Red Hat | DELL EMC

Reference architecture

Red Hat Ceph Storage on Dell EMC servers

Object storage performance and sizing guide

Table of contents

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

redhat.com

Executive summary

Modern organizations are tasked with managing truly enormous amounts of data—up to billions of objects and tens to hundreds of petabytes of data—with no end in sight. Object storage is well suited to addressing these challenges, both in the public cloud and in on-premise infrastructure. While object storage systems like Ceph® convey many advantages, configuring and deploying software, hardware, and network components to serve a diverse range of data-intensive workloads can require a significant investment in time and training.

Object storage is ideal for unstructured data that is written once and read once or many times. By combining metadata within the object itself, object storage eliminates the tiered structure used in file storage, and places everything into a storage pool. Object storage is ideal for data that includes:

Static web content.

- Backup data.
- Archival images.
- Multimedia, including videos, pictures, or music files.
- Geographically distributed back-end storage.

Small-object workloads can include image and video recognition, algorithmic trading, and factor sensor data. Large-object sequential 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.

Based on extensive testing by Red Hat and Dell Technologies, this paper describes a robust object storage infrastructure using a combination of Red Hat® Ceph Storage coupled with Dell EMC storage servers and networking. Both large-object and small-object synthetic workloads were applied to the test system and the results were subjected to performance analysis. Testing also evaluated the ability of the system under test to scale beyond a billion objects. To evaluate performance progress through time, a modern Red Hat Ceph Storage 3.3 system was also compared to an earlier Red Hat Ceph Storage 2.0 configuration (See appendix).

The sections that follow offer a brief overview of performance results realized in this study, with details provided in the remainder of the document.

Answering foundational questions

Before the Red Hat Storage Solution Architectures team begins a new project, we formulate questions that we intend to answer during the course of the project. We prioritize questions after discussing them with various stakeholders who are well versed with Red Hat Ceph Storage, including large customers, our technical sellers, product management, engineering, and support. This effort generated the carefully curated list of questions below for evaluating Red Hat Ceph Storage on Dell EMC hardware:

- What is the optimal CPU/Thread allocation ratio per reliable automatic distributed object store (RADOS) gateway (RGW) instance?
- What is the recommended RGW deployment strategy?
- What is the maximum performance on a fixed-sized cluster?

- How is cluster performance and stability affected as the cluster population approaches 1 billion objects?
- How does bucket sharding lend to increased bucket scalability, and what are the performance implications?
- How does the new Beast.Asio frontend for RGW compare to Civetweb?
- What's the performance delta between Red Hat Ceph Storage 2.0 with a FileStore backend and Red Hat Ceph Storage 3.3 with a BlueStore backend?
- What are the performance effects of erasure coded Fast_Reads vs. Regular Reads?

Performance summary

Red Hat Ceph Storage is able to run on myriad industry-standard hardware configurations and clusters are often used in multi-tenant environments, subjected to a diverse mix of workloads. This flexibility is powerful, but it can be intimidating to those who are not familiar with the most recent advances in both system hardware and in Ceph software components. Designing and optimizing a Ceph cluster requires careful analysis of the applications, their required capacity, and workload profile. Red Hat, Dell EMC, and Intel carefully selected system components that we believed would best affect performance. Tests evaluated workloads in three broad categories:

- **Large-object workloads (throughput, GB per second).** Throughput is the principal performance concern for large-object workloads. Red Hat testing showed near-linear scalability for both read and write throughput with the addition of RGW instances, until performance peaked due to OSD node disk contention. We measured over 6GB/s cluster-wide aggregated bandwidth for both read and write workloads.
- **Small-object workloads (operations per second).** Small-object workloads are more susceptible to metadata input/output (I/O) operations than are large-object workloads. In Red Hat testing, cluster-wide client write operations scaled sub-linearly to 6.3K operations per second (OPS) with 8.8 ms average response time until saturated by OSD node disk contention. Client read operations reached a maximum of 3.9K OPS, which could be attributed to lack of Linux® page cache within the BlueStore OSD backend.
- H**igher object-count workloads (billion objects or more).** As the object population in a Ceph OSD bucket grows, there is corresponding growth in the amount of index metadata that needs to be managed. Index metadata is spread across a number of shards to ensure that it can be accessed efficiently. Red Hat Ceph Storage will dynamically increase the number of shards when the number of objects in a bucket is larger than what can be efficiently managed with the current number of shards. In the course of testing we found that these dynamic sharding events have a detrimental, but temporary impact on the throughput of requests. If a developer has a priori knowledge that a bucket will grow to a given population, that bucket can be created with or manually updated to a number of shards that will be able to efficiently accommodate the amount of metadata. Our testing revealed that buckets that have been pre-shared in this manner deliver more deterministic performance, because they do not need to reorganize bucket index metadata as object count increases.

During both large and small object size workload tests, Red Hat Ceph Storage delivered significant performance improvements relative to previous performance and sizing analysis. Table 1 lists expectations for both large and small object performance based on this study, assuming the cluster is operating in steady state with no resource saturation at any tier. We believe these improvements can be attributed to the combination of the BlueStore OSD backend, the new Beast.Asio[1] web frontend for RGW, the use of Intel Optane SSDs for BlueStore write-ahead log (WAL) and block.db, and the latest generation processing power as provided by Intel Cascade Lake processors.

Table 1. Large and small object performance summary

Objective	Expectations
Large object (32M) performance	~900MBps per RGW instance for both 100% read and 100% write workloads
Small object (64K) performance	~1000 OPS per RGW instance for a 100% write workload ~500 Ops per RGW instance for a 100% read workload

Architectural approaches

Ideal architectural approaches can vary based on workloads, the data being stored, and intended objectives. While specific sizing recommendations are beyond the scope of this document, Table 2 lists several objectives along with architectural design strategies evaluated in Red Hat testing. Detailed test results that evaluate these strategies are provided throughout this document.

Table 2. Architectural approaches to accomplishing various objectives

Objective	Strategy
Achieve optimal Ceph RGW sizing	• 4x logical CPU cores per RGW instance • 512 RGW threads per RGW instance • Beast.Asio web server • RGW co-located with Ceph OSD daemons
Maximize object storage performance for both reads and writes	• Use 1x RGW instance per storage node • If workload demands more performance, scale to 2x RGW instances (assuming cluster is operating in steady state with no resource saturation at any tier, and adequate system resources are available to run 2 x RGW / OSD node)
Store a large number of objects in a bucket	• Pre-shard the buckets for consistent performance

1 *Beast.Asio is a new front-end for the Object Store Gateway (RGW) that provides better I/O performance and has lower resource requirements. Learn more at* https://github.com/boostorg/beast.

Objective	Strategy
Store 1 billion+ objects per cluster	• Use several buckets and distribute objects evenly (100K objects / bucket) • Appropriately size Flash Media for Bluestore metadata (block.db) to avoid spillover of RocksDB levels to slower media. • For bulk delete operations, use either object expiration bucket lifecycles, or the radosgw-admin tool. • Maintain enough free capacity in the cluster as cluster operating at 70% filled ratio and above can suffer degraded performance

Payload selection

Object storage payloads vary widely, and 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. Table 3 lists object payload sizes included in Red Hat testing.[2]

Table 3. Object storage payload sizes tested

Workload category	Payload size	Representative of
Small object	64KB	Thumbnail images, small files
Medium object	1MB	Images, documents, attachments, etc.
Large object	32MB	HD audio/video, backup, logs

Dell EMC test configuration with Red Hat Ceph Storage

The Dell EMC PowerEdge R740xd server is optimized for storage and provides an ideal combination of performance and flexibility for software-defined storage (SDS) applications. The server can accommodate various storage device configurations including 2.5-inch or 3.5-inch hard disk drives (HDDs), solid state drives (SSDs), and non-volatile memory express (NVMe) devices. For this testing, the team chose a PowerEdge R740xd configuration with 12x 3.5-inch HDDs in the front bays and four additional HDDs in the mid-bays (located within the server chassis). The 3.5-inch storage devices

2 *Data processing applications tend to operate on larger sized files. Red Hat testing found that 32MB is the largest object size COSBench can handle reliably at the scale of our testing.*

were chosen for this architecture as they provide high capacity in a cost-effective manner. Two Intel Optane P4800X high-speed storage devices were included to accommodate the BlueStore write-ahead log (WAL) and RocksDB metadata. These Intel Optane devices were 750GB add-in cards (AIC) models.

Using typical Ceph OSD memory sizing guidelines, approximately 100GB of RAM is needed for all of the Ceph daemons and the operating system. Since the Intel Xeon processors provide six memory channels, we were able to achieve the best memory performance by populating six DIMMs per processor. Using 16GB DIMMs, this configuration provides a total of 192GB of RAM (6x 16GB DIMM per processor) per server.

Guidelines for CPU sizing call for roughly one CPU core per OSD with additional cores to service the operating system and other Ceph daemons. Previous studies have indicated that 2.2 core-GHz per HDD-based OSD was appropriate for Intel Xeon 'Skylake' processors with RHCS 3.2.[3] For this architecture, we chose the Intel Xeon Gold 5215 2.5GHz processor (formerly 'Cascade Lake') processor with 10 cores. A dual processor configuration provides a total of 20 cores and a ratio of 0.8 cores/OSD and 2.0 core-GHz per HDD-based OSD.

Testing was conducted in a Dell EMC laboratory, using Dell EMC hardware and CPU and NVMe upgrades contributed by Intel corporation.

Hardware configuration

The rack-level hardware configuration is shown in Figure 1, with server configurations detailed in Table 4. Testing was based on the number of server nodes available in the laboratory (seven), and was sufficient for evaluating performance results as the cluster scaled. Two different types of Dell EMC PowerEdge servers were used in the test configuration:

- **Dell EMC PowerEdge R740xd servers** functioned as Red Hat Ceph Storage nodes. These servers provide storage performance and flexibility in a two rack-unit (2U), 2-socket form factor. The R740xd servers are ideal for workloads like software-defined storage, big data, and high-performance computing. They can be equipped with up to 24 NVMe drives, or a total of 32x 2.5-inch or 18x 3.5-inch drives.
- **Dell EMC PowerEdge R630 servers** provide performance in a compact 1U chassis, ideal for virtualization or large-scale transactional and analytical processing solutions. In Red Hat testing, these servers were used to drive the workload against the storage cluster.
- **Dell EMC Z9100-ON switches** were used to provide redundant 25 GbE networks for the rack. The front-end (public) network carried benchmark traffic from the COSBench worker nodes, while a separate 25 GbE (private) network isolated Red Hat Ceph Storage cluster traffic.

3 *For more information see the* Dell EMC Ready Architecture for Red Hat Ceph Storage 3.2.

Table 4. Dell EMC PowerEdge server configuration details

Function	Quantity	Configuration details
Red Hat Ceph Storage 3.3 storage nodes	7	Dell EMC PowerEdge R740xd server: • 2x Intel Xeon Gold 5215 processors • 192GB system memory • 16x 4TB SAS HDD 7.2K RPM drives • 2x Intel Optane SSD DC P4800X Series, 750GB • 2x Intel Ethernet Network Adapter XXV710 25GbE NICs (dual port)
COSBench worker nodes	7	Dell EMC PowerEdge R630 server: • 2x Intel Xeon E5-2650 v4 processors • 192GB system memory • 1x Intel Ethernet Network Adapter XXV710 25GbE NICs (dual port)

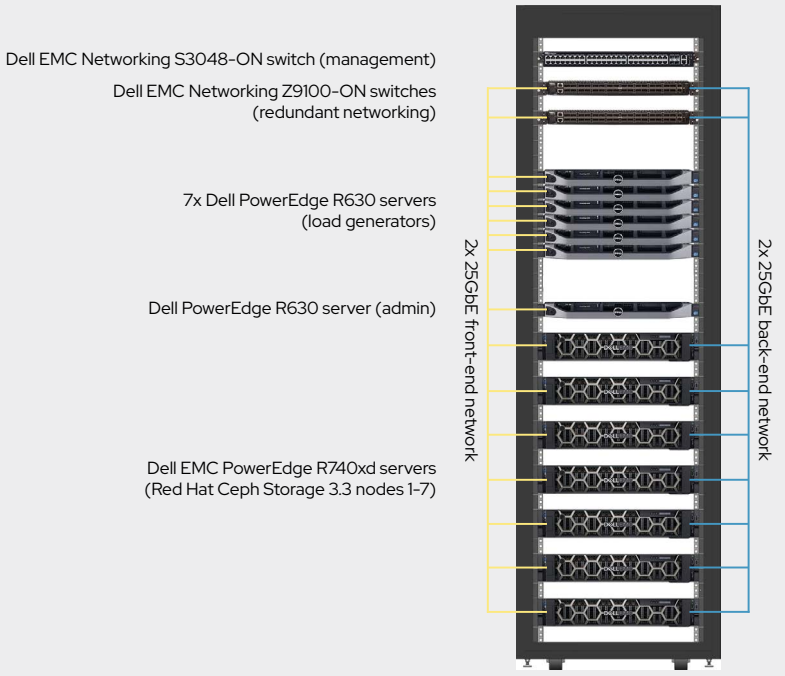

Figure 1. Rack-level configuration for Red Hat testing

Logical architecture

Figure 2 shows the logical architecture used in Red Hat Ceph Storage testing, illustrating the dual network architecture. Testing utilized Containerized Storage Daemons (CSD), a feature that allows all of the Ceph software-defined storage components to run within containers (rather than on dedicated storage nodes). CSD configuration strategies and testing are covered in more depth elsewhere in this document.

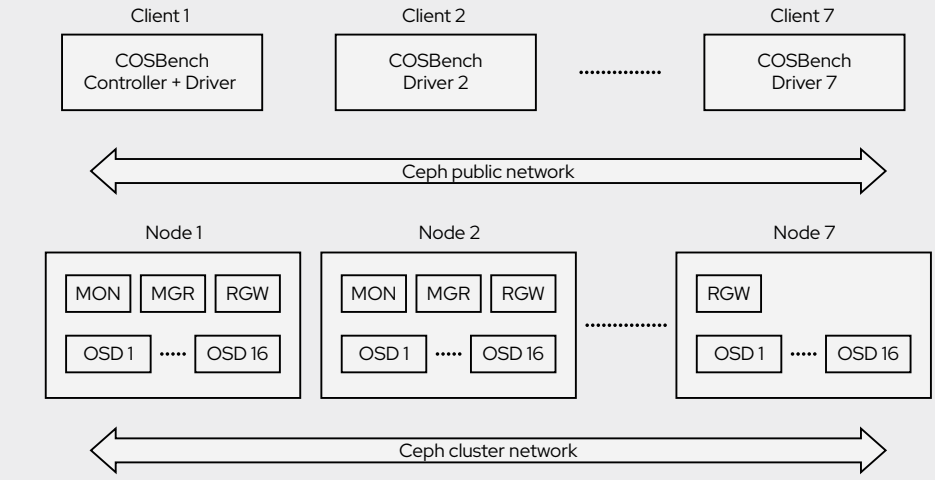

Figure 2. Logical design for Red Hat Ceph Storage testing

Software configuration

Table 5 lists the software configuration used in the test cluster.

Table 5. Test cluster software configuration

Component	Details
Operating system	Red Hat Enterprise Linux 7.6
Ceph	Red Hat Ceph Storage 3.3
Ceph OSD backend	BlueStore
Ceph deployment type	Containerized storage daemon (CSD)
Ceph data population	Erasure coding 4+2 (plugin: jerasure)
COSBench	V0.4.2 (Loadgen to RGW static mapping)
Specter patches	Disabled

Table 6 lists the CPU logical cores and memory configured into the various software-defined storage components.

Table 6. Resource limitations for software-defined storage components

	Ceph MON	Ceph MGR	Ceph OSD	Ceph RGW	OS
CPU logical cores per container instance	3	3	1	1/2/4/8/10	8
Memory limit (GB)	192	192	192	192	192

Ceph cluster baseline performance

To measure native Ceph cluster performance, Red Hat used the Ceph Benchmarking Tool (CBT), an open-source tool for automation of Ceph cluster benchmarks.[4] The Ceph cluster was deployed using Ceph-Ansible®, based on Ansible Playbooks.[5] With Ceph-Ansible playbooks, Red Hat Ansible Automation Platform can automate the creation of a Ceph cluster with default settings.

Baseline testing was conducted to find the cluster aggregate throughput high-water mark of the untuned cluster. This water mark is the point beyond which throughput either remains flat, or decreases. Seven iterations of each benchmark scenario were run using CBT. 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.

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
- **RADOSbench instance per client:** 1
- **Pool data protection:** Erasure coding 4+2 jerasure plugin OR 3x replication
- **Total number of clients:** 7

As shown in Figure 3 and Figure 4, performance scaled sub-linearly in the untuned cluster for both write and read workloads respectively as the number of clients increased. Interestingly, the configuration using an erasure coding data protection scheme delivered a near symmetrical 8.4GBps for both write and read throughput. This symmetry had not been observed in previous studies, and we believe that it can be attributed to the combination of the BlueStore backend and Intel Optane SSDs. We could not identify any system level resource saturation during any of the RADOSbench tests. Had additional client nodes been available to further stress the existing cluster, we hypothesize that we might have observed even higher performance and potentially been able to saturate the underlying storage media.

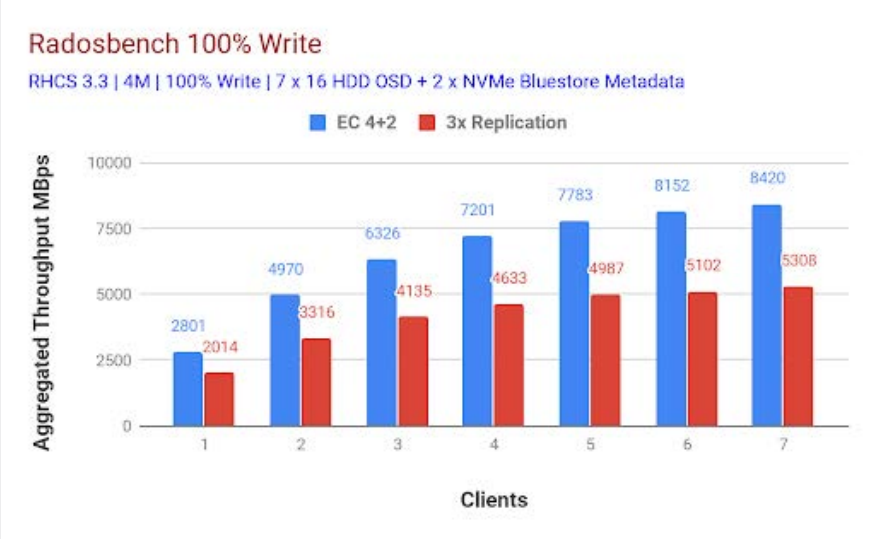

Figure 3. RADOSbench 100% write scalability test (baseline, untuned cluster)

Figure 4. RADOSbench 100% read test (baseline, untuned cluster)

RGW deployment strategies and sizing guidance

As a part of testing, Red Hat engineers wanted to understand how changing available resources for the Red Hat Ceph Storage RGW instances could affect performance. Broadly, these tests involved varying the number of RGW instances that ran on each node, and tuning the RGW thread pool size.

Containerized storage daemons

Beginning with Red Hat Ceph Storage 3.0, Red Hat added support for Containerized Storage Daemons (CSD), allowing all of the Ceph software-defined storage components (MON, OSD, MGR, RGW, etc.) to run within containers. This approach eliminates the need to have dedicated nodes for storage services, thus reducing both capital expenses (CapEx) and operational expenses (OpEx) by co-locating storage components on a single server. Ceph-Ansible provides the required mechanism to put resource fencing around each storage container, useful for running multiple storage daemon containers on one physical node.

Standalone RGW is the traditional way of running Ceph, with dedicated physical or virtual nodes for running each RGW instance as shown in Figure 5. Starting with Red Hat Ceph Storage 3.0 standalone RGW is no longer the preferred deployment method, with co-located RGW being preferable.

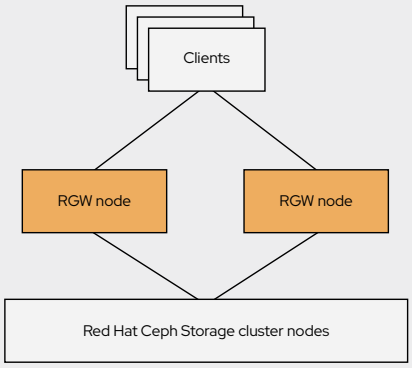

Figure 5: Standalone RGW deployment, with an RGW instance on each physical or virtual node

With co-located RGW, a single instance of a Ceph RGW container is placed on a storage node co-resident with other storage containers (Figure 6). This approach does not require dedicated nodes for RGWs, potentially reducing both CapEx and OpEx.

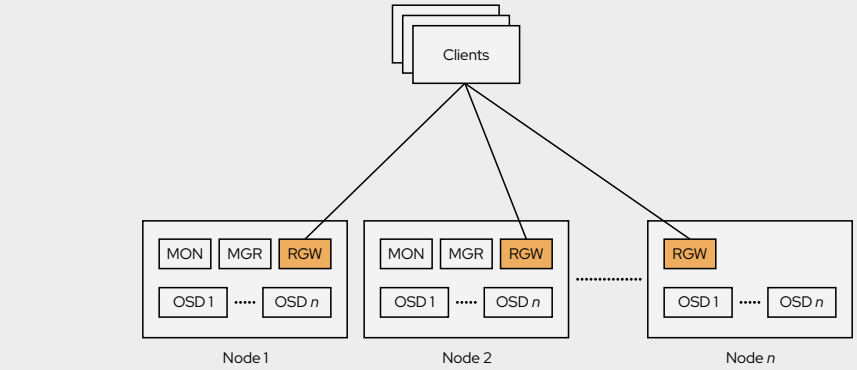

Figure 6. A co-located RGW deployment strategy places an RGW container on each storage server

Multi co-located RGW places multiple Ceph RGW instances co-resident with other storage containers (Figure 7). Red Hat testing evaluated configurations with up to two RGW instances per storage node. Our testing showed this option delivers the highest performance without incurring additional cost.[4]

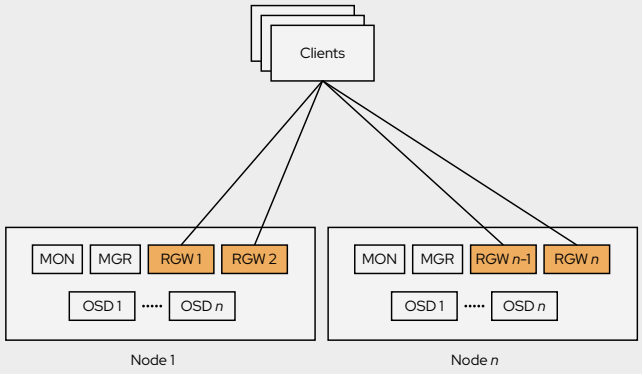

Figure 7. Multiple co-resident deployment strategy with two RGW instances per node

Evaluating RGW deployment and sizing strategies

To gauge the performance of different RGW deployment methods, engineers executed multiple tests by modulating RGW deployment strategy as well as RGW CPU core count across small and large objects for both write and read workloads.

4 *Note: Before deploying this configuration in production, a support exception may be required from Red Hat Ceph Product Management.*

100% write workload performance

Figures 8 and 9 illustrate test results for small (throughput) and large object (bandwidth) tests respectively under a 100% write workload. The charts show the results of various RGW deployment strategies as well as the impact of CPU core count per RGW.

In this testing, co-located RGW instances (with 1x RGW container per node) outperformed a stand-alone RGW deployment for both small and large object sizes. Similarly, two co-resident (2x RGW containers per node) RGW instances outperformed the co-resident (1x) RGW instance deployment. Through these tests, multiple co-resident (2x) RGW instances delivered 2328 Ops and 1879 MBps performance for small and large object sizes respectively. Across multiple tests, a ratio of four CPU cores per RGW instance was found to be optimal. The allocation of more CPU cores to the RGW instance did not deliver higher performance.

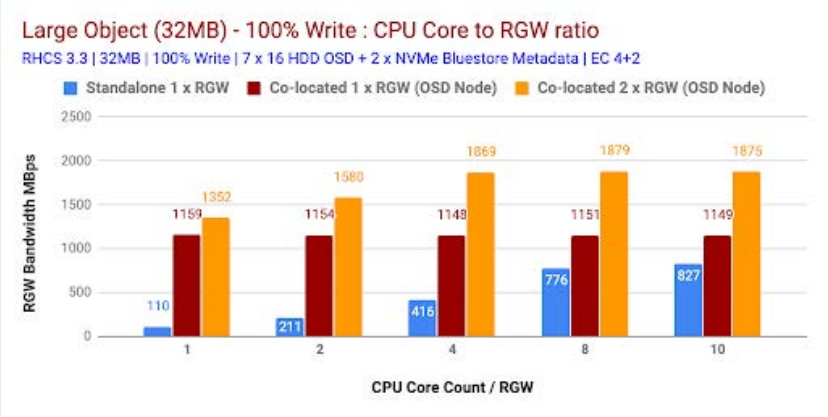

Figure 8. Small object 100% write test

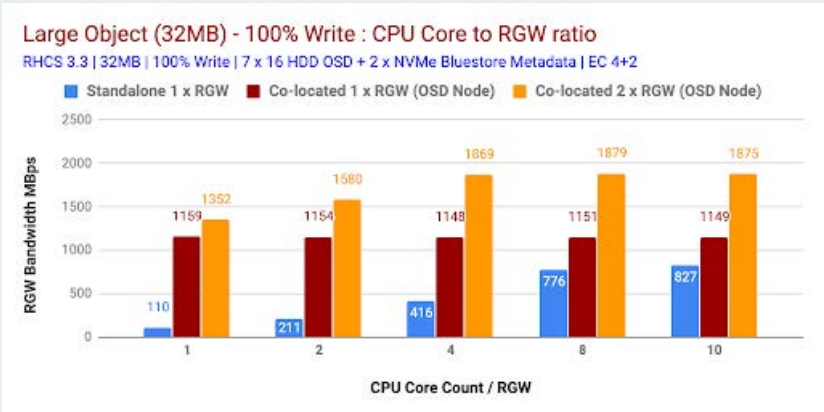

Figure 9. Large object 100% write test

100% read workload performance

When testing read workloads, engineers found that increasing CPU cores per RGW instance did not deliver performance improvements for both small and large object sizes (Figures 10 and 11 respectively). As such, results from configuring one CPU core per RGW instance were similar to results from configurations using 10 CPU cores per RGW instance. When reviewing previous testing, we observed similar results in that read workloads do not consume a significant amount of CPU. This may be due to Ceph making use of systematic erasure coding, and not need to decode chunks during read. If the RGW workload is read-intensive, over-allocating CPU does not help performance.

Comparing performance results from standalone RGW to co-located (1x) RGW showed very similar results. However, by adding one more additional co-located RGW instance (2x) performance improved by roughly 200% in the case of small object sizes and approximately 90% in the case of large object sizes. This testing showed that running multiple co-located (2x) RGW instance could boost overall read performance significantly for read-intensive workloads.

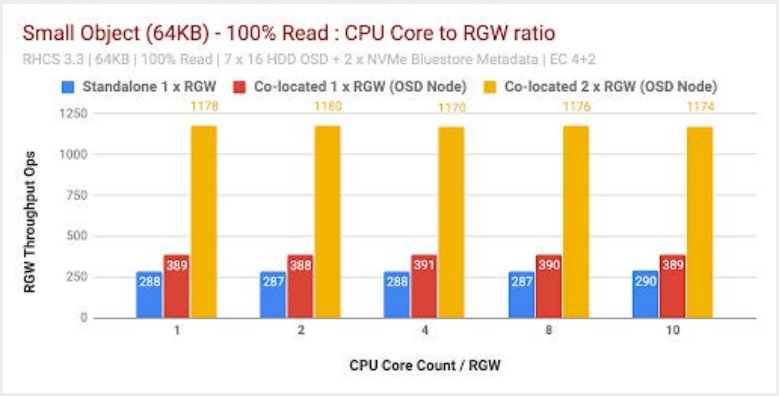

Figure 10. Small object 100% read test

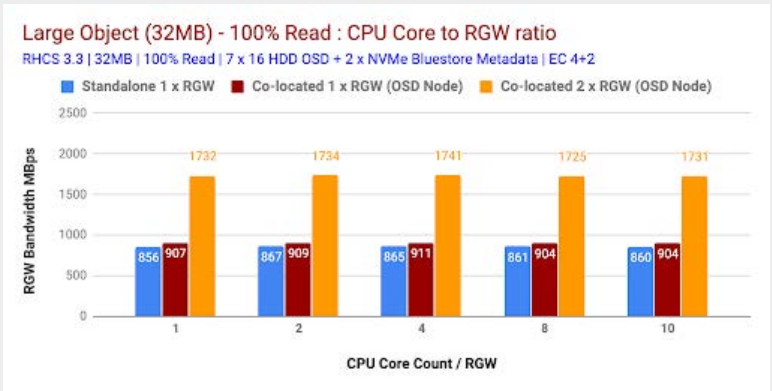

Figure 11. Large object 100% write test

Evaluating RGW thread pool sizing

rgw_thread_pool_size is a tuning parameter that is relevant while deciding CPU core allocation to RGW instances. The parameter is responsible for the number of threads spawned by Beast.Asio to service HTTP requests. Setting the parameter effectively limits the number of concurrent connections that the Beast web frontend can service.

To identify the most appropriate value for this tunable parameter, engineers ran multiple tests, varying the rgw_thread_pool_size together with CPU core count per RGW instance. As shown in Figures 12 and 13, we found that setting rgw_thread_pool_size to 512 delivers maximum performance at four CPU cores per RGW instance. Increasing both CPU core count as well as rgw_thread_pool_size did not increase performance any further.[5]

Based on this testing, and the RGW deployment strategy testing, engineers concluded that multi-collocated (2x) RGW instance with four CPU cores per RGW instance and rgw_thread_pool_size of 512 deliver the maximum performance.

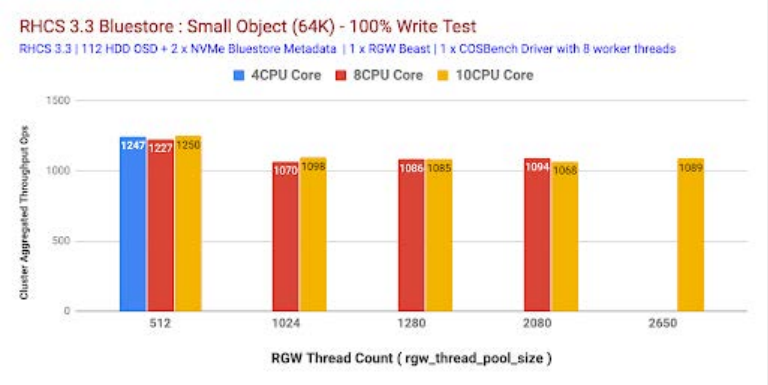

Figure 12. Small object 100% write test

5 *Unfortunately, engineers were unable to run tests with a rgw_thread_pool_size lower than 512. Our hypothesis is that since the Beast web server is based on an asynchronous c10k web server, it does not need a thread per connection and hence should perform as well with lower numbers of threads.*

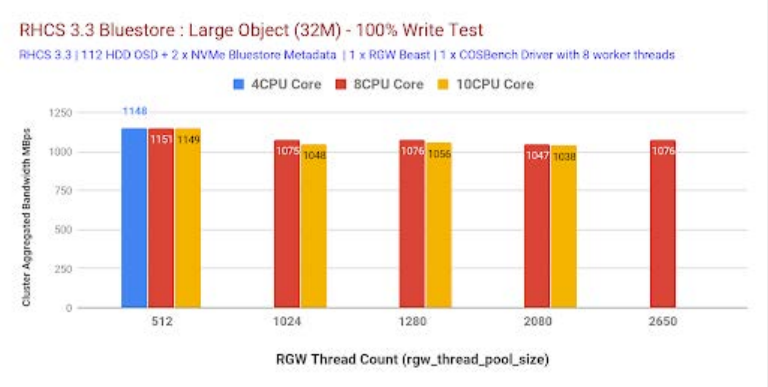

Figure 13. Large object 100% write test

Maximum fixed-sized cluster performance

One of the foundational goals for this study was to determine the maximal performance for a fixed-size Red Hat Ceph Storage cluster. To satisfy this goal, Red Hat testing evaluated a variety of configurations, object sizes, and client worker counts in order to maximize the throughput of a seven-node Ceph cluster for both small and large object workloads.

Large-object workload

Large-object sequential 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. Throughput (in terms of megabytes or gigabytes per second) is the key metric that defines storage performance for these workloads.

As discussed, the Ceph cluster was built using a single OSD configured per HDD, resulting in a total of 112 OSDs for the entire Ceph cluster. Figure 14 illustrates that both large-object 100% read and 100% write workloads exhibited sub-linear scalability when incrementing the number of RGW hosts from one to seven.[6]

6 *The terms "read" and HTTP GET are used interchangeably throughout this piece, as are the terms HTTP PUT and "write."*

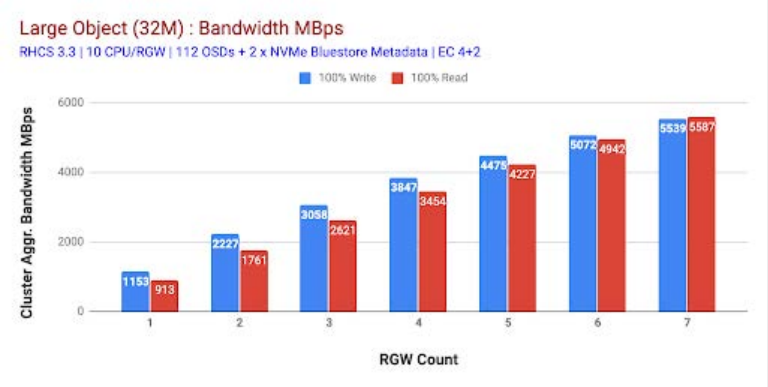

Figure 14. Large-object scalability testing with 7 RGWS (one per node)

With seven RGWs, engineers measured ~5.5 GBps aggregated bandwidth for both HTTP GET and HTTP PUT workloads (Figure 15), and interestingly did not notice resource saturation in Ceph cluster nodes. This finding implied that there was potentially room to increase throughput. Two methods of directing more load to the cluster were considered; adding more client nodes, or adding more RGW nodes.

Since the number of physical client nodes available for testing was fixed, we opted to run another round of tests with 14 RGWs. As shown in Figure 15, the 14-RGW HTTP PUT (write) test yielded a 14% higher write performance as compared to the 7-RGW test, resulting in maximum throughput of ~6.3GBps. Similarly, the HTTP GET (read) workload showed 16% higher read performance, resulting in a maximum throughput of ~6.5GBps.

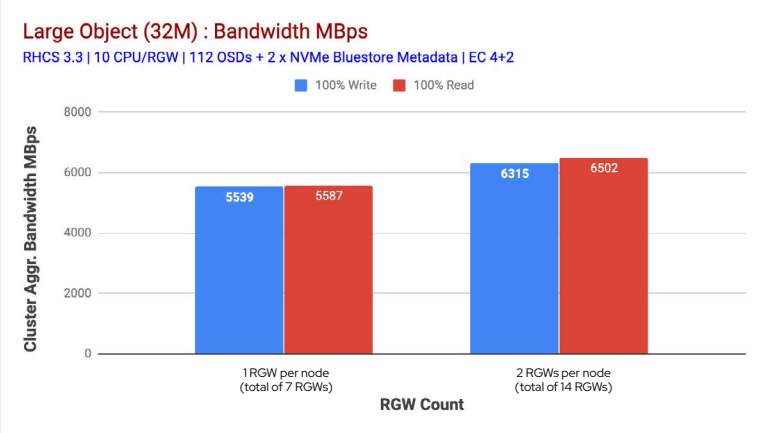

Figure 15. Large object scalability comparing clusters with 7 and 14 RGWs

Importantly, these results demonstrate maximum aggregated throughput observed for the cluster, after which media (HDD) saturation was noticed as depicted in Figure 16. Based on these results, we believe that if more Ceph OSD nodes were added to this cluster, the performance could have been scaled even further, until limited by resource saturation.

Figure 16 Ceph OSD (HDD) media utilization with 14 RGWs

Small-object workload

As shown in Figure 17, both small-object 100% write and 100% read workloads likewise exhibited sub-linear scalability when incrementing the number of RGW hosts. As such, we measured ~6.2K write operations per second (OPS) throughput at 9ms application write latency and ~3.2K read OPS for seven RGW instances. We again did not notice resource saturation at seven RGW instances.

At 14 RGW instances, we observed degraded performance for the HTTP PUT workload which we attributed to media saturation. We hypothesized that write performance could have scaled higher with more Ceph OSD nodes. HTTP GET performance scaled up to a maximum of ~4.5K OPS. We believe that adding more client nodes should have improved read performance, but that was beyond the scope of this testing.

Reduced average response times also fell as the number of RGW instances rose to six and above. As shown in Figure 17, at six RGWs, response times for HTTP PUT workloads were measured at 9ms while HTTP GET workloads showed 17ms of average latency (as measured from the application). Achieving single-digit write average latency from an object storage system is significant. We believe these improved latencies may be attributable to the combined performance improvement coming from the BlueStore OSD backend as well as the high-performance Intel Optane NVMe's used to back the BlueStore metadata devices.

Doubling the number of RGWs to 14 improved 100% read performance (in terms of OPs) but not 100% write performance.

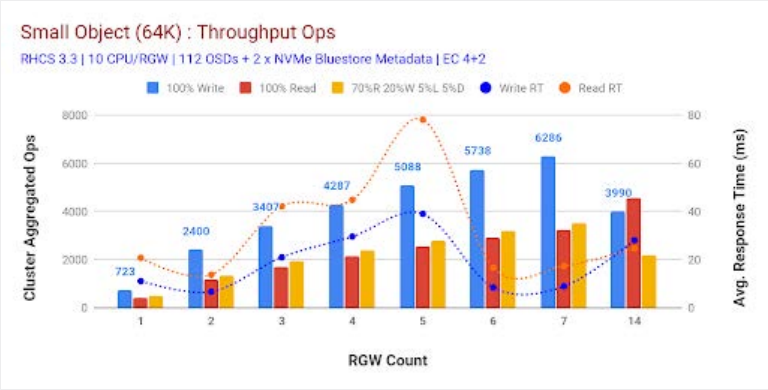

Figure 17 Small object throughput testing

Scaling to 1 billion objects

Ceph has long been established as a solution for true scale-out storage, with the ability to easily add not only capacity, but also performance to an existing cluster. Given this ability, it is important to analyse how Ceph performs when a cluster is filled with large numbers of small objects. Red Hat engineers wanted to see how Red Hat Ceph Storage 3.3 performance characteristics change while filling up the test cluster with over 1 billion objects.

Test configuration

To write out the 1 billion objects, engineers employed the COSbench tool in rounds of object generation and performance measurement. Each round consisted of the following steps:

- Creating 14 new buckets
- Writing (ingesting) 100,000 objects per bucket (64KB payload size)
- Reading as many objects as possible in a 300-second period

Each of the 14 buckets was filled and performance-tested in parallel. The fill procedure was done using the special COSbench work type "prepare" with a single worker thread. This prepare mode allowed us to ensure that each object is written out exactly one time. Reading, as well as the mixed workload, was done using eight worker threads for each RGW using the regular COSbench work modes.

Performance results

Ceph is designed to be an inherently scalable system. By carrying out a 1 billion-object ingestion test, we sought to stress a single, but very important dimension of Ceph's scalability. In this section we will share our findings that we captured while ingesting one billion objects to the Ceph cluster.

Read performance

Figure 18 represents read performance measured in aggregated throughput (OPS) metrics. Figure 19 shows average read latency, measured in milliseconds (blue line). Both of these charts show strong and consistent read performance from the Ceph cluster while the test suite ingested more than one billion objects. The read throughput stayed in the range of 15K OPS - 10K OPS across the duration of the test. This variability in performance could be related to high storage capacity consumption (~ 90%) as well as old large object deletions and re-balancing operations occurring in the background.

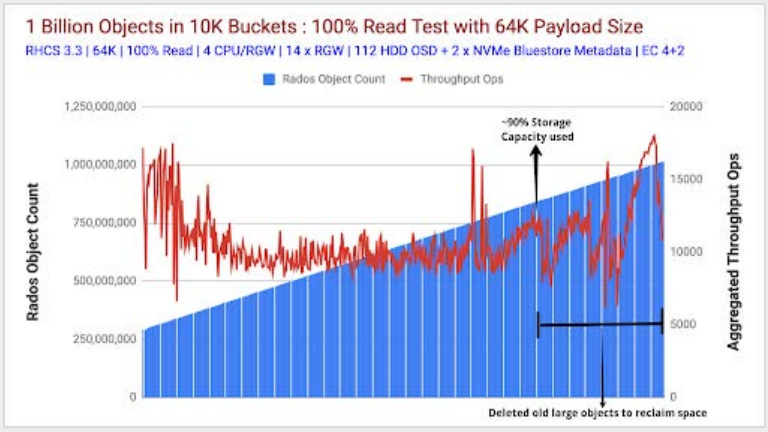

Figure 18 Object Count vs Aggregated Read Throughput Ops

Figure 19 compares the average latency in milliseconds (measured from the client side) for both read and write tests, as we ingested the objects. Given the scale of this test, both read and write latencies stayed very consistent until we ran out of storage capacity and a high amount of Bluestore meta-data spill-over occurred (from flash to slower devices).

The first half of the test shows that write-latencies stayed lower compared to the latency of the read operations. This could possibly be a Bluestore effect. The performance tests we did in the past showed a similar behaviour where Bluestore write latencies were found to be slightly lower than Bluestore read latencies, possibly because Bluestore does not rely on Linux page cache for read aheads and OS level caching.

In the later half of the test, read latency stayed lower compared to write, which could be possibly related to the Bluestore meta-data spill over from flash to slow HDDs (explained in more depth in the next section).

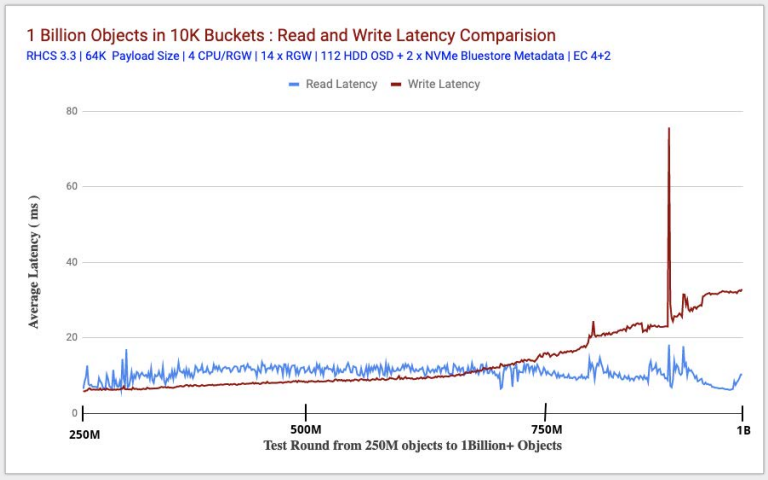

Figure 19. Read and Write Latency Comparison

Write performance

Figure 20 represents ingestion (write operations) of 1 billion 64K objects to our Ceph cluster through its Amazon S3 interface. The test started at around 290 million objects which were already stored in the Ceph cluster.

This data that was created by the previous test runs, which we chose not to delete and started to fill the cluster from this point until we reached more than 1 billion objects. We executed more than 600 unique tests and filled the cluster with 1 billion objects. During the course, we measured metrics like total object count, write and read throughput (Ops), read and write average latency(ms), etc.

At around 500 million objects, the cluster reached 50% of its available capacity and we observed a down trend in aggregated write throughput performance. After several hundreds of tests, aggregated write throughput continues to go down, while cluster used capacity reached an alarming 90%.

From this point, in order for us to reach our goal, i.e., ingesting 1 billion objects, we needed more free capacity, hence we deleted / re-balanced old objects which were larger than 64KB.

Generally, as we know, the performance of storage systems declines gradually as the overall consumption grows. We observed a similar behavior with Ceph, at approx 90% of used capacity, aggregated throughput declined compared to what we started with initially. As such, we believe that had we added more storage nodes to keep the utilized percentage low, the performance might not have suffered at the same rate as we have observed.

Another interesting observation that could potentially explain this declining aggregated performance is the frequent NVMe to HDD spillover of Bluestore metadata. We ingested approximately 1 billion new objects, which generated a lot of Bluestore metadata. By design, Bluestore metadata gets stored in RocksDB, and it's recommended to have this partition on Flash media, in our case we used 80GB NVMe partition per OSD, which is shared between Bluestore RocksDB and WAL.

RocksDB internally uses level style compaction, in which files in RocksDB are organized in multiple levels. For example Level-0(L0), Level-1(L1) and so on. Level-0 is special, where in-memory write buffers (memtables) are flushed to files, and it contains the newest data. Higher levels contain older data.

When L0 files reach a specific threshold (configurable using *level0_file_num_compaction_trigger*) they are merged into L1. All non-0 levels have a target size. RocksDB's compaction goal is to restrict the data size in each level to be under the target. The target size is calculated as level base size x 10 as the next level multiplier. As such, L0 target size is (250MB), L1 (250MB), L2(2,500MB), L3(25,000MB) and so on.

The sum of the target sizes of all levels is the total amount of RocksDB storage you need. As recommended by Bluestore configuration RocksDB storage should use flash media. In case we do not provide enough flash capacity to RocksDB to store its Levels, RocksDB spills the level data on to slow devices such as HDDs. After all, the data has to be stored somewhere. This spilling of RocksDB metadata from flash devices on to HDDs deteriorates performance significantly.

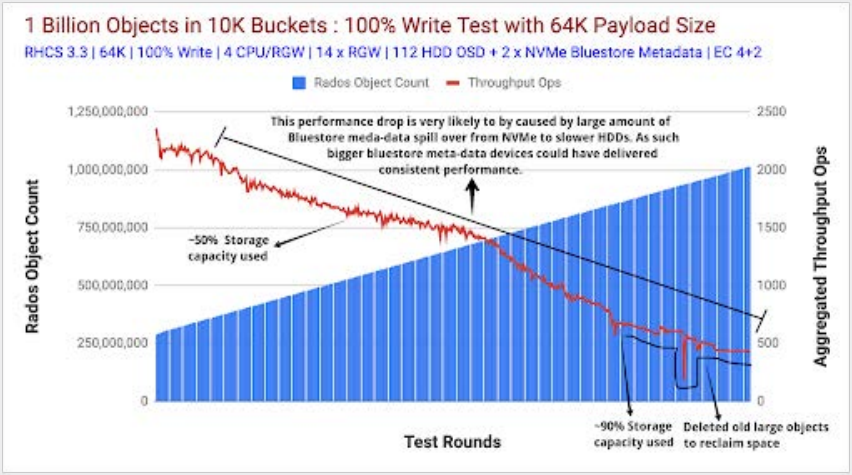

Figure 20. Object Count vs Aggregated Write Throughput Ops

As shown in Figure 21, the spill-over meta-data reached upwards of 80+GB per OSD while we ingested 1 billion+ objects into the system. Our hypothesis is that this frequent spill-over of Bluestore metadata from flash media to slow media is the reason for decremental aggregated performance in our case. As such, if you know that your use case would involve storing several billions of objects on a Ceph cluster, the performance impact could potentially be mitigated by using large flash partitions per Ceph OSD for BlueStore (RocksDB) metadata, accommodating up toRocksDB L4 files on flash.

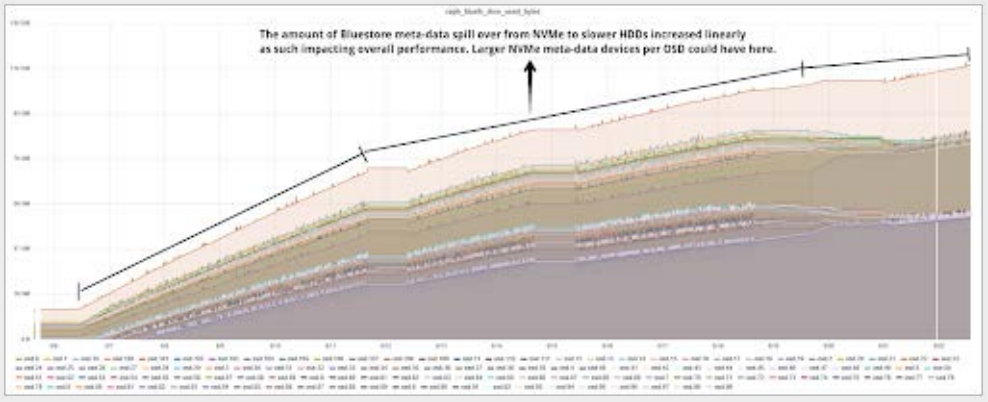

Figure 21. Bluestore metadata spill-over from flash to slower HDDs

Additional findings

In the process of testing the cluster, engineers made additional observations worth noting.

Deleting objects at scale

When we ran out of cluster storage capacity, there was no choice but to delete old large objects stored in buckets, and there were several millions of these objects in the cluster. We initially started with the S3 API DELETE method, but soon realized that it's not applicable for bucket deletion. As such, all the objects from the bucket must be deleted before the bucket itself could be deleted.

Another S3 API limitation we encountered is that it can only delete 1,000 objects per API request. We had several hundreds of buckets and each with 100,000 objects, so it was not practical for us to delete millions of objects using S3 API DELETE method.

Fortunately, deleting buckets loaded with objects is supported using the native RADOS Gateway API which is exposed using radosgw-admin command line interface (CLI) tool. By using the native RADOS Gateway API, it required only a few seconds to delete millions of objects. Ceph's native API is highly useful for deleting objects at any scale.

Adjusting the *bluestore_min_alloc_size_hdd* parameter

As described, our testing was done on an erasure-coded pool with a 4+2 configuration. As such, every 64K payload had to be split into four chunks of 16KB each. The *bluestore_min_alloc_size_hdd* parameter used by Bluestore, represents the minimal size of the blob created for an object stored in Ceph Bluestore objectstore and its default value is 64KB. Therefore, in our case each 16KB erasure-coded chunk would be allocated 64KB of space, which will cause 48KB overhead of unused space that cannot be further utilized.

After our 1 billion object ingestion test, we decided to lower *bluestore_min_alloc_size_hdd* to from 64KB (default) to 18KB, and re-test. As represented in Figure 22, the object creation rate was found to be notably reduced after this change, leading to the following conclusions:

- The default value (64KB) seems to be optimal for objects larger than the *bluestore_min_alloc_size_hdd* parameter.
- Smaller objects require more investigation. Note that the *bluestore_min_alloc_size_hdd* parameter cannot be set lower then the *bdev_block_size* (default 4096 - 4kB)

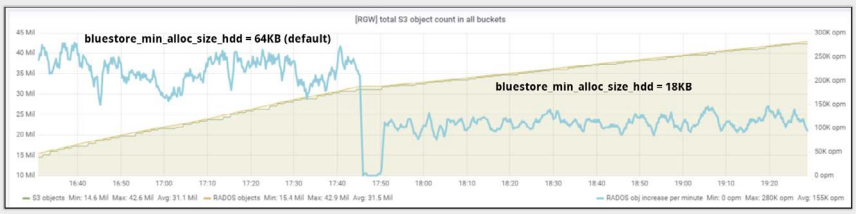

Figure 22. Object ingestion rate per minute, comparing different bluestore_min_alloc_size_hdd values

Performance investigation and guidance

With well over 1,200 workload runs in COSbench, testing measured several additional factors related to storage cluster performance. These factors may be useful in correctly scaling Ceph clusters with Red Hat Ceph Storage 3.3.

RGW dynamic bucket sharding

Red Hat Ceph Storage 3.0 introduced a dynamic bucket resharding capability that automatically spreads bucket indices across multiple RADOS objects. Dynamic resharding is now a default feature, requiring no action by the administrator. With this feature, bucket indices will now reshard automatically as the number of objects in the bucket grows. Also, there is no need to stop the I/O operation that goes to the ongoing resharding bucket. Dynamic resharding is a native RGW feature, where RGW automatically identifies a bucket that needs to be resharded if the number of objects in that bucket is too high. RGW schedules resharding for the buckets by spawning a special thread that is responsible for processing the scheduled reshard operation.

Red Hat engineers wanted to understand the performance ramifications associated with dynamic resharding, as well understanding how sharding can be minimized by using pre-sharded buckets.

Test methodology

To study the performance implications associated while storing a large number of objects in a single bucket as well as dynamic bucket resharding, engineers intentionally used a single bucket for each test type. Also, the buckets were created using default Red Hat Ceph Storage 3.3 tunings. The tests were comprised of two types:

- Dynamic bucket resharding, where a single bucket stored up to 30 million objects
- Pre-sharded bucket, where the bucket was populated with approximately 200 million objects

For each type of test, COSBench testing was divided into 50 rounds, where each round wrote for one hour followed by 15 minutes of read and RWLD (70Read, 20Write, 5List, 5Delete) operations respectively. As such, we wrote over ~245 million objects across two buckets during the entire test cycle.

Performance effects of dynamic bucket resharding

As explained, dynamic bucket resharding is a Red Hat Ceph Storage feature that automatically engages when the number of stored objects in the bucket crosses a certain threshold. Figure 23 shows performance change (in terms of throughput) while continuously filling up the bucket with objects.

- The first round of testing delivered approximately 5.4K OPS while storing roughly 800,000 objects in the bucket under test.
- Test round #44 delivered approximately 3.9K OPS while bucket object count exceeded 30 million.

Corresponding to the growth in object count, the bucket shard count also increased from 16 (default) at test round #1 to 512 at the end of test round #44. The sudden plunge in throughput as represented in the figure can most likely be attributed to RGW dynamic resharding activity on the bucket.

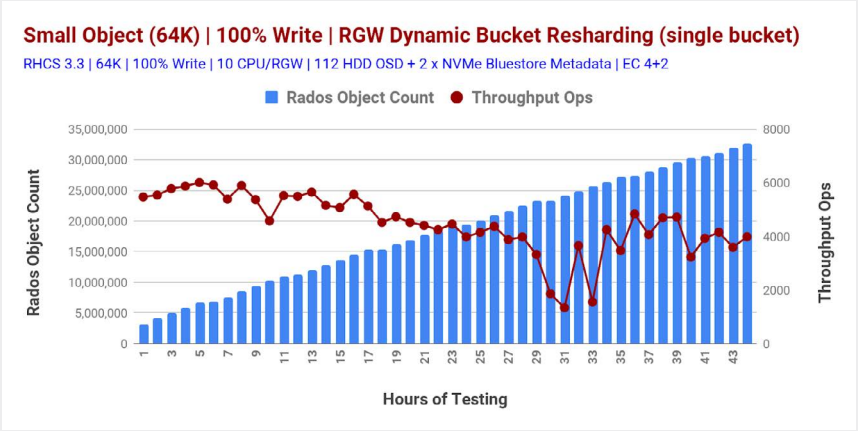

Figure 23. Performance (throughput) effects of RGW dynamic bucket resharding as object count grows

Performance effects of pre-sharded buckets

The non-deterministic performance with an overly populated bucket led engineers to experiment with pre-shardeding the bucket in advance, before storing any objects in it. For this testing we stored (wrote) over 190 million objects in that pre-sharded bucket and measured the performance over time (Figure 24). We observed largely stable performance with the pre-sharded bucket. However, there were two sudden plunges in performance at the 14th and 28th hour of testing, which engineers attributed to RGW dynamic bucket sharding.

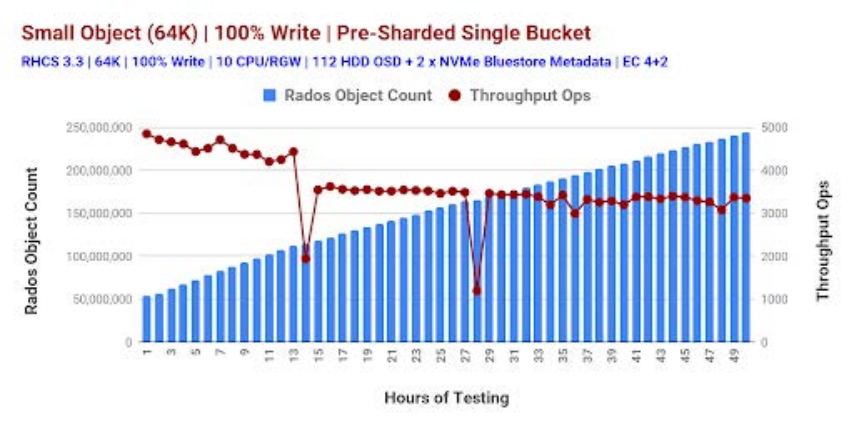

Figure 24. Performance (throughput) effects with a pre-sharded bucket

Figure 25 illustrates a head-to-head performance comparison of dynamically sharded and pre-sharded buckets. Based on post-test statistics, we believe that the sudden performance plunges for both categories were caused by the dynamic resharding event.

Pre-sharding the bucket helped achieve deterministic performance, resulting in the following guidance:

- If the application's object storage consumption pattern is known, (specifically the expected number of objects per bucket) pre-sharding the bucket generally helps normalize performance.
- If the number of objects to be stored per bucket is unknown, dynamic bucket resharding helps to avoid degraded performance associated with overloaded buckets, imposing a minor performance penalty at the time of resharding.

Our testing methodology exaggerates the impact of these events at the cluster level. During the testing, each client writes to a distinct bucket, and each of the clients has a tendency to write objects at a similar rate. As a result, buckets tend to surpass dynamic sharding thresholds with similar timing in our testing, increasing the coincidence of resharding events. In real-world environments, it is more likely that dynamic sharding events would be better distributed over time.

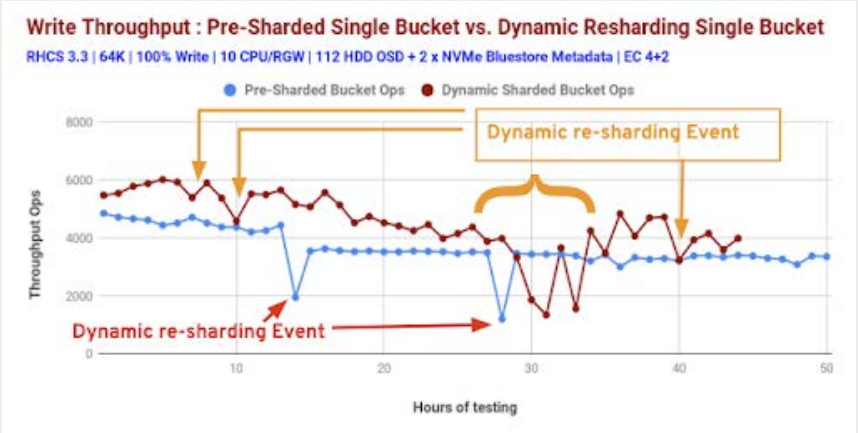

Figure 25. Dynamic bucket resharding and pre-sharding bucket performance comparison (100% write workload)

The read performance of a dynamically resharded bucket was found to be slightly higher compared to a pre-sharded bucket. However the pre-sharded bucket showed deterministic performance as represented in Figure 26.

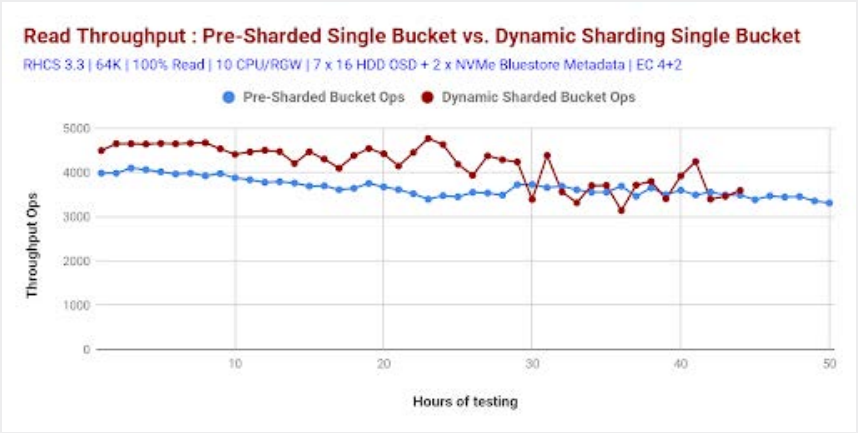

Figure 26. Dynamic bucket resharding and Pre-sharding bucket performance comparison (100% read workload)

Beast vs. Civetweb front-ends

The RGW Beast front-end is one of the new features supported with Red Hat Ceph Storage 3.3. The Beast frontend uses the Boost.Beast library for HTTP parsing and the Boost.Asio library for asynchronous network I/O. Engineers were curious to understand how the Beast frontend compared with with the previously-supported Civetweb frontend in terms of performance.

Configuration

The main difference in configuring the Civetweb and the Beast frontend comes into play when specifying the endpoint. With Civetweb this is combined with the listening port, for Beast you have an extra option for specifying an endpoint.

Performance

As an initial test, engineers sought to understand the impact of thread count on Beast front-end performance. Earlier Civetweb guidance was to increase thread count to obtain better performance. With testing, it was clear that Beast front-end performance was not as sensitive and did not change significantly with thread count (Figures 27 and 28).

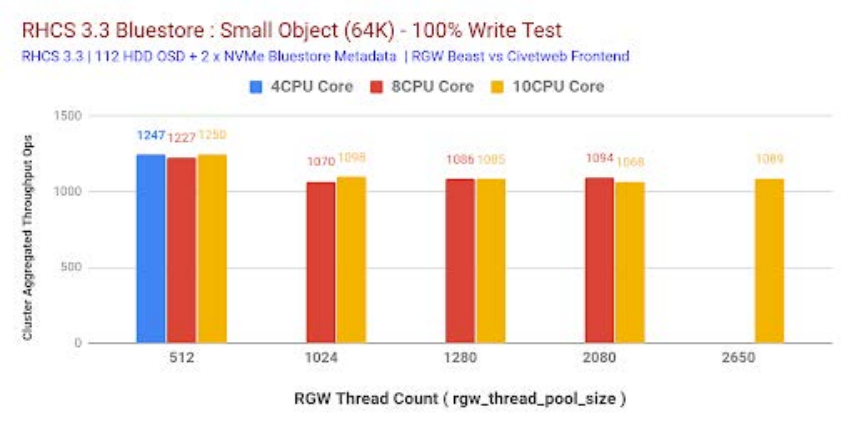

Figure 27. Small-object Beast front-end performance with varying thread count

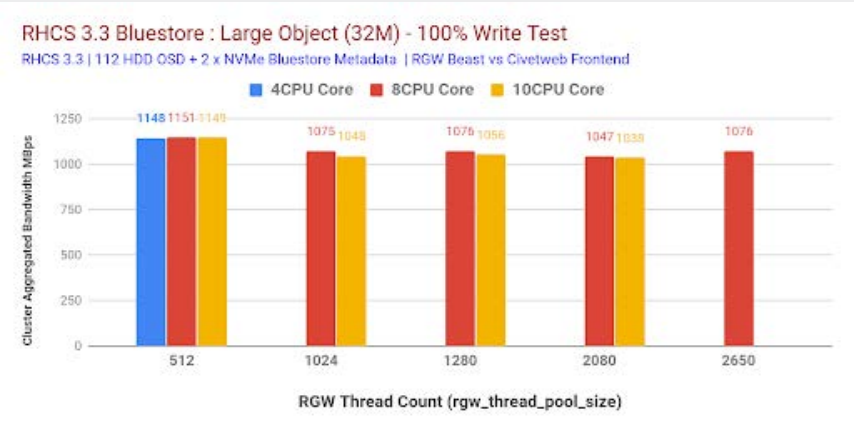

Figure 28. Large-object Beast front-end performance with varying thread count

Next, engineers wanted to analyse how performance would compare between Beast and Civetweb front-ends when using multiple RGW frontends in parallel. As shown in Figure 29, Civetweb is slightly faster than Beast when reading objects from a small number of RGWs in parallel. For most other scenarios, however, Beast performance exceeded that of Civetweb (Figures 30-32).

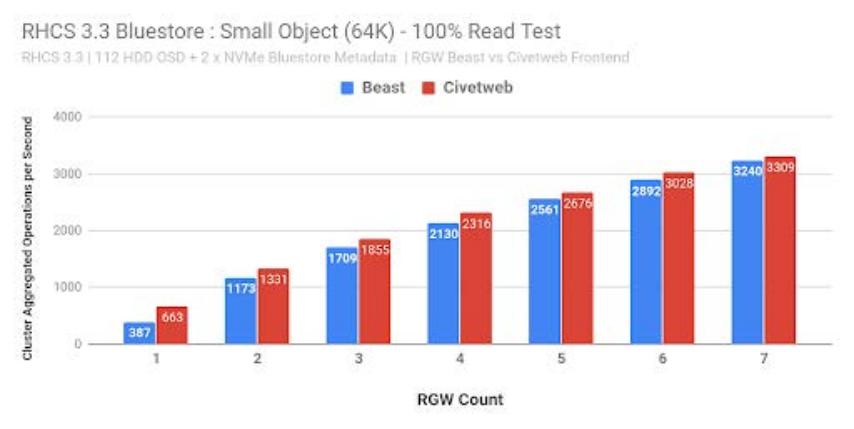

Figure 29. Beast vs. Civetweb front-ends for small objects (100% read test)

Figure 30. Beast vs. Civetweb front-ends for large objects (100% read test)

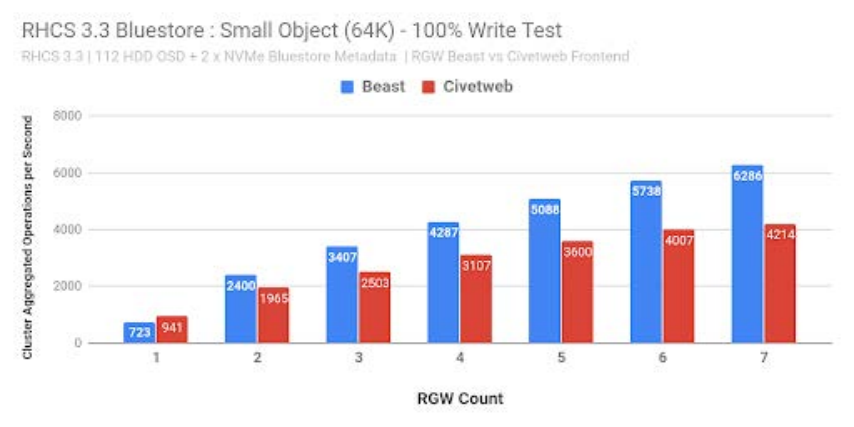

Figure 31. Beast vs. Civetweb front-ends for small objects (100% write test)

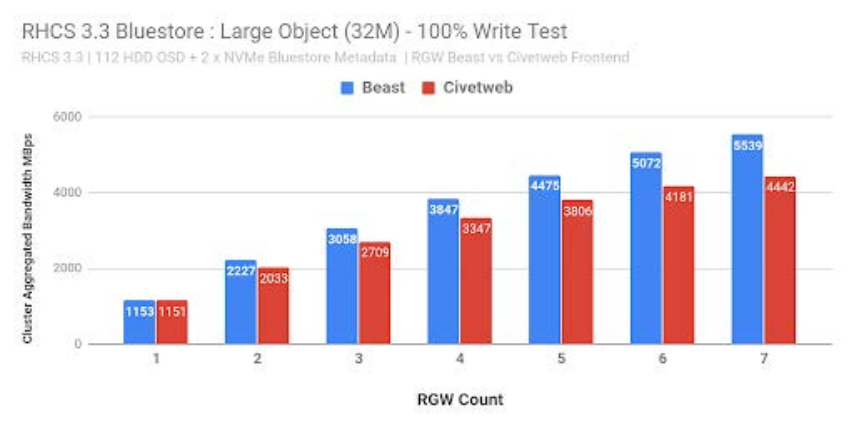

Figure 32. Beast vs. Civetweb front-ends for large objects (100% write test)

Erasure-coding fast_read vs. standard read

Erasure-coded pools can have *fast_read* enabled to better allocate resources and enhance performance.[7] In Red Hat testing, engineers wanted to better understand the scenarios where this enhancement is helpful, and how it affects overall cluster performance.

7 https://access.redhat.com/documentation/en-us/red_hat_ceph_storage/3/html/storage_strategies_guide/pools-1#pool_values

In practice, fast_read queries all erasure-coded shards for data, and returns the data to the client when enough shards have returned. For example, in a *k+n* erasure-coded pool, only *k* shards are needed to reconstruct the original data. As such, our expectation was that bandwidth would probably stay the same, while the latency would improve.

Figure 33 illustrates measures for the small-object workload, combining bandwidth (bars) and latency (depicted as a line). While *fast_read* slightly reduces the bandwidth, the latency is consistent and usually lower than the latency of the regular read.

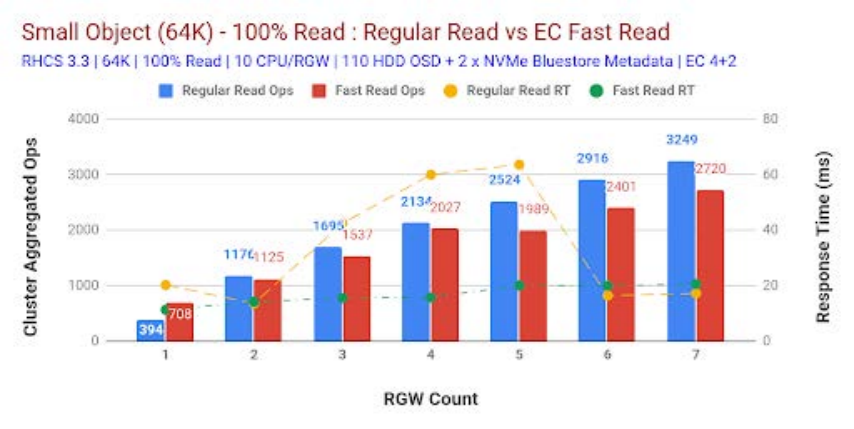

Figure 33. Bandwidth and latency comparison (small objects, regular read vs. fast_read)

Large-object testing revealed the same pattern. Figure 34 shows response time (latency) for large-object workloads. Again, the latency with the *fast_read* option enabled is consistent across all tests, while the regular-read latency is variable and often much higher.

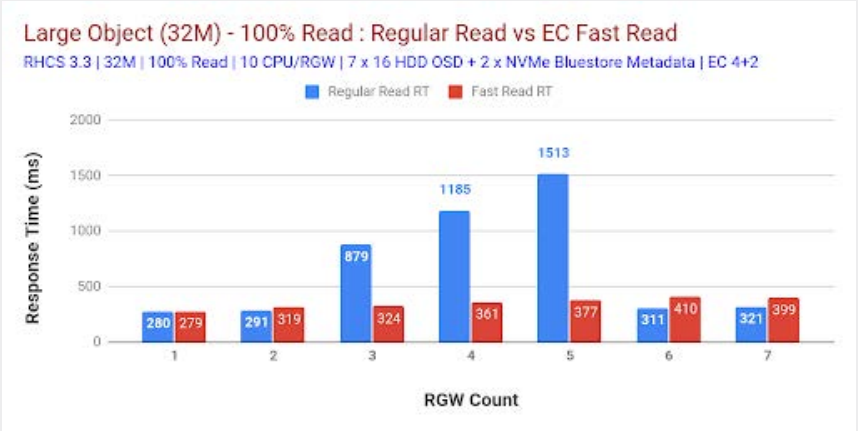

Figure 34. Bandwidth and latency comparison (large objects, regular read vs. fast_read)

The *fast_read* option generally improves overall latency, sometimes more than halving the latency over regular reads. This improvement in latency comes at a cost, however. With the *fast_read* option enabled, we observed a slightly reduced bandwidth.

Summary

Extensive testing described in this document demonstrated that Red Hat Ceph Storage combined with Dell EMC servers and Intel Optane NVMe technology provides a robust and scalable object storage solution. Engineers evaluated a wide range of optimization techniques, including using the Beast.Asio web server, co-locating containerized Ceph OSD daemons, increasing the number of RGW instances, and resharding buckets for consistent performance. The test cluster demonstrated near-linear scalability for large-object workloads (both read and write) with the addition of RGW instances. As suspected, small-object workloads proved to be more susceptible to metadata I/O operations than large-object workloads but performance scaled nonetheless. Importantly, the test cluster was able to scale effectively to store over 1 billion objects.

Appendix: Comparing Red Hat Ceph Storage 3.3 BlueStore/Beast with Red Hat Ceph Storage 2.0 FileStore performance

As a part of Red Hat testing, engineers wanted to compare the performance of the most recent Red Hat Ceph Storage release and technology with an earlier effort. Though we were conscious that results from both these performance studies are not scientifically comparable, we believe that comparing the two should provide performance insights and enables you to take an informed decision when it comes to architecting your Ceph Storage clusters. Table 7 lists the compared systems and components.

Table 7. Current vs. legacy Red Hat Ceph Storage configurations

Software version	RGS front-end Webserver	OSD backend	Hardware
Red Hat Ceph Storage 3.3	Beast	BlueStore	Dell EMC storage servers
Red Hat Ceph Storage 2.0 (mid-2017 release)	Civetweb	FileStore	QCT QuantaPlex T21P-4U storage servers

Test lab configurations

Table 8 lists the configuration details for both environments under comparison.

Table 8. Red Hat Ceph Storage 2.0 and 3.3 configurations

Software environment	
Red Hat Ceph Storage 2.0 • Filestore OSD backend • Civitweb RGW frontend Red Hat Enterprise Linux 7.2 COSBench 0.4.2.c3 Intel CAS 03.01.01	Red Hat Ceph Storage 3.3 • BlueStore OSD backend • Beast RGW frontend Red Hat Enterprise Linux 7.6 COSBench 0.4.2.c3
OSD nodes	
6x QCT QuantaPlex T21P-4U • 2x Intel Xeon E5-2660 v3 • 128GB memory • 35x 6TB SATA HDDs, 7.2K RPM • 2x Intel SSD P3700 800G NVMe AIC • 1x MT27520 Mellanox ConnectX-3 Pro (40GbE)	7x Dell EMC PowerEdge R740xd: • 2x Intel Xeon Gold 5215 (Cascade Lake) • 192GB memory • 16x 4TB SAS HDDs, 7.2K RPM • 2x Intel Optane SSD DC P4800X 750GB NVMe AIC • 2x Intel XXV710 25GbE NICs (dual port)

COSBench worker nodes	
11x QCT QuantaPlex T41S-2U: • 1x MT27520 Mellanox ConnectX-3 Pro (40GbE)	7x Dell EMC PowerEdge R630: • 1x Intel XXV710 25GbE NIC (dual port)

It was no surprise that the Red Hat Ceph Storage 3.3 outperformed RHCS 2.0 for both small and large as well as 100% read and write categories. We believe these improvements can be attributed to the combination of the BlueStore OSD backend, the new Beast web frontend for RGW, the use of Intel Optane SSDs for BlueStore WAL and block.db, and the latest generation processing power as provided by Intel Cascade Lake processors.

Small-object performance

Figure 35 compares the performance of a small-object 100% write workload for both the Red Hat Ceph Storage 3.3 and 2.0 test configurations. As shown in the chart, Red Hat Ceph Storage 3.3 consistently delivered performance in terms of operations per second that is 5x higher than for the Red Hat Ceph Storage 2.0 configuration. We observed over 500% higher throughput OPS with Red Hat Ceph Storage 3.3. We assume that the two sharp performance degradation points in the Red Hat Ceph Storage 3.3 performance line are due to RGW dynamic bucket re-sharding events. These events have been described elsewhere in this document.

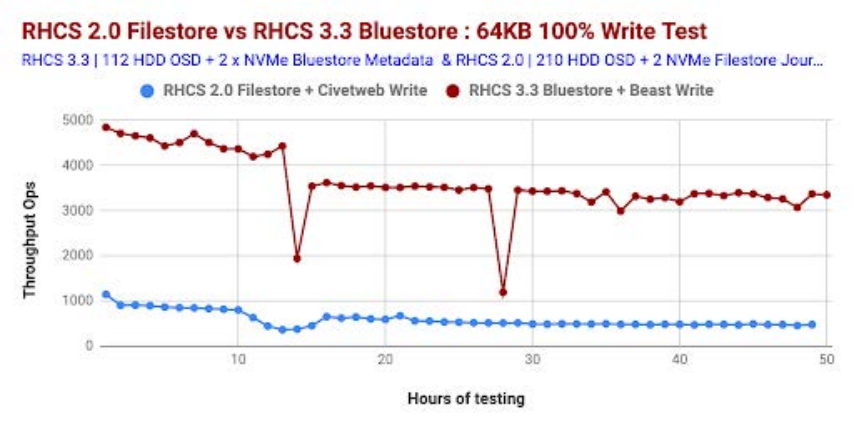

Figure 35. Small-object 100% write performance

The 100% read workload for Red Hat Ceph Storage 3.3 showed near perfect deterministic performance compared to 2.0 test where performance decreased markedly after hours of testing (Figure 36). 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.

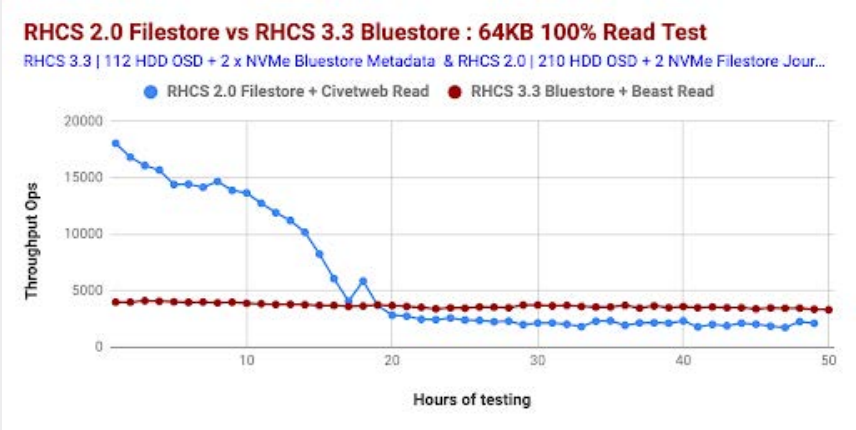

Figure 36. Small-object 100% read performance

Large-object performance

For the large-object 100% write test, Red Hat Ceph Storage 3.3 delivered sub-linear performance improvement, while the Red Hat Ceph Storage 2.0 configuration demonstrated retrograde performance as depicted in Figure 37. As we were lacking a dedicated RGW node for the Red Hat Ceph Storage configuration, we were unable to test beyond four RGWs. Unless saturated by sub-system resource saturation, the Red Hat Ceph Storage 3.3 configuration delivered roughly 5.5 GBps of write bandwidth.

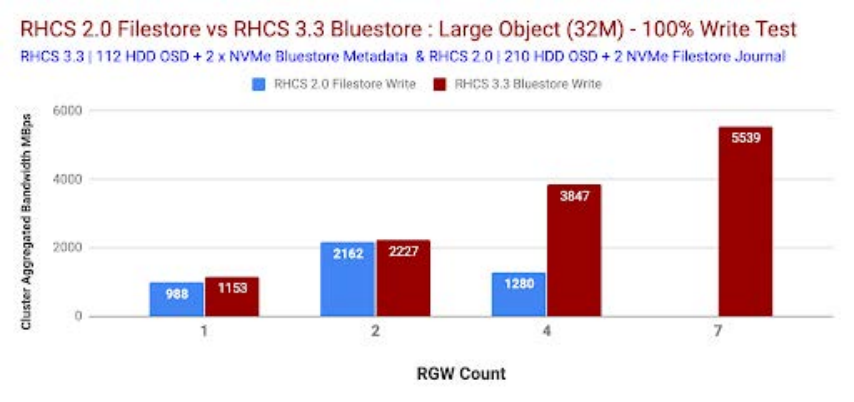

Figure 37. Large-object 100% write performance

The large-object 100% read workload test showed sub-linear performance for both configurations (Figure 38). Again, the Red Hat Ceph Storage 2.0 test was limited to four RGWs. Red Hat Ceph Storage 3.3 showed approximately 5.5 GBps 100% read bandwidth with no bottlenecks observed.

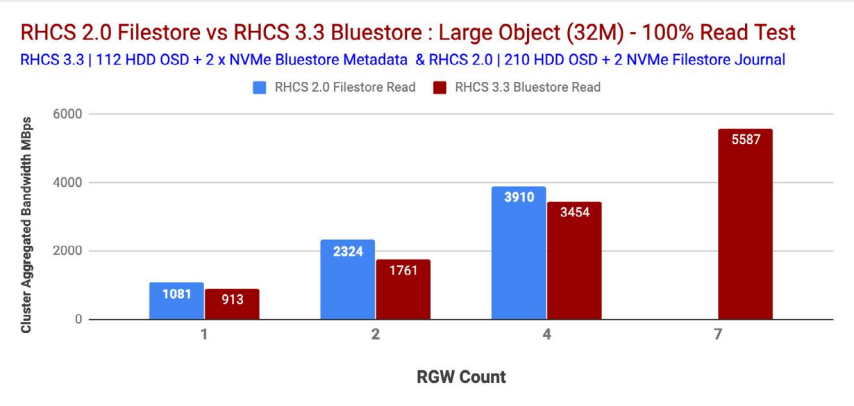

Figure 38. Large-object 100% read performance

Summary

Despite the differences of the two lab test environments, this analysis selected the most common elements of the test environments. Interestingly, the Red Hat Ceph Storage 3.3 test delivered over five-fold higher OPS for small objects and over two-fold higher bandwidth for the large-object 100% write workload. This was accomplished with just half the number of spindles (110 for the Red Hat Ceph Storage 3.3 test vs. 210 for the Red Hat Ceph Storage 2.0 test).

About Red Hat

Red Hat is the world's leading provider of enterprise open source software solutions, using a community-powered approach to deliver reliable and high-performing Linux, hybrid cloud, container, and Kubernetes technologies. Red Hat helps customers integrate new and existing IT applications, develop cloud-native applications, standardize on our industry-leading operating system, and automate, secure, and manage complex environments. Award-winning support, training, and consulting services make Red Hat a trusted adviser to the Fortune 500. As a strategic partner to cloud providers, system integrators, application vendors, customers, and open source communities, Red Hat can help organizations prepare for the digital future.

North America
1 888 REDHAT1
www.redhat.com

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
@RedHat
linkedin.com/company/red-hat

redhat.com
#F22275_0320