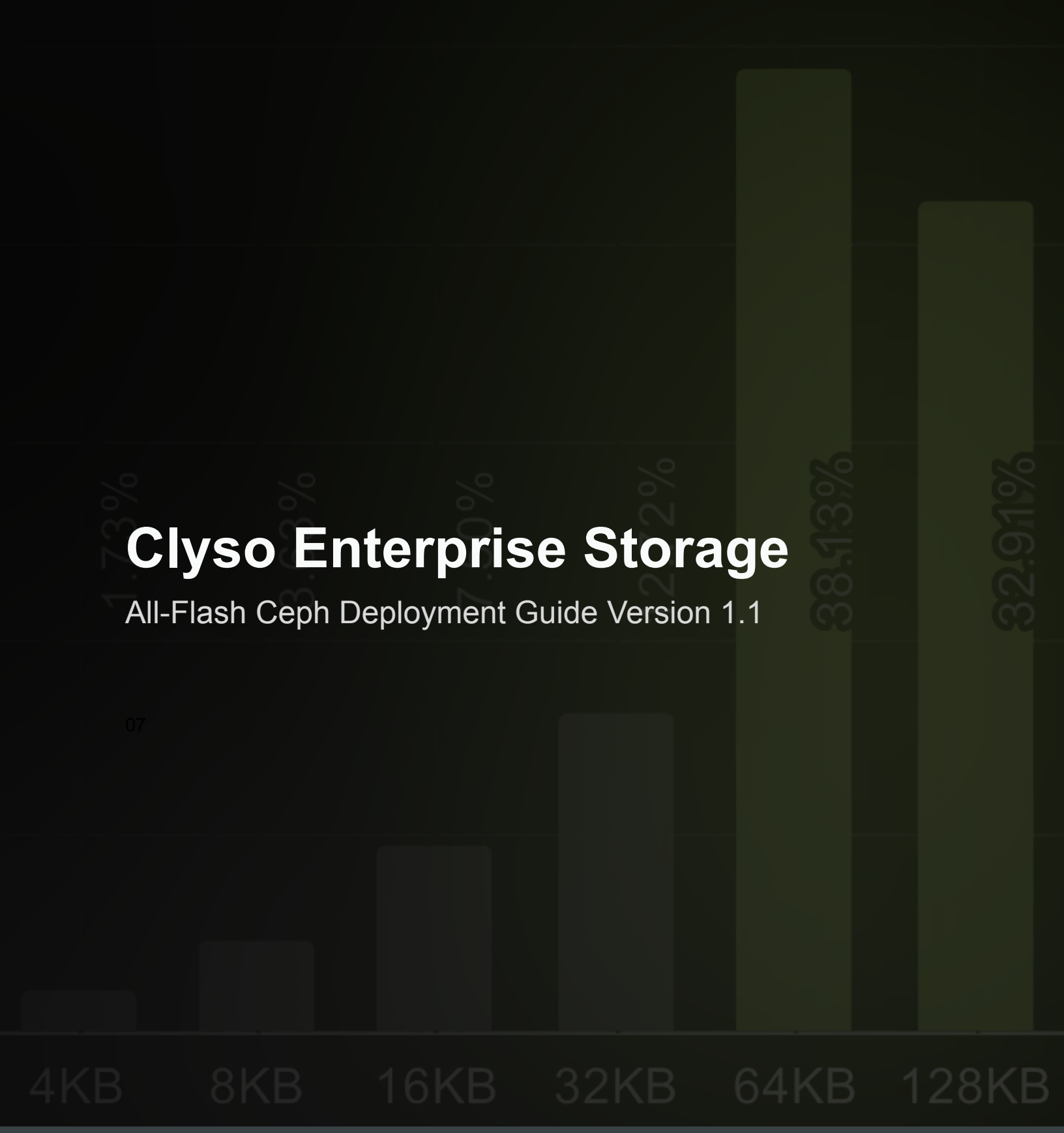


# Clyso Enterprise Storage

## All-Flash Ceph Deployment Guide Version 1.1

07

CLYSO

# Table of Contents

# Forward

This document is intended to provide a sneak preview regarding some of the tuning and optimization ideas that Clyso GmbH is planning for Clyso Enterprise Storage (CES) based on the open source Ceph project. Many of these ideas are also relevant for the upstream community release of Ceph, so we have decided to open this document up for community feedback and discussion as we work on CES. Please keep in mind that this is an initial release of this document and may continue to grow and change over time.

# Introduction

Ceph is a distributed storage platform that provides high scalability and fault tolerance. Using all-flash drives can significantly improve performance and latency for your Ceph cluster.

Before we begin, please note that setting up a Ceph cluster requires knowledge of Linux administration, networking, and storage concepts. It's recommended to have experience with Linux systems and command-line interfaces. It is also helpful to have prior experience designing and tuning high performance server systems.

# Document Version

| Date | Version | Notes |
|---|---|---|
| 2023-06-22 | 1.0 | Initial Version |
| 2023-07-25 | 1.1 | Forward, Branding/Colors, Fixups/Clarifications |

# High Level Strategy and Planning

There are multiple ways to deploy Ceph clusters ranging from container based deployments to bare metal installations using self-compiled binaries or package based installation.

| | |
|---|---|
| * | Container based deployments based on cephadm or Rook have been the official way Ceph is installed since the Pacific release. They offer certain advantages such as reduced OS level packaging dependencies and the isolation, co-location, and migration of Ceph services. The additional complexity of container based deployments can significantly increase the difficulty when debugging and tuning high performance flash based Ceph clusters. |
| * | Bare-metal deployments are more static in nature. Typically the administrator has more control over resource allocation strategies (core/memory pinning, NUMA balancing, etc). Further, it's often easier to attach debugging tools to processes and utilize different tools for performance testing. Upstream Ceph QA testing and most of the upstream Ceph performance testing still takes place on bare metal clusters. Bare-metal deployments may also be a better fit for ansible or other config management systems. It should be noted that cephadm/Rook are the only officially supported and maintained deployment methods as of 2023. Ansible or other bare-metal approaches may still be possible but will require additional work and are not currently supported by the project. |

A container based deployment strategy may be right for you if you already have experience with containers or have little interest in performance tuning and testing. A bare metal deployment may be the right choice if you already have OS and hardware level tuning experience.

# Hardware Recommendations

Ceph has unique hardware needs for each daemon. Below we will briefly explore what each daemon does and how it behaves.

## Ceph Monitors

Ceph monitors maintain information about the state of the cluster including placement groups and OSDs. Clusters should have an odd number of monitors, generally 3 or 5 running on separate hosts. In modern versions of Ceph, monitors store data in RocksDB. Typically the monitors have low system requirements even for all flash deployments and can be co-located

on the same nodes as other daemons, such as OSD servers, provided sufficient RAM and CPU are made available for the MON.

## Ceph Managers

The Ceph manager daemon gathers and maintains run time statistics about the cluster and is responsible for various functions including the dashboard, orchestration, telemetry, and other features. Generally speaking the CPU and memory requirements are low, however it can become quite busy for very large clusters where there is a lot of data to maintain. Generally the manager won't consume more than one CPU core and may benefit from higher CPU clock speeds when busy. Typically for large clusters the rate that statistics are collected can be reduced to also lower resource consumption. These daemons are typically co-located with the Ceph monitors.

## Ceph OSDs

Proper hardware choice for the OSDs is of primary importance for all-flash deployments and will be the key decision that will ultimately govern the overall performance of the cluster. Given the importance of getting this right, we'll break down recommendations into separate sections.

### OSD General Advice

Before getting into the finedetails regarding hardware decisions, a summary of the details will be presented here.

| | |
|---|---|
| 1 | While there is no specific rule regarding how many flash devices to utilize in one node, Clyso has observed that with very high speed NVMe devices it is possible in some cases to hit per-node scaling limits with 4-6 drives. Price, capacity, and performance should be carefully considered when deciding OSD node configurations. Generally smaller nodes with fewer OSDs per node provide higher performance and better fault tolerance while larger nodes provide higher density and lower upfront costs. |
| 2 | OSDs can utilize up to 12+ cores on fast NVMe drives. Expect to potentially be CPU limited for small IO workloads when utilizing fewer (depending on the speed of the flash devices used). |
| 3 | Use Enterprise Class NVMe drives with power loss protection as they are both safer and often faster than consumer class drives. Also consider the TRIM implementation of the device and whether or not online trim is needed and can be used within Ceph. |

| | |
|---|---|
| 4 | Give OSDs enough memory to avoid onode cache misses and potentially more to cache object data. Treat suggestions such as “1GB of memory for 1TB of storage space” as only a very rough estimate. |

## OSD CPU

All-flash Ceph OSDs can utilize significant CPU resources depending on the workload being performed. Unfortunately, some vendors previously recommended that flash based Ceph OSDs only require 2 cores. This is only true in that OSDs can function with 2 cores, but will not perform well. At Clyso we’ve observed that a high performing flash based Ceph cluster can use up to 12 cores per OSD in a multi-node deployment running under maximum load. The number of cores used greatly depends on the workload however. The following table describes the real performance characteristics from a highly tuned, high performing NVMe Ceph cluster tested by Clyso using 3X replication. Please note that this cluster does not represent typical out-of-box Ceph performance. It does however provide a view of what kind of performance can be achieved with careful tuning and hardware planning.

| Workload (3X Rep) | Cores Used / OSD | Perf / OSD |
|---|---:|---:|
| 4MB Read | ~1.9 | 1310 MB/s / OSD |
| 4MB Write* | ~3.2 | 495 MB/s / OSD |
| 128KB Random Read | ~3.8 | 1499 MB/s / OSD |
| 128KB Random Write* | ~7.1 | 359 MB/s / OSD |
| 4KB Random Read | ~7.9 | 79206 IOPS / OSD |
| 4KB Random Write* | ~11.9 | 15966 IOPS / OSD |

** Writes are performing at least 3X the work vs reads due to replication*

For more information about NVMe OSD CPU usage testing, see the Ceph OSD CPU Scaling - Part 1 article posted on the Ceph.io blog.

## OSD NVMe

Ceph OSDs have specific requirements for the underlying storage devices. Specifically, Ceph requires hardware level guarantees that all data for all replicas are safely written before a client is told that a write has succeeded. While many NVMe drives advertise high levels of

performance, far fewer are able to maintain that performance when writes are guaranteed to be persisted. Further, some drives can achieve high burst performance by utilizing a fast tier of persistent SLC flash, but regress under sustained workloads. Generally, enterprise NVMe drives perform better for Ceph than consumer drives as shown below.

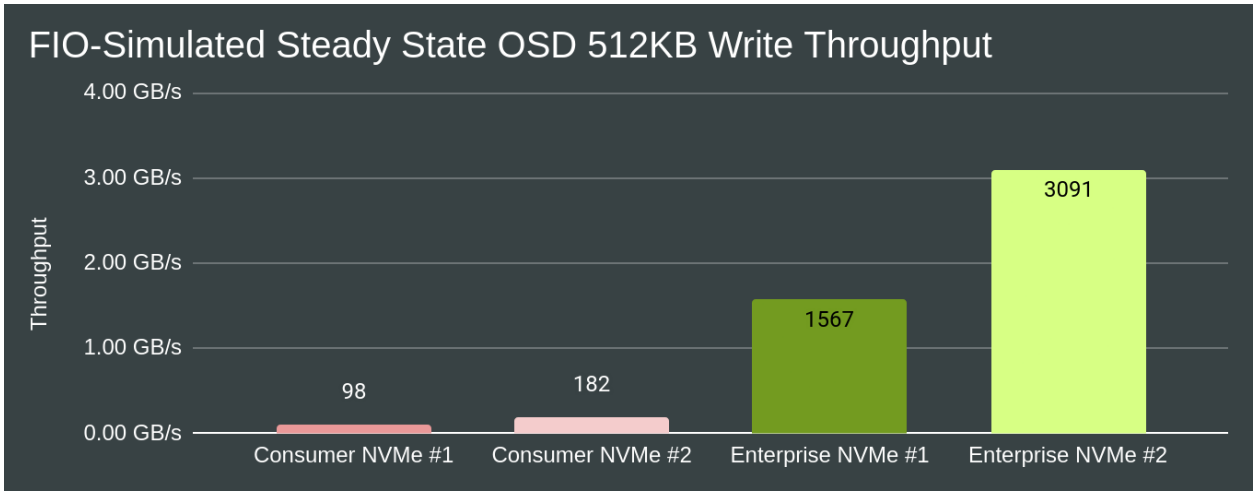


*Chart 1 - FIO-Simulated Steady State OSD 512KB Write Throughput. Enterprise NVMe devices are up to 30x faster than Consumer devices.*

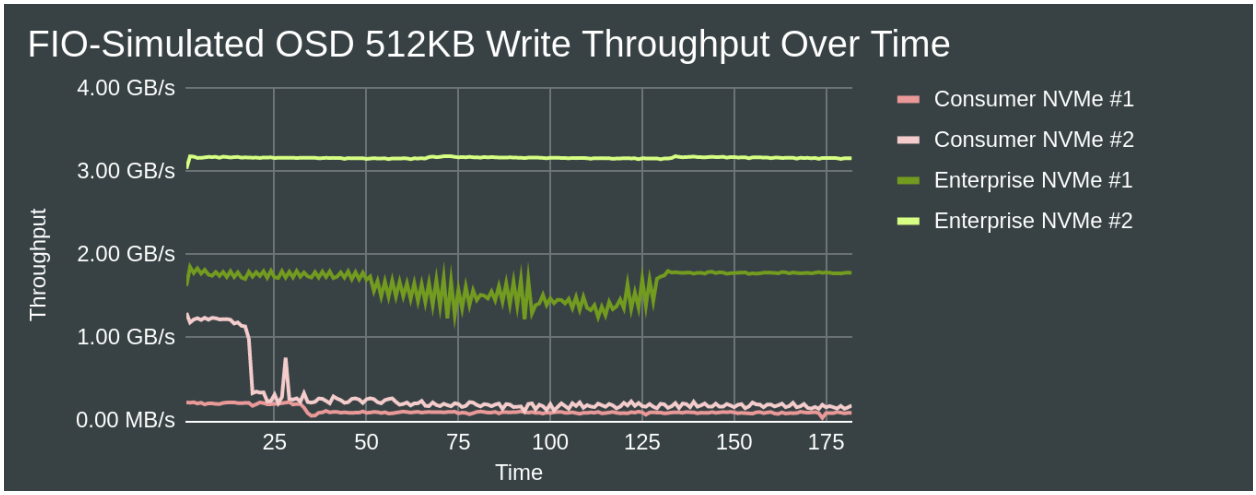


*Chart 2 - FIO-Simulated OSD 512KB Write Throughput Over Time. Enterprise NVMe devices are up to 30x faster than Consumer devices.*

The above charts showcase the performance characteristics of 2 consumer class and 2 enterprise class NVMe drives from several manufacturers running a workload designed to simulate the behavior of Ceph OSDs performing 512KB writes. Both of the enterprise class drives achieve significantly higher performance than the consumer class drives. This is

primarily due to the power loss protection employed on the enterprise class drives that allow them to safely ignore synchronization requests.

Enterprise drives are not only faster, but as their name implies they are more reliable for enterprise use cases such as Ceph. Typically they have higher write endurance than consumer grade drives and better wear-leveling behavior. For read-oriented clusters, enterprise drives that have 1 drive write per day (DWPD) are typically sufficient for multi-year deployments with typical failure rates. For heavy write oriented clusters, drives that have 3 or even 5 DWPD may be desirable depending on the maintenance burden, overall cluster design, and expected service life.

By default, Ceph OSDs do not perform online TRIM/discard operations, as not all drives and firmwares behave well when it is enabled. Carefully consider the impact of trim operations on the device and whether or not it can be enabled on your cluster. Also consider devices that can automatically perform background trim internally.

## OSD Memory

Ceph OSDs use memory for many purposes ranging from keeping in-memory representations of the cluster maps and recent operation logs, to caching object data and metadata. Many Ceph guides recommend 1GB of memory per 1TB of data being stored. While this is a convenient estimation, it doesn't really work well for very small or very large drive sizes. An OSD using a 1TB flash drive will not perform well, or even function consistently, if hard-limited to 1GB of memory. Likewise, the advantage of giving OSDs running on large capacity (30-60TB) drives dozens of GB of memory is very situational. The more accurate answer is that there are multiple consumers of memory inside an OSD. Some consumers have constant needs, some grow with cluster size, and some, like the internal object data and metadata caches, can be scaled based on performance requirements. Early versions of Ceph required users to tune each of these settings individually. Some even had different default settings depending on whether the OSD is running on HDDs or flash devices. This was incredibly confusing for users, so we decided to try to simplify the process and build a memory management system that typically only requires the user to specify a single limit that the OSD should try to stay under called the "osd_memory_target".

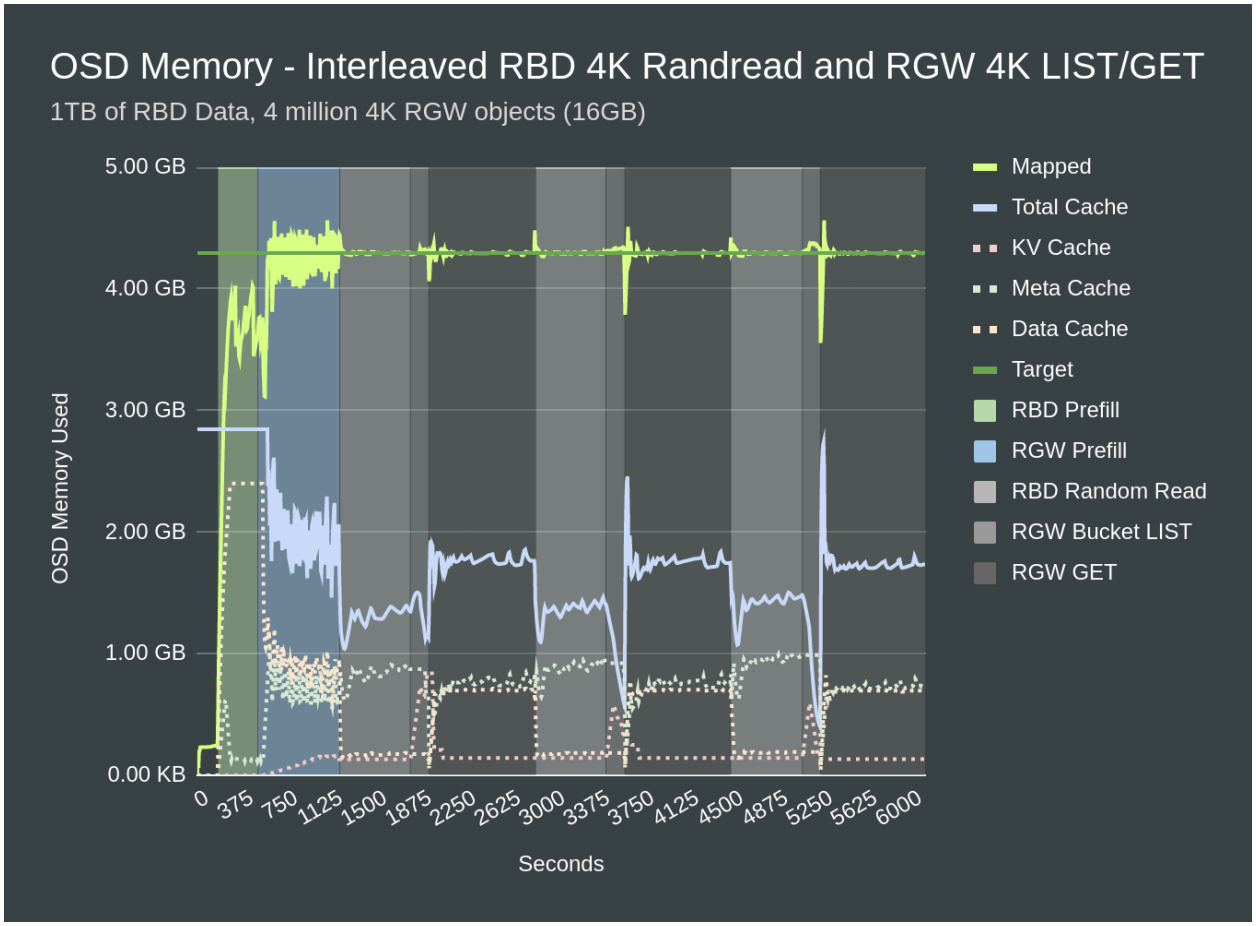


*Chart 3 - OSD Memory - Interleaved RBD 4K Randread and RGW 4K LIST/GET (src)*

The default OSD memory target is 4GB (In containerized deployments, the memory target may be set to a fraction of the container limit when osd_memory_target_autotune is set). That's enough memory to take care of typical OSD memory needs plus around 1-3GB of memory for bluestore caches. The exact amount depends on object and in-memory fragmentation along with other factors. Of the memory assigned to bluestore caches, the most important from a performance perspective is the bluestore meta (onode) cache. On all-flash clusters, onode misses are relatively very expensive and should be minimized as much as possible. As a result, Ceph by default will aggressively prioritize meta cache over data and kv (mostly omap) caches as can be seen in Chart 3. When the workload switches to the RBD 4KB random read phases (shown with a light gray background), the meta cache grows while the data cache shrinks. The overall cache size also shrinks during this period to keep the process' mapped memory below the target. This is a result of increased memory fragmentation during those phases.

To check whether or not onodes are being cached effectively, Ceph provides a variety of per-OSD performance counters that can be checked by running the following command:

```
$ ceph tell osd.N perf dump
```

For example, the following two counters show how effectively the OSD is caching onodes:

```
"onode_hits": 13789150,
"onode_misses": 46427,
```

Clyso recommends that for all-flash clusters, it is beneficial to keep the onode cache hit rate at 90% or higher. As a rule of thumb, scale the osd_memory_size (or the container limit) based on how much hot data you expect to be active on the cluster at the same time. If most of the data stored in the cluster is typically cold, you may be able to stick with the default memory target. If a large percentage of the data is hot, especially on large drives, it may be beneficial to use 8GB or higher memory targets.

In addition to cache hit rate counters, OSDs also provide performance counters showing how Ceph's prioritycache memory management system is prioritizing memory for each cache at different priority levels. For instance, the following snippet shows how memory is being assigned to the bluestore's "meta" cache at different priority levels:

```
"bluestore-pricache:meta":
    "pri0_bytes": 0,
    "pri1_bytes": 29880,
    "pri2_bytes": 178434427,
    "pri3_bytes": 14895681,
    "pri4_bytes": 14940,
    "pri5_bytes": 59761,
    "pri6_bytes": 0,
    "pri7_bytes": 0,
    "pri8_bytes": 0,
    "pri9_bytes": 0,
    "pri10_bytes": 0,
    "pri11_bytes": 0,
    "reserved_bytes": 75000767,
    "committed_bytes": 268435456
```

The priority cache counters provide a detailed view of how Ceph is prioritizing memory allocations for each cache. When viewed over time, you can watch how Ceph changes the priority of different memory allocations based on the current workload.

## RadosGW

With HDD Ceph deployments, a single or modest number of RadosGW instances may be sufficient to provide a balanced architecture. With all-flash deployments, significantly more

RadosGW instances with significantly higher hardware requirements may be necessary to achieve optimal performance. As part of the testing for the Reef release of Ceph, Clyso ran a variety of NVMe backed RGW tests here. What we saw is that when utilizing 20 RGW instances on a 60 NVMe Ceph cluster, we could achieve network saturation during large object tests and relatively heavy OSD load during small object tests. A ratio of 1 RGW daemon for every 3 OSDs is unreasonable at scale however, and the CPU consumption of RGW daemons was significant. In fact the aggregate CPU consumption of RGW daemons was higher than it was for the OSDs in multiple tests:

**4MB Objects:**

| Test | Result | Total RGW Cores | Total OSD Cores | Total Cores | RGW/OSD Core Rato |
|---|---|---|---|---|---|
| 4MB PUT | 18.6GB/s | 184 Cores | 152 Cores | 336 Cores | 6/5 |
| 4MB GET | 53 GB/s | 88 Cores | 55 Cores | 143 Cores | 8/5 |

In the above test, throughput was primarily limited by Network bandwidth, as the RGW daemons were colocated with the OSD nodes and had to share the same 100GbE links.

**4KB Objects:**

| Test | Result | Total RGW Cores | Total OSD Cores | Total Cores | RGW/OSD Core Rato |
|---|---|---|---|---|---|
| 4KB PUT | 178K PUT/s | 269 Cores | 475 Cores | 744 Cores | 11/20 |
| 4KB GET | 312K GET/s | 302 Cores | 102 Cores | 404 Cores | 3/1 |

In small object tests, CPU usage and round-trip latency for bucket metadata played a bigger role in aggregate performance than network throughput. Bucket index contention between the RGW daemons may have played a significant role. As a result, we expect that scaling the number of RGW daemons significantly higher than the 20 tested here may be difficult without careful planning. Clyso believes additional tuning and code optimization may improve RGW CPU usage and performance in the future. RGW may also show higher efficiency (though with lower performance) when fewer daemons are utilized. For now, it may be necessary to allocate significant CPU resources when attempting to achieve maximum performance for RGW daemons backed by a cluster of flash based OSDs. RGW and any HAProxy nodes may also require a significant network bandwidth investment to match the aggregate throughput of the OSD nodes for large object workloads.

## CephFS MDS

Ceph employs one or more metadata servers to service metadata requests for the CephFS distributed file system. A single MDS can be sufficient for all-flash Ceph clusters so long as the number of file and directory metadata operations remains limited. For metadata intensive workloads, multiple active MDSs may need to be deployed. Each MDS has relatively low CPU requirements, typically using no more than 2-3 cores each. High clock speeds are generally beneficial. Ceph MDSes have a distributed cache that can grow quite large with many clients. Under heavy metadata workloads, it may be necessary to assign dozens of GB of memory to the MDS cache. In some heavy metadata use cases a single MDS can use 80GB of memory or more at once.

## Network

With All-Flash Ceph deployments the latency and throughput of the network can have a dramatic impact on overall performance. Clyso's standard recommendation to use 1 interface or 2 interfaces in a bonded configuration with separate VLANs for the front and back network still apply. If two independent interfaces are used, make sure not to put them in the same subnet. It is best to choose high performance NICs with enough throughput to meet recovery and large read/write operation needs. When using NVMe drives, this may mean 1-2GB/s per OSD or more. To the extent possible, attempt to purchase switches that have low latency and full bisection bandwidth. Keep OSDs on the smallest switch topology possible, even just a single switch for smaller deployments. Otherwise, carefully plan the network topology based on overall throughput and latency needs. Use high quality cables and optics and keep runs to the OSD nodes as short as possible.

# Software Configuration

In addition to picking well balanced hardware, it may be necessary to carefully tune certain aspects of both Ceph and the host operating system as well. In container deployments this can be especially tricky to get right as it's not always clear at what layer optimizations should be made. Below we will discuss several high level issues to watch out for.

## Operating System

Ceph can work well with multiple linux distributions, however there are a couple of key issues to consider that are especially important for flash based Ceph deployments.

## TCMalloc

It is critically important that Ceph be compiled with tcmalloc support. Ceph will be both slower and use significantly more memory without it. You can verify this by checking that tcmalloc is included by running the following command:

```
$ ldd /usr/bin/ceph-osd | grep tcmalloc
     libtcmalloc.so.4 => /lib64/libtcmalloc.so.4
```

Note that the ceph-osd executable may be located in a different directory depending on the distribution and method of installation. As of the time of this writing, neither the Ubuntu nor Debian distribution packages for Ceph are compiled with tcmalloc support. Clyso informed Canonical of the issue in May of 2023 and a resolution is in the works. Beyond having Ceph itself compiled with tcmalloc support, it may also be important to have any clients that use the ceph libraries also compiled with tcmalloc support. In the fall of 2022, Clyso's research and development team performed a study on QEMU/KVM client performance and found that compiling QEMU with tcmalloc support made a huge difference in single-client performance when utilzing an all-flash Ceph Cluster. As of the time of this writing, few, if any, distribution packages for QEMU are compiled with tcmalloc support enabled.

| | 16K Randread IOPS | Improvement |
|---|---|---|
| QEMU without TCMalloc | 53.5K IOPS | 0% |
| QEMU with TCMalloc | 80.0K IOPS | 49.5% |

## CPU Power States

Preventing CPUs from transitioning back and forth between low and high power states on OSD nodes can result in a significant latency reduction and performance improvement. One of the easiest ways to accomplish this is to use tuned-adm. Tuned-adm is available on RedHat and Debian based distributions and may be available for others as well. Tuned typically defaults to the throughput-performance profile:

```
$ tuned-adm active
Current active profile: throughput-performance
```

Clyso recommends the latency-performance or network-latency tuned profile as both have several useful default tunings for Ceph. This can be set using the following command:

```
$ sudo tuned-adm profile network-latency
```

## Networking

High speed network interfaces and switches may need additional tuning to achieve the highest performance and lowest latency possible with Ceph. While this guide will not get into every detail of network tuning, we will offer that general advice for tuning high speed networks is applicable. There are several very detailed guides at ESNet that may be of use:

https://fasterdata.es.net/network-tuning/
https://fasterdata.es.net/assets/Papers-and-Publications/100G-Tuning-TechEx2016.tierney.pdf
https://fasterdata.es.net/host-tuning/linux/100g-tuning/

## Cluster Configuration

Most of Ceph's default settings are reasonable, though there are a couple of general details to watch out for.

| | |
|---|---|
| 1 | Debugging, especially at high levels in the messenger or OSD code, can have a dramatic effect on performance and latency with all-flash deployments. It can also very quickly fill up the disk that logs are stored on. Be careful not to leave high debug levels enabled indefinitely. Debugging defaults are generally optimal. |
| 2 | By default RGW records information about every request in the rgw log to improve visibility into access attempts. For high speed flash deployments this leads to extra overhead and disk usage on rgw installations. Disabling the rgw access log may improve performance and efficiency but increase security risk. Clyso does not recommend disabling it for public deployments or in any case where security is a concern. This setting can be controlled with the `debug_rgw_access` config option. |
| 3 | On large clusters with many OSDs (or small clusters where the PG count is higher than normal), the mgr nodes may slow down due to the number of PG stat updates. If this is the case, the frequency of mgr updates can be controlled using the following two options:<br><br>`mgr_tick_period`<br>`mgr_stats_period` |

# OSD Configuration

There are a couple of decisions to make regarding how OSDs are deployed on flash drives. One of the first ones that comes up is whether or not to deploy multiple OSDs on a single flash device. In previous versions of Ceph, if you had enough CPU and memory, performance was almost universally higher on flash drives if you put multiple OSDs on the same device. During the Ceph Pacific development cycle, the performance of the Ceph OSD code was improved to the point where this was not always universally true.

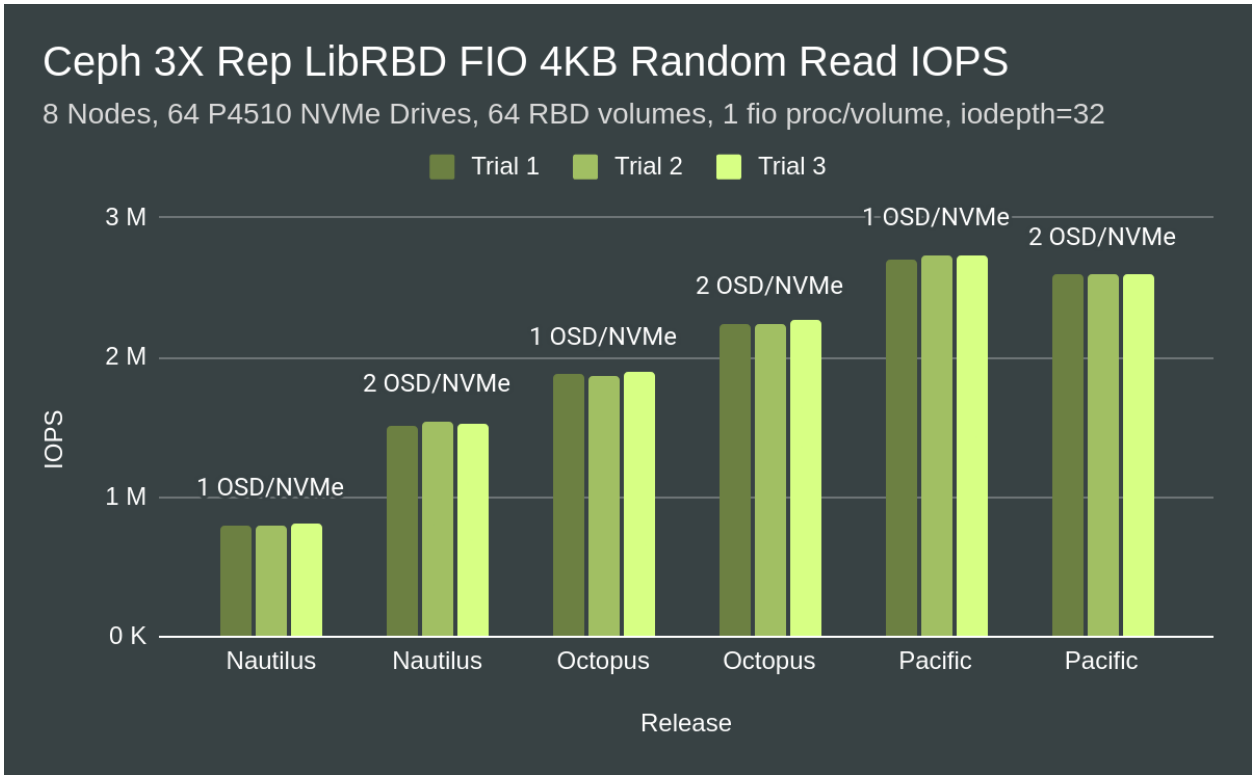


*Chart 4 - Effect of placing multiple OSDs on a single NVMe device by Ceph Release (src)*

Since then, the performance of Ceph OSDs has improved further. Clyso has received reports that on some hardware configurations there are still advantages to putting 2 OSDs on a single NVMe drive, while others show little advantage (and in fact increased memory and potentially CPU consumption with increased context switching overhead). For most SATA/SAS and standard enterprise grade NVMe flash drives, it may no longer be worth the complexity trade-off to double the number of OSDs on each node. For high performance drives, there may still be an advantage. Again, this is likely only true if there are enough resources to fully support doubling the OSDs with the associated increase in resource consumption.

To control the number of OSDs per device, the following orchestration invocation can be used:

```
ceph orch daemon add osd ceph-nvme01:/dev/nvme0n1,osds_per_device=1
```

There are several OSD level tunables that are useful to examine, particularly when Ceph OSDs are CPU constrained or there are many OSDs on a single host.

| Option | Default | Notes |
|---|---|---|
| osd_op_num_shards_ssd | 8 | Large values can improve maximum performance and also improve low-load latency. Small values can improve efficiency and may improve performance when CPU constrained. |
| osd_op_num_threads_per_shard_ssd | 2 | There is little benefit to increasing this in Reef or earlier vs increasing shards. Future benefits may be possible. Decreasing to 1 may increase efficiency in very CPU limited scenarios. |
| ms_async_op_threads | 3 | Little value to increasing this. Decreasing this can improve efficiency and improve performance if there are many OSDs on the same node. Decreasing this may hurt performance when there are fewer OSDs on the same node. |
| osd_op_queue | mclock | For Quincy and earlier, the wpq op queue may behave better during recovery. |

In addition to Ceph OSD tunings, there are a large number of complex RocksDB tunings that can impact Ceph all-flash performance. Clyso R&D researched and authored the new RocksDB tuning that will be released with the Reef release of Ceph in Q2 2023. It may, however, be worthwhile to explore alternate tuning for older releases as well. See the blog post written by Clyso's R&D team here and the new RocksDB tuning we authored for Reef here.

## Replication and Data Placement

Ceph allows several replication strategies for pools. Primarily, the user can choose between replication (typically 3) and erasure coding. There are space amplification, write-amplification, and performance considerations with each.

| **Large Reads** | **Large Writes** |
|---|---|

| | |
|---|---|
| Replication is faster than erasure coding due to all reads serviced from a single OSD instead of fetching chunks from multiple OSDs. Replication has lower network overhead. | Erasure coding is often faster than replication due to lower write-amplification. Erasure coding typically has lower space-amplification as well. |
| **Small Reads**<br><br>Replication is typically much faster than erasure coding due to all reads serviced from a single OSD. Replication has significantly lower latency as no secondary OSDs need to be contacted. | **Small Writes**<br><br>Replication is typically much faster than erasure coding due to lower latency and lower write-amplification. Erasure coding has significant potential for high space-amplification due to Ceph's 4K minimum allocation size. |

Beyond a replication or erasure coding strategy, Ceph pools are configured with a certain number of PGs. By default, ceph will try to automatically scale the number of PGs based on the amount of data in the pool, and further will try to automatically balance the distribution of data across PGs. Unfortunately this is a mixed bag. The process of growing and shrinking the pool PG count imposes extra overhead as a background workload, and often results in low PG counts that hurt performance. Theoretically balancing the distribution of data more evenly across PGs can help, but that's only part of the problem. For all-flash clusters, PG lock contention inside the OSD can itself become a major bottleneck. The following chart showcases how dramatic the PG count in an RBD pool can affect performance:

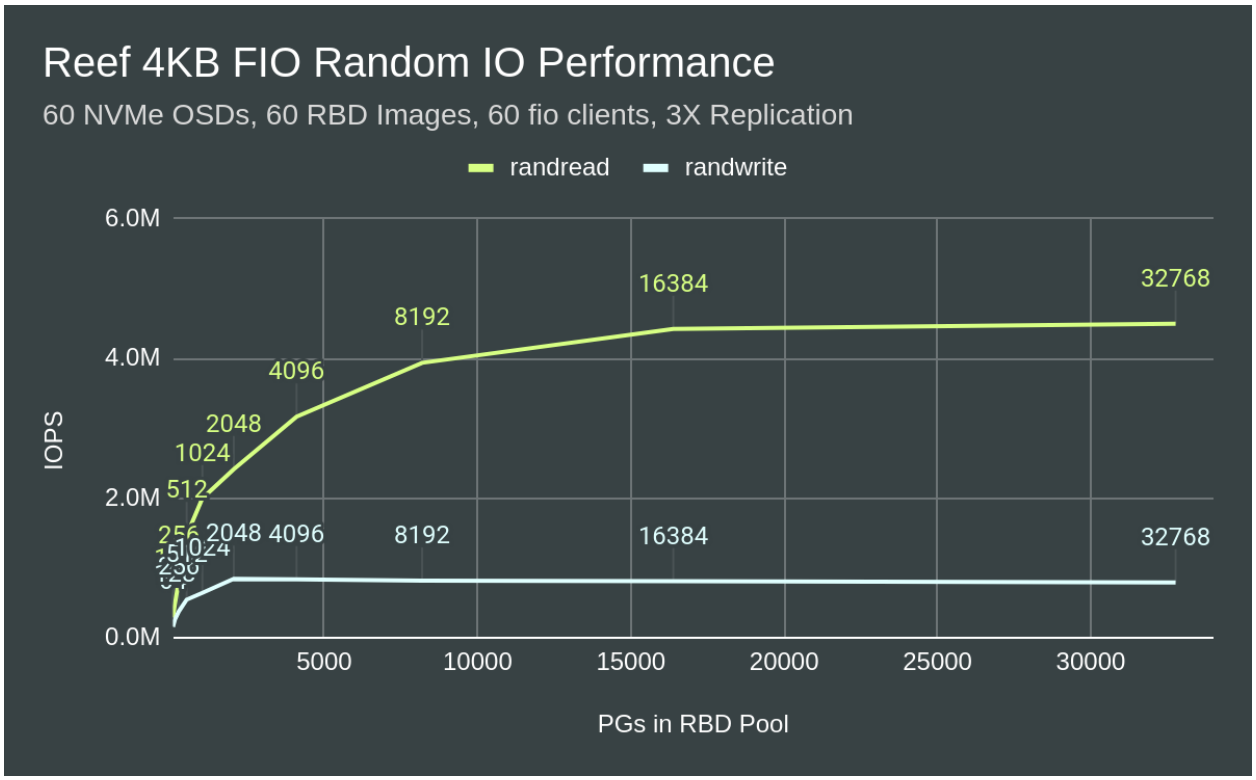


*Chart 5 - The Effect of Pool PG Scaling on RBD Performance (src)*

In the above set of tests, the PG count was increased significantly beyond the default Ceph limits and continued to improve 4K random read performance up to 16384 PGs. There are downsides to setting extremely large PG counts. Without additional tuning, OSDs will consume more memory leaving less for cache and PG stat updates to the manager might cause additional overhead. To set the PG count this high, several options must be set.

For example:

```
osd_pool_default_pg_autoscale_mode = off
mon_pg_warn_max_object_skew = 0.0
mon_pg_warn_min_per_osd = 0
mon_pg_warn_max_per_osd = 32768
mon_max_pg_per_osd = 32768
```

# Testing and Verification

There are many tools that can aid in performance testing for all-flash Ceph clusters. Below is an incomplete list that may be helpful:

| Tool | Description | URL |
|---|---|---|
| cbt | Ceph Benchmarking Tool | https://github.com/ceph/cbt |
| fio | File/Block Benchmark | https://github.com/axboe/fio |
| hsbench | S3 Benchmark | https://github.com/markhpc/hsbench |
| sibench | S3/Ceph Benchmarks | https://github.com/SoftIron/sibench |
| collectl | System Monitoring | https://collectl.sourceforge.net/ |
| perf | CPU Profiler | https://perf.wiki.kernel.org/index.php/Main_Page |
| uwpmp | Wallclock Profiler | https://github.com/markhpc/uwpmp |
| eBPF | Kernel and Userland Tracing | https://ebpf.io/ |

# Monitoring and Maintenance

As the cluster ages and the amount of data stored grows, certain operations in Ceph may become slower. RocksDB for instance may need to traverse through more levels to find object metadata, and the disk allocator may spend slightly more time searching for contiguous free space if bluestore has become fragmented. All-flash Ceph clusters should be more resilient to some of these issues, however there are a couple of things to keep in mind.

| | |
|---|---|
| 1 | Generally it's best to operate below maximum capacity. There is no hard rule regarding how close to full is too close. If you are above 50% capacity however, you may want to carefully watch the growth rate and start planning an expansion. Running near capacity can be dangerous: Losing a rack, node, or even just a drive may cause recovery to fail if there isn't enough capacity remaining to write out new replicas of the data. Recovery itself puts more write load on the devices which increases the probability of a second failure during the recovery process. It can also be slower and harder on the flash devices to run a cluster near maximum capacity when freespace is fragmented. |

| | |
|---|---|
| 2 | Many flash devices require periodic trims/discards to maintain high performance and reduce internal write amplification. The compatibility, overhead, and duration of trim operations is different for every device. Ceph does not enable automatic runtime trimming of devices by default, however does have several options that govern this behavior:<br><br>`bdev_enable_discard`<br>`bldev_async_discard` |
| 3 | It is a good idea to periodically inspect the SMART status of flash based OSD devices and make sure that there are no warnings and/or quickly degrading wear counters shown. Pay attention to the rate of change of the media wearout indicators and plan to replace SSDs well ahead of their wearout. |
| 4 | In rare cases, Clyso has observed that an individual flash device can become extremely slow without showing any signs of poor health via SMART status or other means. This can have a dramatic impact on overall cluster performance. If the cluster shows a sudden degradation in performance without any other explanations, it may be a good idea to individually test OSDs with `ceph osd perf` and `ceph tell osd.X bench` and look for outliers. |

# Conclusion

This concludes our deployment guide for all-flash Ceph clusters. Remember to consult with your vendor or the official Ceph documentation and user community for detailed instructions and configuration references. Deploying an all-flash Ceph cluster requires careful planning and consideration of your workload requirements, so it's essential to thoroughly test and validate the performance and reliability of the setup. Clyso can provide assistance for any step in the process and provide support for your storage needs.