REFERENCE ARCHITECTURE

Service Providers
Data Center

Build a High-Performance Object Storage-as-a-Service Platform with Minio*

Storage-as-a-service (STaaS) based on Minio* with Intel® technology simplifies object storage while providing high performance, scalability, and enhanced security features

What You'll Find in This Solution Reference Architecture: This solution provides a starting point for developing a storage-as-a-service (STaaS) platform based on Minio*.

If you are responsible for:

- **Investment decisions and business strategy:** You'll learn how Minio-based STaaS can help solve the pressing storage challenges facing cloud service providers (CSPs) today.
- **Figuring out how to implement STaaS and Minio:** You'll learn about the architecture components and how they work together to create a cohesive business solution.

Executive Summary

Emerging cloud service providers (CSPs) have an opportunity to build or expand their storage-as-a-service (STaaS) capabilities and tap into one of today's fastest-growing markets. However, CSPs who support this market face a substantial challenge: how to cost effectively store an exponentially growing amount of data while exposing the data as a service with high performance, scalability and security.

File and block protocols are complex, have legacy architectures that hold back innovation, and are limited in their ability to scale. Object storage, which was born in the cloud, solves these issues with a reduced set of storage APIs that are accessed over HTTP RESTful services. Hyperscalers have looked into many options when building the foundation of their cloud storage infrastructure and they all have adopted object storage as their primary storage service.

In this paper, we take a deeper look into an Amazon S3*-compatible object storage service architecture—a STaaS platform based on Minio* Object Storage and optimized for Intel® technology. An object storage solution should handle a broad spectrum of use cases including archival, application data, big data and machine learning. Unlike other object storage solutions that are built for archival-only use cases, the Minio platform is built for CSPs to deliver a high-performance cloud storage alternative to the hyperscalers. The Minio platform offers several benefits:

- **Hyperscale** architecture that enables multi-data center expansion through federation
- **High performance** object store to serve the most demanding cloud-native workloads
- **Ease of use** with non-disruptive upgrades, no tuning knobs and simple support requirements
- **High availability** so objects continue to be available despite multiple disk and node failures
- **Security-enabled** storage by encrypting each object with a unique key

Minio provides a compelling STaaS object storage platform when combined with Intel's broad selection of products and technologies, such as Intel® Solid State Drive Data Center Family for NVM Express* (NVMe*), Intel® Ethernet products and Intel® Xeon® Scalable processors, augmented by Intel® Advanced Vector Extensions

Table of Contents

512 (Intel® AVX-512) single instruction multiple data (SIMD) instructions for x86 architecture. A collaborative open source developer community is also available for Minio.

Using the information and reference architecture (see Figure 1) presented here, you can take the next step toward unleashing the power of STaaS.

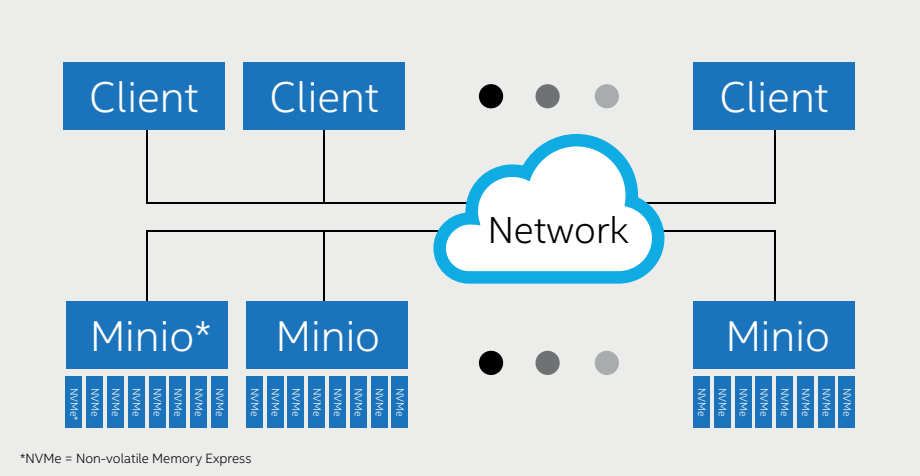


Figure 1. Our tests were run on an eight-node Minio* cluster.

High-Performance Object Storage

STaaS is the second-fastest growing cloud workload worldwide, representing a USD 4.8 billion annual market[1]. And yet a mere handful of CSPs control the majority of that market. With data growing exponentially every year—by 2025, experts predict that the world will create and replicate 163 zettabytes (ZB) of data[2]—there is tremendous potential for emerging CSPs to benefit from the 30-percent annual growth of STaaS[3].

Fueling the growth is an increasing focus on big data applications, Internet of Things (IoT) and artificial intelligence (AI) workloads. Object storage is the primary medium for storing big data because it is designed to provide high rates of throughput, offers excellent data integrity, and has a cost-effective deployment model. Although file and block storage solutions are being shoehorned into use with big data workloads, these solutions are limited in their ability to provide what is required. They were designed for enterprise applications like databases and file shares. High-performance object storage has a completely different design goal, which is to provide the extreme rates of throughput required by big data workloads, along with namespaces that span data centers. Neither file nor block storage solutions can perform as fast as, nor scale as large as, object storage.

Minio is an object storage solution that provides performance and scalability without suffering from the compromises of file and block storage. By following the methods and design philosophy of hyperscale computing providers, Minio is an object storage solution that provides both high performance and massive scalability. With Minio, which is optimized for Intel® architecture, you can build a fast, reliable STaaS platform with the scalability and flexibility you need to thrive in a data-centric world.

Solution Architecture

Minio consists of a server, an optional client, and an optional software development kit (SDK):

- **Minio Server.** Minio is a distributed object storage server released under Apache* License v2.0. It is compatible with the Amazon S3 API. Minio is feature-complete, providing enterprise-grade encryption, identity management, access control and data protection capabilities, including erasure code and bitrot protection.
- **Minio Client.** Called mc, the Minio Client is a modern and cloud-native alternative to the familiar UNIX* commands like `ls, cat, cp mirror, diff, find` and `mv`. This client provides advanced functionality that is suitable for web-scale object storage deployments. For example, powerful object mirroring tools that synchronize objects between multiple sites and tools for generating shared, time bound links for objects.
- **Minio SDKs.** The Minio Client SDKs provide simple APIs to access any Amazon S3-compatible object storage. Minio repositories on Github offer SDKs for popular development languages such as Golang*, JavaScript*, .Net*, Python* and Java*.

Minio* Overview and Benefits

Minio is unique in that it was built from the ground up to be simple, fast and highly scalable. With the belief that a complex solution cannot be scalable, a minimalist design philosophy forms the foundation of the Minio architecture design.

Minio is designed to deliver multiple benefits to object storage:

- **Performance.** With its focus on high performance, Minio enables enterprises to support multiple use cases with the same simple, scalable platform. Minio performance means you can run multiple Spark*, Presto* and Hive* queries, or quickly test, train and deploy AI algorithms, and not encounter a storage bottleneck. Minio object storage is used as the primary storage for cloud-native applications that require higher throughput and lower latency than traditional object storage can provide. Efficient code underlies Minio's fast performance. By using Intel® Xeon® Scalable processors in combination with 40 GbE Intel® Ethernet Network Adapters network interface cards (NICs) and Intel SSD Data Center Family for NVMe, Minio delivers highly performant object storage (see "Latest Intel Technology Drives Performance Advantage for Minio" for performance details.)
- **Scalability.** A design philosophy that "simple things scale" means that scaling starts with a single cluster that can be federated with other Minio clusters to create a global namespace—one that can span multiple data centers. Gradual expansion of the namespace is possible by adding more clusters, racks and/or data centers. Minio takes advantage of the hard-won knowledge of the hyperscalers to bring a simple scaling model to object storage.
- **Ease of use.** Minio can be installed and configured within minutes simply by downloading a single binary and then executing it. The amount of configuration options and variations has been kept to a minimum, which results in near-zero system administration tasks and few paths to failures. Upgrading Minio is done with a single command, which is non-disruptive and incurs zero downtime.
- **Encryption and WORM.** For production applications, security is critical. Minio provides per-object encryption using a unique key-per-object protection of data at rest. Minio supports external key management servers to securely manage encryption keys outside of the storage system. WORM (write once, read many) mode prevents tampering with data once written.
- **Identity and access management.** Minio provides Amazon Identity and Access Management* (IAM*)-compatible identity and access management. Minio plugs into industry-standard OpenID*-compatible identity management servers. Minio also provides temporary rotating credentials that help secure data access by eliminating the need to embed long-term credentials within an application.
- **High availability.** With Minio's high-availability design, a server can lose up to half of its drives and a cluster can lose up to half of its servers and Minio will continue to serve objects. It can also withstand a total rack failure when the cluster spans across racks. This is achieved by Minio's distributed erasure code that protects data with multiple redundant parity blocks. Minio also supports continuous mirroring to remote sites for disaster recovery purposes.

- **Metadata architecture.** Minio has no separate metadata store. All operations are performed atomically at object-level granularity. This approach isolates any failures, containing them within an object, and prevents spillover to larger system failures. Each object is strongly protected with erasure code and bitrot hash. You can crash a cluster in the middle of a busy workload and still not lose any data. Another advantage of this design is strict consistency, which is important for distributed machine-learning and big-data workloads.
- **Geographic namespace.** Multi-data center scaling is no longer limited to hyperscalers. Minio enables enterprises to adopt a scaling model that starts small and keeps expanding a cluster across racks and data centers around the globe, using Minio's federation feature. Minio is deployed in units of scale where the size is limited by the failure domain.
- **Cloud-native design.** The multi-instance, multi-tenant design of Minio enables Kubernetes*-like orchestration platforms to seamlessly manage storage resources just like compute resources. Each instance of Minio is provisioned on demand through self-service registration. Traditional storage systems are monolithic and compete with Kubernetes resource management. Minio is lightweight and container-friendly so you can pack many tenants simultaneously on the same shared infrastructure.
- **Lambda* function support.** Minio supports Amazon-compatible Lambda event notifications, which enables applications to be notified of individual object actions such as access, creation and deletion. The events can be delivered using industry-standard messaging platforms like Kafka*, NATS*, AMQP*, MQTT*, Webhooks* or a database such as Elasticsearch*, Redis, Postgres* and MySQL*.

Minio Object Storage Architecture

Traditionally, storage architects built a multi-layer storage architecture with a durable block layer at the bottom, an abstract file system as a middle layer, and multiple API gateways implementing various protocols for file, block and object. The problem with this approach to storage is that it requires too many compromises. That is why public cloud architecture provides separate object, file and block storage.

Minio follows a fundamentally different architecture when compared to other storage systems. Because Minio is purpose-built to serve only objects, a single-layer architecture achieves all of the necessary functionality without compromise. The advantage of this design is an object server that is high-performance and lightweight.

Another distinct feature of Minio is its resilient design resulting from the elimination of the metadata datastore. Data and metadata are always written together and all operations, like erasure code, encryption and compression, are performed at object-level granularity. This metadata design has the advantage that, in case of damage to an object, the damage can be healed/corrected for the individual object.

Linear Scaling

The designers of the Minio platform believe that building large-scale architectures is best done by combining simple building blocks. They learned this approach by studying the hyperscalers and their methods of scaling. A single Minio cluster can be deployed with anywhere from four to 32 nodes and Minio federation allows many clusters to be joined into a single global namespace (see Figure 2).

There are multiple benefits to Minio's cluster and federation architecture:

- Each node is an equal member of a Minio cluster. There is no master node.
- Each node can serve requests for any object in the cluster, even concurrently.
- Each cluster uses a Distributed Locking Manager (DLM) to manage updates and deletes to objects.
- The performance of an individual cluster remains constant as you add more clusters to the federation.
- Failure domains are kept within the cluster. An issue with one cluster does not affect the entire federation.

When deploying a cluster, it is recommended that you use a programmable domain name service (DNS), such as coreDNS*, to route HTTP(S) requests to the appropriate cluster. Also, use a load balancer to balance the load across the servers in a cluster. Global configuration parameters can be stored and managed in etcd (an open-source distributed key-value store).

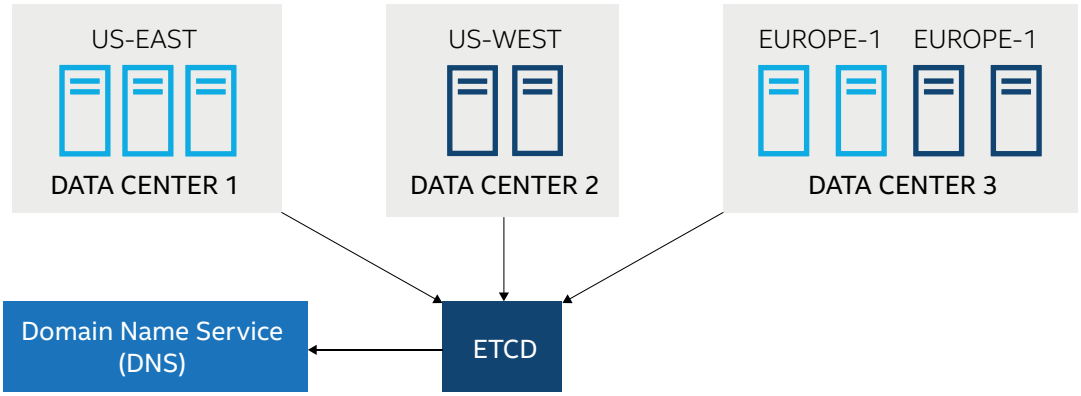


Figure 2. Minio* federation provides heterogeneous scalability and a planet-scale namespace with federation; the failure domain is limited to 32 servers maximum.

Erasure Code

Minio protects the integrity of object data with erasure coding and bitrot protection checksums (see Figure 3). Erasure code is a mathematical algorithm used to reconstruct missing or corrupted data. By applying a Reed-Solomon code to shard objects into data and parity blocks, and hashing algorithms to help protect individual shards, Minio is able to guard against hardware failures and silent data corruption.

Erasure code helps protect data without the high storage overhead of using RAID configurations or data replicas. For example, RAID-6 helps protect only against a two-drive failure whereas erasure code allows Minio to continue to serve data even with the loss of up to 50 percent of the drives and 50 percent of the servers. Minio applies erasure code to individual objects, which allows the healing of one object at a time. For RAID-protected storage solutions, healing is done at the RAID volume level, which impacts the performance of every file stored on the volume until the healing is completed.

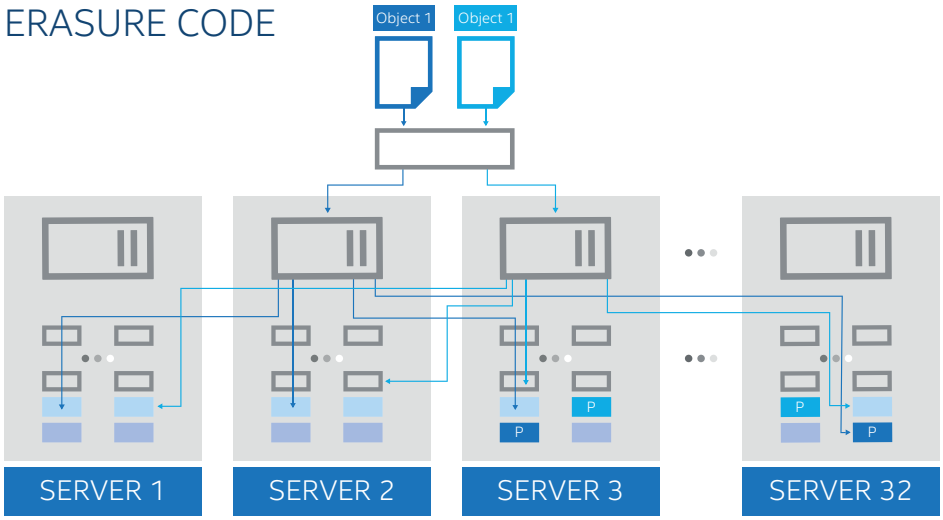


Figure 3. The Minio* erasure code protects data without the high storage overhead of using RAID configurations or data replicas.

Minio—The Perfect Fit for Open Source STaaS

Minio is distributed under the terms of the Apache License v2.0 and is actively developed on Github. The Minio development community starts with the Minio engineering team and includes all of the 4,000 members of Minio's Slack* Workspace. Since 2015, Minio has gathered 15K stars on Github, making it one of the top 15 Golang projects based on number of stars.

As an open source solution, Minio-based STaaS avoids vendor lock-in and is characterized by innovation that is driven by the open source community. For instance, the Minio engineering team frequently gets Pull Requests and/or ideas from the community to improve the Minio object server. All Pull Requests

are thoroughly reviewed by the Minio engineering team before deciding whether to commit the change into the master branch. The large size of the Minio community also helps the Minio engineering team to quickly detect source code errors and potential intrusion vulnerabilities.

Better Together: Intel® Technology Accelerates Minio Performance

Minio takes advantage of Intel technologies to create a high-performance, high-bandwidth and durable object storage solution. Here are examples of how Minio uses Intel® hardware:

- **Data durability and performance boost.** Minio's erasure coding and bitrot protection checksum algorithms are accelerated using single instruction multiple data (SIMD) instructions on Intel architecture using Intel® Advanced Vector Extensions 2 (Intel® AVX2) and Intel AVX-512. Offloading these calculations has a positive impact on overall system performance (see "Latest Intel Technology Drives Performance Advantage for Minio" for details).
- **Enhanced storage performance.** Intel SSD Data Center Family for NVMe and Intel® Optane™ SSDs provide performance, stability, efficiency and low power consumption. This reference architecture uses Intel SSD Data Center Family for NVMe to provide an extremely fast storage layer. In turn, Minio is able to translate the performance of the NVMe-based SSD into significantly fast object storage read and write throughput (see Figures 5 and 6).
- **Linear scaling with Intel® processors.** Testing shows that Minio servers scale performance linearly. The Intel Xeon Scalable processor family provides a wide range of performance options that deliver energy efficiency and high performance for intensive STaaS workloads, and lower performance options for less demanding STaaS workloads (see Figure 7).
- **Increased server bandwidth.** Intel Ethernet Network Adapters provide flexible bandwidth options. They are available with 10, 25 and 40 GbE ports to support whatever network infrastructure may be deployed in your data center. This reference architecture uses the 40 GbE Intel Ethernet Network Adapters to provide fast, low-latency networking between the clients and Minio, and between Minio servers.

Figure 4 provides an overview of Intel's broad portfolio of STaaS-enabling technologies.

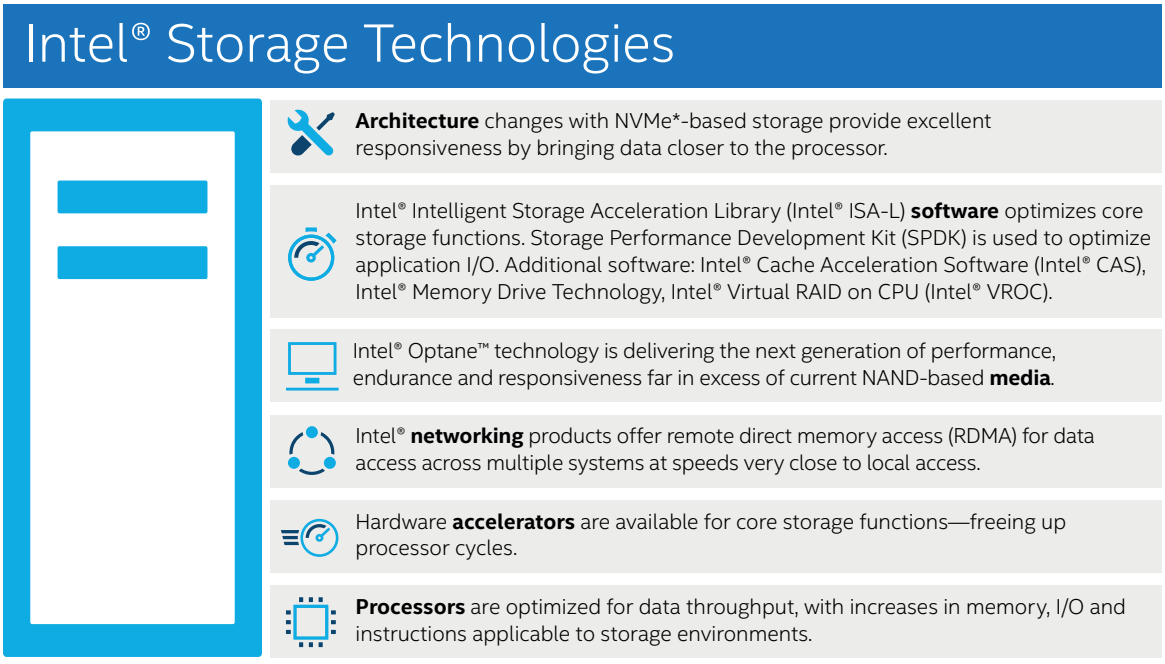


Figure 4. From hardware to software, Intel offers many technologies that can benefit storage-as-a-service (STaaS) solutions.

Latest Intel Technology Drives Performance Advantage for Minio

One of the benefits of SDS is the ability to rapidly integrate the latest advances in technology, which creates an advantage in the highly competitive CSP market. To demonstrate this, Intel evaluated the performance of an all-NVMe Minio cluster with storage nodes based on the following:

- Intel® Xeon® Gold 6240 processors
- Intel® SSD Data Center Family P4510 and P4610 Series PCIe* for data storage
- Intel® XL710-QDA2 Dual-Port NIC (40 GbE)
- Minio release.2019-02-12T21-58-47Z

Read and write benchmark testing was performed on an eight-node server configuration driven by eight clients. To confirm the linearity of the Minio cluster performance, the cluster throughput was tested with four, six and eight Minio servers. For test configuration details, see Table 3 later in this document.

Read and Write Performance Testing

We used the COSBench* benchmarking tool to test the read and write performance of an eight-node Minio cluster (see Table 1). The COSBench tool was run on eight clients against eight Minio servers that were configured as a single Minio cluster. Each client directed its tests against one Minio server in the cluster, thereby distributing the load equally across all servers. The COSBench tool ran with 256 threads per client for a total of 2,048 threads.

Each test had three stages which ran for a total of 90 minutes. The first 40-minute stage prepared the cluster by writing objects and data to the Minio cluster. This was followed by a 30-minute stage where objects were read from the cluster. The last stage was a cleanup stage. To avoid any memory caching by the servers, we used a data set size of 10 TB. This data set is much larger than the 192 GB of RAM on the Minio servers.

Table 1. Aggregated Read and Write Throughput

Object size	10 MB	20 MB	32 MB	64 MB
Read (GB/s)	19.6	19.5	19.4	19.6
Write (GB/s)	5.1	7.2	9.0	9.8

The read performance of the eight-node Minio cluster shows a sustained level of approximately 20 GB/second for all object sizes (see Figure 5). This performance is approximately half of the aggregated bandwidth of all eight servers[4]. Based on these results, it is clear that Minio is able to fully utilize all available bi-directional bandwidth on the single 40 GbE NIC port in the servers.

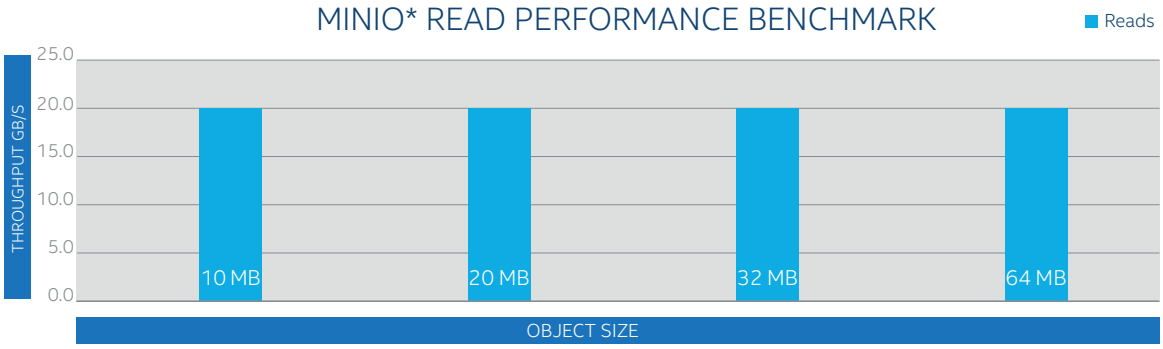


Figure 5. The Minio* cluster achieved almost 20 GB/s, regardless of object size (10 MB, 20 MB, 32 MB and 64 MB).

During object writes, the Minio server calculates erasure codes, parity blocks and bitrot hash, then distributes this data across multiple disks and across multiple servers. When the COSBench test was run with large object sizes of 32 MB and 64 MB, the write performance approached 10 GB/s (see Figure 6). With smaller object sizes, the performance achieved was 5 GB/sec (10 MB object) and 7 GB/sec (20 MB object).

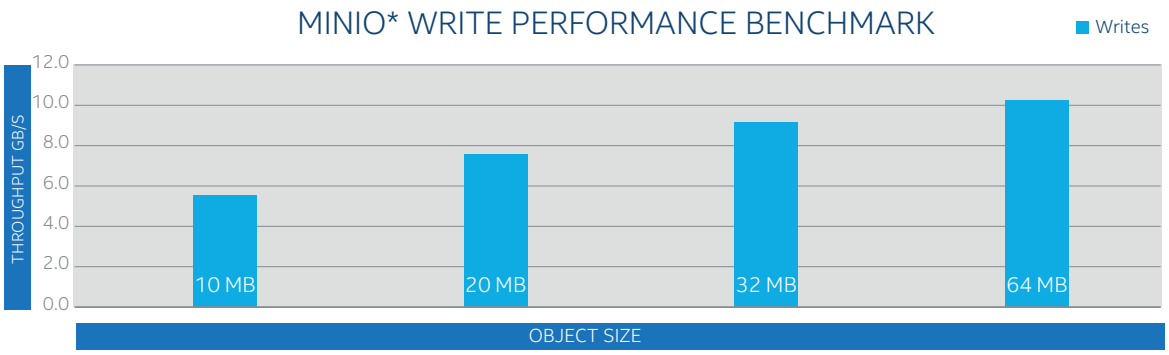


Figure 6. Write performance increased with larger objects.

Linear Scalability Testing

To demonstrate the linear scalability of the Minio clusters we ran COSBench performance tests with clusters of four, six and eight nodes. These tests used a fixed object size of 32 MB (see Table 2).

Table 2. Aggregated Read and Write Throughput for Different Cluster Sizes

Cluster size (nodes)	4	6	8
Read (GB/s)	9.0	13.6	19.4
Write (GB/s)	4.7	6.7	9.0

These test results demonstrate that Minio clusters increase read and write performance linearly as the number of servers in the Minio cluster is increased (see Figure 7). One reason for this linear scaling capability is a design that stores metadata with the object itself instead of using a metadata database server. Minio was purposely designed to avoid the bottleneck that a central metadata database can create. As a result, there is no need to consider the performance, or lack of performance, of a central metadata database when scaling Minio object storage.

Scalability beyond a single cluster is achieved by federating multiple clusters together to create a global namespace. Expansion of the federated namespace is possible simply by adding another Minio cluster, whether the servers are located in the same data center, or in another data center across the globe.

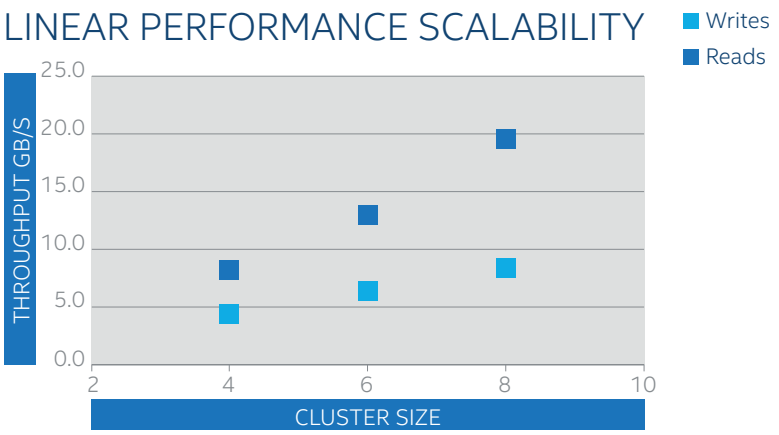


Figure 7. Minio* read and write performance increases linearly as the cluster size increases.

Architecture Design Considerations

This section presents general guidelines for Minio setup.

General Guidelines

A high-performance Minio object storage cluster is achieved only when the underlying components are capable of providing high performance. Using Intel Xeon processors, Intel SSD Data Center Family for NVMe and high-speed Intel® networking products makes delivering high performance possible. Before describing the specific reference architecture for Minio, here are several concepts common to Minio deployments:

- **Server architecture.** The foundation of a Minio cluster starts with a high-performance server architecture. Intel Xeon processor-based platforms combined with a high-bandwidth PCIe bus provides a strong foundation.
- **Memory.** 128 to 192 GB of RAM per server provides sufficient memory for Minio.
- **Disk.** Minio servers can take advantage of as many disk drives as can be deployed in a server. The choice of disk type depends on the type of Minio cluster you want to create. For the highest performance Minio cluster with the highest throughput, choose NVMe-based SSDs. In most cases, SSDs provide the perfect balance of performance and cost when compared to low-performing hard disk drives (HDDs).
- **Network.** The Minio servers communicate between servers and clients using Ethernet. For this reference architecture we used a single 40 GbE Intel Ethernet Network Adapter per server. However, Minio can take advantage of multiple NICs and network speeds up to 100 GbE. Some deployments use a separate NIC for internode communication. Note that the top-of-rack switch should be sized appropriately to support the maximum throughput speeds that you expect to achieve.
- **Failure domain.** Minio clusters can include up to 32 servers. This means it is possible to build a large and dense Minio cluster with tens of petabytes (PBs) of object storage in a single rack. However, we recommend starting with a cluster that keeps the failure domain small, and then scale by federating clusters into a global namespace. A moderately sized cluster is in the range of 8 to 16 nodes.
- **Server chassis.** Special care should be taken to ensure that the server chassis layout is balanced and sufficient PCIe* bandwidth is allocated for the devices.
- **Hardware support and personnel costs.** Minio uses erasure code parity calculations to provide data durability. The number of drives that are used for parity writes for each object can be as high as N/2. This parity level means you can lose up to 50 percent of the drives and continue to serve data. Therefore, you can balance hardware support and personnel costs by changing your operational model and elect to ignore disk failures. This can reduce your operational costs and avoids human error when replacing failed disks.

Although we did not test them, there are several ways that performance could be increased:

- Add an additional dedicated 40 GbE NIC for the Minio internode communications.
- Keep the existing network configuration but add additional 40 GbE NIC ports to a multi-chassis link aggregation (MLAG) pair.
- Or, if 100 GbE is desired, up to 4x25 GbE NIC ports could be combined into a link aggregation group (LAG).
- Cache remote S3 objects for faster access using an Intel Optane SSD. Once caching is configured, a remote object will be cached locally once fetched. Each subsequent request for that object gets served directly from the Minio cache until it expires.

Recommended Configuration

This performance test used an eight-node Minio cluster (see Figure 1). Although we used eight servers, Minio distributed clusters can be deployed on anywhere from four to 32 servers. The configuration details for our eight-node cluster are provided in Table 3. See Appendix A for tuning details.

Table 3. Reference Configuration Details

Component	Details
Client Nodes (8x)	2x Intel® Xeon® Gold 6140 processor @ 2.30 GHz 128 GB RAM 1x Intel® XL710-QDA2 Dual-Port NIC (40 GbE)
Storage Nodes (8x)	2x Intel Xeon Gold 6240C processor @ 2.60 GHz 192 GB RAM 5x Intel® SSD DC P4510 Series 4TB 2x Intel SSD DC P4510 Series 2TB 1x Intel SSD DC P4610 Series 1.6TB 1x Intel XL710-QDA2 Dual-Port NIC (40 GbE) Minio release.2019-02-12T21-58-47Z with Intel® Advanced Vector Extensions 512 (Intel® AVX-512) optimizations Parity configuration: data=12 and parity=4
Other Details	CentOS* release 7.5.1804 Linux* kernel: 3.10.0-862.14.4.el7.x86_64 XV710 Driver Version: 2.7.27 XV710 Firmware Version: 6.80 0x80003cfb 1.2007.0 COSbench* version 0.4.2.20160615 Switch: Extreme Networks* X770 Series

Obtaining Support and Fixing Bugs

There are multiple options for deploying and supporting a Minio object storage-based STaaS:

- Self-support using Minio documentation: https://docs.minio.io/
- Support by the Minio Slack channel and open source community: https://www.minio.io/community.html
- Contribute to the Minio product, report bugs and request features on Github: https://github.com/minio
- Minio SUBNET Support subscription for production deployments of Minio: https://www.minio.io/subscription.html

The Minio open source community provides significant support through the Minio Slack channel and through the online documentation. Although you can purchase a support subscription, if you are working with a skilled Linux* team then using the free open source distribution of Minio along with self-support may be the correct choice. A large production deployment of Minio could warrant the peace of mind that comes with the Minio SUBNET support subscription.

Summary

STaaS offerings can help you expand your customer base and achieve economies of scale. SDS solutions such as Minio, running on Intel architecture and technology, provide many benefits, including:

- Improved resource utilization and performance
- Shortened application time to market
- Efficiencies associated with automated deployment and operations

Minio is easy to use and fast enough to support transactional workloads such as big data analytics, streaming workloads, AI and machine learning. The reference architecture shown here will enable you to confidently offer a simple yet powerful STaaS solution using Minio. As a result, you can successfully compete against much larger CSPs and tap into a marketplace worth billions of dollars.

Appendix A - System Tuning Details

Getting the best performance out of Minio on a Linux system doesn't require extensive system tuning, but some basic best practices for SDS platforms were applied to the systems under test, as shown below.

Network and Kernel Tuning Parameters in /etc/sysctl.conf

```
###-------------------------------------------------------###
###    Settings to tune 40Gb NICs and system perf.
###-------------------------------------------------------###
kernel.pid _ max=4194303
fs.fle-max=4194303
vm.swappiness = 1
vm.vfs _ cache _ pressure = 10
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
net.core.somaxconn=65535
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 _ fn _ timeout=5
net.ipv4.udp _ rmem _ min=8192
net.ipv4.udp _ wmem _ min=8192
net.ipv4.conf.all.send _ redirects=0
net.ipv4.conf.all.accept _ redirects=0
net.ipv4.conf.all.accept _ source _ route=0
net.ipv4.tcp _ mtu _ probing=1
vm.min _ free _ kbytes=1000000
```

Setting IRQ Affinity

Configuring Interrupt Request (IRQ) affinity so that interrupts for different network queues are affinitized to different CPU cores can have a huge impact on performance, especially with highly concurrent workloads like Minio. A script is provided with the Intel NIC driver packages to facilitate this and was run on all of the systems under test. We set the driver to use all cores with the following command:

```
#  set _ irq _ affinity -x all <interfacename>
```

Solutions Proven by Your Peers

An STaaS platform, powered by Intel technology, provides high performance and easy manageability. This and other solutions are based on real-world experience gathered from customers who have successfully tested, piloted, and/or deployed the solutions in specific use cases. The solutions architects and technology experts for this solution reference architecture include:

- **Daniel Ferber,** Solutions Architect, Intel Sales & Marketing Group
- **Karl Vietmeier,** Solutions Architect, Intel Sales & Marketing Group

Intel Solutions Architects are technology experts who work with the world's largest and most successful companies to design business solutions that solve pressing business challenges. These solutions are based on real-world experience gathered from customers who have successfully tested, piloted, and/or deployed these solutions in specific business use cases.

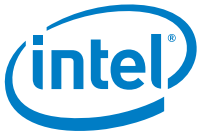
Find the solution that is right for your organization. Contact your Intel representative or visit **intel.com/CSP**.

Learn More

Be sure to bookmark these resources:

- Minio* home page
- Intel® Cloud Insider Program
- Intel® Storage Builders Program
- Intel® Xeon® Scalable Processors
- Intel® Solid State Drives
- Intel® Converged Network Adapters
- Intel® Rack Scale Design

Solution Provided By:



[1] IDC, 2017 H1, "Worldwide Semiannual Public Cloud Services Tracker."
https://www.idc.com/tracker/showproductinfo.jsp?prod_id=881

[2] Seagate, April 2017, "Data Age 2025."
https://www.seagate.com/files/www-content/our-story/trends/files/Seagate-WP-DataAge2025-March-2017.pdf

[3] BusinessWire, 2016, "Global Storage as a Service Market 2016-2020 - Market to Grow at a CAGR of 29.59% - Research and Markets."
https://www.businesswire.com/news/home/20160715005348/en/Global-Storage-Service-Market-2016-2020---Market

[4] Calculation: (8 servers * 40 Gbit * 50% = 160 Gbit/s = 20 GB/sec)

0319/JS/CAT/PDF 338821-001EN