drive performance.

For IOPS-optimized workloads, multiple OSDs mapped to a single SSD have shown to improve performance of small block random IO. As shown in Figure 13, in a configuration with 4 SSDs per OSD node, the random read performance for small IOs increases as the number of OSDs mapped to each SSD increased from 1 to 4. However, at higher number of OSDs per OSD node such mapping does not always deliver higher performance and in some cases reduces performance due to increased overhead associated with context switching. As illustrated, optimal price-performance for small block random IO configurations is achieved with 4 SSDs per OSD node (with 4 OSDs mapped to each SSD).

Red Hat Ceph Storage on Samsung NVMe SSDs

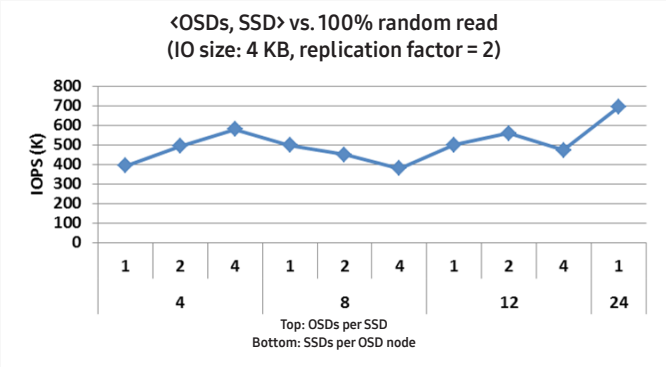


Figure 13: 3-node cluster read IOPS – SSD/OSD scaling

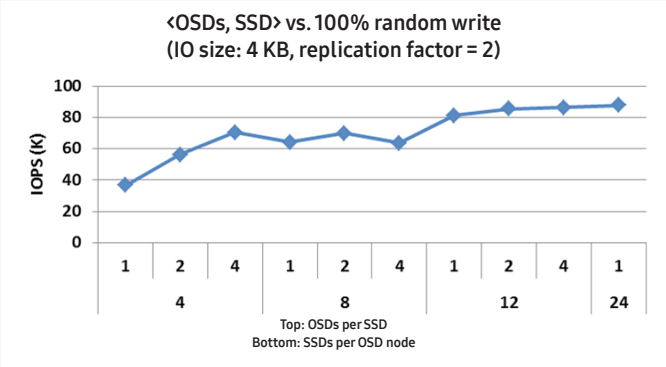


Figure 14: 3-node cluster write IOPS – SSD/OSD scaling

For random writes of small IO sizes (4 KB) 3-node Ceph cluster running on a Samsung NVMe reference platform delivers 87.8K IOPS; this accounts for OSD journal writes and 2 x replication for data protection. As shown in Figure 14, the IOPS performance scales from 34.4K IOPS to 87.8K IOPS as the SSD count is increased in the cluster. For random IOPs tests, CPUs on the OSD nodes saturate before achieving the raw drive performance.

For IOPS-optimized workloads, multiple OSDs mapped to a single SSD have shown to improve performance of small block random IO. As shown in Figure 14, in a configuration with 4 SSDs per OSD node, the random write performance for small IOs increases as the number of OSDs mapped to each SSD increased from 1 to 4. However, such mapping does not always deliver higher performance and in some cases reduces performance due to increased overhead due to context switching at higher number of SSDs per OSD node. As illustrated, optimal price-performance for small block random IO configurations is achieved with 4 SSDs per OSD node (with 4 OSDs mapped to each SSD).

IOPS for 8 KB IOs

While random IOPS performance is often presented for 4 KB IOs as listed above, databases and other IO intensive applications use 8 KB and larger IO sizes as transaction size increases.

As shown in Figure 15 and Figure 16 random read and write IOPS performance for 8 KB IOs is 529K and 82.7K respectively. 8 KB IO random read performance is ~ 23% lower than 4 KB IO random read performance, and 8 KB IO random write performance is ~ 5% lower than 4 KB IO random write performance.

Figure 15: 3-node cluster read IOPS – 4 KB vs. 8 KB

Red Hat Ceph Storage on Samsung NVMe SSDs

Figure 16: 3-node cluster write IOPS – 4 KB vs. 8 KB

Capacity sizing

Samsung PM953 NVMe SSDs of 960 GB capacity are used in the reference test cluster. As the number of SSDs are increased from 4 per OSD node to 24 per OSD node, the effective capacity of the cluster accounting for 2 x replication increases from 5.76 TB to 34.56 TB.

Figure 17 shows the increased random read IOPS for small IOs (4 KB) to scale from 488.9K (average IOPS across 1 to 4 OSDs mapped to a single SSD) to 693K (average IOPS with 1 OSD mapped 1 SSD) as the cluster capacity increases. The dotted line in the figure shows read IOPS per GB of effective storage capacity in the cluster. Figure 18 shows the increased random write IOPS for small IOs (4 KB) to scale from 54.4K (average IOPS across 1 to 4 OSDs mapped to a single SSD) to 87.8K (average IOPS with 1 OSD mapped 1 SSD) as the cluster capacity increases. The dotted line in the figure shows write IOPS per GB of effective storage capacity in the cluster.

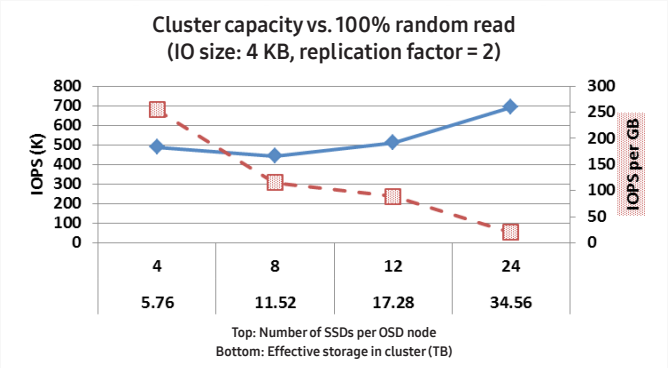


Figure 17: 3-node cluster read IOPS – Capacity sizing

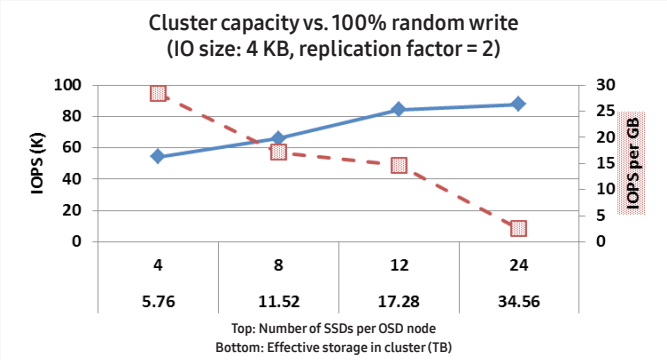


Figure 18: 3-node cluster write IOPS – Capacity sizing

Testing methodology and details

This section provides details on the testing methodology used in establishing the optimized configurations in this Reference Architecture.

Figure 19 lists the steps testing for the Reference Architecture progressed through.

Step 1:
Baseline storage node performance

Step 2:
Network bandwidth consistency across nodes

Step 3:
RADOS baseline: CBT/ radosbench over 2 x replicated pool

Step 4:
RBD baseline: CBT/ librbdfio over 2 x replicated pool

Figure 19: Test methodology steps in Reference Architecture

Baseline storage node performance

The purpose of these test runs is to establish a single node raw storage IO performance of the storage media present in the node. Steps followed to run the tests are listed below:

- Each of the SSDs in the storage node is mounted using XFS and with the same options as they would be mounted in a Ceph configuration.

```
o mkfs.xfs -f -i size=2048 <SSD device>
o mount -t xfs <SSD device> -o noatime,inode64,lo
  gbsize=256k,delaylog  /var/lib/ceph/osd/ceph-
  <osd#>
```

Red Hat Ceph Storage on Samsung NVMe SSDs

- SSDs are pre-conditioned as appropriate to the device.
- FIO® test tool is used to exercise the drive; typically, with
  - 4 FIO jobs with a queue depth of 32
  - IO Sizes of 4 KB, 8 KB, 128 KB, and 4 MB
  - Read write workloads: 100% random read, 100% random write for small IOs (4 KB and 8 KB), and 100% sequential read, 100% sequential write for large IOs (128 KB and 4 MB).
- IOPS for small IO sizes, throughput for large IO sizes and latency across the tests are measured.
- Aggregate drive performance across all drives in the storage node is also measured.

Network bandwidth

Ceph uses a message based protocol with all participants in a Ceph cluster exchanging messages with each other across the cluster. It is critical that any potential network performance issues are identified and resolved to ensure a high performance Ceph cluster.

Steps followed to run the tests are listed below:

- Run bi-directional iperf tests across two end points and ensure that the bandwidth rates are close to link rate.
- Verify the network connectivity performance from each client to each storage node in the cluster on the public network.
- Verify the network connectivity and performance from each client to each of the monitor nodes in the cluster on the public network.
- Verify the network connectivity and performance from each storage node in the cluster to the each of the monitor nodes in the cluster on the public network.
- Verify the network connectivity and performance from each storage node in the cluster to all other storage nodes in the cluster on the cluster network.

Jumbo frames of MTU = 9000B are enabled on the NIC interfaces in the network.

System tunables

In addition, the following NIC and system tunables are set:

- /etc/sysctl.conf

```
# System default settings live in /usr/lib/sysctl.
d/00-system.conf.
# To override those settings, enter new settings
here, or in an /etc/sysctl.d/<name>.conf file
#
# For more information, see sysctl.conf(5) and
sysctl.d(5).
kernel.pid_max=4194303
fs.file-max=4194303
# VM Settings
vm.swappiness = 1
vm.vfs_cache_pressure = 1
# Increase Linux autotuning TCP buffer limits
# Set max to 16MB for 1GE and 32M (33554432) or 54M
(56623104) for 10GE
# Don't set tcp_mem itself! Let the kernel scale it
based on RAM.
# Use 128M buffers
net.core.rmem_max=268435456
net.core.wmem_max=268435456
net.core.rmem_default=67108864
net.core.wmem_default=67108864
net.core.netdev_budget=1200
net.core.optmem_max=134217728
net.ipv4.tcp_rmem=67108864 134217728 268435456
net.ipv4.tcp_wmem=67108864 134217728 268435456
net.ipv4.tcp_low_latency=1
net.ipv4.tcp_adv_win_scale=1
#
# Make room for more TIME_WAIT sockets due to more
clients,
# and allow them to be reused if we run out of
sockets
# Also increase the max packet backlog
net.core.somaxconn=32768
# Increase the length of the processor input queue
net.core.netdev_max_backlog=250000
net.ipv4.tcp_max_syn_backlog=30000
net.ipv4.tcp_max_tw_buckets=2000000
net.ipv4.tcp_tw_reuse=1
net.ipv4.tcp_tw_recycle=1
net.ipv4.tcp_fin_timeout=5
#
# # Disable TCP slow start on idle connections
net.ipv4.tcp_slow_start_after_idle=0
#
# # If your servers talk UDP, also up these limits
net.ipv4.udp_rmem_min=8192
net.ipv4.udp_wmem_min=8192
#
# # Disable source routing and redirects
net.ipv4.conf.all.send_redirects=0
net.ipv4.conf.all.accept_redirects=0
```

Red Hat Ceph Storage on Samsung NVMe SSDs

```
net.ipv4.conf.all.accept_source_route=0
#
# # Recommended when jumbo frames are enabled
net.ipv4.tcp_mtu_probing=1
```

- Disable firewall and start ntpd service in all nodes.

```
# rpm -qa firewalld | grep firewalld && sudo
systemctl stop firewalld && sudo systemctl disable
firewalld
# systemctl start ntpd.service
# systemctl enable ntpd.service
```

- For all CPUs in the system, set cpu scaling governor as 'performance'.

```
For e.g., for CPU 0,
# echo performance > /sys/devices/system/cpu/cpu0/
cpufreq/scaling_governor
```

- For each NVMe device, enable rq_affinity,

```
# echo 2 > /sys/block/nvme0/queue/rq_affinity
```

- For each NVMe device, enable read_ahead,

```
For radosbench tests, set read_ahead value to 128,
# echo 128 > /sys/block/nvme0/queue/read_ahead_kb
For librbdfio tests, set read_ahead to 0,
# echo 0 > /sys/block/nvme0/queue/read_ahead_kb
```

CBT tests

CBT framework provides radosbench and librbdfio that will be used to perform benchmark runs on the Ceph cluster. Make sure size of /tmp is adequate to hold the results of CBT tests. For each test like read, random-read, random-write, etc., the results may take up to 40 MB of drive space.

radosbench is used for sequential workloads for large IO sizes and librbdfio is used for random workloads for small IO sizes.

Appendix 10.1 presents ceph.conf file with the Ceph configuration tunables used in the configuration.

Appendix 10.2 presents CBT test.yaml file with the test configurations used in the test runs.

Summary

Samsung NVMe Reference Design provides a highly flexible high-performance platform that can be configured for serving both high IOPS and high throughput Ceph cluster deployments using Samsung NVMe SSDs. Combined with Ceph flexibility, a scalable Ceph cluster can be built to deliver an operational cluster that can deliver per node:

- Up to 11.52 TB of effective capacity accounting for 2 x replication
- Up to 231K 100% random read of 4 KB IOs
- Up to 29.2K 100% random write of 4 KB IOs accounting for OSD journal and 2x replication writes
- Up to 9.5 GB/s 100% sequential read of 1 MB IOs
- Up to 2.06 GB/s 100% sequential write of 1 MB IOs accounting for OSD journal and 2x replication writes

Red Hat Ceph Storage on Samsung NVMe SSDs

Appendix

Reference ceph. conf

```
[global]
        auth_cluster_required = none
        auth_service_required = none
        auth_client_required = none
        crushtool = /usr/local/bin/crushtool
        debug_lockdep = 0/1
        debug_context = 0/1
        debug_crush = 1/1
        debug_buffer = 0/1
        debug_timer = 0/0
        debug_filer = 0/1
        debug_objecter = 0/1
        debug_rados = 0/5
        debug_rbd = 0/5
        debug_ms = 0/5
        debug_monc = 0/5
        debug_tp = 0/5
        debug_auth = 1/5
        debug_finisher = 1/5
        debug_heartbeatmap = 1/5
        debug_perfcounter = 1/5
        debug_rgw = 1/5
        debug_asok = 1/5
        debug_throttle = 1/1

        debug_journaler = 0/0
        debug_objectcatcher = 0/0
        debug_client = 0/0
        debug_osd = 0/0
        debug_optracker = 0/0
        debug_objclass = 0/0
        debug_filestore = 0/0
        debug_journal = 0/0
        debug_mon = 0/0
        debug_paxos = 0/0

        osd_crush_chooseleaf_type = 0
        filestore_xattr_use_omap = true
        osd_pool_default_size = 1
        osd_pool_default_min_size = 1

        mon_pg_warn_max_object_skew = 10000
        mon_pg_warn_min_per_osd = 0
        mon_pg_warn_max_per_osd = 32768
        rbd_cache = true
        mon_compact_on_trim = false
        log_to_syslog = false
        log_file = /var/log/ceph/$name.log
perf = true
        mutex_perf_counter = true
        throttler_perf_counter = false
        ms_nocrc = true

[mon]
        mon data = /tmp/cbt/ceph/mon.$id
        mon_max_pool_pg_num = 166496
        mon_osd_max_split_count = 10000

[mon.a]
        host = mon-host-1
        mon_addr = 40.10.10.150:6789

[client]
        rbd_cache = true
        rbd_cache_writethrough_until_flush = false
        admin_socket = /var/run/ceph/$cluster-
        $type.$id.$pid.$cctid.asok
        log_file = /var/log/ceph/


[osd]
        filestore_wbthrottle_enable = false
        filestore_queue_max_bytes = 1048576000
        filestore_queue_committing_max_bytes =
        1048576000
        filestore_queue_max_ops = 5000
        filestore_queue_committing_max_ops = 5000
        filestore_max_sync_interval = 10
        filestore_fd_cache_size = 64
        filestore_fd_cache_shards = 32
        filestore_op_threads = 6

        filestore flusher = false
        osd_crush_update_on_start = false
        osd_max_backfills = 1
        osd_recovery_priority = 1
        osd_client_op_priority = 63
        osd_recovery_max_active = 1
        osd_recovery_max_start = 1

        osd_mount_options_xfs = "rw,noatime,inode64,
        logbsize=256k,delaylog"
```

```
        osd_mkfs_options_xfs = "-f -i size=2048"

        journal_max_write_entries = 2000
        journal_queue_max_ops = 3000
        journal_max_write_bytes = 4194304000
        journal_queue_max_bytes = 4194304000

        osd_enable_op_tracker = false

        osd_client_message_size_cap = 0
        osd_client_message_cap = 0
objecter_inflight_ops = 102400
        objecter_inflight_op_bytes = 1048576000

        ms_dispatch_throttle_bytes = 1048576000

        osd_op_threads = 32
        osd_op_num_shards = 5
        osd_op_num_threads_per_shard = 2

[osd.0]
        host = osd-host-1
        public_addr = 42.10.10.111
        cluster_addr = 42.10.10.111
        osd_data = /tmp/cbt/mnt/osd-device-0-data
        osd_journal = /dev/disk/by-partlabel/osd-
device-0-journal
        ms_bind_port_min = 7000
        ms_bind_port_max = 8000

[osd.1]
        host = osd-host-1
        public_addr = 42.10.10.111
        cluster_addr = 42.10.10.111
        osd_data = /tmp/cbt/mnt/osd-device-1-data
        osd_journal = /dev/disk/by-partlabel/osd-
device-1-journal
        ms_bind_port_min = 8001
        ms_bind_port_max = 9000

[osd.2]
        host = osd-host-1
        public_addr = 42.10.10.111
        cluster_addr = 42.10.10.111
        osd_data = /tmp/cbt/mnt/osd-device-2-data
        osd_journal = /dev/disk/by-partlabel/osd-
        device-2-journal
        ms_bind_port_min = 9001
        ms_bind_port_max = 10000

[osd.3]
        host = osd-host-1
        public_addr = 42.10.10.111
        cluster_addr = 42.10.10.111
        osd_data = /tmp/cbt/mnt/osd-device-3-data
        osd_journal = /dev/disk/by-partlabel/osd-
        device-3-journal
        ms_bind_port_min = 10001
        ms_bind_port_max = 11000

[osd.4]
        host = osd-host-1
        public_addr = 42.10.10.111
        cluster_addr = 42.10.10.111
        osd_data = /tmp/cbt/mnt/osd-device-4-data
        osd_journal = /dev/disk/by-partlabel/osd-
device-4-journal
        ms_bind_port_min = 11001
        ms_bind_port_max = 12000

[osd.5]
        host = osd-host-1
        public_addr = 42.10.10.111
        cluster_addr = 42.10.10.111
        osd_data = /tmp/cbt/mnt/osd-device-5-data
        osd_journal = /dev/disk/by-partlabel/osd-
device-5-journal
        ms_bind_port_min = 12001
        ms_bind_port_max = 13000

[osd.6]
        host = osd-host-1
        public_addr = 43.10.10.111
        cluster_addr = 43.10.10.111
        osd_data = /tmp/cbt/mnt/osd-device-6-data
        osd_journal = /dev/disk/by-partlabel/osd-
device-6-journal
        ms_bind_port_min = 13001
        ms_bind_port_max = 14000

[osd.7]
        host = osd-host-1
        public_addr = 43.10.10.111
        cluster_addr = 43.10.10.111
```

Red Hat Ceph Storage on Samsung NVMe SSDs

```
        osd_data = /tmp/cbt/mnt/osd-device-7-data
        osd_journal = /dev/disk/by-partlabel/osd-
device-7-journal
        ms_bind_port_min = 14001
        ms_bind_port_max = 15000

[osd.8]
        host = osd-host-1
        public_addr = 43.10.10.111
        cluster_addr = 43.10.10.111
        osd_data = /tmp/cbt/mnt/osd-device-8-data
        osd_journal = /dev/disk/by-partlabel/osd-
device-8-journal
        ms_bind_port_min = 15001
        ms_bind_port_max = 16000

[osd.9]
        host = osd-host-1
        public_addr = 43.10.10.111
        cluster_addr = 43.10.10.111
        osd_data = /tmp/cbt/mnt/osd-device-9-data
        osd_journal = /dev/disk/by-partlabel/osd-
device-9-journal
        ms_bind_port_min = 16001
        ms_bind_port_max = 17000

[osd.10]
        host = osd-host-1
        public_addr = 43.10.10.111
        cluster_addr = 43.10.10.111
        osd_data = /tmp/cbt/mnt/osd-device-10-data
        osd_journal = /dev/disk/by-partlabel/osd-
device-10-journal
        ms_bind_port_min = 17001
        ms_bind_port_max = 18000
[osd.11]
        host = osd-host-1
        public_addr = 43.10.10.111
        cluster_addr = 43.10.10.111
        osd_data = /tmp/cbt/mnt/osd-device-11-data
        osd_journal = /dev/disk/by-partlabel/osd-
device-11-journal
        ms_bind_port_min = 18001
        ms_bind_port_max = 19000

[osd.12]
        host = osd-host-1
        public_addr = 40.10.10.111
        cluster_addr = 40.10.10.111
        osd_data = /tmp/cbt/mnt/osd-device-12-data
        osd_journal = /dev/disk/by-partlabel/osd-
device-12-journal
        ms_bind_port_min = 19001
        ms_bind_port_max = 20000

[osd.13]
        host = osd-host-1
        public_addr = 40.10.10.111
        cluster_addr = 40.10.10.111
        osd_data = /tmp/cbt/mnt/osd-device-13-data
        osd_journal = /dev/disk/by-partlabel/osd-
device-13-journal
        ms_bind_port_min = 20001
        ms_bind_port_max = 21000

[osd.14]
        host = osd-host-1
        public_addr = 40.10.10.111
        cluster_addr = 40.10.10.111
        osd_data = /tmp/cbt/mnt/osd-device-14-data
        osd_journal = /dev/disk/by-partlabel/osd-
device-14-journal
        ms_bind_port_min = 21001
        ms_bind_port_max = 22000

[osd.15]
        host = osd-host-1
        public_addr = 40.10.10.111
        cluster_addr = 40.10.10.111
        osd_data = /tmp/cbt/mnt/osd-device-15-data
        osd_journal = /dev/disk/by-partlabel/osd-
device-15-journal
        ms_bind_port_min = 22001
        ms_bind_port_max = 23000

[osd.16]
        host = osd-host-1
        public_addr = 40.10.10.111
        cluster_addr = 40.10.10.111
        osd_data = /tmp/cbt/mnt/osd-device-16-data
        osd_journal = /dev/disk/by-partlabel/osd-
device-16-journal
        ms_bind_port_min = 23001
        ms_bind_port_max = 24000
[osd.17]
        host = osd-host-1
```

```
        public_addr = 40.10.10.111
        cluster_addr = 40.10.10.111
        osd_data = /tmp/cbt/mnt/osd-device-17-data
        osd_journal = /dev/disk/by-partlabel/osd-
device-17-journal
        ms_bind_port_min = 24001
        ms_bind_port_max = 25000

[osd.18]
        host = osd-host-1
        public_addr = 41.10.10.111
        cluster_addr = 41.10.10.111
        osd_data = /tmp/cbt/mnt/osd-device-18-data
        osd_journal = /dev/disk/by-partlabel/osd-
device-18-journal
        ms_bind_port_min = 25001
        ms_bind_port_max = 26000

[osd.19]
        host = osd-host-1
        public_addr = 41.10.10.111
        cluster_addr = 41.10.10.111
        osd_data = /tmp/cbt/mnt/osd-device-19-data
        osd_journal = /dev/disk/by-partlabel/osd-
device-19-journal
        ms_bind_port_min = 26001
        ms_bind_port_max = 27000

[osd.20]
        host = osd-host-1
        public_addr = 41.10.10.111
        cluster_addr = 41.10.10.111
        osd_data = /tmp/cbt/mnt/osd-device-20-data
        osd_journal = /dev/disk/by-partlabel/osd-
device-20-journal
        ms_bind_port_min = 27001
        ms_bind_port_max = 28000

[osd.21]
        host = osd-host-1
        public_addr = 41.10.10.111
        cluster_addr = 41.10.10.111
        osd_data = /tmp/cbt/mnt/osd-device-21-data
        osd_journal = /dev/disk/by-partlabel/osd-
device-21-journal
        ms_bind_port_min = 28001
        ms_bind_port_max = 29000

[osd.22]
        host = osd-host-1
        public_addr = 41.10.10.111
        cluster_addr = 41.10.10.111
        osd_data = /tmp/cbt/mnt/osd-device-22-data
        osd_journal = /dev/disk/by-partlabel/osd-
device-22-journal
        ms_bind_port_min = 29001
        ms_bind_port_max = 30000

[osd.23]
        host = osd-host-1
        public_addr = 41.10.10.111
        cluster_addr = 41.10.10.111
        osd_data = /tmp/cbt/mnt/osd-device-23-data
        osd_journal = /dev/disk/by-partlabel/osd-
device-23-journal
        ms_bind_port_min = 30001
        ms_bind_port_max = 31000

[osd.24]
        host = osd-host-2
        public_addr = 42.10.10.112
        cluster_addr = 42.10.10.112
        osd_data = /tmp/cbt/mnt/osd-device-0-data
        osd_journal = /dev/disk/by-partlabel/osd-
device-0-journal
        ms_bind_port_min = 7001
        ms_bind_port_max = 8000

[osd.25]
        host = osd-host-2
        public_addr = 42.10.10.112
        cluster_addr = 42.10.10.112
        osd_data = /tmp/cbt/mnt/osd-device-1-data
        osd_journal = /dev/disk/by-partlabel/osd-
device-1-journal
        ms_bind_port_min = 8001
        ms_bind_port_max = 9000

[osd.26]
        host = osd-host-2
        public_addr = 42.10.10.112
        cluster_addr = 42.10.10.112
        osd_data = /tmp/cbt/mnt/osd-device-2-data
        osd_journal = /dev/disk/by-partlabel/osd-
device-2-journal
        ms_bind_port_min = 9001
        ms_bind_port_max = 10000
```

```
[osd.27]
        host = osd-host-2
        public_addr = 42.10.10.112
        cluster_addr = 42.10.10.112
        osd_data = /tmp/cbt/mnt/osd-device-3-data
        osd_journal = /dev/disk/by-partlabel/osd-
device-3-journal
        ms_bind_port_min = 10001
        ms_bind_port_max = 11000

[osd.28]
        host = osd-host-2
        public_addr = 42.10.10.112
        cluster_addr = 42.10.10.112
        osd_data = /tmp/cbt/mnt/osd-device-4-data
        osd_journal = /dev/disk/by-partlabel/osd-
device-4-journal
        ms_bind_port_min = 11001
        ms_bind_port_max = 12000

[osd.29]
        host = osd-host-2
        public_addr = 42.10.10.112
        cluster_addr = 42.10.10.112
        osd_data = /tmp/cbt/mnt/osd-device-5-data
        osd_journal = /dev/disk/by-partlabel/osd-
device-5-journal
        ms_bind_port_min = 12001
        ms_bind_port_max = 13000

[osd.30]
        host = osd-host-2
        public_addr = 43.10.10.112
        cluster_addr = 43.10.10.112
        osd_data = /tmp/cbt/mnt/osd-device-6-data
        osd_journal = /dev/disk/by-partlabel/osd-
device-6-journal
        ms_bind_port_min = 13001
        ms_bind_port_max = 14000

[osd.31]
        host = osd-host-2
        public_addr = 43.10.10.112
        cluster_addr = 43.10.10.112
        osd_data = /tmp/cbt/mnt/osd-device-7-data
        osd_journal = /dev/disk/by-partlabel/osd-
device-7-journal
        ms_bind_port_min = 14001
        ms_bind_port_max = 15000

[osd.32]
        host = osd-host-2
        public_addr = 43.10.10.112
        cluster_addr = 43.10.10.112
        osd_data = /tmp/cbt/mnt/osd-device-8-data
        osd_journal = /dev/disk/by-partlabel/osd-
device-8-journal
        ms_bind_port_min = 15001
        ms_bind_port_max = 16000

[osd.33]
        host = osd-host-2
        public_addr = 43.10.10.112
        cluster_addr = 43.10.10.112
        osd_data = /tmp/cbt/mnt/osd-device-9-data
        osd_journal = /dev/disk/by-partlabel/osd-
device-9-journal
        ms_bind_port_min = 16001
        ms_bind_port_max = 17000

[osd.34]
        host = osd-host-2
        public_addr = 43.10.10.112
        cluster_addr = 43.10.10.112
        osd_data = /tmp/cbt/mnt/osd-device-10-data
        osd_journal = /dev/disk/by-partlabel/osd-
device-10-journal
        ms_bind_port_min = 17001
        ms_bind_port_max = 18000

[osd.35]
        host = osd-host-2
        public_addr = 43.10.10.112
        cluster_addr = 43.10.10.112
        osd_data = /tmp/cbt/mnt/osd-device-11-data
        osd_journal = /dev/disk/by-partlabel/osd-
device-11-journal
        ms_bind_port_min = 18001
        ms_bind_port_max = 19000

[osd.36]
        host = osd-host-2
        public_addr = 40.10.10.112
        cluster_addr = 40.10.10.112
        osd_data = /tmp/cbt/mnt/osd-device-12-data
        osd_journal = /dev/disk/by-partlabel/osd-
```

```
device-12-journal
        ms_bind_port_min = 19001
        ms_bind_port_max = 20000

[osd.37]
        host = osd-host-2
        public_addr = 40.10.10.112
        cluster_addr = 40.10.10.112
        osd_data = /tmp/cbt/mnt/osd-device-13-data
        osd_journal = /dev/disk/by-partlabel/osd-
device-13-journal
        ms_bind_port_min = 20001
        ms_bind_port_max = 21000

[osd.38]
        host = osd-host-2
        public_addr = 40.10.10.112
        cluster_addr = 40.10.10.112
        osd_data = /tmp/cbt/mnt/osd-device-14-data
        osd_journal = /dev/disk/by-partlabel/osd-
device-14-journal
        ms_bind_port_min = 21001
        ms_bind_port_max = 22000

[osd.39]
        host = osd-host-2
        public_addr = 40.10.10.112
        cluster_addr = 40.10.10.112
        osd_data = /tmp/cbt/mnt/osd-device-15-data
        osd_journal = /dev/disk/by-partlabel/osd-
device-15-journal
        ms_bind_port_min = 22001
        ms_bind_port_max = 23000

[osd.40]
        host = osd-host-2
        public_addr = 40.10.10.112
        cluster_addr = 40.10.10.112
        osd_data = /tmp/cbt/mnt/osd-device-16-data
        osd_journal = /dev/disk/by-partlabel/osd-
device-16-journal
        ms_bind_port_min = 23001
        ms_bind_port_max = 24000

[osd.41]
        host = osd-host-2
        public_addr = 40.10.10.112
        cluster_addr = 40.10.10.112
        osd_data = /tmp/cbt/mnt/osd-device-17-data
        osd_journal = /dev/disk/by-partlabel/osd-
device-17-journal
        ms_bind_port_min = 24001
        ms_bind_port_max = 25000

[osd.42]
        host = osd-host-2
        public_addr = 41.10.10.112
        cluster_addr = 41.10.10.112
        osd_data = /tmp/cbt/mnt/osd-device-18-data
        osd_journal = /dev/disk/by-partlabel/osd-
device-18-journal
        ms_bind_port_min = 25001
        ms_bind_port_max = 26000

[osd.43]
        host = osd-host-2
        public_addr = 41.10.10.112
        cluster_addr = 41.10.10.112
        osd_data = /tmp/cbt/mnt/osd-device-19-data
        osd_journal = /dev/disk/by-partlabel/osd-
device-19-journal
        ms_bind_port_min = 26001
        ms_bind_port_max = 27000

[osd.44]
        host = osd-host-2
        public_addr = 41.10.10.112
        cluster_addr = 41.10.10.112
        osd_data = /tmp/cbt/mnt/osd-device-20-data
        osd_journal = /dev/disk/by-partlabel/osd-
device-20-journal
        ms_bind_port_min = 27001
        ms_bind_port_max = 28000

[osd.45]
        host = osd-host-2
        public_addr = 41.10.10.112
        cluster_addr = 41.10.10.112
        osd_data = /tmp/cbt/mnt/osd-device-21-data
        osd_journal = /dev/disk/by-partlabel/osd-
device-21-journal
        ms_bind_port_min = 28001
        ms_bind_port_max = 29000

[osd.46]
        host = osd-host-2
```

```
        public_addr = 41.10.10.112
        cluster_addr = 41.10.10.112
        osd_data = /tmp/cbt/mnt/osd-device-22-data
        osd_journal = /dev/disk/by-partlabel/osd-
device-22-journal
        ms_bind_port_min = 29001
        ms_bind_port_max = 30000

[osd.47]
        host = osd-host-2
        public_addr = 41.10.10.112
        cluster_addr = 41.10.10.112
        osd_data = /tmp/cbt/mnt/osd-device-23-data
        osd_journal = /dev/disk/by-partlabel/osd-
device-23-journal
        ms_bind_port_min = 30001
        ms_bind_port_max = 31000

[osd.48]
        host = osd-host-3
        public_addr = 42.10.10.115
        cluster_addr = 42.10.10.115
        osd_data = /tmp/cbt/mnt/osd-device-0-data
        osd_journal = /dev/disk/by-partlabel/osd-
device-0-journal
        ms_bind_port_min = 7001
        ms_bind_port_max = 8000

[osd.49]
        host = osd-host-3
        public_addr = 42.10.10.115
        cluster_addr = 42.10.10.115
        osd_data = /tmp/cbt/mnt/osd-device-1-data
        osd_journal = /dev/disk/by-partlabel/osd-
device-1-journal
        ms_bind_port_min = 8001
        ms_bind_port_max = 9000

[osd.50]
        host = osd-host-3
        public_addr = 42.10.10.115
        cluster_addr = 42.10.10.115
        osd_data = /tmp/cbt/mnt/osd-device-2-data
        osd_journal = /dev/disk/by-partlabel/osd-
device-2-journal
        ms_bind_port_min = 9001
        ms_bind_port_max = 10000

[osd.51]
        host = osd-host-3
        public_addr = 42.10.10.115
        cluster_addr = 42.10.10.115
        osd_data = /tmp/cbt/mnt/osd-device-3-data
        osd_journal = /dev/disk/by-partlabel/osd-
device-3-journal
        ms_bind_port_min = 10001
        ms_bind_port_max = 11000

[osd.52]
        host = osd-host-3
        public_addr = 42.10.10.115
        cluster_addr = 42.10.10.115
        osd_data = /tmp/cbt/mnt/osd-device-4-data
        osd_journal = /dev/disk/by-partlabel/osd-
device-4-journal
        ms_bind_port_min = 11001
        ms_bind_port_max = 12000

[osd.53]
        host = osd-host-3
        public_addr = 42.10.10.115
        cluster_addr = 42.10.10.115
        osd_data = /tmp/cbt/mnt/osd-device-5-data
        osd_journal = /dev/disk/by-partlabel/osd-
device-5-journal
        ms_bind_port_min = 12001
        ms_bind_port_max = 13000

[osd.54]
        host = osd-host-3
        public_addr = 43.10.10.115
        cluster_addr = 43.10.10.115
        osd_data = /tmp/cbt/mnt/osd-device-6-data
        osd_journal = /dev/disk/by-partlabel/osd-
device-6-journal
        ms_bind_port_min = 13001
        ms_bind_port_max = 14000

[osd.55]
        host = osd-host-3
        public_addr = 43.10.10.115
        cluster_addr = 43.10.10.115
        osd_data = /tmp/cbt/mnt/osd-device-7-data
        osd_journal = /dev/disk/by-partlabel/osd-
device-7-journal
        ms_bind_port_min = 14001
```

```
        ms_bind_port_max = 15000

[osd.56]
        host = osd-host-3
        public_addr = 43.10.10.115
        cluster_addr = 43.10.10.115
        osd_data = /tmp/cbt/mnt/osd-device-8-data
        osd_journal = /dev/disk/by-partlabel/osd-
device-8-journal
        ms_bind_port_min = 15001
        ms_bind_port_max = 16000

[osd.57]
        host = osd-host-3
        public_addr = 43.10.10.115
        cluster_addr = 43.10.10.115
        osd_data = /tmp/cbt/mnt/osd-device-9-data
        osd_journal = /dev/disk/by-partlabel/osd-
device-9-journal
        ms_bind_port_min = 16001
        ms_bind_port_max = 17000

[osd.58]
        host = osd-host-3
        public_addr = 43.10.10.115
        cluster_addr = 43.10.10.115
        osd_data = /tmp/cbt/mnt/osd-device-10-data
        osd_journal = /dev/disk/by-partlabel/osd-
device-10-journal
        ms_bind_port_min = 17001
        ms_bind_port_max = 18000

[osd.59]
        host = osd-host-3
        public_addr = 43.10.10.115
        cluster_addr = 43.10.10.115
        osd_data = /tmp/cbt/mnt/osd-device-11-data
        osd_journal = /dev/disk/by-partlabel/osd-
device-11-journal
        ms_bind_port_min = 18001
        ms_bind_port_max = 19000

[osd.60]
        host = osd-host-3
        public_addr = 40.10.10.115
        cluster_addr = 40.10.10.115
        osd_data = /tmp/cbt/mnt/osd-device-12-data
        osd_journal = /dev/disk/by-partlabel/osd-
device-12-journal
        ms_bind_port_min = 19001
        ms_bind_port_max = 20000

[osd.61]
        host = osd-host-3
        public_addr = 40.10.10.115
        cluster_addr = 40.10.10.115
        osd_data = /tmp/cbt/mnt/osd-device-13-data
        osd_journal = /dev/disk/by-partlabel/osd-
device-13-journal
        ms_bind_port_min = 20001
        ms_bind_port_max = 21000

[osd.62]
        host = osd-host-3
        public_addr = 40.10.10.115
        cluster_addr = 40.10.10.115
        osd_data = /tmp/cbt/mnt/osd-device-14-data
        osd_journal = /dev/disk/by-partlabel/osd-
device-14-journal
        ms_bind_port_min = 21001
        ms_bind_port_max = 22000

[osd.63]
        host = osd-host-3
        public_addr = 40.10.10.115
        cluster_addr = 40.10.10.115
        osd_data = /tmp/cbt/mnt/osd-device-15-data
        osd_journal = /dev/disk/by-partlabel/osd-
device-15-journal
        ms_bind_port_min = 22001
        ms_bind_port_max = 23000

[osd.64]
        host = osd-host-3
        public_addr = 40.10.10.115
        cluster_addr = 40.10.10.115
        osd_data = /tmp/cbt/mnt/osd-device-16-data
        osd_journal = /dev/disk/by-partlabel/osd-
device-16-journal
        ms_bind_port_min = 23001
        ms_bind_port_max = 24000

[osd.65]
        host = osd-host-3
        public_addr = 40.10.10.115
        cluster_addr = 40.10.10.115
```

```
        osd_data = /tmp/cbt/mnt/osd-device-17-data
        osd_journal = /dev/disk/by-partlabel/osd-
device-17-journal
        ms_bind_port_min = 24001
        ms_bind_port_max = 25000

[osd.66]
        host = osd-host-3
        public_addr = 41.10.10.115
        cluster_addr = 41.10.10.115
        osd_data = /tmp/cbt/mnt/osd-device-18-data
        osd_journal = /dev/disk/by-partlabel/osd-
device-18-journal
        ms_bind_port_min = 25001
        ms_bind_port_max = 26000

[osd.67]
        host = osd-host-3
        public_addr = 41.10.10.115
        cluster_addr = 41.10.10.115
        osd_data = /tmp/cbt/mnt/osd-device-19-data
        osd_journal = /dev/disk/by-partlabel/osd-
device-19-journal
        ms_bind_port_min = 26001
        ms_bind_port_max = 27000

[osd.68]
        host = osd-host-3
        public_addr = 41.10.10.115
        cluster_addr = 41.10.10.115
        osd_data = /tmp/cbt/mnt/osd-device-20-data
        osd_journal = /dev/disk/by-partlabel/osd-
device-20-journal
        ms_bind_port_min = 27001
        ms_bind_port_max = 28000

[osd.69]
        host = osd-host-3
        public_addr = 41.10.10.115
        cluster_addr = 41.10.10.115
        osd_data = /tmp/cbt/mnt/osd-device-21-data
        osd_journal = /dev/disk/by-partlabel/osd-
device-21-journal
        ms_bind_port_min = 28001
        ms_bind_port_max = 29000

[osd.70]
        host = osd-host-3
        public_addr = 41.10.10.115
        cluster_addr = 41.10.10.115
        osd_data = /tmp/cbt/mnt/osd-device-22-data
        osd_journal = /dev/disk/by-partlabel/osd-
device-22-journal
        ms_bind_port_min = 29001
        ms_bind_port_max = 30000

[osd.71]
        host = osd-host-3
        public_addr = 41.10.10.115
        cluster_addr = 41.10.10.115
        osd_data = /tmp/cbt/mnt/osd-device-23-data
        osd_journal = /dev/disk/by-partlabel/osd-
device-23-journal
        ms_bind_port_min = 30001
        ms_bind_port_max = 31000
```

Sample CBT `test.yaml` file

```
cluster:
  user: 'root'
  head: "cbt-host-1"
  clients: ["cbt-client-1","cbt-client-2","cbt-
client-3","cbt-client-4","cbt-client-5","cbt-
client-6","cbt-client-7","cbt-client-8","cbt-
client-9"]
   osds: ["osd-host-1", "osd-host-2", "osd-host-3"]
  mons:
    mon-host-1:
      a: "40.10.10.150:6789"
  osds_per_node: 24
  fs: 'xfs'
  mkfs_opts: '-f -i size=2048 -n size=64k'
  mount_opts: '-o inode64,noatime,nodiratime,logbufs
=8,logbsize=256k'
  conf_file: '/usr/local/ceph-cbt/ceph.conf'
  iterations: 1
  use_existing: False
  clusterid: "ceph"
  tmp_dir: "/tmp/cbt"
  pool_profiles:
    rbd:
      pg_size: 16384
      pgp_size: 16384
      replication: 2
    radosbench:
      pg_size: 8192
      pgp_size: 8192
```

Red Hat Ceph Storage on Samsung NVMe SSDs

```
      replication: 2
benchmarks:
  librbdfio:
    time: 300
    ramp: 100
    vol_size: 65536
    mode: ['randread', 'randwrite', 'randrw']
    rwmixread: [70]
    numjobs: 4
    use_existing_volumes: False
    procs_per_volume: [1]
    volumes_per_client: [10]
    op_size: [4096, 8192, 16384]
    concurrent_procs: [1]
    iodepth: [768]
    osd_ra: [9000]
    cmd_path: '/usr/local/bin/fio'
    log_avg_msec: 20000
    pool_profile: 'rbd'
  radosbench:
    op_size: [131072, 4194304, 1048576]
    write_only: False
    time: 300
    pool_per_proc: True
    concurrent_procs: 4
    concurrent_ops: [ 32, 64, 128 ]
    osd_ra: [1108]
    pool_profile: 'radosbench'
```

Red Hat Ceph Storage on Samsung NVMe SSDs

Bill of materials

Table 12, Table 13, and Table 14 present the hardware configurations used in this Ceph Reference Architecture.

Samsung NVMe SSD PM953			
Model	PM953	**Interface**	PCIe Gen3 x 4
Form factor	2.5 inch	**Capacity**	Up to 1.92 TB
Sequential read (128 KB)	Up to 1,000 MB/s	**Sequential write (128 KB)**	Up to 870 MB/s
Random read IOPS (4 KB)	Up to 240K IOPS	**Random write IOPS (4 KB)**	Up to 19K IOPS
DWPD	1.3 DWPD	**Production status**	Mass Production

Table 12: Samsung NVMe SSD PM953 960 GB

Samsung NVMe reference platform	
Component	**Configuration details**
2x Intel® Xeon® CPU E5-2699 v3 @ 2.30 GHz	With 18 cores per socket, this will have 36 cores per system. With HT on, OS will list 72 hardware threads.
256 GB	To avoid memory limitations in using high-performance SSDs and NICs.
Up to 24x PM953	Each drive with 960 GB raw capacity.
Mellanox ConnectX®-4 EN adapters	1 x to 4 x 40 GbE ports; to avoid any network bottlenecks in delivering high IOPS in a cluster with replication or recovery traffic.

Table 13: Samsung NVMe Reference Platform (OSD node) configuration

Client and monitor nodes	
Component	**Configuration details**
2 x Intel® Xeon® CPU E5-2670 v3 @ 2.30 GHz	With 12 cores per socket, this will have 24 cores per system. With HT on, OS will list 48 hardware threads.
128 GB	To avoid memory limitations in using high-performance NICs.
Mellanox ConnectX®-3 EN adapters	2 x 40 GbE port

Table 14: Generic dual-socket x86 Server (client and monitor nodes) configuration

Legal and additional information

About Samsung Electronics Co., Ltd.

Samsung Electronics Co., Ltd. inspires the world and shapes the future with transformative ideas and technologies. The company is redefining the worlds of TVs, smartphones, wearable devices, tablets, cameras, digital appliances, printers, medical equipment, network systems, and semiconductor and LED solutions. For the latest news, please visit the Samsung Newsroom at news.samsung.com.

About Red Hat

Red Hat is the world's leading provider of open source software solutions, using a communitypowered approach to provide reliable and high-performing cloud, Linux, middleware, storage, and virtualization technologies. Red Hat also offers award-winning support, training, and consulting services. As a connective hub in a global network of enterprises, partners, and open source communities, Red Hat helps create relevant, innovative technologies that liberate resources for growth and prepare customers for the future of IT.

For more information

For more information about the Samsung NVMe Reference Design, please visit at samsung.com/semiconductor/afard/.

Samsung Electronics Co., Ltd.
129 Samsung-ro,
Yeongtong-gu,
Suwon-si, Gyeonggi-do 16677,
Korea

www.samsung.com

2016-10