RED HAT® STORAGE

REFERENCE ARCHITECTURE

RED HAT CEPH STORAGE: SCALABLE OBJECT STORAGE ON QCT SERVERS

A performance and sizing guide

QCT (Quanta Cloud Technology) offers industry-standard servers for scalable object storage clusters based on Red Hat Ceph Storage.

The combination of Red Hat Ceph Storage and QCT storage servers provides a compelling platform for flexible and scalable object storage.

Extensive Red Hat testing and tuning has measured and validated object storage performance for large and small objects as well as for higher object count storage workloads, ranging to hundreds of millions of objects.

ABSTRACT

With applications for object storage growing rapidly, organizations need to understand how to best configure and deploy software, hardware, and network components to serve a range of diverse workloads. This reference architecture describes the combination of Red Hat® Ceph Storage coupled with QCT (Quanta Cloud Technology) storage servers and networking as object storage infrastructure. Testing, tuning and performance are described for both large-object and small-object workloads. Testing also evaluated the ability of configurations to scale to host hundreds of millions of objects.

TABLE OF CONTENTS

INTRODUCTION

As new applications increasingly move toward cloud models, they have evolved to use cloud storage primitives. Object storage is an effective way to provision flexible and massively-scalable data storage without the arbitrary limitations of traditional proprietary or scale-up storage solutions. While most are familiar with deploying block or file storage, object storage expertise is less common. Before building object storage infrastructure at scale, organizations need to understand the performance and scalability they can expect from given hardware, software, and network configurations.

Though it is often used for block storage, Ceph is fundamentally an object storage platform at its core. To understand Ceph's object storage capabilities, Red Hat conducted an extensive workload-centric testing process for Red Hat Ceph Storage on QCT storage servers, focused on identifying ideal hardware and software configurations for diverse kinds of object storage. The main areas of testing focus included:

- **Large-object performance (throughput).** Large-object sequential input/output (I/O) workloads are one of the most common use cases for Ceph object storage. These high-throughput workloads include big data analytics, backup and archival systems, image storage, and streaming audio, and video. For these types of workloads throughput (MB/s or GB/s) is the key metric that defines storage performance.
- **Small-object performance (object operations per second).** Thumbnail images, small files, documents, and static website pages are all examples of small-object workloads that can be accommodated on object storage. Measuring operations per second (OPS) is key for these workloads.
- **High object count (scalability).** With exponential data growth, organizations need to be able to seamlessly scale from tens to hundreds of millions of objects. In response, object storage platforms need to scale predictably in terms of read and write operations without accumulating unacceptable amounts of latency.

The combination of QCT servers and Red Hat Storage software is well suited for object storage, and both are already at the heart of many public and private cloud deployments. QCT is reinventing datacenter server technology to boost storage capacity and density, and creatively designing scalable hardware for cloud applications. Together, QCT servers and Red Hat Ceph Storage provide software-defined storage solutions for both private and public clouds, helping to accelerate the shift away from costly, proprietary external storage.

CEPH ARCHITECTURE OVERVIEW

A Ceph storage cluster is built from large numbers of nodes for scalability, fault-tolerance, and performance. Each node is based on industry-standard server 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 upon cluster expansion or hardware failure (backfilling and recovery).
- 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.

When a Ceph client reads or writes data, 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.

- ***Client interface layer.*** Writing and reading data in a Ceph storage cluster is accomplished using the Ceph client architecture. Ceph supports a range of storage methods. The RADOS gateway (RADOSGW) is an object storage gateway service that provides S3-compatible and OpenStack® Swift compatible RESTful interfaces. LIBRADOS provides direct access to RADOS with libraries for most programming languages. RBD offers a Ceph block storage device that mounts like a physical storage drive for both physical and virtual systems.
- ***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. For data protection, 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—enabling optimize storage for different workloads.
- ***Placement groups.*** Ceph maps objects to placement groups (PGs), which are shards or fragments of a logical object pool composed of a group of Ceph OSD daemons in a peering relationship. Peer OSDs each receive an object replica (or erasure-coded chunk) upon a write. Fault domain policies within the CRUSH ruleset can force OSD peers to be selected on different servers, racks, or rows. Placement groups enable the creation of 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.

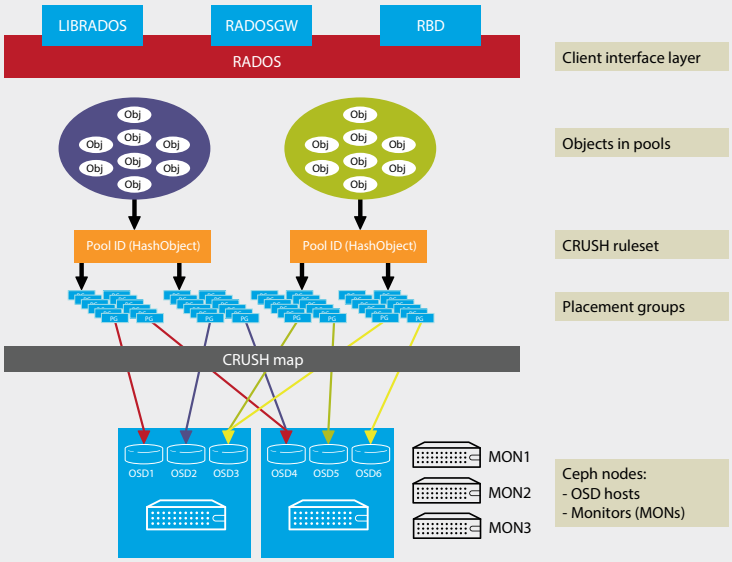

Figure 1. Clients write to Ceph storage pools while the CRUSH ruleset determines how placement groups are distributed across object storage daemons (OSDs).

- ***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 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 OSDs store data and handle data replication, recovery, backfilling, and rebalancing. They also provide some cluster state information to Ceph monitors by checking other Ceph OSD daemons with a heartbeat mechanism. A Ceph storage cluster configured to keep three replicas of every object requires a minimum of three Ceph OSD daemons, two of which need to be operational to successfully process write requests. Ceph OSD daemons roughly correspond to a file system on a physical hard disk drive (HDD) or flash. Multiple OSDs can exist on a physical OSD node.

TEST RESULTS SUMMARY

Organizations need to understand how to configure hardware for optimized Ceph object storage clusters that meet their unique needs. Red Hat Ceph Storage is able to run on myriad industry-standard hardware configurations. However, designing a successful Ceph cluster for a specific use case requires careful analysis of issues related to application, capacity, and workload. The ability to address dramatically different kinds of I/O workloads within a single Ceph cluster makes understanding these issues paramount to a successful deployment.

Red Hat and QCT established that careful platform choice and selective tuning can have a profound effect on performance. Tests evaluated workloads in three broad categories:

- **Large-object workloads (throughput).** Red Hat testing showed near linear throughput scalability for both reads and writes. Read throughput scaled easily with the addition of RADOSGW (RGW) hosts, while write throughput scaled until it peaked due to OSD node disk contention. Write throughput was increased by an additional 40% by tuning Ceph chunk size, thereby reducing disk I/O requests.
- **Small-object workloads (operations per second).** Small-object workloads are more heavily impacted by metadata I/O than are large-object workloads. Client read operations scaled linearly with the addition of RGW hosts, limited only by the number of load-generating clients in the test environment. Client write operations scaled sublinearly, and improved up to 50% by placing the Ceph bucket index pool on solid state device (SSD) media.
- **Higher object-count workloads (100M+ objects).** As the cluster object count grew, read operations per second (OPS) declined due to an increase in kernel slab cache miss rate for filesystem metadata lookups. However, write IOPS maintained a consistent level. Optimal results for both read and write OPS were obtained using Intel Cache Acceleration Software (Intel CAS) caching XFS filesystem metadata to SSDs.

Efficiency and price/performance analysis of the complete results yielded recommended configurations for various object workloads, as shown in Table 1.

TABLE 1. OBJECT STORAGE OPTIMIZATION SUMMARY

WORKLOAD	PERFORMANCE AND SIZING GUIDANCE
LARGE-OBJECT WORKLOADS	Standard-density OSD servers (12 HDDs) offered more consistent performance at scale. High-density OSD servers (35 HDDs) offered superior price/performance. Tuned Ceph chunk size increased write performance by 40%.
SMALL-OBJECT WORKLOADS	Standard-density OSD servers with bucket index pools configured on SSDs
HIGHER OBJECT-COUNT WORKLOADS	Standard-density OSD nodes with default OSD filestore configurations and Intel CAS software to cache XFS filesystem metadata*

* *High object-count testing was only conducted on high-density servers with small-object workloads. However, standard-density servers are expected to deliver optimal performance, consistent with both small and large workload test results.*

OBJECT STORAGE ON QCT SERVERS

In Red Hat and QCT testing, the solution architecture included Red Hat Ceph Storage installed on industry-standard QCT storage servers.

RED HAT CEPH STORAGE

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

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

QCT SERVERS FOR CEPH

Scale-out software-defined storage requires capable and scalable server platforms that can be selected and sized to meet the needs of specific workloads. Driven by social media, mobile applications, and the demands of hyperscale datacenters, storage servers and storage platforms must provide increasing capacity and performance to store growing volumes of data with ever-longer retention periods. QCT servers are offered in a range of configurations to allow optimization for diverse application workloads.

QCT servers used in Red Hat testing provide massive storage capacity in a small space and also include PCIe Generation 3 (Gen3) slots, allowing NVM Express (NVMe) SSDs for Ceph journaling. SSD caching is important to accelerate both IOPS and throughput in many software-defined storage technologies. In this reference architecture, Red Hat and QCT have tested the following servers optimized for Ceph object storage workloads:

- **QCT QuantaPlex T21P-4U server.** Capable of delivering up to 780TB of storage in just one four rack-unit (4U) system, the QuantaPlex T21P-4U efficiently serves the most demanding cloud storage environments. This server maximizes storage density to meet the demand for growing storage capacity in hyperscale datacenters. Two models are available: A single storage node can be equipped with 78 HDDs to achieve ultra-dense capacity and low cost per gigabyte, or the system can be configured as dual nodes, each with 35 HDDs to optimize rack density. Along with support for four PCIe Gen3 slots (for two SSDs per node), the server offers flexible and versatile I/O expansion capacity. The ratio of regular drives to SSDs is 17.5:1 in a dual-node configuration, helping to boost Ceph performance and IOPS. The QCT QuantaPlex T21P-4U server also features a unique, innovative screwless hard drive carrier design to let operators rapidly complete system assembly, significantly reducing deployment and service time. The server offers flexible networking options to support workload requirements based on application needs. QCT mezzanine cards provide varied Ethernet connectivity, ranging from Gigabit Ethernet (GbE) to 10 GbE and 40 GbE.

- **QCT QuantaPlex T41S-2U server.** The QuantaPlex T41S-2U is an ultra-dense server chassis equipped with four independent nodes. Powered by the Intel Xeon E5-2600 v3 and v4 product family, it provides cost-effective datacenter performance per dollar and per rack unit. By sharing system infrastructure, such as cooling and power supply, the total cost of ownership (TCO) of the T41S-2U is lower than four regular 1U servers. The QuantaPlex T41S-2U also provides sufficient I/O expansion for storage and networking and uses a modular design concept to optimize system interoperability, flexibility, and serviceability. For example, the T41S-2U system is designed with a dual-rotor fan that keeps the chassis' internal air flowing in the right direction even if one rotor fails, enhancing stability and durability.

QCT QuantaPlex T21P-4U
(OSD nodes)

QCT QuantaPlex T41S-2U
(Ceph MON, RGW, and client nodes)

Figure 2. QuantaPlex T21P-4U and T41S-2U servers were used in Red Hat and QCT testing.

Seagate Enterprise Capacity HDDs

Seagate and QCT have worked closely to deliver the benefits of Ceph object storage on Seagate's award-winning Enterprise Capacity 3.5-inch HDDs. Utilizing high-performance, low-power 6TB SAS drives, the QCT solution ensures compelling host object storage throughput under Red Hat Ceph Storage. With QCT's rigorous proofs of concept and Seagate's advanced caching technology, organizations can confidently and consistently store data without sacrificing performance.

Seagate Enterprise Capacity 3.5-inch HDDs offer:

- Storage infrastructure that scales to meet growing capacity needs while offering users a consistent and predictable response rate.
- Support for enterprise-class nearline workloads of 550TB per year (10 times the rated workload of desktop HDDs) and backed by a 2-million hour mean time between failure (MTBF) rating and a five-year limited warranty.
- Highly scalable 12Gb/s SAS storage for replicated and redundant array of independent disk (RAID) bulk storage systems—perfect for the growth of unstructured data.
- The industry's best response times, enabling the fastest data transfers thanks to comprehensive advanced caching technology.
- User-definable innovative technology advancements like PowerBalance, PowerChoice, and RAID Rebuild, give organizations the control to tailor bulk storage requirements for even greater improvements in lowering TCO.
- Seagate Secure drives with self-encrypted technology protect data where it lives—on the drive.

LABORATORY CONFIGURATION

Figure 3 and Table 2 detail the QCT server-based testbed used to build the Ceph cluster in this study.

TABLE 2. QCT QUANTAPLEX SERVERS AND SWITCHES USED IN RED HAT TESTING.

COMPONENT	QUANTITY/CONFIGURATION
QUANTAPLEX T21P-4U	3x two-node servers (six nodes total) each equipped with: • 2x Intel Xeon E5-2660 v3@ 2.60 GHz • 35x 3.5-inch Seagate Enterprise Capacity 6TB 7.2K RPM 12Gb/s SAS • 2x Intel P3700 800G NVMe SSD • 40GbE Mellanox ConnectX-3 Pro
QUANTAPLEX T41S-2U	4x four-node servers (16 nodes total) each equipped with: • 2x Intel Xeon E5-2670 v3 @2.30 GHz, 96GB memory, and up to 40 GbE
QUANTAMESH SWITCHES	• QuantaMesh T3048-LY8: 48 SFP+ and 6 QSFP+ ports • QuiantaMesh T5032-LY6: 32 QSFP+ 10/40GbE ports

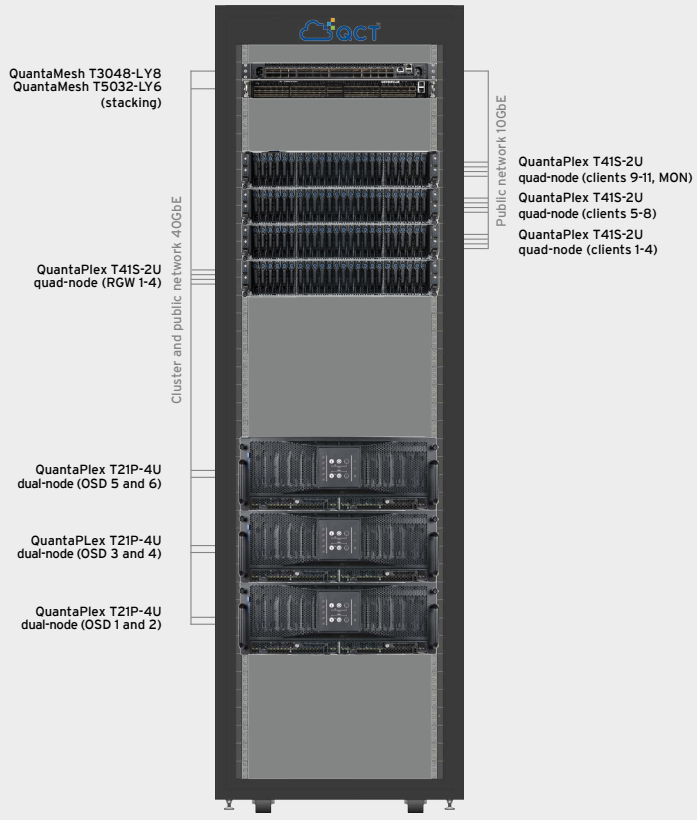

Figure 3. Red Hat tested a cluster with six OSD nodes hosted on three QCT QuantaPlex T21P-4U nodes.

STANDARD-DENSITY AND HIGH-DENSITY SERVERS IN CEPH CLUSTERS

As a part of Red Hat testing, different OSD node configurations were analyzed to help evaluate both performance and price/performance. Specifically, the OSD node I/O subsystem configuration was altered to help determine the best performing configuration for specific usage scenarios and to compare price/performance. The same QCT QuantaPlex servers were configured to simulate two different OSD node configurations:

- **High-density servers.** Dual-node QCT QuantaPlex T21P-4U servers with 35 disks per node.
- **Standard-density servers.** Dual-node QCT QuantaPlex T21P-4U servers with 12 disks per node[1].

Tables 3 and 4 list the configuration details used for both kinds of servers evaluated in Red Hat testing.

TABLE 3. TESTED SERVER CONFIGURATIONS

CONFIGURATIONS	HIGH-DENSITY STORAGE SERVER	STANDARD-DENSITY STORAGE SERVER
PROCESSOR	2x Intel Xeon E5-2660 v3, 2.60 GHz	1x Intel Xeon E5-2660 v3, 2.60 GHz
MEMORY	128GB	
NETWORK	1x 40GbE Mellanox ConnectX-3	
OS DISK TYPE	1x Intel SSD DC S3510 series, 200GB	
STORAGE DRIVES	3.5-inch Seagate Enterprise Capacity 6TB 7.2K RPM 12Gb/s SAS	
I/O CONTROLLER	Quanta SAS 3008 Mezz card 12Gb/s	
I/O CONTROLLER CONFIGURATION	Pass through (JBOD) mode	
NUMBER OF DRIVES PER NODE	35	12
JOURNAL DEVICE TYPE	Intel SSD P3700 800G NVMe AIC	
JOURNAL DEVICE COUNT	2	1
OSD TO JOURNAL RATIO	18:1, 17:1	12:1
JOURNAL PARTITION SIZE	10GB	
NUMBER OF OSDS IN CEPH CLUSTER	210	72
RAW CAPACITY PER NODE	210TB	72TB
RAW CLUSTER CAPACITY	1.2PB	430TB

1 *Red Hat testing used T21P-4U servers with partially-filled media bays to simulate a standard-density server with 12 media bays.*

TABLE 4. CEPH CLUSTER CONFIGURATION DETAILS

COMPONENT	OSD	MON	RGW	CLIENTS
QCT SYSTEM	QuantaPlex T21P-4U	QuantaPlex T41S-2U		
CHASSIS	3	4		
NODE COUNT	6	1*	4	11
STORAGE	1x 2.5" Intel S3510 200GB SATA (OS) 35x 3.5" Seagate Enterprise Capacity 6TB 7.2K RPM 12Gb/s SAS (Ceph OSDs)	1x 2.5-inch Intel S3510 480GB SATA (OS)		
NETWORK	1x MT27520 Mellanox ConnectX-3 Pro (40Gbe) 2x Intel I350 (GbE)	1x Intel 82599ES 10GbE SFP+	1x MT27520 Mellanox ConnectX-3 Pro (40GbE)	1x Intel 82599ES 10GbE SFP+

* *For testing only. For production environments, Red Hat recommends at least three monitor nodes for redundancy and high availability.*

SOFTWARE COMPONENTS

The following software versions were used in Red Hat testing:

- Red Hat Ceph Storage 2.0
- Red Hat Enterprise Linux® 7.2
- Benchmarking tools:
  - Ceph Benchmarking Tool (CBT) (github.com/ceph/cbt)
  - FIO 2.11-12 (github.com/axboe/fio)
  - iPerf3 (iperf.fr/iperf-download.php)
  - COSBench 0.4.2.c3 (https://github.com/intel-cloud/cosbench)
  - Intel Cache Acceleration Software (CAS) 03.01.01 (intel.com/content/www/us/en/software/intel-cache-acceleration-software-performance.html)

TESTING METHODS AND PERFORMANCE SUMMARY

In order to fully understand the performance characteristics of the Ceph clusters under test, Red Hat performed extensive baseline testing that included a range of payloads. Beyond raw performance, results were evaluated in terms of both efficiency and price/performance.

BASELINE TESTING SUMMARY

Before performing object benchmarks that utilized higher-level Ceph protocols, Red Hat executed a series of simpler performance tests to establish known performance baselines for all relevant subsystems. Baseline testing included:

- Hard disk drive and NVMe tesing using FIO (4KB random read/write and 4MB sequential read/write on top of XFS)
- Network testing using iPerf3 (multiple TCP-stream benchmarks, all-to-all)
- Ceph baseline testing using CBT to evaluate RADOS performance (4M sequential read/write)

The full baseline testing methodology is documented in Appendix A.

PAYLOAD SELECTION

Object storage payloads vary widely, payload size is a crucial consideration in designing benchmarking test cases. In an ideal benchmarking test case, the payload size should be representative of a real-world application workload. Unlike testing for block-storage workloads that typically read or write only a few kilobytes, testing for object storage workloads needs to cover a wide spectrum of payload sizes. For Red Hat testing, the following object payload sizes were selected:

- Small-object payload (64KB) representing thumbnail images, small files, etc.
- Medium-object payload (1MB) representing images, documents, etc.
- Medium/large-object payload (32MB), representing of high-definition (HD) images or log files.
- Large-object payload (64MB), representing backup files, videos, etc.

As a real-world example, the 32MB object size is representative of big data and analytics workloads. Various Apache Hadoop distributions typically use 32MB as the default multipart size for write operations.

EFFICIENCY-BASED RESULTS REPORTING

Rather than focus purely on maximum throughput reporting, the ultimate goal of Red Hat testing was to evaluate the efficiency of a hardware investment against a variety of configurations and workloads. To achieve this goal, results were calculated as total throughput divided by the number of disks in the storage solution, regardless of the chosen data protection scheme. This approach yielded a performance-per-disk metric. As a result, performance can be related directly to the required investment in storage media.

MEASURING PRICE/PERFORMANCE

Red Hat further refines this model by offering a price/performance metric, where performance values are considered against the total storage solution hardware and software costs. This approach allows workload and cost to be cross-referenced, arriving at the best server configuration for a given set of needs.

OPTIMIZING FOR LARGE-OBJECT THROUGHPUT

Red Hat tested a variety of configurations, object sizes, and client worker counts in order to maximize throughput of a six-node Ceph cluster for large-object workloads. The Ceph cluster was built with a single OSD configured per HDD. Testing first exercised the dual-node QCT QuantaPlex T21P-4U configured as high-density servers (6x nodes * 35x HDDs = 210 OSDs). The same servers were then reconfigured as standard-density servers (6x nodes * 12x HDD = 72 OSDs).

Note: The terms "read" and HTTP GET are used interchangeably throughout this document, as are the terms HTTP PUT and "write".

LARGE-OBJECT HTTP GET WORKLOAD

Large-object 100% HTTP GET workload testing exhibited near linear scalability when incrementing the number of RGW hosts together with increasing client load. In other words, for each client making 32MB read requests at 10Gbps, an additional RGW host allowed cluster read throughput to scale nearly linearly.[2] Adding more RGW hosts and corresponding client load would have increased aggregate cluster throughput, until limited by Ceph cluster hardware resources (disk saturation on OSD hosts).

Figures 4 and 5 shows client read throughput normalized per OSD and cluster-wide aggregate throughput respectively for 1MB, 32MB, and 64MB objects. The highest observed read throughput for 32MB object reads with four RGWs and four clients was 4GB/s on the cluster built from standard-density servers. Additional clients did not result in increased cluster throughput. Because cluster throughput was constrained by the four RGW hosts, similar tests against the high-density cluster did not result in increased cluster throughput. This result illustrates that read performance was throttled by RGW host bandwidth and not OSD host disk utilization.

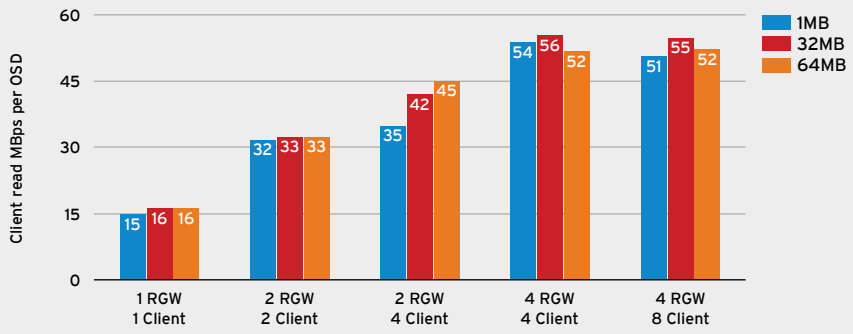

Figure 4. Large-object read test, normalized per OSD on standard-density servers

2 *Each Ceph RGW host used in this benchmarking had a single 40GbE NIC. However, test results only showed the ability to utilize 10GbE effective bandwidth.*

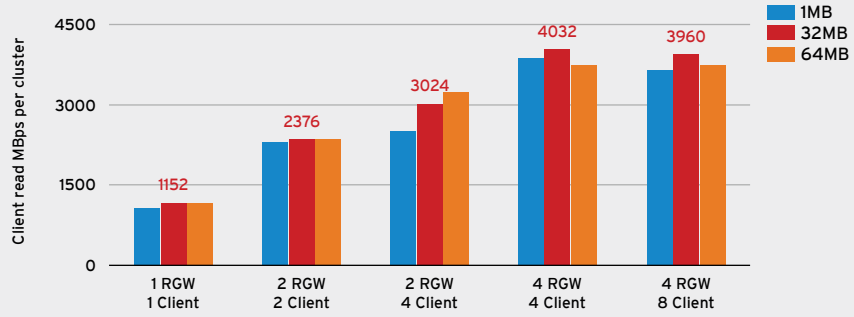

Figure 5. Large-object read test, cluster-wide aggregate performance, on standard-density servers

Similarly, as shown in Figure 6, a high-density cluster with a workload configuration of four RGWs and eight clients also did not result in increased cluster throughput. Instead, the performance was constrained by the four RGW hosts while no bottleneck was observed for the Ceph cluster itself. Again, addition of more RGW hosts and corresponding client load would have increased aggregate cluster throughput, until limited by Ceph cluster hardware resources.

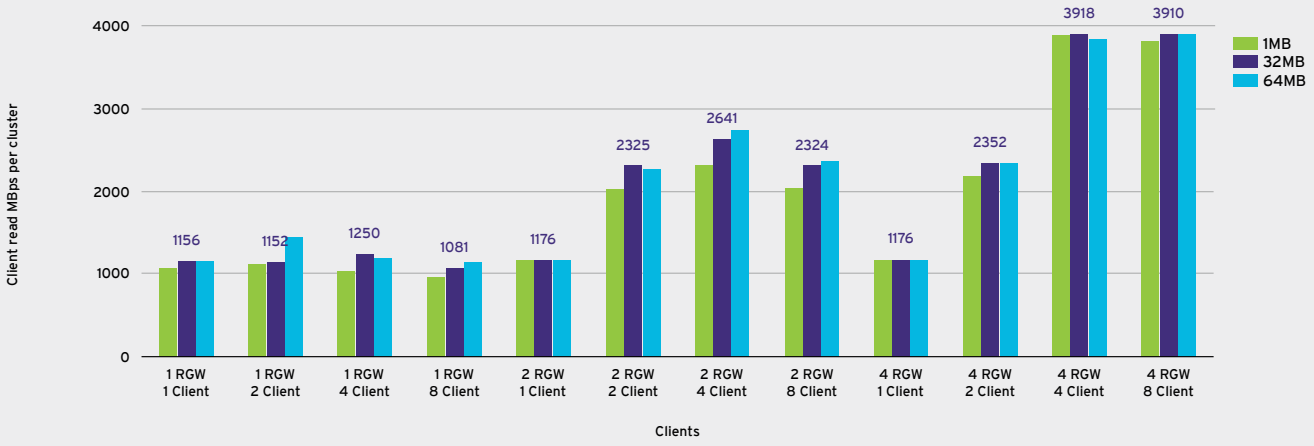

Figure 6. Large-object read test, cluster-wide aggregate performance, on high-density servers

A separate test was then conducted to compare RGW deployment strategies. As shown in Figure 7, the test compared three RGW configurations:

- Dedicated RGW with 10GbE
- Dedicated RGW with 40GbE
- 40GbE RGWs co-located with OSDs[3]

3. Co-location of RGW services on OSD hosts is not officially supported within Red Hat Ceph Storage as of this writing.

The results revealed that with the 10GbE dedicated RGW configuration, each RGW host was able to make use of entire network bandwidth available to it—yielding better resource utilization. This was especially true with larger 32MB and 64MB objects (Figure 7). Both dedicated and colocated RGWs with access to 40GbE were unable to reach network saturation, despite sufficient load generation from client nodes.

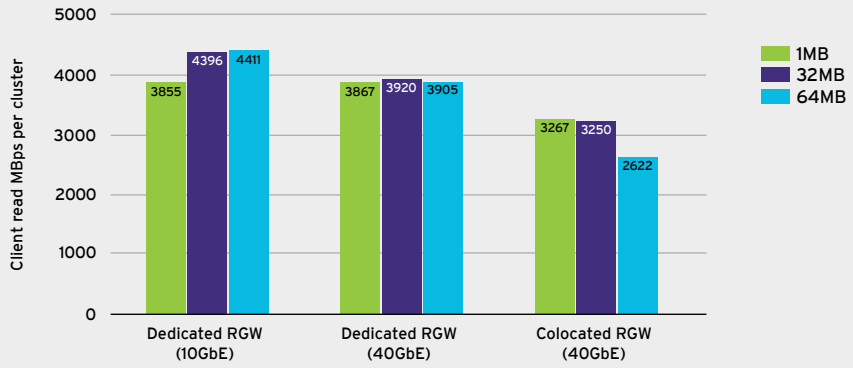

Figure 7. Dedicated RGWs with 10GbE interfaces provided better performance and price/performance (4x RGW, 8x client nodes, high-density servers).

LARGE-OBJECT HTTP PUT WORKLOAD

Similar to HTTP GET workloads, large-object HTTP PUT workload tests also exhibited near-linear scalability when incrementing the number of RGW hosts together with increasing client load (Figure 8). However, the write-focused workload was able to reach cluster saturation, limited by disk contention on Ceph OSD nodes. Each client generating a write workload at a 10Gbps line rate could saturate an RGW host with a 10GbE interface. The peak observed write throughput with 32MB object workload was just over 2100MBps on the high-density cluster as shown in Figure 9.

Throughput was constrained by Ceph OSD disk contention after this point. Unlike read tests, incrementing the number of RGW hosts together with client hosts did not materially increase aggregate cluster throughput. Overall, the results indicate that cluster write throughput would have continued to increase had we added more OSD hosts.

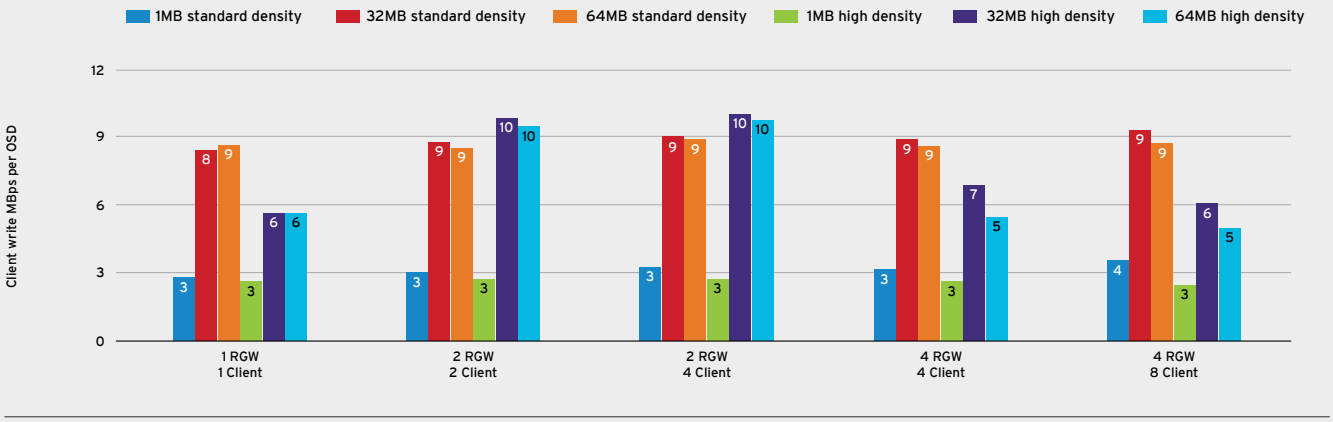

Figure 8. Large-object write test, per-OSD normalized performance, on both standard-density and high-density servers (40GbE RGWs, 10GbE clients)

Contrary to the HTTP GET workload tests, peak cluster write performance was achieved with only two RGW hosts. Incrementing the number of RGW hosts together with client hosts further resulted in retrograde performance (Figure 9).

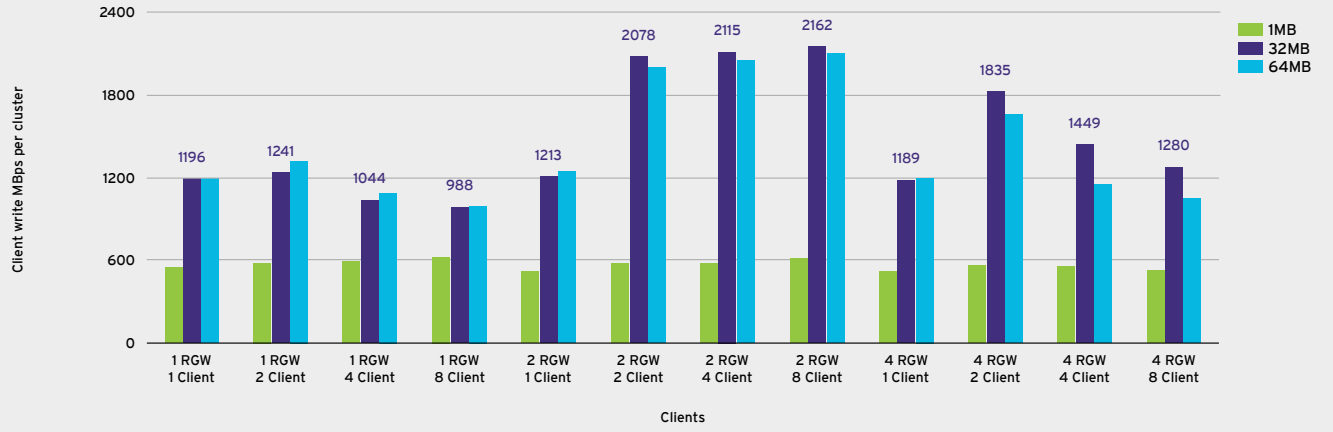

Figure 9. Large-object write test, cluster-wide aggregate performance on high-density servers

Analysis of system-level metrics taken during testing revealed that the disks on the OSD hosts had indeed reached 100% utilization. Figure 10 shows Ceph OSD host disk utilization during both 2-RGW/4-client and 2-RGW/8-client tests. This graph demonstrates that write throughput was clearly limited by disk contention. To scale the cluster's ability to service additional write workload would require adding OSD hosts.

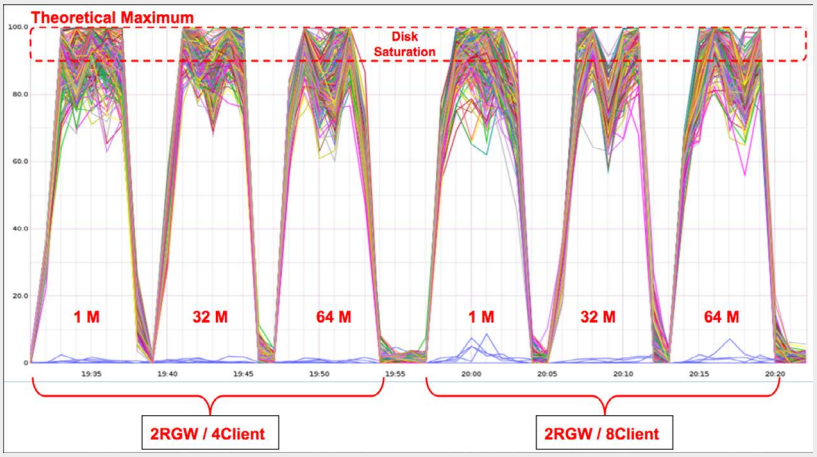

Figure 10. Ceph OSD disk utilization clearly showed disk saturation with only two RGWs and a variable numbers of clients.

To better understand this performance limitation, Red Hat engineers conducted an investigation into the hardware subsystem and Ceph configuration. Figure 11 shows subsystem metrics for the 2-RGW/2-client configuration at the time of 32MB and 64MB object write test execution as captured during the benchmark run. The subsystem graph captures aggregated network statistics for clients, RGWs, and OSDs; aggregated CPU stats for RGWs and OSDs; and nonaggregated disk utilization for 210 OSD disks (each color line represents one OSD).

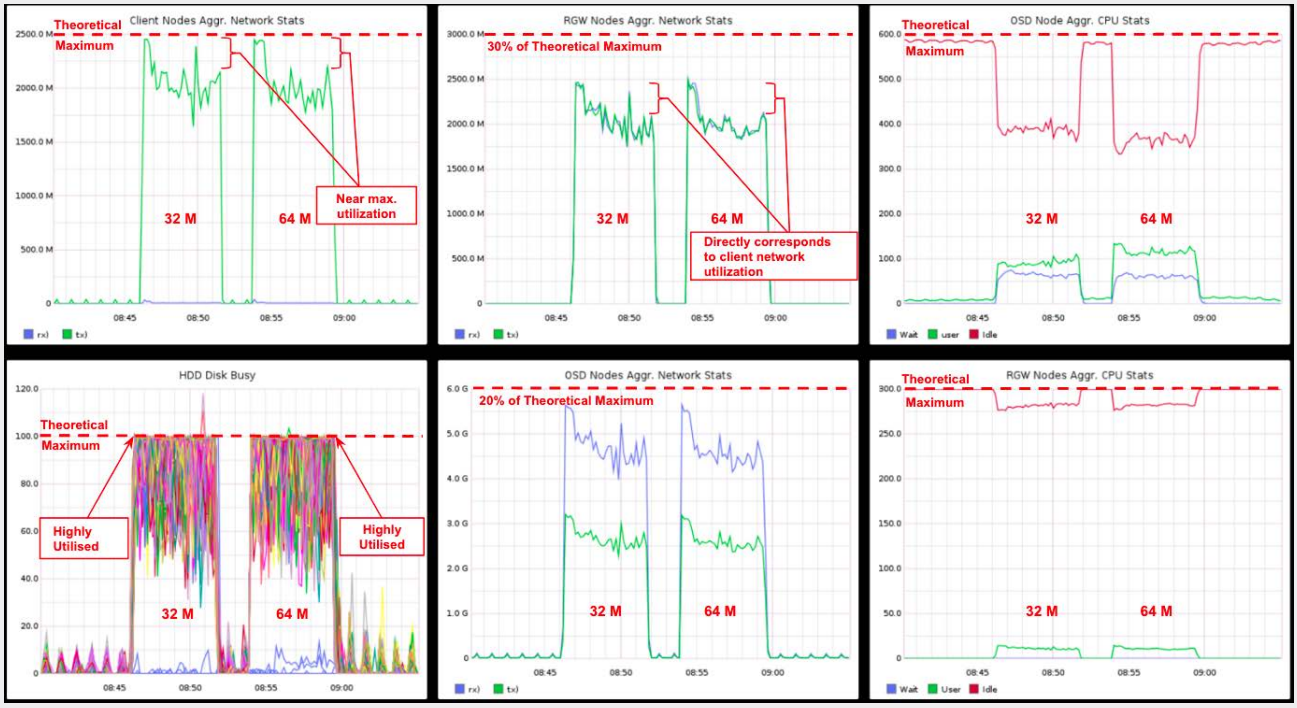

Figure 11. 2-RGW/2-client test, 32MB and 64MB object write, default RGW configuration

From the subsystem resources shown in Figure 11, it can be concluded that:

- CPU resources for both OSD and RGW hosts have enough free capacity and are not constraining performance.
- Network resources for OSD and RGW hosts likewise have sufficient unused bandwidth, showing no bottleneck there as well. Also utilization is not sustained.
- Network resources for client nodes are nearing full utilization.
- All of the HDDs (OSDs) are nearing full utilization serving write requests. Clearly they are causing some performance bottleneck during the write test.

OBJECT CHUNKING AND MINIMIZING I/O AMPLIFICATION

Based on Figure 10 and Figure 11, it is clear that the performance bottleneck was caused by HDD saturation. There are two ways to improve this situation:

- Add more OSD hosts to handle the write workload requirements. While increasing the total solution cost, this will also provide increased performance and storage capacity.
- Tune the `rgw_max_chunk_size` parameter to minimize I/O amplification.

I/O amplification is attributed to the data protection method used (erasure coding in these experiments) and object chunking. Object chunking involves breaking down a single large-object write request into small pieces for optimal handling by Ceph.

To understand how `rgw_max_chunk_size` affects write performance, it is helpful to explore the RGW write request life cycle as shown in Figure 12. The X-axis represents different stages that a single write request goes through after being issued by the client. As per Ceph design principles, each of these stages introduces some kind of object chunking. Once a client issues a write request for an object of a certain size, it goes through several rounds of chunking until it is finally written to the disk. The different stages involved in a client write (PUT) request include:

- **Stage 1.** The client issues a single write request of a 32M object to the RGW.
- **Stage 2.** The RGW instance receives the client write request as a single operation.
- **Stage 3.** Based on the `rgw_obj_stripe_size` parameter (default 4MB), RGW breaks the write request into multiple stripes of 4MB each (eight stripes in this case).
- **Stage 4.** Each of the eight stripes then gets further broken down into chunks of 512KB (default) based on configurable `rgw_max_chunk_size` setting. After this stage, the original object is broken into 64 chunks of 512KB in size.
- **Stage 5.** Since the data protection scheme used for the RGW pool (`default.rgw.buckets.data`) is erasure coding[4] (K=4, M=2), each of the 64 chunks of size 512K gets further broken down into four 128KB data chunks (i.e. 512K/4), and then an additional two erasure coding chunks are added. As a result, the 64-chunk write requests from Stage 4 are amplified to 384 write I/O operations of 128K size to the OSD backing store.
- **Stage 6.** All 384 write requests of 128KB in size are then distributed across all the OSDs of the Ceph cluster and are then written to the media.

Importantly, every storage solution providing data protection exhibits I/O amplification. Most storage solutions also chunk client write requests into smaller, homogeneous I/O requests. Ceph provides sufficient flexibility to control I/O amplification through chunking behavior. By increasing the RGW chunk size, Red Hat engineers were able to dramatically reduce the number of disk operations on OSD hosts.

4 docs.ceph.com/docs/master/dev/osd_internals/erasure_coding/

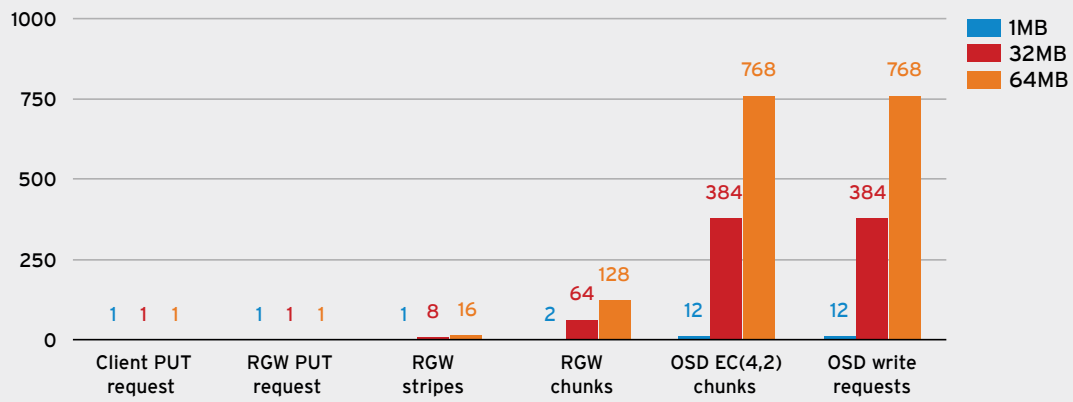

Figure 12. Write-request comparison showing the effects of object chunking for 1MB, 32MB, and 64MB write requests with default RGW settings (lower is better)

Tuning RGW chunk size

Ceph provides sufficient flexibility to improve I/O amplification. As shown, by default each 32MB client object write request gets amplified into 384 write requests of 128KB in size for Ceph OSDs. Ceph allows tuning RGW and it provides a tunable parameter (**rgw_max_chunk_size)** that can drastically reduce the write requests that are submitted to disk drives.

In Red Hat testing, increasing `rgw_max_chunk_size` to 4MB (matching it with `rgw_obj_stripe_size`) immediately resulted in reducing requests sent to disk drives by 87%. As shown in Figures 13 and 14, single client write requests of various sizes were dramatically reduced. For example, I/O requests to write a single 32MB object were reduced to 48 disk operations from 384 disk operations under previous default conditions.

Note: Testing also experimented with altering the Ceph parameter `rgw_obj_stripe_size`, which defaults to 4MB. Increasing the value of this parameter caused OSD flapping and slow requests.

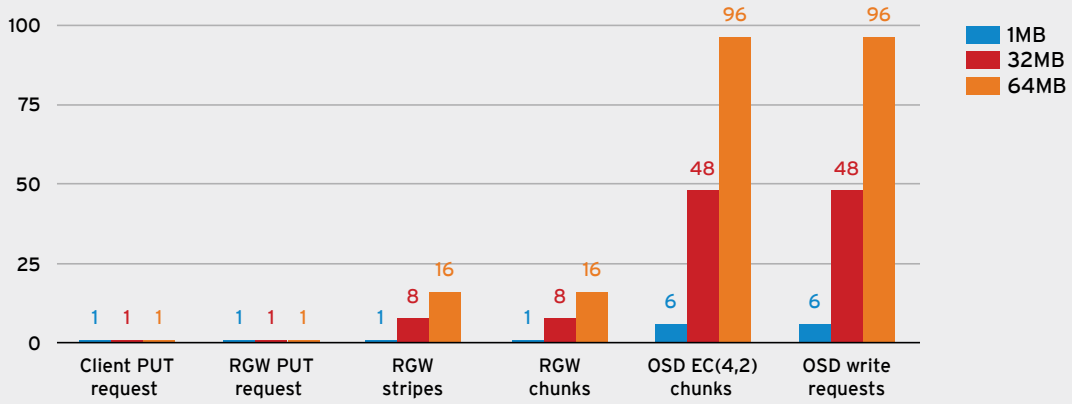

Figure 13. Disk write requests were greatly reduced by tuning rgw_obj_stripe_size (compare to defaults in Figure 11; lower is better).

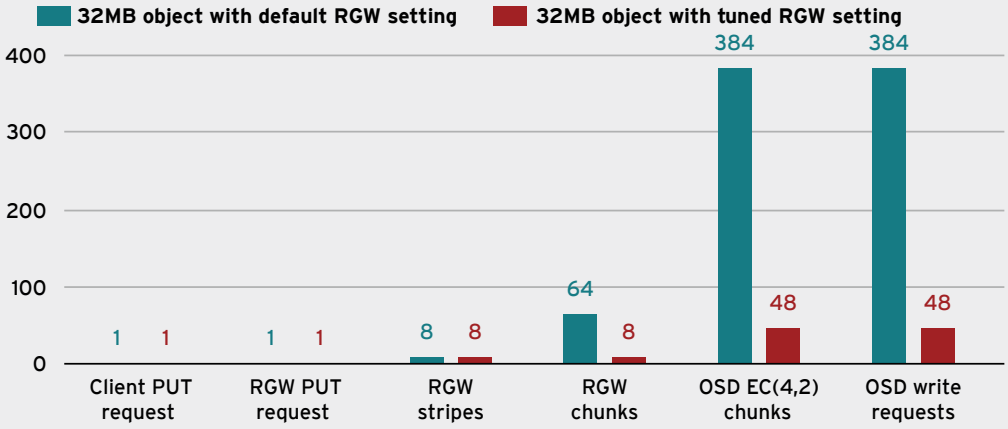

Figure 14. Write requests for a 32MB object are dramatically reduced in the tuned RGW configuration.

Re-evaluating performance with reduced object chunking

After tuning RGWs, engineers repeated 32MB write test with 2-RGW/2-client and 4-RGW/4-client configurations. As shown in Figure 15, they observed an approximate 40% performance improvement as compared to default RGW settings.

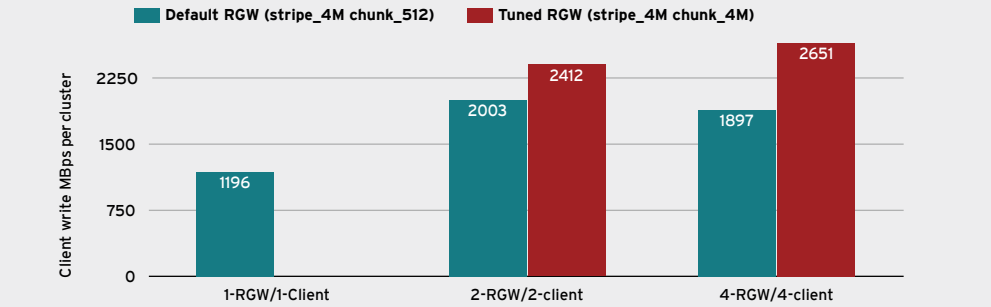

Figure 15.Tuning RGW object stripe size resulted in a roughly 40% improvement in write performance (high-density nodes, 32MB objects write workload; higher is better).

The performance improvement after tuning RGW can easily be seen in subsystem metrics, as shown in Figure 16. Comparing these results with the untuned metrics shown in Figure 11 shows consistently higher aggregate network utilization for clients, RGWs, and OSD nodes, representing significantly better resource utilization after RGW tuning. The disk subsystem section in the tuned configuration shows that disks are more consistently utilized, as these tests are 100% write-intensive tests designed to keep all disks busy during the test run.

This change resulted in roughly a 40% increase in performance, or 2650 MBps. It is likewise clear that if the application workload still needed more write performance then disks would be the next bottleneck. Overall, the results indicate that the cluster write throughput would have continued to increase if additional OSD hosts had been added to the Ceph cluster.

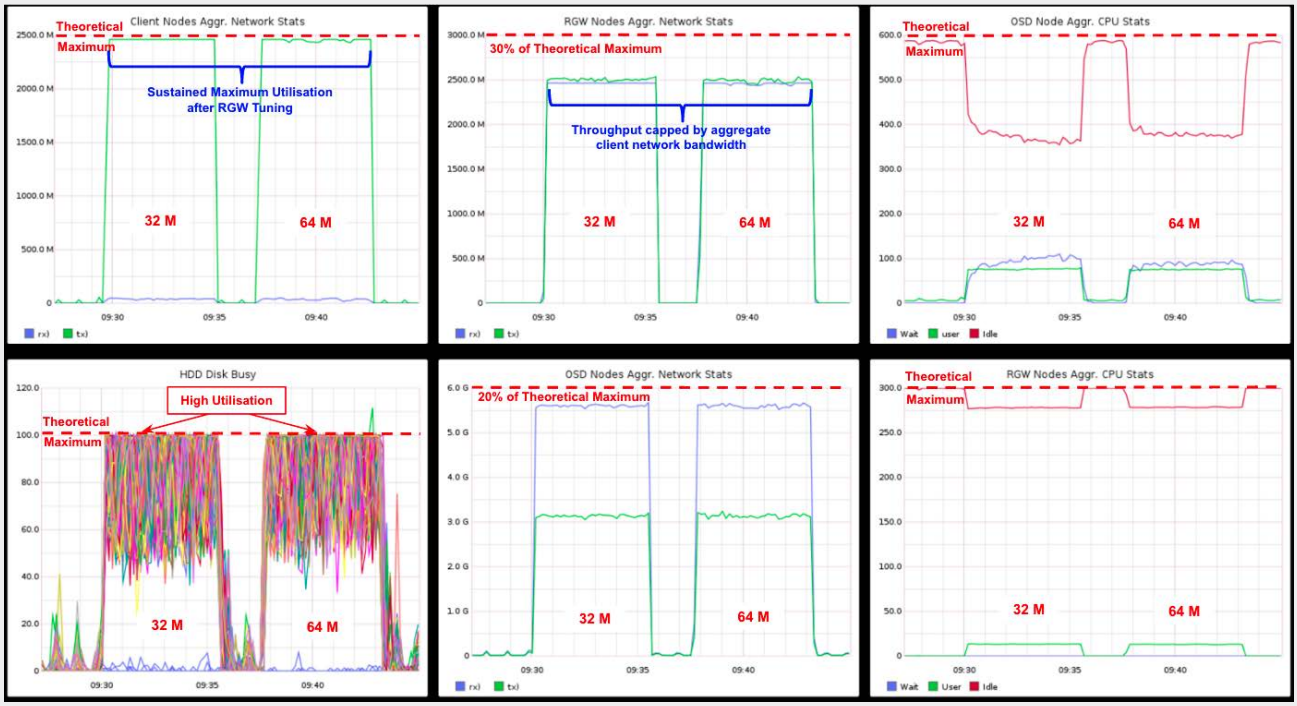

Figure 16. 2-RGW /2-client test, 32MB and 64MB object write, with the tuned RGW configuration

STANDARD VERSUS HIGH-DENSITY STORAGE SERVERS

For large-object write workloads, high-density servers offer better price-performance when compared with standard-density servers as represented in Figure 17. For large-object read workloads, price/performance data was inconclusive, due to a lack of additional RGW hosts in the test configuration required to saturate cluster resources. Since testing was unable to saturate the Ceph cluster for large-object read workloads, engineers could not draw definitive price/performance assumptions to share.

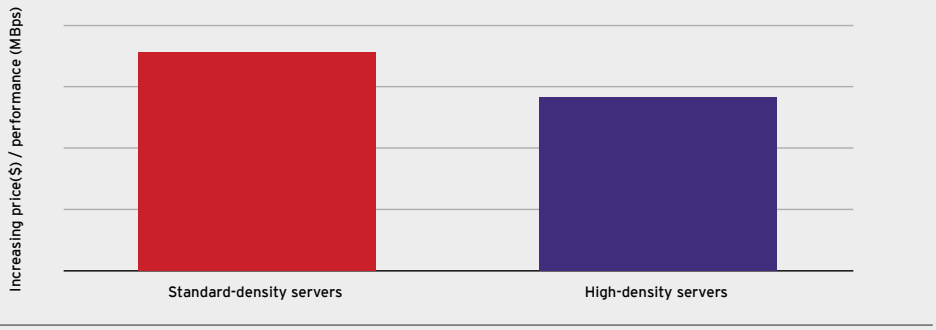

Figure 17. High-density servers offer advantages in price/performance for large-object write-intensive workloads (32MB write price/performance; lower is better).

OPTIMIZING FOR SMALL-OBJECT OPERATIONS

Red Hat tested a variety of configurations, index placement, and client worker counts in order to maximize operations per second for a six-node Ceph cluster using small-object workloads. In addition, the Ceph cluster was built first with high-density servers (6x nodes * 35x HDD = 210 OSDs), and then with standard-density servers (6x nodes * 12x HDD = 72 OSDs) to evaluate for ideal server density.

SMALL-OBJECT HTTP GET WORKLOAD

Small-object HTTP GET (read) workloads scaled super-linearly when incrementing the number of RGW hosts together with load-generating clients. During HTTP GET workloads on both standard-density and high-density configurations, no bottleneck was observed on the Ceph cluster itself. Performance was limited only by the number of load-generating clients available in the test environment. Subsystem testing verified that the Ceph cluster could have delivered more performance for HTTP GET workloads if more client load was applied.

Tuning: Bucket index configuration

During small-object read workload tests, engineers observed a significant percentage of disk activity attributed to bucket index lookups. In order to isolate the effect of the bucket index, a variety of different configurations were tested, including:

- Bucket indices on HDDs (default).
- Bucket indices on flash media (NVMe).
- Indexless buckets.
- Bucket indices on disks accelerated by SSDs with Intel CAS[5].

As shown in Figure 18, the best observed read operations per second (OPS) performance on a standard-density cluster was 8,900 read operations with a bucket index pool configured on NVMe.

5 To learn more about Intel CAS refer to Appendix D: Intel Cache Acceleration Software (Intel CAS).

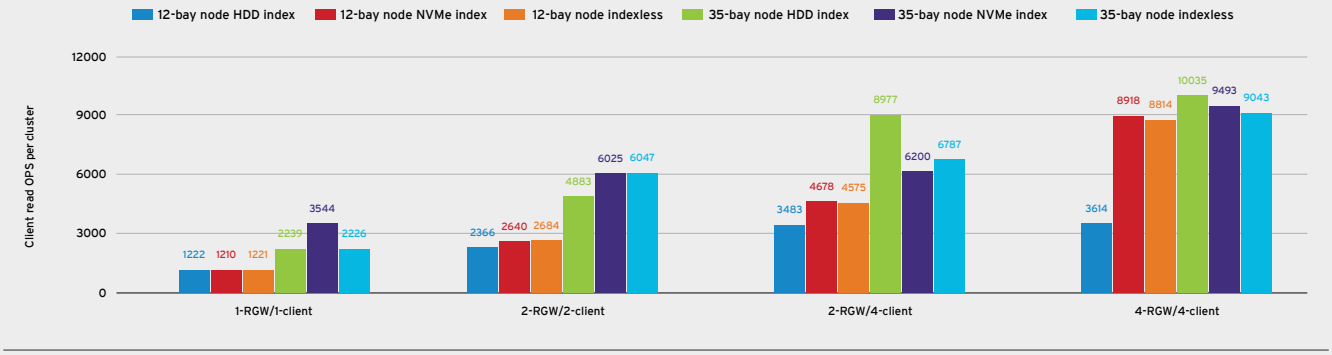

Figure 18. Aggregate small-object read (HTTP GET) performance per cluster across a variety of index configurations on standard-density versus high-density servers (64K object size; higher is better)

A significant percentage of disk activity during read-heavy, small-object workloads is attributed to object lookups in the local filesystems of the OSD hosts. The standard-density and high-density clusters had the same amount of memory per OSD host. With fewer OSDs per host, however, the standard-density servers had more memory available for slab and page cache. As such, the standard-density configuration lent itself to faster object location and delivered better aggregate read performance per OSD as shown in Figure 19.

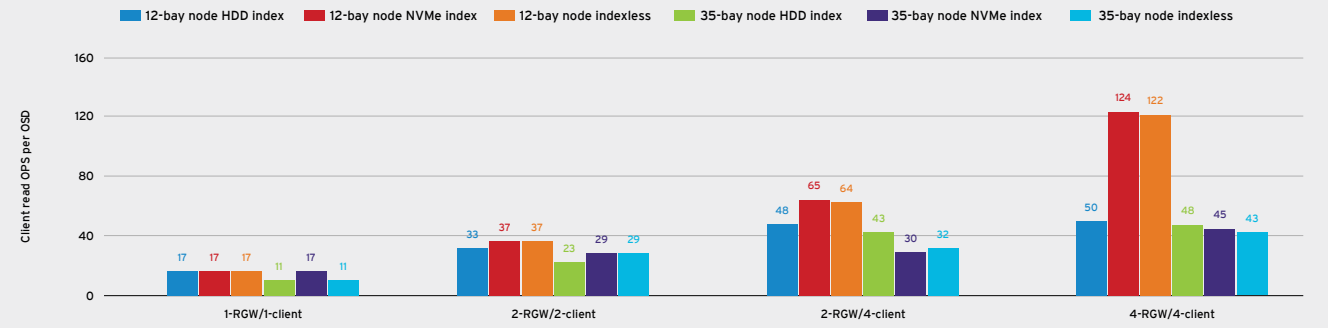

Figure 19. Aggregate small-object read (HTTP GET) performance per OSD across a variety of index configurations on standard-density versus high-density servers (64KB object size; higher is better)

If individual OSD hosts have a large amount of free memory, it can help read-heavy, small-object workloads deliver better performance (if the workload fits inside the page cache). In Red Hat testing, high-density nodes had more OSDs per host and thus a lower RAM:HDD ratio. For high-density servers, read-heavy small object workload with the bucket index on NVMe together with Intel CAS delivers optimal performance as compared to default index configuration (Figure 20). In this configuration, Intel CAS caches the most frequently accessed filesystem metadata on flash storage, which improves read performance materially. Importantly, Intel CAS was configured to cache only metadata, and object data was not cached at any layer during the tests.

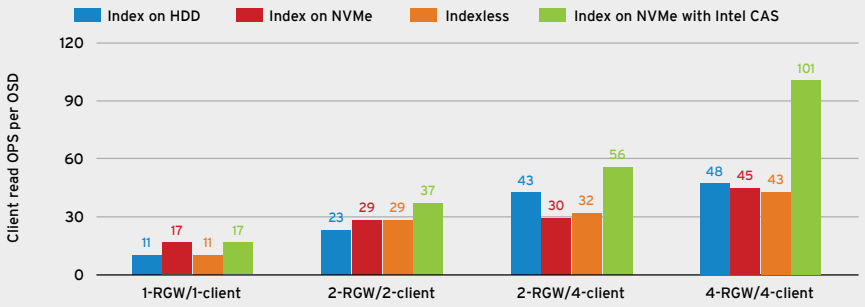

Figure 20. Aggregate small-object read (HTTP GET) performance (normalized per OSD) improves dramatically with Intel CAS on high-density servers (64KB object size).

Subsystem-level testing

Subsystem-level telemetry verified that read performance was limited only by the number of load generating clients available in the test environment. Standard-density servers show no saturation in either HDD or NVMe read IOPS (Figure 21).

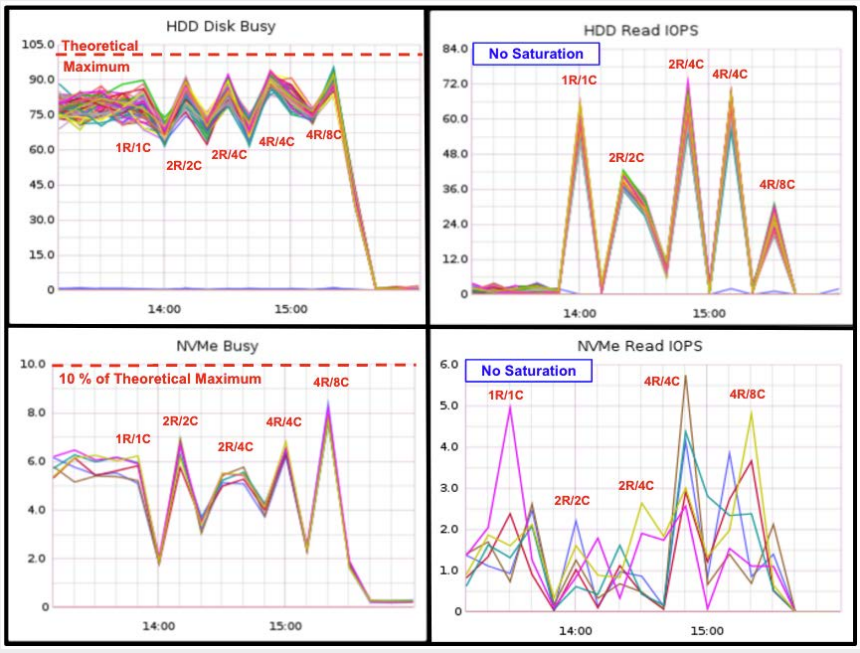

Figure 21. No read saturation was seen in HDDs or NVMes for standard-density servers (#R/#C indicates number of RGW per number of clients).

Figure 22 shows subsystem-level testing for high-density servers, indicating better NVMe usage with the application of Intel CAS software.

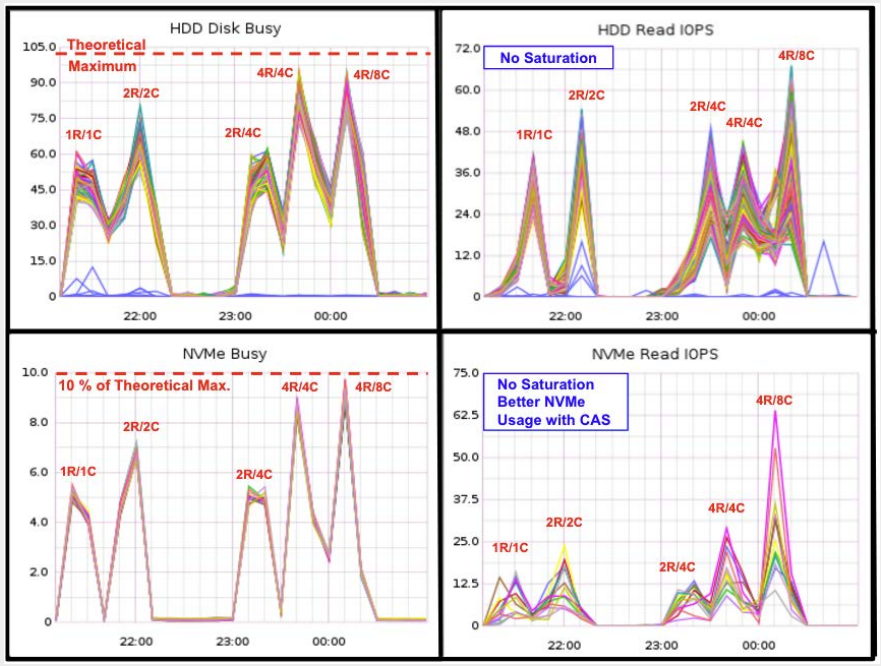

Figure 22. Using Intel CAS provided better NVMe usage for high-density servers with less memory per OSD (#R/#C indicates number of RGW per number of clients).

SMALL-OBJECT HTTP PUT WORKLOAD

In contrast to reads, cluster write IOPS scaled sub-linearly when incrementing the number of RGW hosts, until disks became saturated on the OSD hosts. Figure 23 illustrates small-object write performance normalized per OSD. As shown, both standard-density and high-density server configurations encountered disk saturation and performance and did not scale even after incrementing RGW hosts and load-generating client hosts. Overall, the results indicate that small-object write OPS would have continued to increase if more OSD hosts had been added to the Ceph cluster.

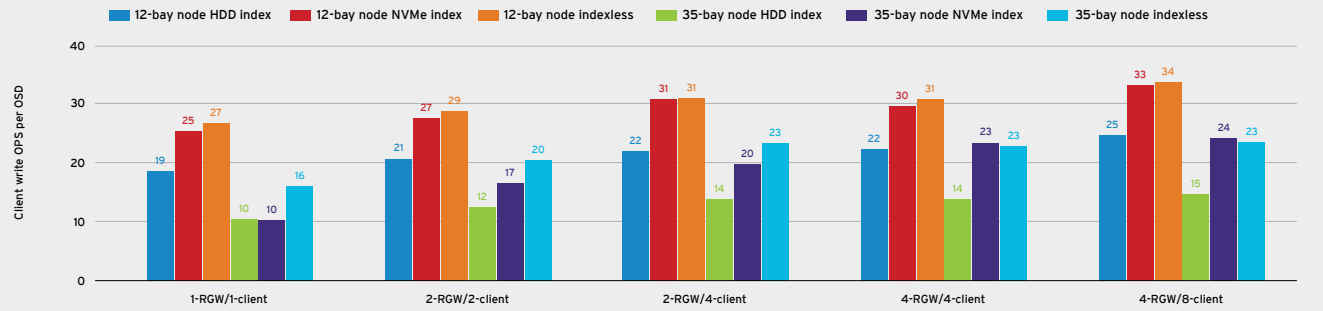

Figure 23. Small-object write (PUT) performance normalized per OSD, demonstrating disk write saturation (64KB object size, standard-density versus high-density servers).

Subsystem resource utilization (Figure 25 and 26) provides evidence that during small-object write workload tests on both standard-density and high-density configurations, cluster disks were saturated. For a detailed view of subsystem utilization refer to Figures 37 and 38 (Appendix B).

Tuning: Moving bucket indices to flash media

During small-object write workload tests, engineers observed that a significant percentage of disk activity was attributed to updating the bucket indices corresponding to objects written. In order to better understand the effect of the bucket index on performance, Red Hat tested a variety of different configurations:[6]

- Bucket indices on HDDs (default).
- Bucket indices on flash media (NVMes).
- Indexless buckets.

Moving bucket indexes to faster media, or avoiding indexes altogether, both removes load from disk-based OSDs and shrinks a critical section in the workload that cannot otherwise be accelerated by parallelization. As represented by Figure 24, optimal performance was achieved by placing bucket index metadata (RGW bucket index pool) on flash media. With this configuration, write performance exceeded 5000 OPS on the high-density cluster with four RGW hosts. The unindexed (indexless) bucket configuration produced similar performance, but at the cost of eliminating bucket-listing capability. Generally, each RGW host sending 64KB write requests at 10GbE line speed was able to saturate the disks representing 50 OSDs in the erasure-coded pool.

6 *To understand the mechanics of RGW write operation and bucket index configuration, refer to Appendix C: RGW object writes and bucket index configuration.*

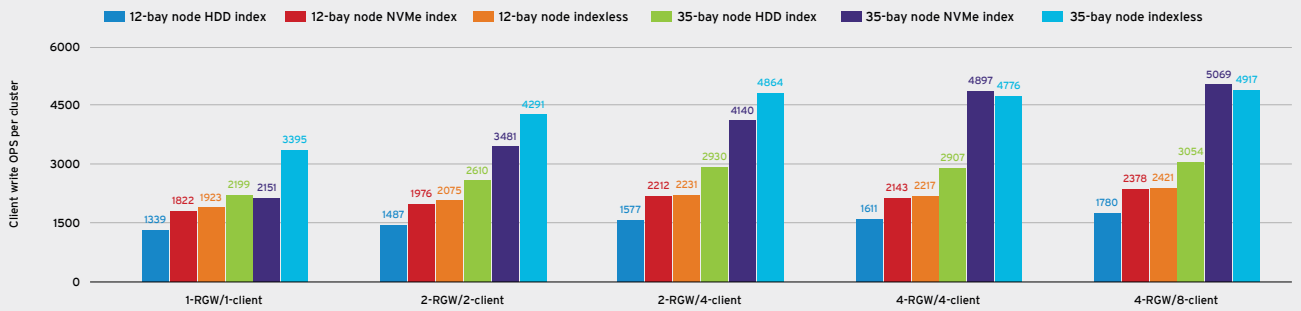

Figure 24. Small-object write (HTTP PUT) performance was greatly accelerated by placing bucket indices on NVMe (64KB object size, aggregate write performance per cluster).

Subsystem-level testing

Subsystem resource telemetry provided further evidence that cluster disks were saturated during small object write workload test on both standard-density and high-density configurations. Overall the results indicate that small-object write OPS would have continued to increase had more OSD hosts been added to the Ceph cluster. Figure 25 illustrates standard-density server subsystem resource telemetry during the HTTP PUT test. High HDD write OPS and high NVMe aggregate wait times point to high HDD utilization and saturation.

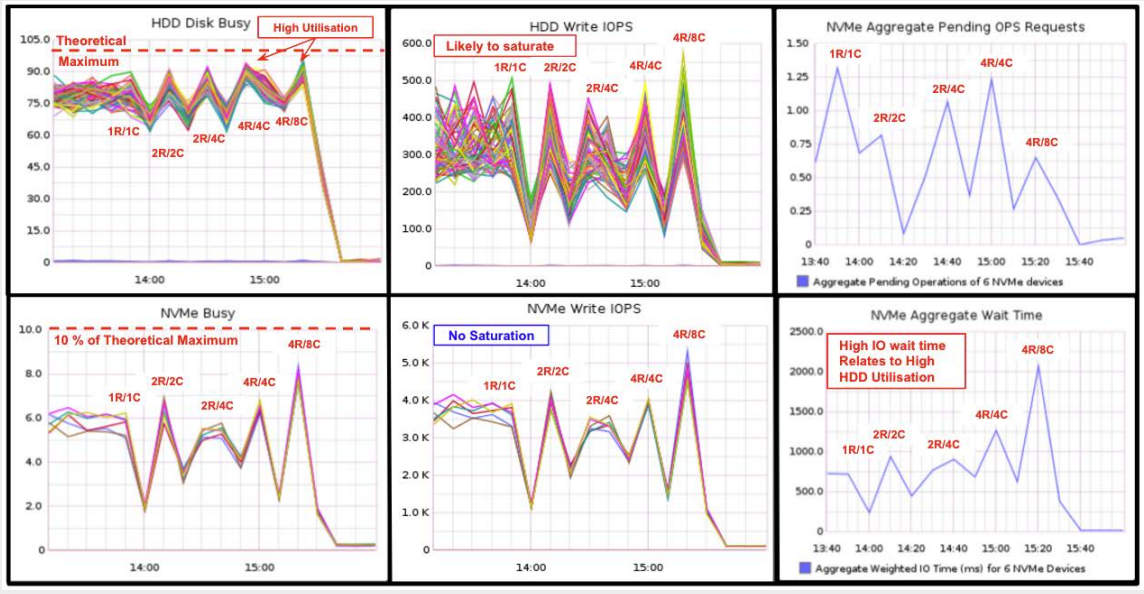

Figure 25. Small-object write (PUT) for standard-density servers was initially limited by HDD saturation, as evidenced by high NVMe aggregate wait time (#R/#C indicates number of RGW per number of clients).

Figure 26 shows high-density server subsystem resource telemetry during small-object HTTP PUT tests. Again, high HDD write IOPS coupled with high aggregate NVMe wait times demonstrated HDD saturation.

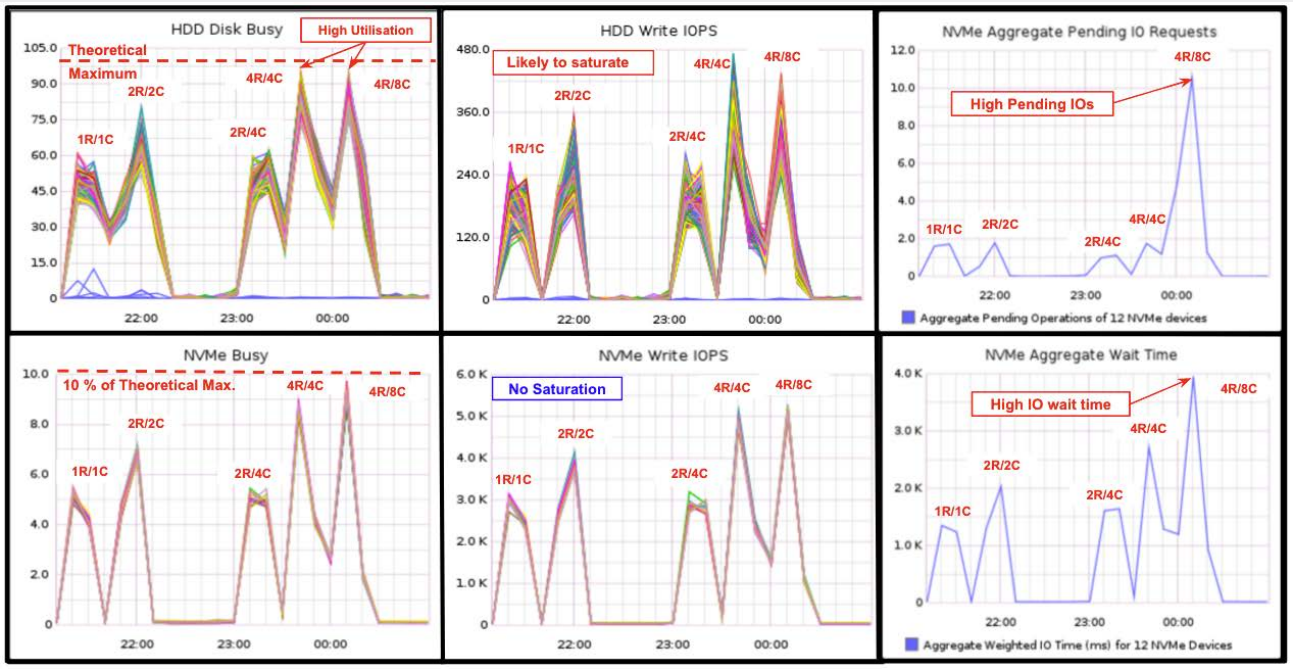

Figure 26. Small-object write (PUT) for high-density servers was likewise initially limited by HDD saturation, as evidenced by high NVMe aggregate wait time.

STANDARD VERSUS HIGH-DENSITY STORAGE SERVERS

For small-object read workloads, price/performance data was inconclusive, since additional load-generating client nodes would have been required to saturate cluster resources. For the tested small-object write workloads, price/performance for both standard-density and high-density servers were similar, as shown in Figure 27.

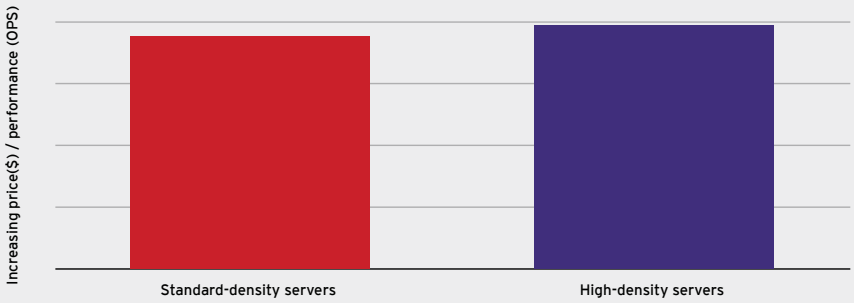

Figure 27. Standard-density and high-density servers demonstrate similar price/performance for small-object writes (64KB objects; smaller is better).

OPTIMIZING FOR HIGHER OBJECT COUNTS

Object storage cluster performance can vary significantly based on total cluster object count. To characterize this important difference, Red Hat conducted hundreds of hours of testing to fill and test the cluster with over 100 million objects. Testing then recorded the highest measured throughput with these higher object counts.

TUNING: OBJECT STORAGE USING METADATA CACHING

Tests were conducted on high-density servers with four RGW hosts, eight load-generating clients, and a small-object size payload of 64KB. In order to increase cluster fill level and minimize the metadata that could fit in memory, tests were executed with long runtime cycles of 49 hours. For this testing, the bucket index pool remained in the default configuration on the on HDDs. Various configurations of Ceph OSD filestore were then tested, evaluating both the split of OSD PG directories into subdirectories[7] and testing both with and without Intel CAS software.[8]

- **Default Ceph OSD filestore.** This configuration featured out-of-the-box default Ceph settings without any tuning applied. It should be noted that default values for `filestore split multiple` and `filestore merge threshold` were 2 and 10 respectively. These parameters play a significant role in Ceph performance.
- **Tuned Ceph OSD filestore.** In this test configuration, `filestore split multiple` and `filestore merge threshold` values have been tuned to 16 and 48 respectively. This change resulted in significant performance improvement as compared to the default configuration.
- **Default Ceph OSD filestore + Intel CAS (for metadata caching).** In this configuration, an additional metadata caching layer was introduced in front of each OSD using Intel CAS software on NVMe flash media.
- **Tuned Ceph OSD filestore + Intel CAS (for metadata caching).** In this configuration, tuned Ceph OSD filestore settings were used as above. An additional metadata caching layer was introduced in front of each OSD using Intel CAS software on NVMe flash media.

7 *For more information on how these parameters affect Ceph performance, refer to Appendix E.*
8 *For more details on Intel CAS and its configuration, refer to Appendix D.*

SMALL-OBJECT HTTP GET WORKLOAD

As represented in Figures 28 and 29, performance for the **default Ceph OSD filestore with Intel CAS** configuration was optimal as compared to the other three configurations. Note, however, that even in this optimal configuration, client read OPS declined by approximately 50% as the cluster grew to over 100 million objects (Figure 47, Appendix E). This decline in read OPS was caused due to an increase in time taken for filesystem metadata lookup as the cluster object-count grew. When the cluster held fewer objects, a greater percentage of filesystem metadata was cached by the kernel in memory. However, when the cluster grew to millions of objects, a smaller percentage of metadata was cached. Disks were then forced to perform I/O operations specifically for metadata lookups, adding additional disk seeks and resulting in lower read OPS.

To reduce disk I/Os caused by filesystem metadata lookups, the optimal test configuration used Intel CAS software to cache XFS filesystem metadata in flash media. Again, Intel CAS was used only to cache metadata while workload data itself was not cached by Intel CAS. With approximately 60 million total objects in the cluster, the Intel CAS configuration improved small-object read OPS by over approximately 550% when compared with the default Ceph configuration (13590 versus 2100 OPS) as shown in Figure 41 and 47 (Appendix E).

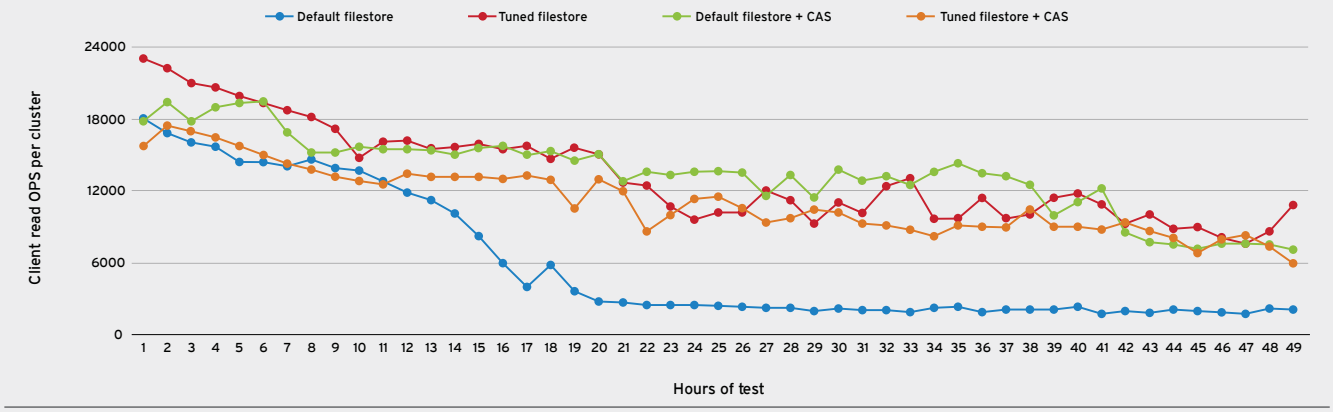

Figure 28. Read OPS per cluster for various filesystem configurations on high-density server-based cluster.

In a separate test, Ceph OSD filestore parameters were tuned for delaying the split of OSD PG directories into subdirectories, using the **tuned Ceph OSD filestore** configuration (without Intel CAS). With approximately 60 million objects in the cluster, read OPS improved by approximately 400% compared to the default Ceph configuration (10200 vs. 2100 IOPS), as shown in Figure 41 and 44 (Appendix E). However, the long tail read latency increased by over two-fold with the tuned Ceph OSD filestore configuration. It is worth noting that tuning OSD filestore split/merge parameters only postpones OSD PG directory splits until they grow larger. Once the directory splits do occur, read IOPS performance is expected to drop to levels similar to the default Ceph configuration.

Figure 29 represents long tail read latency (99th percentile read latency) comparison for first 10 hours of testing, across different tested configurations. As discussed, long tail read latency has increased after tuning the OSD filestore. These tunings postpone OSD PG directory splits, so each directory now stores a greater number of files as compared to default Ceph OSD filestore configuration. The larger number of files per PG directory may have lead to higher read latency.

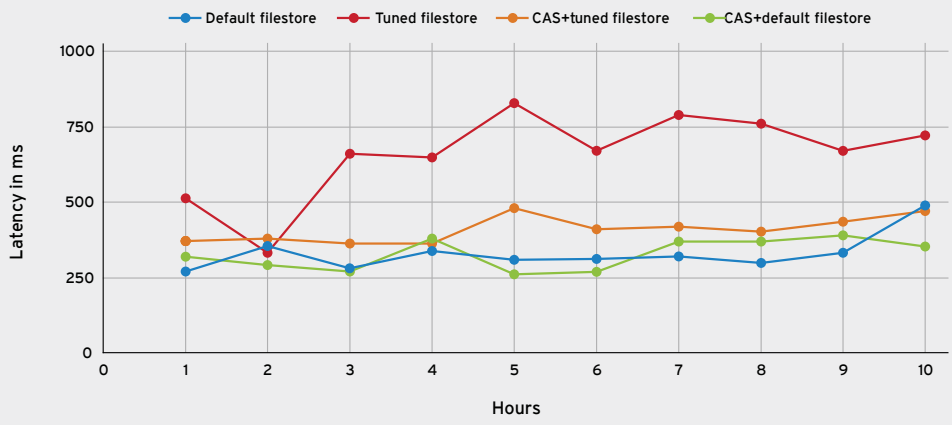

Figure 29. 99th percentile read latency for the first 10 hours of testing, high-density cluster, 64KB objects

In conclusion, the combination of the Ceph OSD default filestore settings with metadata caching provided by Intel CAS resulted in an optimal configuration for use cases storing millions of objects where read OPS and latency are key concerns. As illustrated by the subsystem telemetry tests shown in Figure 30, the default configuration, together with Intel CAS, enabled better utilization of available NVMe hardware during GET operations.

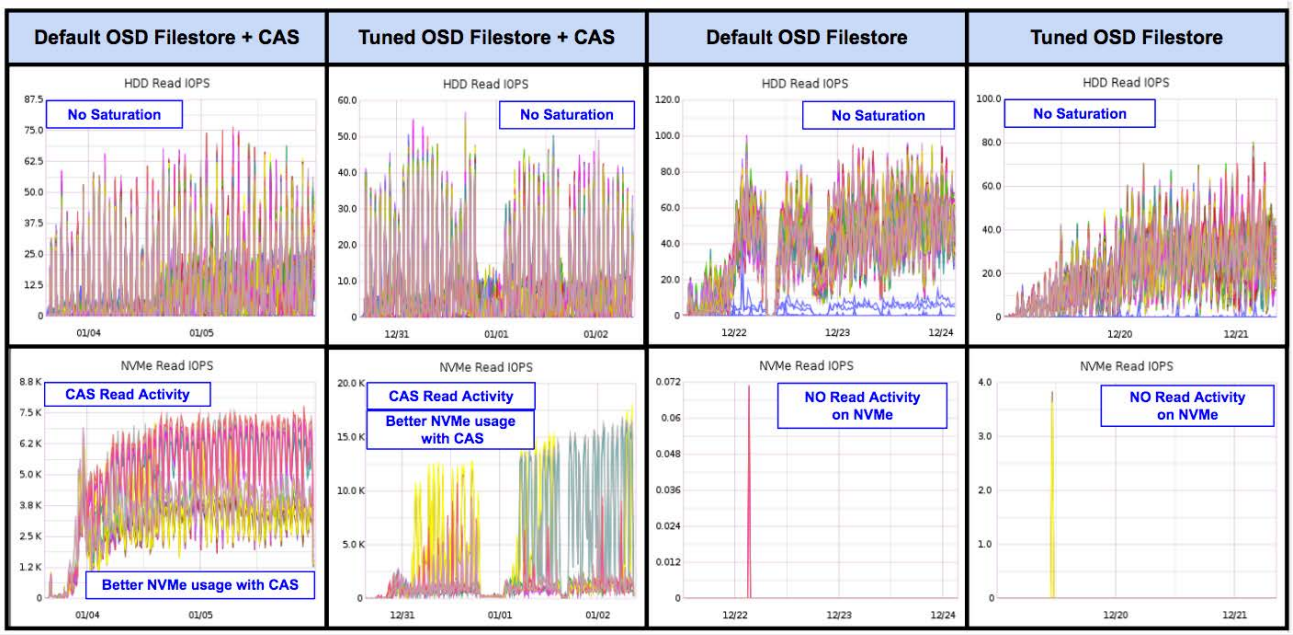

Figure 30. HDD/NVMe read OPS during different configurations of the OSD filestore, with and without Intel CAS

SMALL-OBJECT HTTP PUT WORKLOAD

As with small-object read workloads, the optimal write performance was achieved with default Ceph OSD filestore settings with Intel CAS, represented in Figure 31 and Figure 46 (Appendix E). In this optimal configuration, client write OPS held steady as the cluster grew to 130 million objects. The optimal test configuration used Intel CAS for caching XFS filesystem metadata in flash media. Intel CAS uses a classification-based algorithm that intelligently prioritizes I/O. Called I/O classification, this unique capability allows further performance optimization based on I/O types (data or metadata), size, and additional parameters.

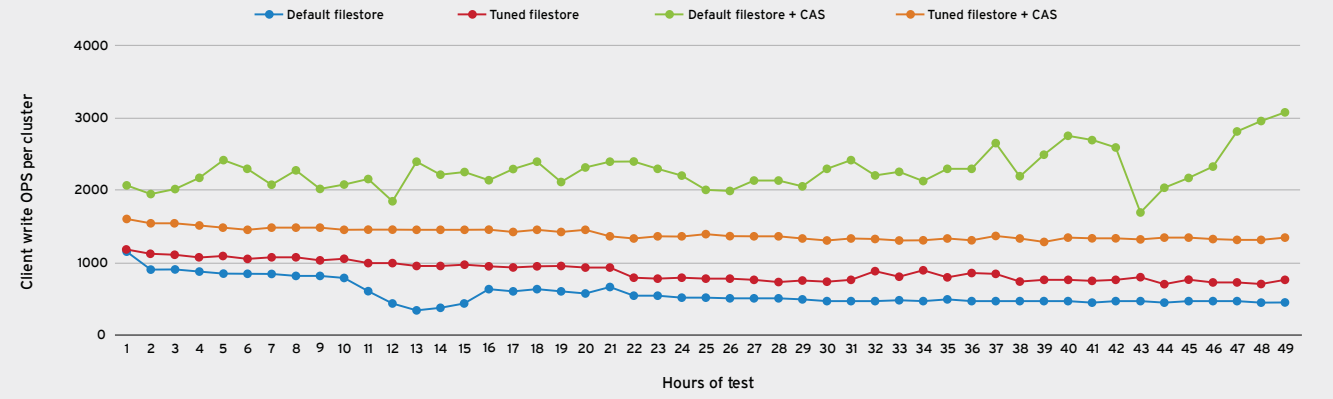

Figure 31. Write OPS per cluster for various filesystem configurations on high-density servers, 64KB objects

As represented in Figure 40 (Appendix E), testing with the default Ceph OSD filestore settings showed an unusual write performance drop during test hours 11-15. This anomaly is presumably associated with Ceph OSD PG directory splitting. The Ceph OSD filestore backend uses the XFS filesystem to store client objects as files inside PG directories. By default, the maximum numbers of files that can be stored in a PG subdirectory is 320. As soon as the file count increases beyond that point, the Ceph OSD splits PG directories into sub-directories.

The filestore tunable parameters that control PG split and merge behavior are **filestore merge threshold** and **filestore split multiple** with default values of 10 and 2 respectively. During test hours 11-15, PG directory splitting triggered additional disk I/O, resulting in a performance drop. Results also showed that write performance decreased over time with default Ceph OSD filestore settings as the cluster object count increased. This continuous performance drop is associated with a higher amount of XFS metadata in the form of filesystem Extended Attributes (XATTRs), *inodes*, and *dentries*. Ceph performance relies heavily upon the stability and performance of the underlying filesystem. By caching the XFS file system metadata on faster media, Intel CAS effectively masks the filesystem limitations and dramatically improves write performance.

With approximately 60 million objects in the cluster, the Intel CAS configuration improved client write operations per second by over 400% compared to the default configuration (2400 versus 475 operations per second). Latency of client write OPS held steady as the cluster grew to 100+ million objects, and was 100% better (lower) compared to the default Ceph configuration (Figure 32).

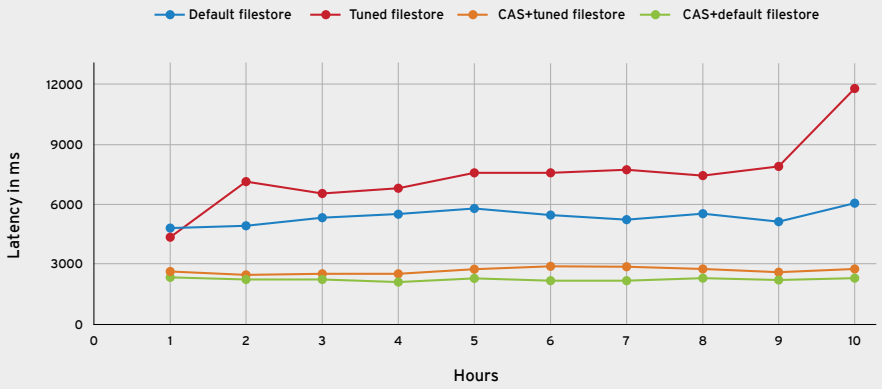

Figure 32. 99th percentile write latency during the first 10 hours of testing, high-density servers, 64K object size

Detailed subsystem telemetry has been captured while different test configurations were benchmarked. As represented in Figure 33, Figure 34, and Figure 50 (Appendix E) with default Ceph OSD filestore setting together with Intel CAS, HDD utilization dropped by approximately 25% and NVMe hardware utilization improved by approximately 60%. Both of these factors contribute to sustained write performance while increasing object count in the cluster, at lower tail latency.

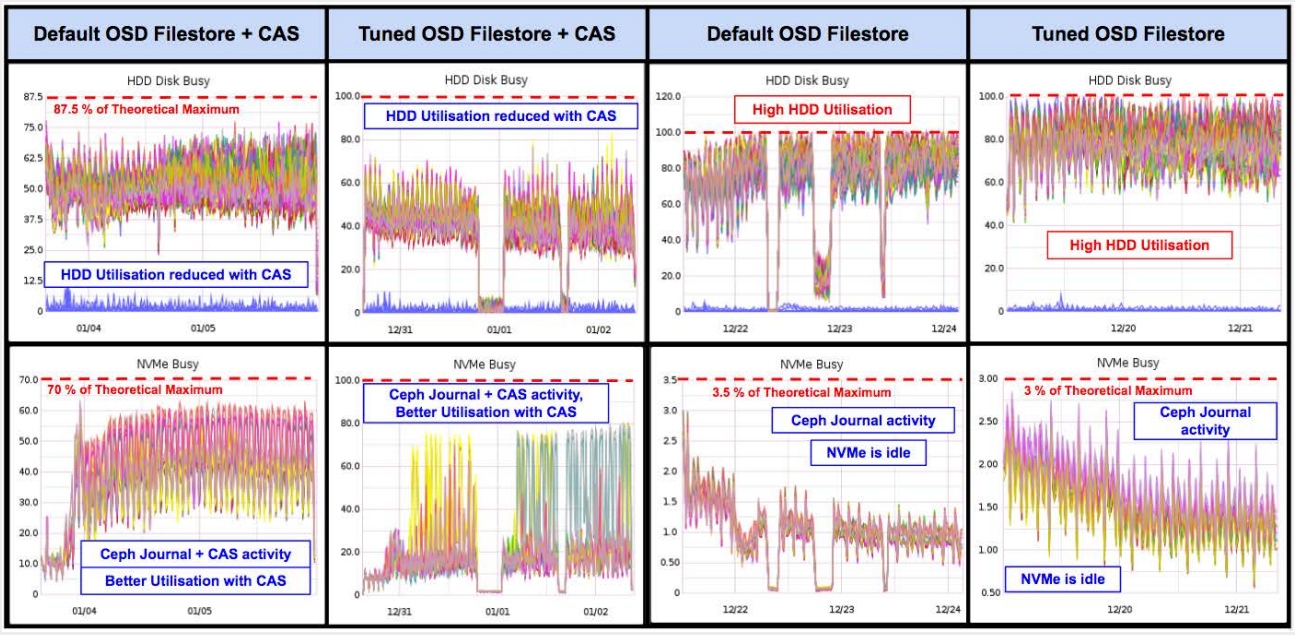

Figure 33. HDD and NVMe utilization using different filestore configurations, with and without Intel CAS

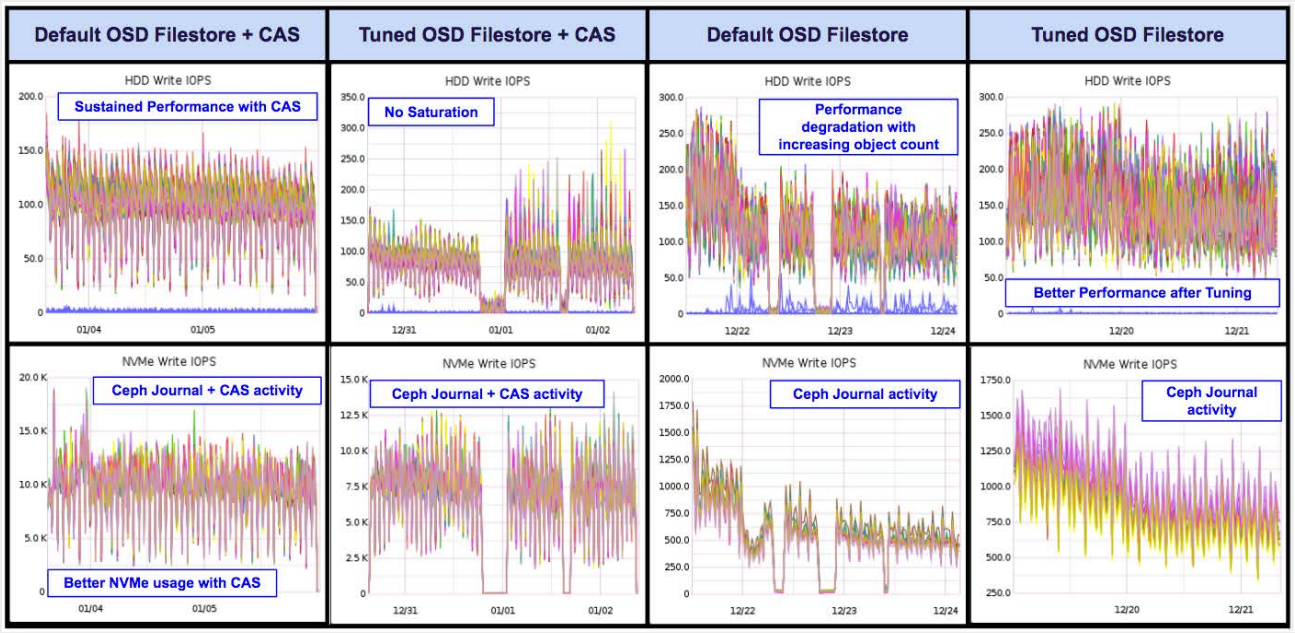

Figure 34. HDD and NVMe write operations per second using different filestore configurations, with and without Intel CAS.

In a separate test on the tuned Ceph OSD filestore (without Intel CAS), the Ceph OSD filestore parameters *filestore merge threshold* and *filestore split multiple* were tuned for delaying the split of OSD PG directories into subdirectories. As represented in Figure 31 and Figure 43 (Appendix E), with 60 million objects in the cluster, tuning these filestore parameters improved write OPS by over 80% when compared to the default Ceph configuration (870 versus 475 OPS). Note, however, that tuning these OSD filestore parameters resulted in a significant write latency increase of 50-100% compared to the default Ceph configuration.

Based on results of various configurations tested by Red Hat, it is evident that a configuration blending the default Ceph OSD filestore settings with Intel CAS provides both optimal write OPS performance at 10s of millions of objects, as well as the lowest write latency (compared to other tested configurations). Additionally, if a solution strictly demands that no caching mechanism be used, then some improvement in write OPS performance can be achieved by tuning Ceph OSD filestore split and merge thresholds. However, this approach might increase write latency.

APPENDIX A: BASELINE TESTING

Before attempting benchmark scenarios that utilize higher-layer Ceph protocols, it is helpful to establish a known performance baseline of all relevant subsystems. In advance of Ceph object testing, Red Hat conducted:

- Single-node disk baseline testing.
- Full mesh network baseline testing.
- Red Hat Ceph Storage native testing (RADOS bench).

Table 5 illustrates the baseline tests that were a part of this testing.

TABLE 5. BASELINE TESTING TOOLS AND METHODOLOGIES

SUBSYSTEM	BENCHMARKING TOOL	METHODOLOGY
HDD	FIO 2.11-12	4KB random read/write and 4MB sequential read/write on top of XFS
NVME	FIO 2.11-12	4KB random read/write and 4MB sequential read/write on top of XFS
NETWORK	iPerf3	Multiple TCP-stream benchmark, all-to-all
CEPH BASELINE	Ceph Benchmark Tool (CBT)	4MB sequential read/write

SINGLE-NODE DISK BASELINE

As a part of testing, disk storage performance was measured thoroughly in order to determine the maximum performance of each individual component. The tests were run on all devices in the system in parallel to ensure that the backplanes, I/O expanders, and PCI bridges do not pose an I/O bottleneck.

In order to run the tests as closely as possible to the way Ceph utilizes the systems, the tests were run on files in an XFS file system, on the block devices under test, using the formatting options from the ceph.conf file: -f –i size=2048 (Appendix F). Before each benchmark was run, all drives were filled with random data to prewarm the devices and ensure steady-state performance. Write benchmarks were executed before read benchmarks to compensate for different NAND behavior during reads. The data is reported on a per-device average and device-level results are listed in Table 6.

TABLE 6. DEVICE-LEVEL BASELINE TEST RESULTS

NODE TYPE	HDD	NVME
SEQUENTIAL READ (4MB)	187MB/s	2234MB/s
SEQUENTIAL WRITE (4MB)	169MB/s	1684MB/s
RANDOM READ (4KB)	252 IOPS	329,695 IOPS
RANDOM WRITE (4KB)	227 IOPS	145,470 IOPS

FULL MESH NETWORK BASELINE

Network performance measurements were taken by running point-to-point connection tests following a fully meshed approach (Table 7). In other words, each server's connection was tested toward each available other server endpoint. The tests were run individually, as well as in parallel, to measure individual NIC performance and bisectional bandwidth. The physical line rate was 10000 Mbps between client, RGW, and monitor nodes, and 40000 Mbps between the Ceph OSDs and the RGWs. The measured results were within approximately 1.5% of the expected TCP/IP overhead. The TCP window size was 325Kb as the determined default by the Iperf tool, and MTU was kept at 1500.

TABLE 7. NETWORK BASELINE TEST RESULTS

NODE TYPE	CEPH OSD	CLIENT
CEPH OSD	39.6 Gbps	9.88 Gbps
RGW	39.5 Gbps	9.88 Gbps
CEPH MON	9.75 Gbps	9.88 Gbps

CEPH NATIVE PERFORMANCE BASELINE

To record native Ceph cluster performance, Red Hat engineers used the Ceph Benchmarking Tool (CBT), an open source tool for automation of Ceph cluster benchmarks. CBT is written in Python and takes a modular approach to Ceph benchmarking. For more information on CBT, visit github.com/ceph/cbt.

The Ceph cluster was deployed using ceph-ansible. Ansible creates a Ceph cluster with default settings and Ceph native performance. As such, baseline testing was done with default Ceph configurations and no special tunings applied. CBT was used to gauge Ceph cluster native performance. Sixteen iterations of each benchmark scenario were run. The first iteration executed the benchmark from a single client, the second from two clients, the third from three clients in parallel, and so on.

For the Ceph native performance test, the following CBT configuration was used:

- Workload: Sequential write tests followed by sequential read tests
- Block Size: 4MB
- Runtime: 300 seconds
- Concurrent threads per client: 128
- RADOS bench instance per client: 1
- Pool data protection: EC 4+2 jerasure
- Total number of clients: 16

Another testing goal was to find the aggregate throughput high-water mark for the cluster. This watermark is the point beyond which performance either ceases to increase or drops. As shown in Figure 35, six storage nodes and 210 OSDs can deliver approximately 11900 MB/s of top-line 4MB aggregate read performance with 16 clients.

Figure 35. RADOS bench cluster baseline, erasure-coded (4,2) pool, high-density storage servers

With increasing client load, write performance scaled linearly up to six clients, and performance remained steady with any further client load on the cluster. The aggregate write throughput did not improve even after increasing client load indicated that had become hardware bound. As such, approximately 6800 MB/s is the high-water mark for 4MB write performance for this cluster. Adding more storage nodes to this cluster will definitely increase the high-water mark for both reads and writes.

Just as it is important for performance to scale out, individual servers within the cluster must also be able to scale up as storage devices are added. Figure 36 shows 4MB RADOS bench sequential reads and writes, demonstrating that Ceph can achieve the same performance per disk on dense servers as it can on sparse servers. In other words, as more disks per server are deployed, Ceph can achieve consistent performance per disk.

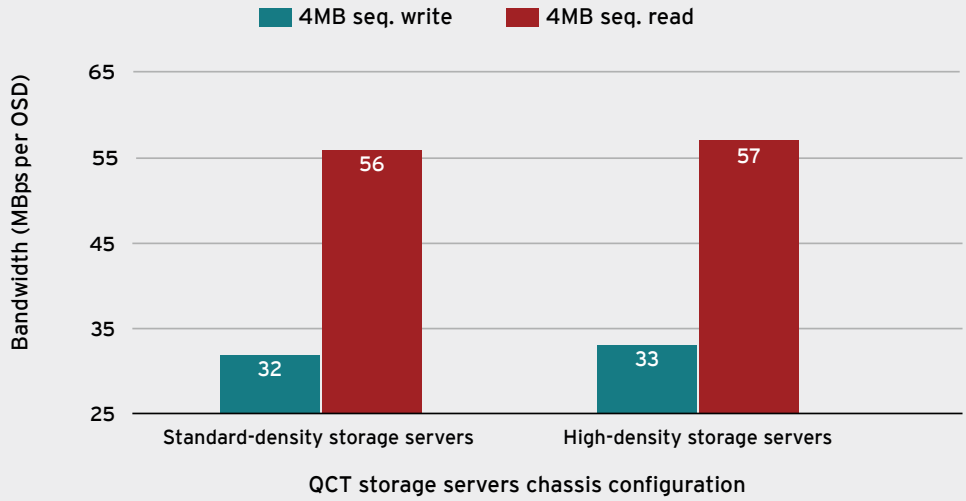

Figure 36. RADOS bench cluster baseline normalized per OSD, showing OSD-level scalability ((EC 4,2) pool)

APPENDIX B: SMALL-OBJECT GRAPHS

Figure 37 and 38 show subsystem telemetry for small-object testing.

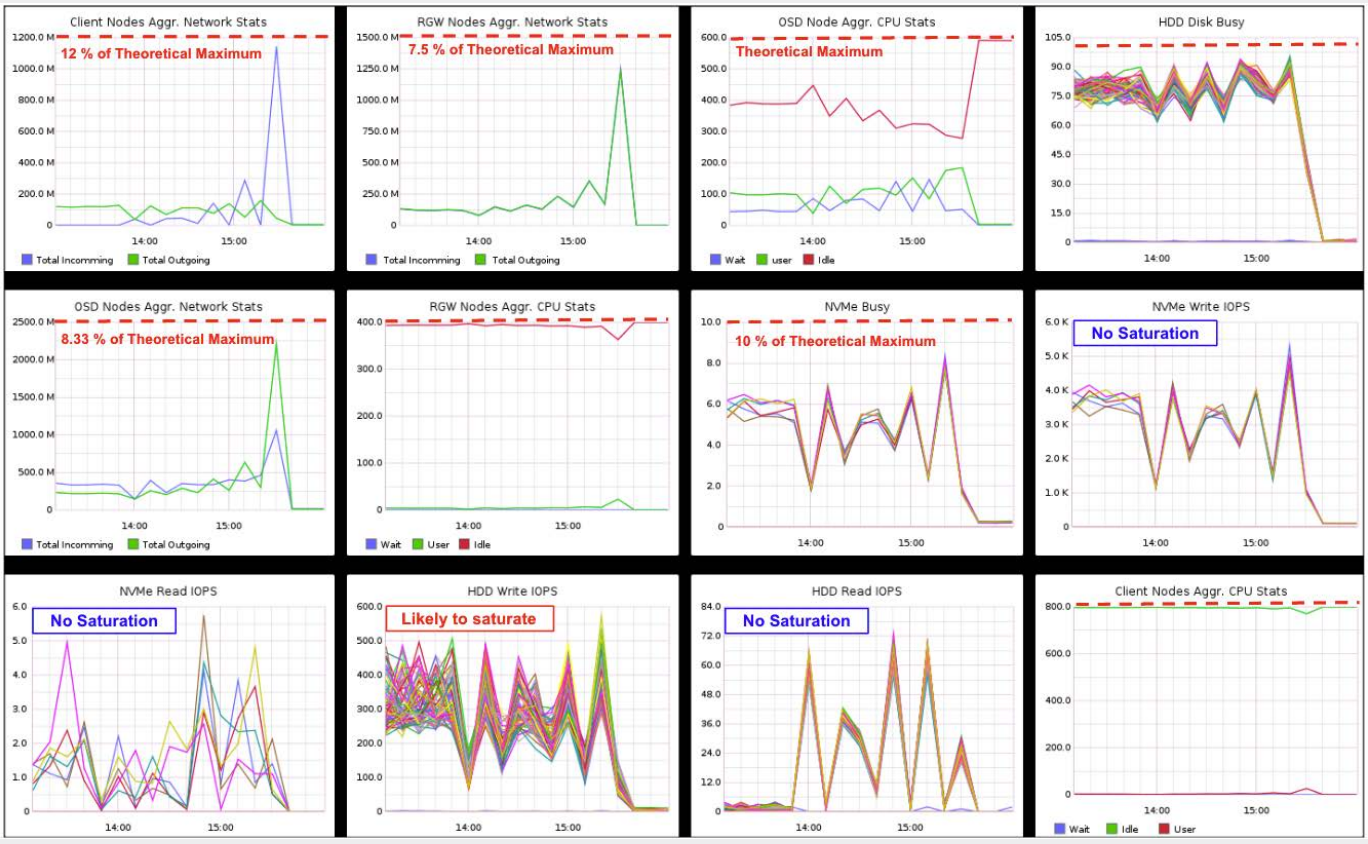

Figure 37. Standard-density servers system performance during small-object HTTP PUT/GET tests

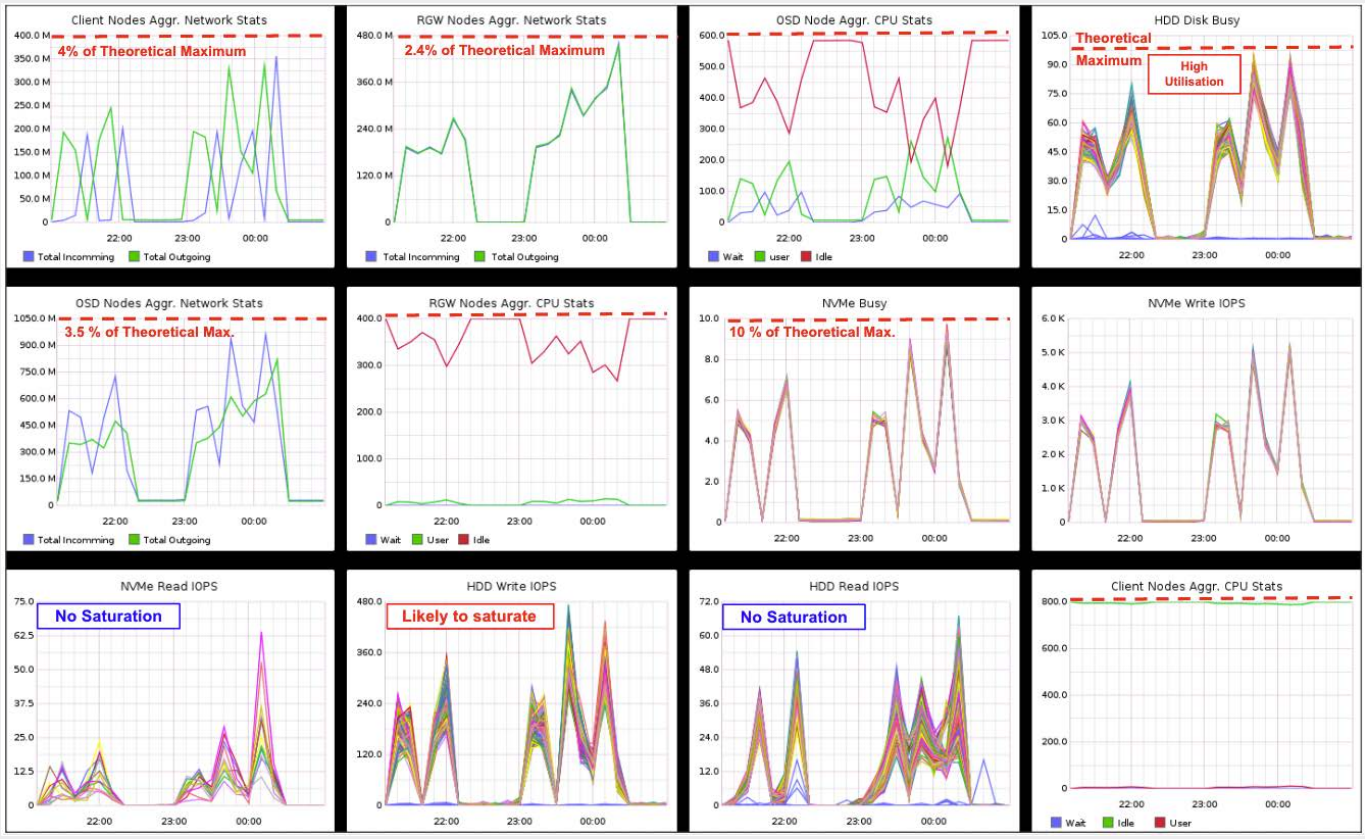

Figure 38. High-density servers system performance during small-object HTTP PUT/GET tests

APPENDIX C: RGW OBJECT WRITES AND BUCKET INDEX CONFIGURATION

To better understand the inner workings of Ceph object storage, it is advantageous to know about RGW write operation sequence. The RGW object body consists of two sections: **HEAD** and **TAIL**. The **HEAD section** consists of the first stripe of the object and the metadata, while the TAIL section consists of subsequent object stripes. When RGW receives write request from the clients, it takes the following actions:

- RGW stripes the object, based on the `rgw_obj_stripe_size` setting of RGW.
- It then divides these stripes into smaller chunks based on `rgw_max_chunk_size` setting of RGW.
- RGW then opens RADOS handles (threads) to write these chunks to Ceph cluster.
- RGW synchronously writes the **HEAD** section containing first chunk of the object to the data pool and, in parallel, performs the first phase of bucket index update by writing bucket index metadata to the index pool.
- Each object is then written in chunks to the data pool. The first chunk written creates the HEAD object, and subsequent chunks (TAIL writes) append to the HEAD object up to the object stripe boundary.
- Finally RGW asynchronously does a second phase bucket index update to record that the write has been completed. The second phase bucket index update again gets written to the index pool.

As such, each RGW object write operation undergoes one HEAD write, a number of TAIL writes, plus two index update write operations. Independent of the object size, the nature of bucket index updates is small-sized IOPS intensive. Especially for small object workloads, improving overall object storage write performance depends on the actual use case, and the following options must be considered:

- **Bucket index pool on flash media**. Since flash media is significantly faster than spinning media, using flash media for storing the bucket index pool can dramatically boost overall object write OPS. This configuration will also remove the index update load from the spinning media. Because the bucket index pool does not store actual object data, it does not require a large amount of storage capacity. As such, making use of excess capacity on OSD journal flash devices to store the bucket index pool is a cost-effective way to boost small-object performance. Keep in mind that flash media must be performant enough to be used simultaneously for OSD journals as well as bucket index pools.
- **Indexless buckets:** When the indexless bucket feature is enabled, the only index operation that occurs is the creation of a blank index during bucket creation. No index updates are done after that, and bucket indices never get updated on subsequent object writes to that bucket. This option saves quite a lot of small I/O writes, which can improve object storage performance. One trade-off is that the buckets do not store any metadata about the objects stored into them. As a result, clients cannot list objects. Indexless buckets are a use-case-specific feature. If an application does not require bucket listing or if bucket metadata is maintained at the application tier, indexless buckets can provide a performance improvement.

APPENDIX D: INTEL CACHE ACCELERATION SOFTWARE (INTEL CAS)

Intel CAS increases storage performance via intelligent caching, especially when combined with high-performance Intel SSDs.[9] Unlike inefficient conventional caching techniques that rely solely on data temperature, Intel CAS employs a classification-based solution that intelligently prioritizes I/O. This unique capability allows organizations to further optimize their performance based on I/O types (data versus metadata), size, and additional parameters. Intel CAS is depicted logically in (Figure 39).

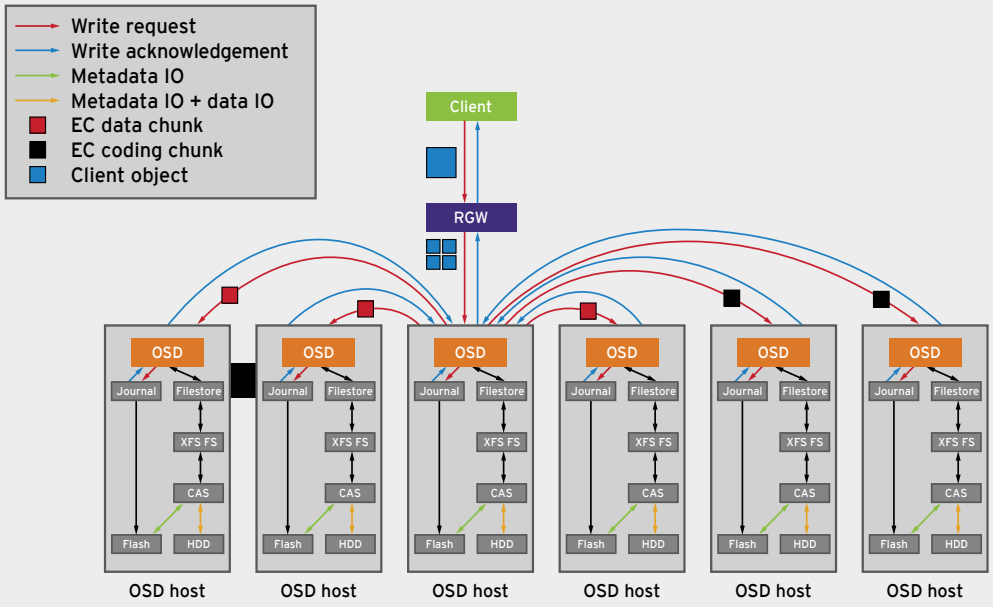

Figure 39. Intel CAS helps accelerate Ceph workloads.

The advantage of Intel's approach is that logical block-level storage volumes can be configured with multiple performance requirements in mind. For example, the file system journal could receive a different class of service than regular file data, allowing workloads to be better tuned for specific applications. Intel and Red Hat have worked with organizations to use Intel CAS I/O classification to address disk seek times on OSD hosts. These efficiencies have resulted in as much as a doubling of cluster throughput while reducing latency by as much as 50%. Efficiencies can be increased incrementally as higher performing Intel NVMe SSDs are used for data caching.

In addition, Intel CAS has several enhancements created to improve deployment and usability for Ceph installations. The software uses a small default memory footprint that can be further reduced by using a feature called selective cache line size, which may provide cost savings benefits especially as servers move to higher densities of storage. Caching mode can be changed in-flight and immediately affects different modes. Other features such as in-flight upgradability allow Intel CAS to be upgraded without stopping applications or rebooting servers, providing further operational efficiencies.

9 *Intel CAS software is licensed on a perpetual basis per SSD and includes one year of support at the time of purchase. For more details on Intel Cache Acceleration Software and a free 120-day trial version, visit intel.com/cas and intel.com/content/www/us/en/solid-state-drives/solid-state-drives-ssd.html.*

APPENDIX E: HIGHER OBJECT-COUNT PERFORMANCE GRAPHS

The sections that follow contain the full data for the higher object-count testing.

DEFAULT CEPH OSD FILESTORE

Figures 40 and 41 display write and read OPS results, respectively, for the default Ceph OSD filestore, as compared to RADOS object count.

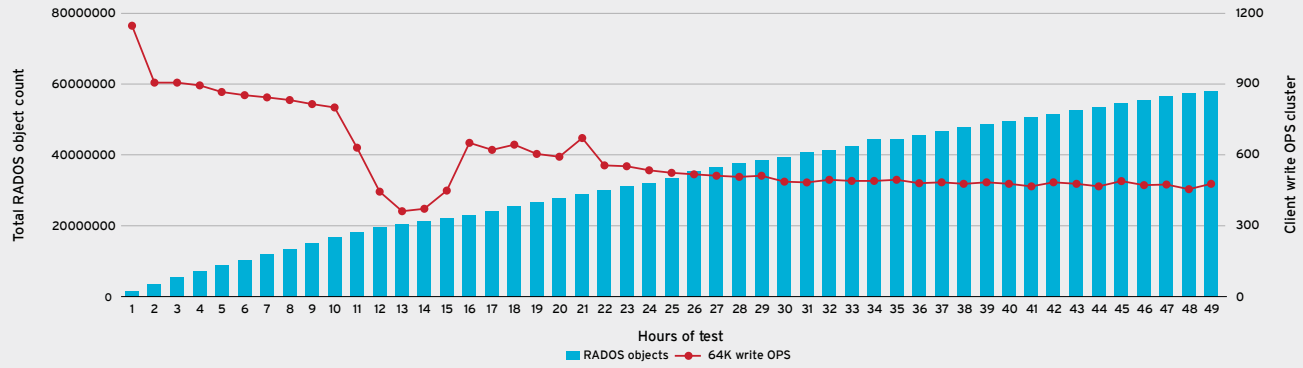

Figure 40. Default Ceph OSD filestore (split:merge 2:10) 64KB object write OPS versus RADOS object count on high-density servers

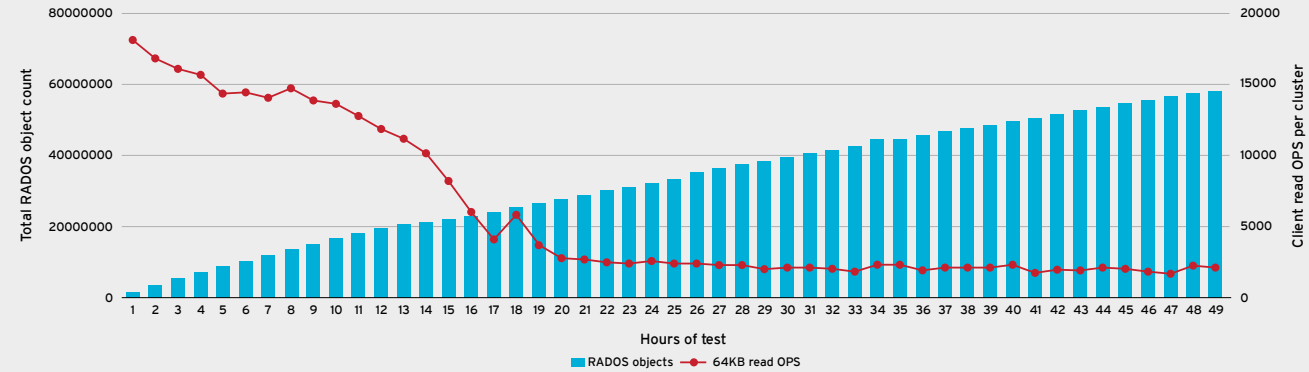

Figure 41. Default Ceph OSD filestore (split:merge 2:10) 64KB object read OPS versus RADOS object count on high-density servers

Figure 42 shows subsystem telemetry during the defualt Ceph OSD filestore testing.

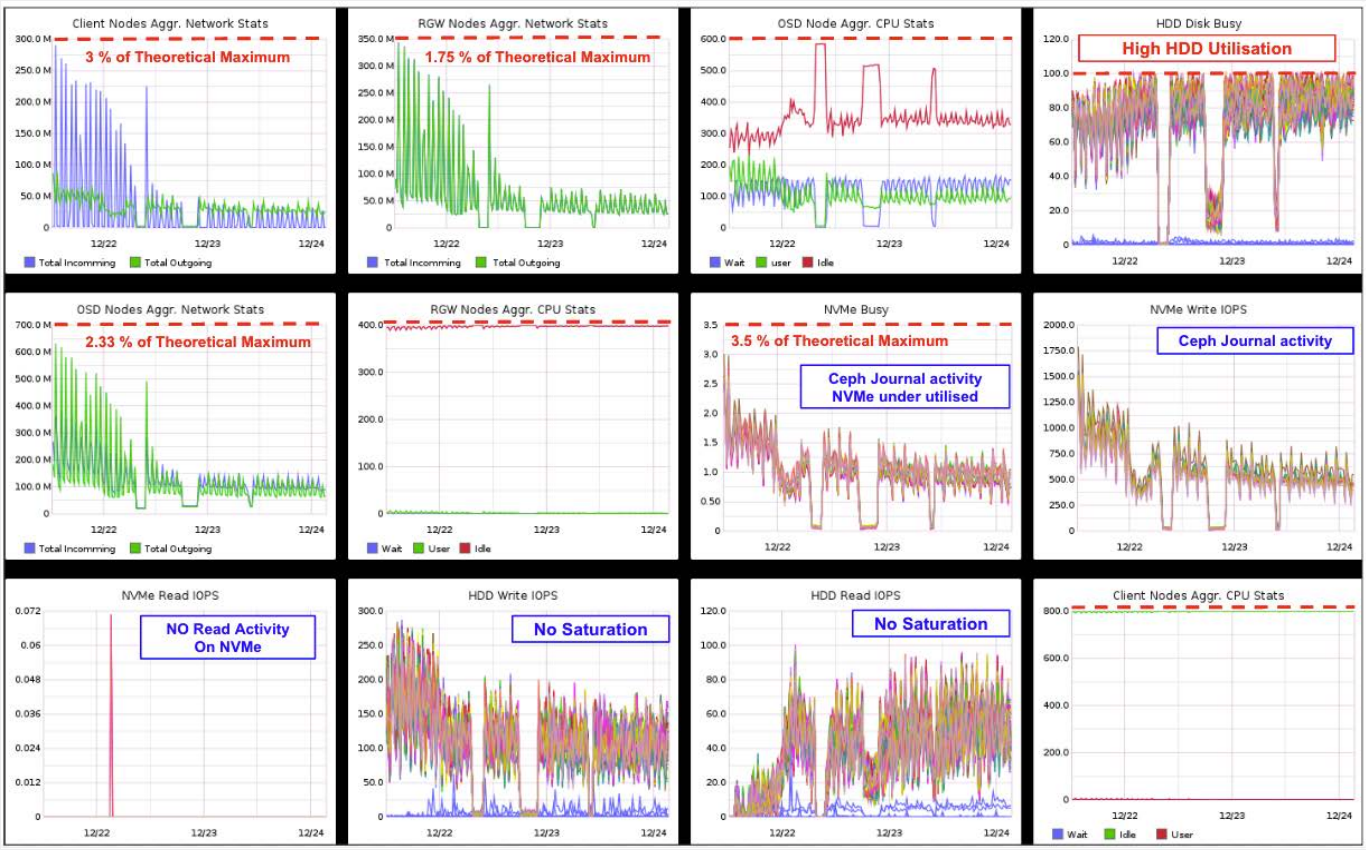

Figure 42. Subsystem telemetry during default Ceph OSD filestore test

TUNED CEPH OSD FILESTORE

Figures 43 and 44 display write and read OPS results respectively for the tuned Ceph OSD filestore, as compared to RADOS object count.

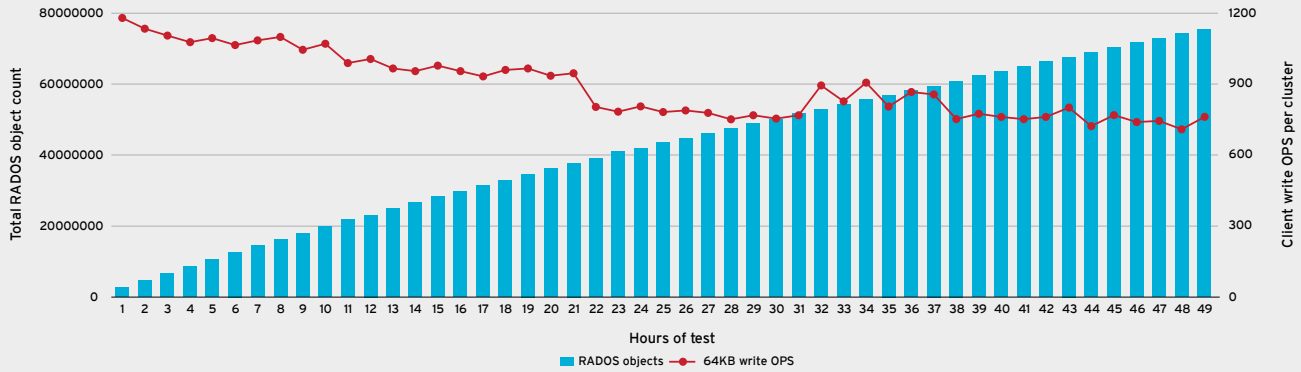

Figure 43. Tuned Ceph OSD filestore (split:merge 16:48) 64KB object write OPS versus RADOS object count on high-density servers

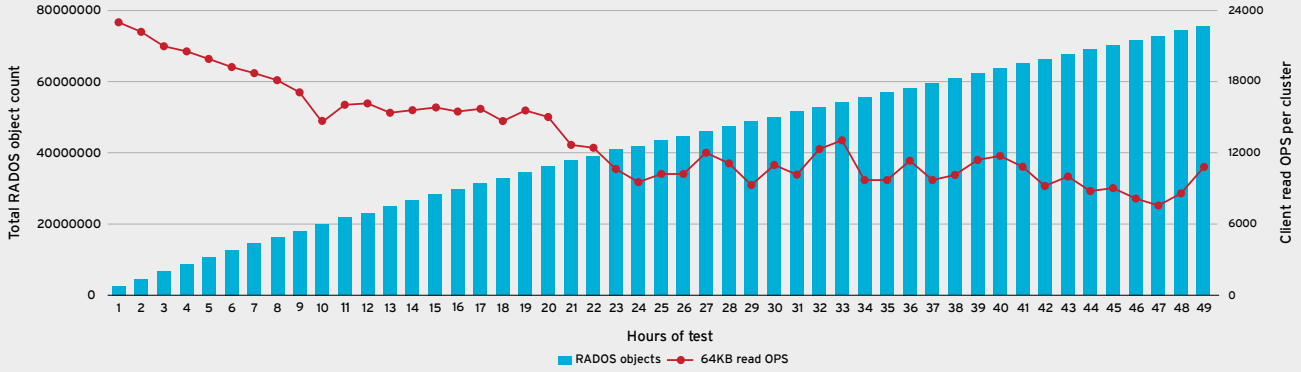

Figure 44. Tuned Ceph OSD filestore (split:merge 16:48) 64KB object read OPS versus RADOS object count on high-density servers

Figure 45 shows subsystem telemetry during the tuned Ceph OSD filestore testing.

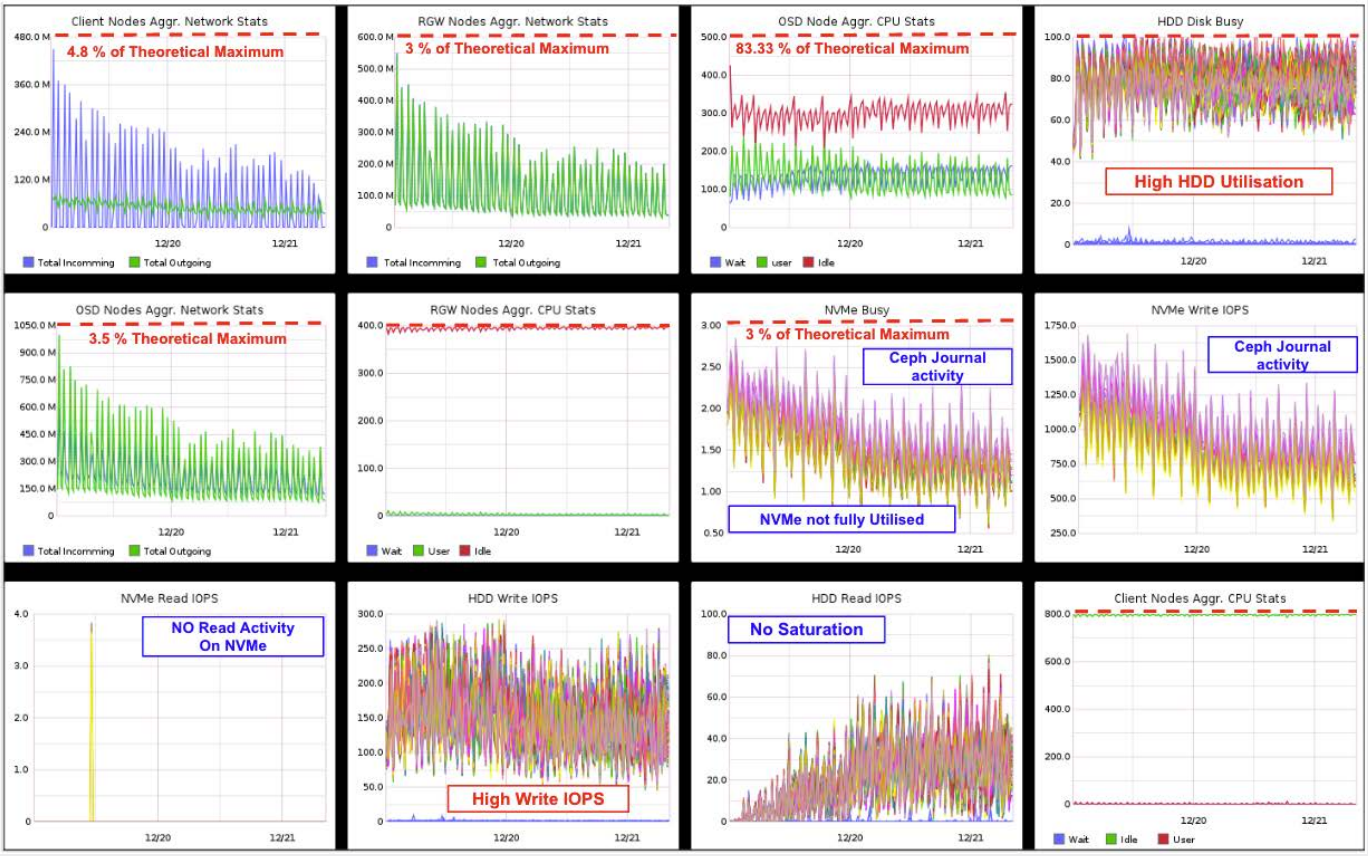

Figure 45. Subsystem telemetry during tuned Ceph OSD filestore test

DEFAULT CEPH OSD FILESTORE + INTEL CAS

Figures 46 and 47 display write and read OPS results respectively for the default Ceph OSD filestore combined with Intel CAS, as compared to RADOS object count.

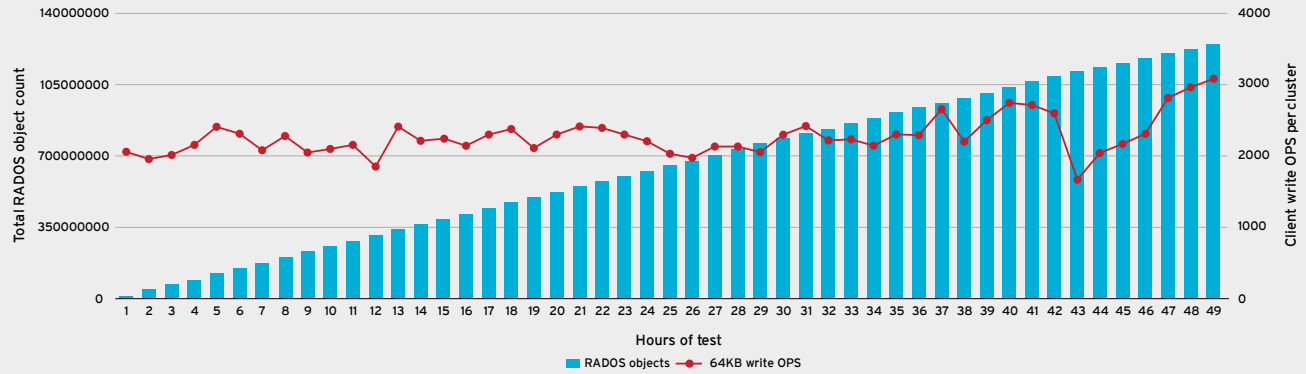

Figure 46. Default Ceph OSD filestore + Intel CAS (split:merge 2:10) 64KB object write OPS versus RADOS object count on high-density servers

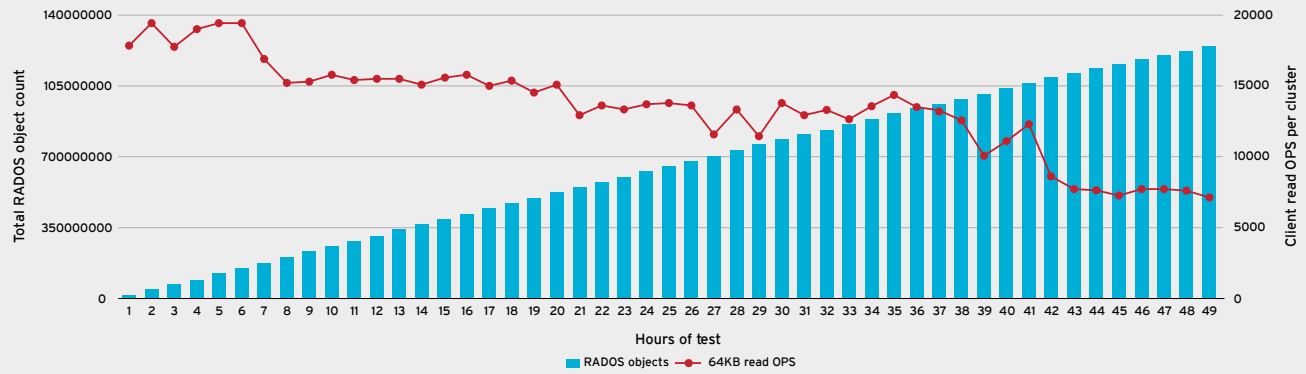

Figure 47. Default Ceph OSD filestore + Intel CAS (split:merge 2:10) 64KB object read OPS versus RADOS object count on high-density servers

Figure 48 shows subsystem telemetry during testing with the default Ceph OSD filestore with Intel CAS.

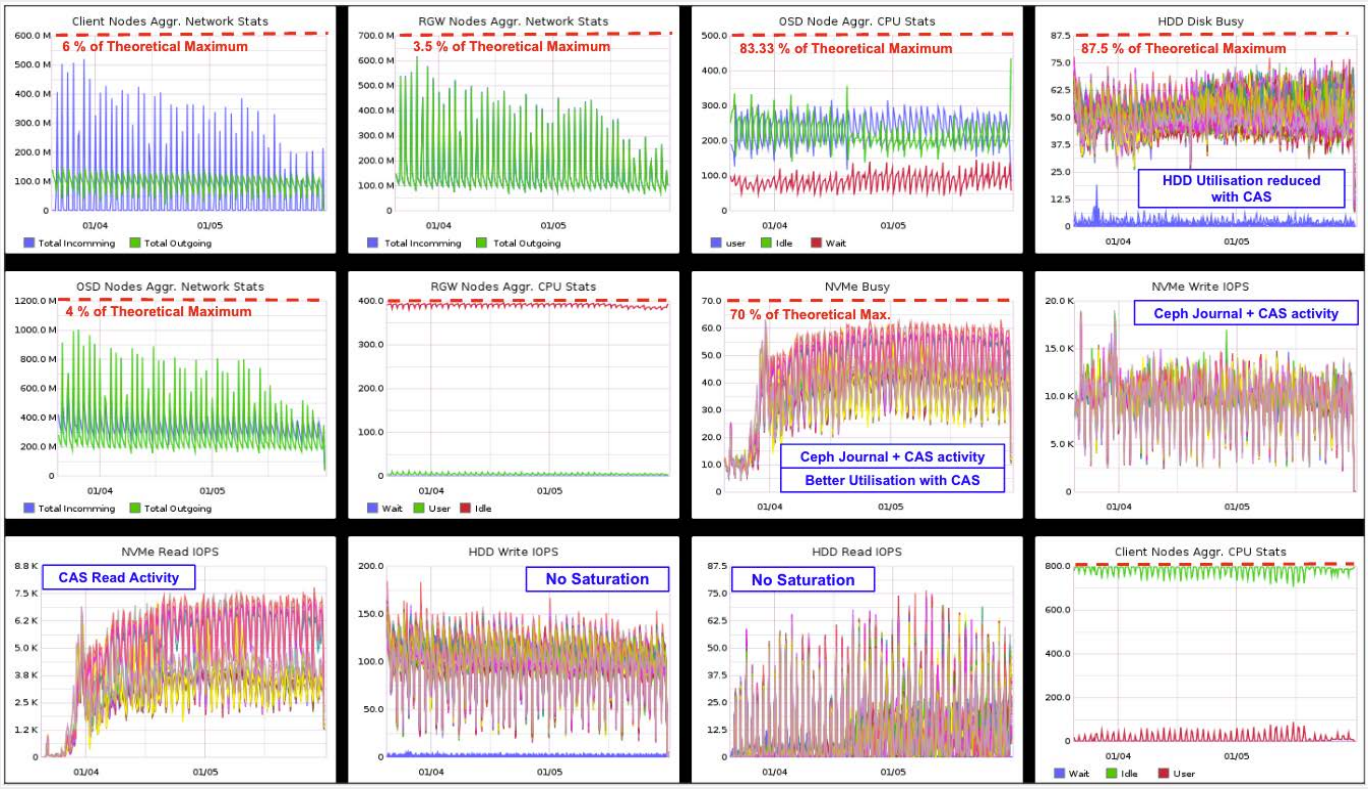

Figure 48. Subsystem telemetry during default Ceph OSD filestore + Intel CAS test

TUNED CEPH OSD FILESTORE + INTEL CAS

Figures 49 and 50 display write and read OPS results respectively for the tuned Ceph OSD filestore combined with Intel CAS, as compared to RADOS object count.

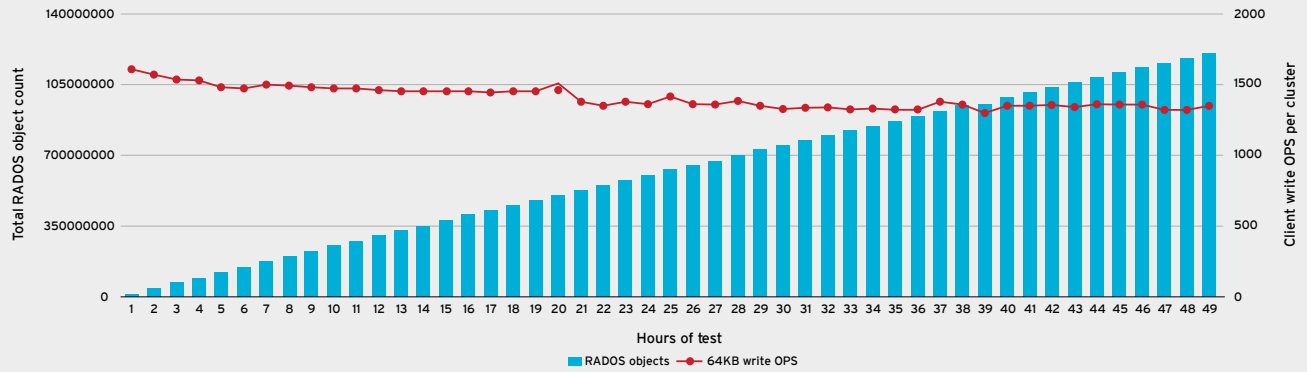

Figure 49. Tuned Ceph OSD filestore + Intel CAS (split:merge 16:48) 64KB object write OPS versus RADOS object count on high-density servers

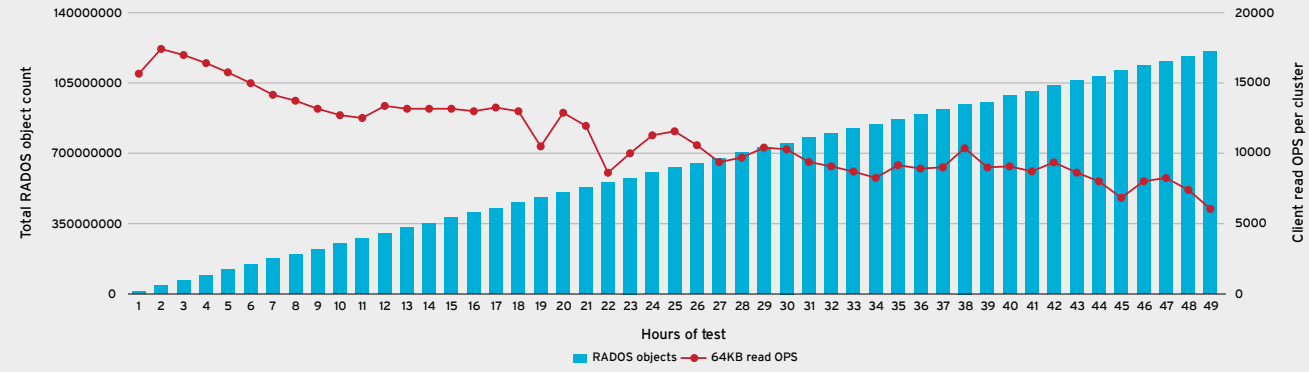

Figure 50. Tuned Ceph OSD filestore + Intel CAS (split:merge 16:48) 64KB object read OPS versus RADOS object count on high-density servers

Figure 51 shows subsystem telemetry during testing with the tuned Ceph OSD filestore with Intel CAS.

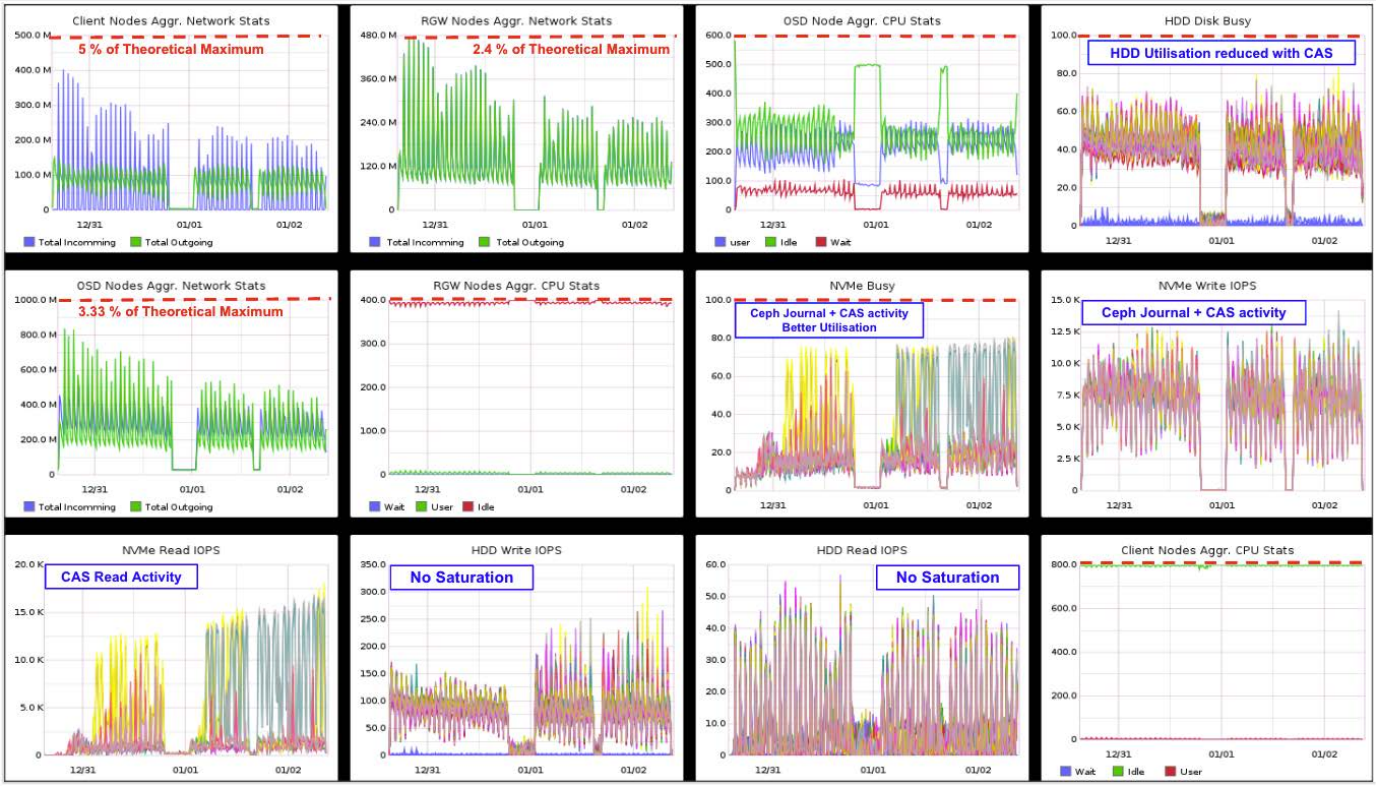

Figure 51. Subsystem utilization during tuned Ceph OSD filestore + Intel CAS test

APPENDIX F: CONFIGURATION DETAILS

The contents of the ceph.conf configuration file are as follows:

```
# Please do not change this file directly since it is managed by Ansible and
will be overwritten


[global]

fsid = 747aeff4-51e6-43f9-a7dd-6f6db0504213

max open files = 131072

objecter_inflight_ops = 40960

objecter_inflight_op_bytes = 2147483648


[client]

admin socket = /var/run/ceph/$cluster-$type.$id.$pid.$cctid.asok # must be
writable by QEMU and allowed by SELinux or AppArmor

log file = /var/log/ceph/qemu-guest-$pid.log # must be writable by QEMU and
allowed by SELinux or AppArmor


[mon]

[mon.ceph-mon1]

host = ceph-mon1

# we need to check if monitor_interface is defined in the inventory per host or
if it’s set in a group_vars file

mon addr = 10.5.13.143


[osd]

osd mkfs type = xfs

osd mkfs options xfs = -f -i size=2048

osd mount options xfs = noatime,largeio,inode64,swalloc

osd journal size = 10240

cluster_network = 10.5.13.0/24

public_network = 10.5.13.0/24

filestore_queue_max_ops = 500

filestore_op_threads = 8
```

```
filestore_max_sync_interval = 10

filestore_wbthrottle_xfs_bytes_start_flusher = 1073741824

osd_op_threads = 8

osd_enable_op_tracker = False


[client]

admin socket = /var/run/ceph/$cluster-$type.$id.$pid.$cctid.asok # must be
writable by QEMU and allowed by SELinux or AppArmor

log file = /var/log/ceph/qemu-guest-$pid.log # must be writable by QEMU and
allowed by SELinux or AppArmor


[client.rgw.ceph-rgw1]

host = ceph-rgw1

rgw_dns_name = ceph-rgw1

keyring = /var/lib/ceph/radosgw/ceph-rgw.ceph-rgw1/keyring

rgw socket path = /tmp/radosgw-ceph-rgw1.sock

log file = /var/log/ceph/ceph-rgw-ceph-rgw1.log

rgw data = /var/lib/ceph/radosgw/ceph-rgw.ceph-rgw1

rgw frontends = civetweb port=10.5.13.140:8080 num_threads=4096
request_timeout_ms=50000

rgw_num_rados_handles = 64

rgw_bucket_index_max_aio = 128

rgw_override_bucket_index_max_shards = 128

rgw_enable_gc_threads = False

rgw_cache_lru_size = 100000

rgw_thread_pool_size = 4096

rgw_list_buckets_max_chunk=999999

rgw_max_chunk_size = 4194304


[client.rgw.ceph-rgw2]

host = ceph-rgw2

keyring = /var/lib/ceph/radosgw/ceph-rgw.ceph-rgw2/keyring

rgw socket path = /tmp/radosgw-ceph-rgw2.sock
```

```
log file = /var/log/ceph/ceph-rgw-ceph-rgw2.log
rgw data = /var/lib/ceph/radosgw/ceph-rgw.ceph-rgw2
rgw frontends = civetweb port=10.5.13.141:8080 num_threads=4096
request_timeout_ms=50000
rgw_num_rados_handles = 64
rgw_bucket_index_max_aio = 128
rgw_override_bucket_index_max_shards = 128
rgw_enable_gc_threads = False
rgw_cache_lru_size = 100000
rgw_thread_pool_size = 4096
rgw_list_buckets_max_chunk=999999
rgw_max_chunk_size = 4194304

[client.rgw.ceph-rgw3]
host = ceph-rgw3
keyring = /var/lib/ceph/radosgw/ceph-rgw.ceph-rgw3/keyring
rgw socket path = /tmp/radosgw-ceph-rgw3.sock
log file = /var/log/ceph/ceph-rgw-ceph-rgw3.log
rgw data = /var/lib/ceph/radosgw/ceph-rgw.ceph-rgw3
rgw frontends = civetweb port=10.5.13.142:8080 num_threads=4096
request_timeout_ms=50000
rgw_num_rados_handles = 64
rgw_bucket_index_max_aio = 128
rgw_override_bucket_index_max_shards = 128
rgw_enable_gc_threads = False
rgw_cache_lru_size = 100000
rgw_thread_pool_size = 4096
rgw_list_buckets_max_chunk=999999
rgw_max_chunk_size = 4194304

[client.rgw.client10]      ##Client node used as RGW
host = client10
keyring = /var/lib/ceph/radosgw/ceph-rgw.ceph-client10/keyring
```

```
rgw socket path = /tmp/radosgw-ceph-client10.sock
log file = /var/log/ceph/ceph-rgw-ceph-client10.log
rgw data = /var/lib/ceph/radosgw/ceph-rgw.ceph-client10
rgw frontends = civetweb port=10.5.13.110:8080 num_threads=4096
request_timeout_ms=50000
rgw_num_rados_handles = 64
rgw_bucket_index_max_aio = 128
rgw_override_bucket_index_max_shards = 128
rgw_enable_gc_threads = False
rgw_cache_lru_size = 100000
rgw_thread_pool_size = 4096
rgw_list_buckets_max_chunk=999999
rgw_max_chunk_size = 4194304
```

ABOUT QCT

QCT (Quanta Cloud Technology) is a global datacenter solution provider extending the power of hyperscale datacenter design in standard and open SKUs to all datacenter customers. Product lines include servers, storage, network switches, integrated rack systems, and cloud solutions, all delivering hyperscale efficiency, scalability, reliability, manageability, serviceability, and optimized performance for each workload. QCT offers a full spectrum of datacenter products and services from engineering, integration, and optimization to global supply chain support, all under one roof. The parent of QCT is Quanta Computer Inc., a Fortune Global 500 technology engineering and manufacturing company. www.qct.io

ABOUT RED HAT

Red Hat is the world's leading provider of open source software solutions, using a community-powered 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.

NORTH AMERICA
1 888 REDHAT1

EUROPE, MIDDLE EAST, AND AFRICA
00800 7334 2835
europe@redhat.com

ASIA PACIFIC
+65 6490 4200
apac@redhat.com

LATIN AMERICA
+54 11 4329 7300
info-latam@redhat.com

facebook.com/redhatinc
@redhatnews
linkedin.com/company/red-hat

redhat.com
#US131488_0617