SUSE Enterprise Storage 6

Tuning Guide

Tuning Guide

SUSE Enterprise Storage 6

by David Byte, Lars Marowsky-Bree, Igor Fedotov, Jan Fajerski, Abhishek Lekshmanan, and Alexandra Settle

Publication Date: 08 Jul 2022

https://documentation.suse.com

Contents

I Introduction

1 User Privileges and Command Prompts

As a Ceph cluster administrator, you will be configuring and adjusting the cluster behavior by running specific commands. There are several types of commands you will need:

1.1 Salt/DeepSea Related Commands

These commands help you to deploy or upgrade the Ceph cluster, run commands on several (or all) cluster nodes at the same time, or assist you when adding or removing cluster nodes. The most frequently used are **`salt`**, **`salt-run`**, and **`deepsea`**. You need to run Salt commands on the Salt master node (refer to *Book "Deployment Guide", Chapter 5 "Deploying with DeepSea/Salt", Section 5.2 "Introduction to DeepSea"* for details) as `root`. These commands are introduced with the following prompt:

```
root@master #
```

For example:

```
root@master # salt '*.example.net' test.ping
```

1.2 Ceph Related Commands

These are lower level commands to configure and fine tune all aspects of the cluster and its gateways on the command line, for example **`ceph`**, **`rbd`**, **`radosgw-admin`**, or **`crushtool`**.

To run Ceph related commands, you need to have read access to a Ceph key. The key's capabilities then define your privileges within the Ceph environment. One option is to run Ceph commands as `root` (or via **`sudo`**) and use the unrestricted default keyring 'ceph.client.admin.key'.

Safer and recommended option is to create a more restrictive individual key for each administrator user and put it in a directory where the users can read it, for example:

```
~/.ceph/ceph.client.USERNAME.keyring
```

Tip: Path to Ceph Keys

To use a custom admin user and keyring, you need to specify the user name and path to the key each time you run the **`ceph`** command using the `-n client.USER_NAME` and `--keyring PATH/TO/KEYRING` options.

To avoid this, include these options in the `CEPH_ARGS` variable in the individual users' `~/.bashrc` files.

Although you can run Ceph related commands on any cluster node, we recommend running them on the Admin Node. This documentation uses the `cephadm` user to run the commands, therefore they are introduced with the following prompt:

```
cephadm@adm >
```

For example:

```
cephadm@adm > ceph auth list
```

Tip: Commands for Specific Nodes

If the documentation instructs you to run a command on a cluster node with a specific role, it will be addressed by the prompt. For example:

```
cephadm@mon >
```

1.3 General Linux Commands

Linux commands not related to Ceph or DeepSea, such as **`mount`**, **`cat`**, or **`openssl`**, are introduced either with the `cephadm@adm >` or `root #` prompts, depending on which privileges the related command requires.

1.4 Additional Information

For more information on Ceph key management, refer to *Book "Administration Guide", Chapter 19 "Authentication with* `cephx`*", Section 19.2 "Key Management"*.

2 General Notes on System Tuning

This manual discusses how to find the reasons for performance problems and provides means to solve these problems. Before you start tuning your system, you should make sure you have ruled out common problems and have found the cause for the problem. You should also have a detailed plan on how to tune the system, because applying random tuning tips often will not help and could make things worse.

PROCEDURE 2.1: GENERAL APPROACH WHEN TUNING A SYSTEM

1. Specify the problem that needs to be solved.
2. In case the degradation is new, identify any recent changes to the system.
3. Identify why the issue is considered a performance problem.
4. Specify a metric that can be used to analyze performance. This metric could for example be latency, throughput, the maximum number of users that are simultaneously logged in, or the maximum number of active users.
5. Measure current performance using the metric from the previous step.
6. Identify the subsystem(s) where the application is spending the most time.
7.
    a. Monitor the system and/or the application.
    b. Analyze the data and categorize where time is being spent.
8. Tune the subsystem identified in the previous step.
9. Remeasure the current performance without monitoring using the same metric as before.
10. If performance is still not acceptable, start over with *Step 3*.

2.1 Understand The Problem You are Solving

Before starting to tuning a system, try to describe the problem as exactly as possible. A statement like “The system is slow!” is not a helpful problem description. For example, it could make a difference whether the system speed needs to be improved in general or only at peak times.

Furthermore, make sure you can apply a measurement to your problem, otherwise you cannot verify if the tuning was a success or not. You should always be able to compare "before" and "after". Which metrics to use depends on the scenario or application you are looking into. Relevant Web server metrics, for example, could be expressed in terms of:

Latency
: The time to deliver a page.

Throughput
: Number of pages served per second or megabytes transferred per second.

Active Users
: The maximum number of users that can be downloading pages while still receiving pages within an acceptable latency.

2.2 Rule Out Common Problems

A performance problem often is caused by network or hardware problems, bugs, or configuration issues. Make sure to rule out problems such as the ones listed below before attempting to tune your system:

- Check the output of the `systemd` journal for unusual entries.
- Check (using **`top`** or **`ps`**) whether a certain process misbehaves by eating up unusual amounts of CPU time or memory.
- Check for network problems by inspecting `/proc/net/dev`.
- In case of I/O problems with physical disks, make sure they are not caused by hardware problems (check the disk with the `smartmontools`) or by a full disk.
- Ensure that background jobs are scheduled to be carried out in times the server load is low. Those jobs should also run with low priority (set via **`nice`**).
- If the machine runs several services using the same resources, consider moving services to another server.
- Last, make sure your software is up-to-date.

2.3 Finding the Bottleneck

Finding the bottleneck very often is the hardest part when tuning a system. SUSE Enterprise Storage offers many tools to help you with this task. If the problem requires a long-time in-depth analysis, the Linux kernel offers means to perform such analysis.

Once you have collected the data, it needs to be analyzed. First, inspect if the server's hardware (memory, CPU, bus) and its I/O capacities (disk, network) are sufficient. If these basic conditions are met, the system might benefit from tuning.

2.4 Step-by-step Tuning

Make sure to carefully plan the tuning itself. It is of vital importance to only do one step at a time. Only by doing so can you measure whether the change made an improvement or even had a negative impact. Each tuning activity should be measured over a sufficient time period to ensure you can do an analysis based on significant data. If you cannot measure a positive effect, do not make the change permanent: it might have a negative effect in the future.

3 Introduction to Tuning SUSE Enterprise Storage Clusters

Tuning a distributed cluster is a foray into the use of the scientific method, backed with iterative testing. By taking a holistic look at the cluster and then delving into all the components, it is possible to achieve dramatic improvements. Over the course of the work that contributed to the authoring of this guide, the authors have seen performance more than double in some specific cases.

This guide is intended to assist the reader in understanding the what and how of tuning a SUSE Enterprise Storage cluster. There are topics that are beyond the scope of this guide, and it is expected that there are further tweaks that may be performed to an individual cluster in order to achieve optimum performance in a particular end user environment.

This reference guide is targeted at architects and administrators who need to tune their SUSE Enterprise Storage cluster for optimal performance. Familiarity with Linux and Ceph are assumed.

3.1 Philosophy of Tuning

Tuning requires looking at the entire system being tuned and approaching the process with scientific rigor. An exhaustive approach requires taking an initial baseline and altering a single variable at a time, measuring the result, and then reverting it back to default while moving on to the next tuning parameter. At the end of that process, it is then possible to examine the results, whether it be increased throughput, reduced latency, reduced CPU consumption, and then decide which are likely candidates for combining for additive benefit. This second phase should also be iterated through in the same fashion as the first. This general process would be continued until all possible combinations were tried and the optimal settings discovered.

Unfortunately, few have the time to perform such an exhaustive effort. This being reality, it is possible to utilize some knowledge and begin an iterative process of combining and measuring well-known candidates for performance improvements and measuring the resulting changes. That is the process followed during the research that produced this work.

3.2 The Process

A proper process is required for effective tuning to occur. The tenets of this process are:

Measure

Start with a baseline and measure the same way after each iteration. Make sure you are measuring all relevant dimensions. Discovering that you are CPU-bound only after working through multiple iterations invalidates all the time spent on the iterations.

Document

Document all results for future analysis. Patterns may not be evident until later review.

Discuss

When possible, discuss the results you are seeing with others for their insights and feedback.

Repeat

A single measurement is not a guarantee of repeatability. Performing the same test multiple times helps to better establish an outcome.

Isolate variables

Having only a single change affect the environment at a time may cause the process to be longer, but it also helps validate the particular adjustment being tested.

3.3 Hardware and Software

This work leveraged for SUSE Enterprise Storage was performed on two models of servers. Any results referenced in this guide are from this specific hardware environment. Variations of the environment can and will have an effect on the results.

Storage nodes:

- 2U Server
  - 1x Intel Skylake 6124
  - 96 GB RAM
  - Mellanox Dual Port ConnectX-4 100 GbE

- 12x Intel SSD D3-S4510 960 GB
- RAID-1 480 GB M.2 Boot Device

Admin, monitor, and protocol gateways:

- 1U Server
  - 1x Intel Skylake 4112
  - 32 GB RAM
  - Mellanox Dual Port ConnectX-4 100 GbE
  - RAID-1 480 GB M.2 Boot Device

Switches:

- 2x 32-port 100 GbE

Software:

- SUSE Enterprise Storage 5.5
- SUSE Linux Enterprise Server 15 SP1

Note

Limited-use subscriptions are provided with SUSE Enterprise Storage as part of the subscription entitlement.

3.3.1 Performance Metrics

The performance of storage is measured on two different, but related axes: `latency` and `throughput`. In some cases, one is more important than the other. An example of throughput being more important is that of backup use cases, where throughput is the most critical measurement and maximum performance is achieved with larger transfer sizes. Conversely, for a high-performance database, latency is the most important measurement.

Latency is the time from when the request was made to when it was completed. This is usually measured in terms of milliseconds. This performance can be directly tied to CPU clock speed, the system bus, and device performance. *Throughput* is the measure of an amount of data that can be written or retrieved within a particular time period. The measure is usually in MB/s and GB/s, or perhaps MiB/s and GiB/s.

A third measurement that is often referred to is `IOPS`. This stands for Input/output Operations Per Second. This measure is somewhat ambiguous as the result is reliant on the size of the I/O operation, what type (read/write), and details about the I/O pattern: fresh write, overwrite, random write, mix of reads and writes. While ambiguous, it is still a valid tool to use when measuring the changes in the performance of your storage environment. For example, it is possible to make adjustments that may not affect the latency of a single 4K sequential write operation, but would allow many more of the operations to happen in parallel, resulting in a change in throughput and IOPS.

3.4 Determining What to Measure

When tuning an environment, it is important to understand the I/O that is being tuned for. By properly understanding the I/O pattern, it is then possible to match the tests to the environment, resulting in a close simulation of the environment.

Tools that can be useful for understanding the I/O patterns are:

- **`iostat`**
- **`blktrace`**, **`blkparse`**
- **`systemtap (stap)`**
- **`dtrace`**

While discussing these tools is beyond the scope of this document, the information they can provide may be very helpful in understanding the I/O profile that needs to be tuned.

3.4.1 Single Thread vs Aggregate IO

Understanding the requirements of the workload in relation to whether it needs scale-up or scale-out performance is often a key to proper tuning as well, particularly when it comes to tuning the hardware architecture and to creating test scenarios that provide valid information regarding tuning for the application(s) in question.

3.5 Testing Tools and Protocol

Proper testing involves selection of the right tools. For most performance test cases, use of `fio` is recommended, as it provides a vast array of options for constructing test cases. For some use cases, such as S3, it may be necessary to use alternative tools to test all phases of I/O.

Tools that are commonly used to simulate I/O are:

- `fio`
- `iometer`

When performing testing, it is imperative that sound practices be utilized. This involves:

- Pre-conditioning the media.
- Ensuring what you are measuring is what you intend to measure.
- Validate that the results you see make sense.
- Test each component individually and then in aggregate, one layer at a time.

II SUSE Enterprise Storage Tuning

4 Architecture and Hardware Tuning

Architectural tuning includes aspects that range from the low-level design of the systems being deployed, up to macro-level decisions about network topology and cooling. Not all of these can be covered in this work, but guidance will be given where possible.

From the top-down view, it is important to think about the physical location of nodes, the connectivity available between them, and implications of such items as power routing and fire compartments. However, from a performance perspective, it is most important to think in terms of the connectivity and what a write actually looks like from the cluster perspective.

The intention of architectural tuning is to take control of where the performance bottlenecks lie. In most cases, it is preferable for the bottleneck to lie with the storage device, as that creates a predictable pattern of performance degradation and associated limitations. Placing the bottleneck in other areas, such as CPU or network, may create inconsistent behavior when resources are stressed. This is due to the possibility of other processes running, whether recovery, rebalancing, garbage collection, and causing inconsistent behavior due to being network or CPU bound. When storage is the bottleneck, the degradation results in longer response times to queued requests, making this the most desireable point at which to place the bottleneck.

4.1 Network

The network is the backbone of the cluster, and a failure to implement it in a robust manner can cripple the performance of the cluster, no matter what other steps are taken. From a best practices perspective, this entails implementing a fault-tolerant switching infrastructure that can scale the aggregate core bandwidth as the cluster grows. For very large clusters, this may entail a leaf and spine architecture, while for smaller clusters, it may look like a more familiar hub-and-spoke or mesh network.

No matter which network architecture is chosen, it is important to carefully examine each hop along the path and consider the maximum network traffic that could occur during adverse conditions.

An example of a bad decision here would be to have a multi-rack cluster where each rack contains 16 storage nodes of 24× 7200rpm drives where each node is connected to a pair of stacked Top-of-Rack (TOR) switches via 2x25Gb connections. This connectivity is sufficient for 3 GB/s per connection or 6 GB/s total for the node, which is near the maximum that a 7200rpm

drive will sustain. The bad decision comes in *only* using 4× of the 25 GB interfaces for the uplink. While 100 Gb may seem like a substantial amount of bandwidth, it translates into only about 10 GB/s, while the rack is capable of an aggregate of around 48 GB/s.

During an adverse situation, it is possible that network congestion will result in dramatically increased latency, packet delays, drops, etc. A simple rememdy would have been to select switches with 4× 100 Gb uplink ports, resulting in each switch being able to transmit the complete, aggregate bandwidth load to the network core.

In reality, some level of over-subscription is going to happen in most networks, but the smaller the ratio, the better the resulting cluster will be able to handle adverse conditions.

4.1.1 Network Tuning

Here are some general principals to follow when planning a Ceph cluster network:

- Utilize 25/50/100 GbE connections - The signaling rate is 2.5 times faster than 10/40 GbE resulting in lower latency over the wire. The impact from the faster signaling rate would be minimal with spinning media and more impactful with faster technologies such as NVMe.
- Network bandwidth should be at least the total bandwidth of all storage devices present in the storage node
- Usage of VLANs on LACP-bonded Ethernet provides the best balance of bandwidth aggregation + fault tolerance
- The network should utilize jumbo-frame Ethernet if all nodes connecting to the storage are able to do so, otherwise use the standard MTU.

4.2 Node Hardware Recommendations

4.2.1 CPU

There are a lot of options when it comes to processor selection in the market today. There are multiple varieties available from the major vendors in the x86 space and a wide array of options in the 64-bit Arm space as well. Without regard to the core architecture, there is one rule that

is always true, and that is that higher clockspeed allows more work to be done in the same amount of time. This consideration is most important when working with higher speed storage and network devices.

CPU selection is also an important consideration for specific services. Some services, such as metadata servers, NFS, Samba, and ISCSI gateways benefit from a smaller number of much faster cores, while the OSD nodes need a more core dense solution.

A second consideration is whether to use a single socket or multiple sockets. The answer to this will depend on the device density, type of network hardware being utilized, etc. In many nodes, a single socket will provide better performance as the processor interlink is a bottleneck, though this would most likely be noticed in an all NVMe based node type. The general recommendation is to use a single socket whenever possible.

When considering which processor to choose, there are several considerations aside from clock-speed that should be taken into consideration. They include:

- *Memory bandwidth*: Ceph is a heavy user of RAM, thus the more memory bandwidth available, the more performant the node can be.
- *Memory layout*: Even if the memory selected is fast, if all memory channels are not leveraged, performance is being left untapped. It is advantageous to ensure that RAM is distributed evenly across all channels.
- *Offload capabilities*: For example, Intel CPUs offer `zlib` and Reed-Solomon offloads, the latter being used with erasure coding when the ISA plugin is specified.
- *PCIe bus speed and lanes*: This is particularly important when looking at devices with a large number of PCIe devices, like NVMe. The bus speed also affects the performance of network devices as well.

4.2.2 Storage Device

Storage device selection can dramatically affect the performance and reliability of a Ceph cluster. When building for performance, it is important to understand the nature of the device in regard to reads/writes and the workloads that will be applied. This is particularly true with flash media.

4.2.2.1 Device Type

The first recommendation is to ensure that systems utilize Enterprise-class storage media. With NVMe and SSD devices, this implies a few key items.

- More spare cells to deal with media wear-out
- Battery/capacitor to allow completion of buffer dumps during unexpected power events

It is also important to ensure the media being utilized will support the workloads of the cluster. For example, if the applications using the cluster have a read/write mix of 90:10, it is likely acceptable to utilize a read-intensive NVMe device. However, if the ratio is flipped or even 50:50, it is a better choice to at least consider mixed-use, or write intensive media. This selection goes beyond just the media durability, but also includes considerations around the design. Write-intensive media typically allocate more PCIe lanes to handling write requests to the media, ensuring faster commits than a read-intensive device would provide under load. Also, write intensive devices will most often use faster classes of non-volitaile memory technology and/or have large, supercap backed caches.

4.2.3 Device Buses

It is also important to understand the impact of the storage bus and hardware pieces along the way. Clearly, 6 Gb/s is slower than 12 Gb/s, and 12 Gb/s is slower than PCIe Gen3 (8 Gb/s per lane), but what about mixing SATA 3 Gb/s and SATA 6 Gb/s, or mixing 6 Gb/s and 12 Gb/s SAS?

The general rule is not to mix. When a 6 Gb/s device is introduced to a 12 Gb/s bus, the entire bus slows down to 6 Gb/s, greatly reducing the overall throughput capability. Where this really would hurt is in a dense SAS SSD system. If there are 24 SAS SSDs on a two-channel, 12 Gb/s bus and one of the devices is only 6 Gb/s, then the 12 Gb/s SAS drives that can push 850 MB/s now oversubscribe the bus due to the reduced data rate.

Another consideration is the presence of bus expanders. Bus expanders allow a system to multiplex multiple devices on a single channel. The result is higher density at lower performance maximum. In some cases, the expanders may work acceptably, such as with HDDs, but with SSDs, they are likely to quickly become a bottleneck.

4.2.4 General Recommendations

Below are some generic tuning options applicable to performance tuning for server platforms:

- Set firmware Power/Performance controls to the performance profile. This should eliminate frequency scaling and ensure that there is no added latency caused by it.
- Enable multi-threading on SMT-capable CPUs. This extra processing power is utilized effectively by Ceph.
- Ensure all add-in cards are in the optimal slots for performance.

4.3 Ceph

4.3.1 RocksDB and WAL

Ceph makes use of both a Write-Ahead-Log (WAL) and `RocksDB`. The WAL is the internal journal where small writes are queued before commiting to the backend storage. RocksDB is where Ceph stores metadata associated with the objects written to BlueStore. When using spinning media, or perhaps even SSDs, it generally makes sense to locate the RocksDB and WAL on a faster device, such as NVMe. When doing so, proper sizing of these is critical to ensuring a stable performance profile of the cluster over time.

From a performance perspective, the rule of thumb is to divide the write performance of the WAL/RocksDB device by the write performance of the data device. This yields what should be considered to be the maximum ratio of data devices per WAL/RocksDB device.

4.3.1.1 WAL

The WAL maxes out a bit under two gigabytes. In order to leave room for maintenance activities, having about four gigabytes of space allocated/allowed is optimal.

4.3.1.2 RocksDB

RocksDB operates with a series of tiered levels, each being an order of magnitude larger than the last. Levels one through four are 256 MB, 2.56 GB, 25.6 GB, 256 GB respectively. Allocating appropriate space for these is an act of aggregation. Given that few installations utilize enough

metadata to require the fourth tier, allocating for the first three and associated maintenance is sufficient. 25.6 + 2.56 + .256 GB = 28.416 GB. Rounding up to 30 GB and providing 100% overhead to allow for maintenance takes the space allocation suggested for the first three tiers to 60 GB.

Note

Keep in mind that we recommend reserving 4 GB for the WAL device. The recommended size for DB is a total of 64 GB for most workloads. See *Book "Deployment Guide", Chapter 2 "Hardware Requirements and Recommendations", Section 2.4.3 "Recommended Size for the BlueStore's WAL and DB Device"* for more information.

Making the decision to provision fast space for the fourth tier of RocksDB is entirely related to the expected metadata load. Protocols like RBD use little metadata, while CephFS is somewhere from a mild to moderate amount. S3 and native RADOS can utilize the highest amounts of metadata and are generally the cases where it starts making sense to evaluate whether it makes sense to move the fourth tier makes to faster media.

5 Operating System Level Tuning

A significant amount of the performance tuning for Ceph clusters can be done at the operating system (OS) layer. This tuning involves ensuring that unnecessary services are not running, and extends down to ensuring buffers are not being overrun and interrupts are being spread properly. There are many additional tuning options for the OS that are not included here, either due to statistically-insignificant performance changes, or not being deemed candidates for significant performance improvement at the time of this work.

5.1 SUSE Linux Enterprise Install and Validation of Base Performance

During the OS installation, do *not* select an install pattern that includes an X Server. Doing so utilizes RAM and CPU resources that would be better allocated to tuning storage-related daemons. We recommend a pattern that includes the minimal pattern with the addition of the `YaST management` pattern.

After the OS is installed, it is proper to evaluate the various individual components that are critical to the overall performance of the storage cluster.

Note

Check performance of individual components before SUSE Enterprise Storage is setup to ensure they are performing as desired.

5.1.1 Network Performance

To perform **`iperf3`** tests for network performance, consider increasing the window size (with the `-w` option) and running multiple streams to fully test the bandwidth capability. If using the standard MTU, your NICs should be capable of running at approximately 70-80% of the advertised bandwidth. If you move up to jumbo frames, the NIC should be able to saturate the link.

This is not necessarily true for faster topologies such as 100 Gb. In those topologies, saturating the NIC can require substantial OS and driver tuning, in combination with ensuring the hardware has the appropriate CPU clock speeds and settings.

This is a sample of the **`iperf3`** commands used on a 100 Gb network. In the command line, the `-N` disables Nagle's buffering algorithm and the `-l` sets the buffer length to higher than the default 128 kB, resulting in slightly higher throughput.

```
server# iperf3 -s

client# iperf3   -c server -N -l 256k
Connecting to host sr650-1, port 5201
[  4] local 172.16.227.22 port 36628 connected to 172.16.227.21 port 5201
[ ID] Interval           Transfer     Bandwidth       Retr  Cwnd
[  4]   0.00-1.00   sec  4.76 GBytes  40.9 Gbits/sec    0   1.48 MBytes
[  4]   1.00-2.00   sec  4.79 GBytes  41.1 Gbits/sec    0   2.52 MBytes
[  4]   2.00-3.00   sec  4.73 GBytes  40.6 Gbits/sec    0   2.52 MBytes
[  4]   3.00-4.00   sec  4.73 GBytes  40.6 Gbits/sec    0   2.52 MBytes
[  4]   4.00-5.00   sec  4.74 GBytes  40.7 Gbits/sec    0   2.52 MBytes
[  4]   5.00-6.00   sec  4.73 GBytes  40.6 Gbits/sec    0   2.52 MBytes
[  4]   6.00-7.00   sec  4.72 GBytes  40.6 Gbits/sec    0   2.52 MBytes
[  4]   7.00-8.00   sec  4.72 GBytes  40.6 Gbits/sec    0   2.52 MBytes
[  4]   8.00-9.00   sec  4.73 GBytes  40.7 Gbits/sec    0   2.52 MBytes
[  4]   9.00-10.00  sec  4.73 GBytes  40.6 Gbits/sec    0   2.52 MBytes
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bandwidth       Retr
[  4]   0.00-10.00  sec  47.4 GBytes  40.7 Gbits/sec    0             sender
[  4]   0.00-10.00  sec  47.4 GBytes  40.7 Gbits/sec                  receiver
```

5.1.2 Storage Performance

Use **`fio`** to test individual storage devices to understand the per-device performance maxima. Do this for all devices to ensure that none are out of specification, or are connected to expanders which lower bandwidth. Doing an exhaustive study of different I/O sizes and patterns would provide the most information about performance expectations, but is beyond the scope of this document.

We recommend testing at least random 4 kB, random 64 kB, and sequential 64 kB and 1 MB buffers. This should give a reasonable overall view of the device's performance characteristics. When testing, it is important to use the raw device (`/dev/sd`*X*) and to use the `direct=1` option with multiple jobs to maximize device performance under stress.

Make sure that the test size (dataset) is large enough to over-run any caches that may apply. We recommend using the `ramp-time` parameter to allow for enough time for the cache overrun to occur before performance measurements are made. This helps to ensure that performance numbers are not tainted by cache-only performance.

Run **fio** against all devices in an OSD node simultaneously to identify bottlenecks. It should scale very near linearly; if not, check controller firmware, slot placement, and if necessary, split devices across multiple controllers. This simulates the node under heavy load.

We recommend using the same I/O patterns and block sizes that the individual devices were tested with. The job count should be a multiple of the total number of devices in the system, to allow for even distribution across all devices and buses.

5.1.2.1 Latency Bound and Maximum

There is value in performing both latency-bound and worst-case-scenario tests. Changing a particular value may not improve latency, but rather enable the cluster to handle even more total load, even though latencies continue to rise. The inverse may also be true, where a change affects latency of operations in a measurable way. To identify both possibilities, we recommend that tests be performed that represent both positions. Latency-bounded tests in **fio** have the following set:

```
latency_target=10ms
latency_window=5s
latency_percentile=99
```

The settings above cause **fio** to increase the I/O queue depth until 1% or more of IOPS no longer maintain a 10 ms average during a sliding five-second window. It then backs down the queue depth until the latency average is maintained.

5.2 Kernel Tuning

There are several aspects of the kernel that can be tuned on both the cluster and some of the clients. It is important to understand that most tuning is about gaining very small incremental improvements, which in aggregate represent a measureable (and hopefully meaningful) improvement in performance. For this document, information on tuning comes from a variety of sources. The primary source is https://documentation.suse.com/sles/15-SP1/html/SLES-all/book-sle-tuning.html. We also used numerous other references, including documentation from hardware vendors.

5.2.1 CPU Mitigations

One key area of kernel performance tuning is to disable its side-channel attack mitigations. The bulk of the benefits from this occur with smaller I/Os, ranging in size from 4 kB to 64 kB. In particular, 64 kB random read and sequential writes doubled in performance in a limited test environment using only two client nodes.

Changing these options requires a clear understanding of the security implications, as they involve disabling mitigations for side-channel attacks on some CPUs. The need to disable these mitigations may be minimized with newer processors. You must carefully evaluate whether this is something needed for the particular hardware being utilized. In the test configuration, a Salt state was utilized to apply these changes. The Salt state should be in a subdirectory of `/srv/salt` on the Salt master and is applied by using a **`salt state.apply`** command similar to below:

```
salt '*' state.apply my_kerntune
```

The Salt state and steps used in this testing can be found in *Appendix A, Salt State for Kernel Tuning*. This needs to be adjusted to work in each customer environment. An example of adjusting the `grub2` configuration file can also be found in the appendix.

5.2.2 I/O tuning - Multiqueue Block I/O

The first aspect to tune is to ensure that I/O is flowing in the most optimal pattern. For the test cluster used in this test, that means enabling multi-queue block I/O. This is done through adding a boot-time kernel parameter, as found in Section 12.4 https://documentation.suse.com/sles/15-SP1/html/SLES-all/cha-tuning-io.html#cha-tuning-io-barrier. This is not an action that should be taken unilaterally on clusters which contain spinning media devices, due to potential performance degradation for those devices. The general result is that there multiple I/O queues are assigned to each device, allowing more jobs to be handled simultaneously by those device that can service large numbers of requests, such as SSD and NVMe drives.

5.2.3 SSD Tuning

To get the best read performance, it may be necessary to adjust the `read_ahead` and write cache settings for the SSD devices. In our particular testing environment, disabling write cache and forcing `read_ahead` to 2 MB resulted in the best overall performance.

Before tuning, it is important to check and record the default values, and measure any differences in performance compared to that baseline.

By placing the following file in `/etc/udev/rules.d`, devices will be detected according to the model name shown in `/sys/block/{devname}/device/model`, and instructed to disable write caching and set the `read_ahead_kb` option to 2 MB.

```
  /etc/udev/rules.d/99-ssd.rules

  # Setting specific kernel parameters for a subset of block devices (Intel SSDs)
  SUBSYSTEM=="block", ATTRS{model}=="INTEL SSDS*", ACTION=="add|change", ATTR{queue/
read_ahead_kb}="2048"
  SUBSYSTEM=="block", ATTRS{model}=="INTEL SSDS*", ACTION=="add|change", RUN+="/sbin/
hdparm -W 0 /dev/%k"
```

5.2.4 Network Stack and Device Tuning

Proper tuning of the network stack can substantially assist improving the latency and throughput of the cluster. A full script for the testing we performed can be found in *Appendix C, Network Tuning*.

The first change, and one with the highest impact, is to utilize jumbo frame packets. For this to be done, all interfaces utilizing the cluster must be set to use an MTU of 9000 bytes. Network switches are often set to use 9100 or higher. This is suitable, as they are only passing packets, not creating them.

5.2.4.1 Network Device Tuning

5.2.4.1.1 Jumbo Frames

The following Salt command ensures the bonded interfaces on all nodes under control (including the test load generation nodes) were utilizing an MTU of 9000:

```
salt '*' cmd.run 'ip link set bond0 mtu 9000'
```

To set this persistently, utilize YaST to set the MTU for the bonded interface.

5.2.4.1.2 PCIe Bus Adjustment

Adjusting the PCIe maximum read request size can provide a slight boost to performance. Be aware that this tuning is card- and slot-specific and must only be done in conjunction with the conditions and instructions supplied by the manufacturer. The maximum PCIe read request size was set with the following Salt commands:

Warning

This should only be done with guidance from the NIC manufacturer and is specific to bus location, driver version and hardware.

```
salt '*' cmd.run 'setpci -s 5b:00.0 68.w=5936'
salt '*' cmd.run 'setpci -s 5b:00.1 68.w=5936'
```

5.2.4.1.3 TCP RSS

The next item on the tuning list is helpful in ensuring that a single CPU core is not responsible for all packet processing. A small script is used to spread the I/O across multiple local (from a NUMA perspective) cores.

Note

This is not necessary if the number of queues returned by **`ls /sys/class/net/{if-name}/queues/rx-*|wc -l`** is equal to the number of physical cores in a single CPU socket.

```
salt '*' cmd.run 'for j in `cat /sys/class/net/bond0/bonding/slaves`;do \
LOCAL_CPUS=`cat /sys/class/net/$j/device/local_cpus`;echo $LOCAL_CPUS > \
/sys/class/net/$j/queues/rx-0/rps_cpus;done'
```

5.2.4.1.4 Ring Buffers

Many NIC drivers start with a default value for the receive (RX) and transmit (TX) buffers that is not optimal for high-throughput scenarios, and does not allow enough time for the kernel to drain the buffer before it fills up.

The current and maximum settings can be revealed by issuing the following command to the proper NICs:

```
ethtool -g eth4
```

The output from this command should look similar to this:

```
  Ring parameters for eth4:
  Pre-set maximums:
  RX:  8192
  RX Mini: 0
  RX Jumbo: 0
  TX:  8192
  Current hardware settings:
  RX:  1024
  RX Mini: 0
  RX Jumbo: 0
  TX:  1024
```

Here we can see that the NIC can allocate buffers of up to 8 kB, but it iscurrently using ones of only 1 kB. To adjust this for the cluster, issue the following command:

```
  salt '*' cmd.run 'ethtool -G eth4 rx 8192 tx 8192'
  salt '*' cmd.run 'ethtool -G eth5 rx 8192 tx 8192'
```

Setting this value persistently can be achieved via the YaST configuration module:

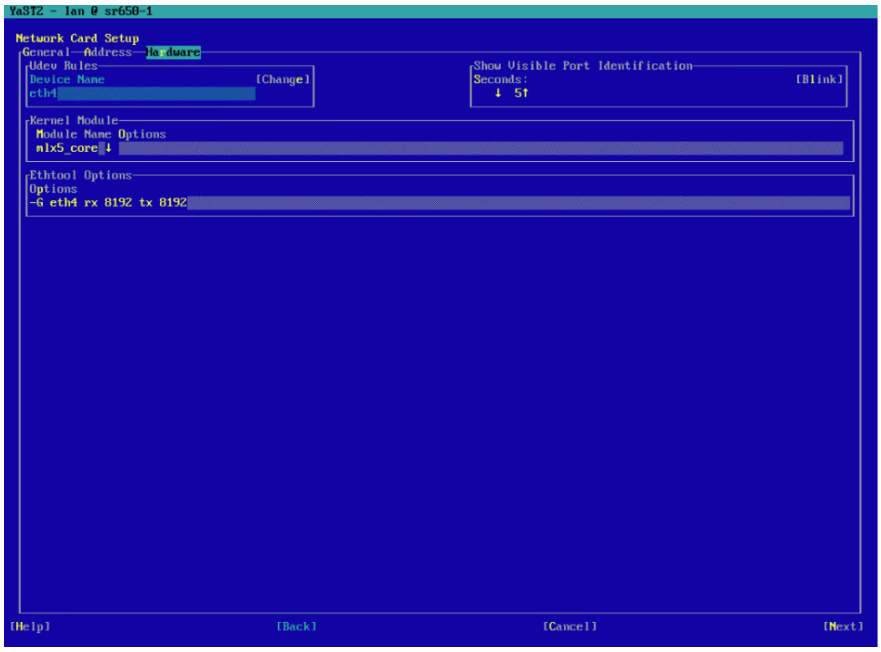

FIGURE 5.1: YAST CONFIGURATION MODULE

Additionally, the settings can be made persistent by editing the configuration files for the physical interfaces found in `/etc/sysconfig/network`. A script can be found in Appendix B that will change all interfaces to the maximum ring buffer value.

5.2.4.2 Network Stack

The following settings can all be made persistent by modifying `/etc/sysctl.conf`. They are represented as arguments in a Salt command to allow testing and validation in your environment before making them permanent.

5.2.4.2.1 Lower TCP Latency

Setting the `TCP low latency` option disables IPv4 TCP pre-queue processing and improves latency. We recommend experimenting with setting this to both `0` and `1`. In laboratory testing, setting the value to `1` provided slightly better performance:

```
salt '*' cmd.run 'sysctl -w net.ipv4.tcp_low_latency=1'
```

The TCP `fastopen` option allows the sending of data in the first `syn` packet, resulting in a slight improvement in latency:

```
salt '*' cmd.run 'sysctl -w net.ipv4.tcp_fastopen=1'
```

5.2.4.2.2 TCP Stack Buffers

Ensure that the TCP stack has sufficent buffer space to queue both inbound and outbound traffic:

```
salt '*' cmd.run 'sysctl -w net.ipv4.tcp_rmem="10240 87380 2147483647"'
salt '*' cmd.run 'sysctl -w net.ipv4.tcp_wmem="10240 87380 2147483647"'
```

5.2.4.2.3 TCP Sequence and Timestamps

In fast networks, TCP sequence numbers can be re-used in a very short timeframe. The result is that the system thinks a packet has been received out of order, resulting in a drop. TCP timestamps were added to help ensure that packet sequence could be tracked better:

```
salt '*' cmd.run 'sysctl -w net.ipv4.tcp_timestamps=1'
```

`TCP Selective Acknowledgement` is a feature that is primarily useful for WAN or lower-speed networks. However, disabling it may have negative effects in other ways:

```
salt '*' cmd.run 'sysctl -w net.ipv4.tcp_sack=1'
```

5.2.4.2.4 Kernel Network Buffers and Connections

Providing plenty of buffer space is a recurring theme in tuning networks for high performance. The `netdev_max_backlog` is where traffic is queued after it has been received by the NIC, but before it is processed by the protocol stack (IP, TCP, etc):

```
salt '*' cmd.run 'sysctl -w net.core.netdev_max_backlog=250000'
```

Other preventative measures for the system and gateway nodes include ensuring that the maximum connection count is high enough to prevent the generation of `syn` cookies. This is useful to set on all nodes involved:

```
salt '*' cmd.run 'sysctl -w net.core.somaxconn=2048'
```

Increasing the network stack buffers is useful to ensure that sufficient buffers exist for all transactions:

```
salt '*' cmd.run 'sysctl -w net.core.rmem_max=2147483647'
salt '*' cmd.run 'sysctl -w net.core.wmem_max=2147483647'
```

6 Ceph Tuning

Ceph includes a telemetry module that provides anonymized information back to the Ceph developer community. The information contained in the telemetry report provides information that helps the developers prioritize efforts and identify areas where more work may be needed. It may be necessary to enable the telemetry module before turning it on. To enable the module:

```
ceph mgr module enable telemetry
```

To turn on telemetry reporting use the following command:

```
ceph telemetry on
```

Additional information about the Ceph telemetry module may be found in the *Book "Administration Guide"*.

6.1 Obtaining Ceph Metrics

Before adjusting Ceph tunables, it is helpful to have an understanding of the critical metrics to monitor and what they indicate. Many of these parameters are found by dumping raw data from the daemons. This is done by means of the **`ceph daemon dump`** command. The following example shows the dump command being utilized for `osd.104`.

```
ceph --admin-daemon /var/run/ceph/ceph-osd.104.asok perf dump
```

Starting with the Ceph Nautilus release, the following command may be used as well:

```
ceph daemon osd.104 perf dump
```

The output of the command is quite lengthy and may benefit from being redirected to a file.

6.2 Making Tuning Persistent

To make parameter adjustment persistent requires modifying the `/etc/ceph/ceph.conf` file. This is best done through modifying the source component files that DeepSea uses to manage the cluster. Each section is identified with a header such as:

```
[global]
[osd]
[mds]
[mon]
```

```
[mgr]
[client]
```

The section of the configuration is tuned by modifying the correct `[sectionname].conf` in the `/srv/salt/ceph/configuration/files/ceph.conf.d/` directory. After modifying the configuration file, replace `master` with the master minion-name (usually the admin node). The result is that the changes are pushed to all cluster nodes.

```
salt 'master' state.apply ceph.configuration.create
salt '*' state.apply ceph.configuration
```

Changes made in this way require the affected services to be restarted before taking effect. It is also possible to deploy these files before running stage 2 of the SUSE Enterprise Storage deployment process. It is especially desirable to do so if changing the settings that require node or device re-deployment.

6.3 Core

6.3.1 Logging

It is possible to disable all logging to reduce latency in the various codepaths.

Warning

This tuning should be used with caution and understanding that logging *will* need to be re-enabled should support be required. This implies that an incident would need to be reproduced *after* logging is re-enabled.

```
debug ms=0
debug mds=0
debug osd=0
debug optracker=0
debug auth=0
debug asok=0
debug bluestore=0
debug bluefs=0
debug bdev=0
debug kstore=0
debug rocksdb=0
debug eventtrace=0
```

```
  debug default=0
  debug rados=0
  debug client=0
  debug perfcounter=0
  debug finisher=0
```

6.3.2 Authentication Parameters

Under certain conditions where the cluster is physically secure and isolated inside a secured network with no external exposure, it is possible to disable `cephx`. There are two levels at which `cephx` can be disabled. The first is to disable signing of authentication traffic. This can be accomplished with the following settings:

```
cephx_require_signatures = false
cephx_cluster_require_signatures = false
cephx_sign_messages = false
```

The second level of tuning completely disables `cephx` authentication. This should only be done on networks that are isolated from public network infrastructure. This change is achieved by adding the following three lines in the global section:

```
auth cluster required = none
auth service required = none
auth client required = none
```

6.3.3 RADOS Operations

The backend processes for performing RADOS operations show up in `throttle-*objecter_ops` when dumping various daemons. If there is too much time being spent in `wait`, there may be some performance to gain by increasing the memory for in-flight ops or by increasing the total number of inflight operations overall.

```
objecter inflight op bytes = 1073741824 # default 100_M
objecter inflight ops = 24576
```

6.3.4 OSD parameters

Increasing the number of `op threads` may be helpful with SSD and NVMe devices, as it provides more work queues for operations.

```
osd_op_num_threads_per_shard = 4
```

6.3.5 RocksDB or WAL device

In checking the performance of BlueStore, it is important to understand if your metadata is spilling over from the high-speed device, if defined, to the bulk data storage device. The parameters useful in this case are found under bluefs `slow_used_bytes`. If `slow_used_bytes` is greater than zero, the cluster is using the storage device instead of the RocksDB/WAL device. This is an indicator that more space needs to be allocated to RocksDB/WAL.

Starting with the Ceph Nautilus release, spillover is shown in the output of the **`ceph health`** command.

The process of allocating more space depends on how the OSD was deployed. If it was deployed by a version prior to SUSE Enterprise Storage 6, the OSD will need to be re-deployed. If it was deployed with version 6 or after, it may be possible to expand the LVM that the RocksDB and WAL reside on, subject to available space.

6.3.6 BlueStore parameters

Ceph is thin provisioned, including the Write-Ahead Log (WAL) files. By pre-extending the files for the WAL, time is saved by not having to engage the allocator. It also potentially reduces the likelihood of fragmentation of the WAL files. This likely only provides benefit during the early life of the cluster.

```
bluefs_preextend_wal_files=1
```

BlueStore has the ability to perform buffered writes. Buffered writes enable populating the read cache during the write process. This setting, in effect, changes the BlueStore cache into a write-through cache.

```
bluestore_default_buffered_write = true
```

To prevent writes to the WAL when using a fast device, such as SSD and NVMe, set:

```
prefer_deferred_size_ssd=0 (pre-deployment)
```

6.3.7 BlueStore Allocation Size

Important

The following settings are not necessary for fresh deployments. Apply only to upgraded deployments or early SUSE Enterprise Storage 6 deployments as they may still benefit.

The following settings have been shown to slightly improve small object write performance under mixed workload conditions. Reducing the `alloc_size` to 4 kB helps reduce write amplification for small objects and with erasure coded pools of smaller objects. This change needs to be done before OSD deployment. If done after the fact, the OSDs will need to be re-deployed for it to take effect.

It is advised that spinning media continue to use 64 kB while SSD/NVMe are likely to benefit from setting to 4 kB.

```
min_alloc_size_ssd=4096
min_alloc_size_hdd=65536
```

Warning

Setting the `alloc_size_ssd` to 64 kB may reduce maximum throughput capability of the OSD.

6.3.8 BlueStore Cache

Increasing the BlueStore cache size can improve performance with many workloads. While FileStore OSDs cache data in the kernel's page cache, BlueStore OSDs cache data within the memory allocated by the OSD daemon itself. The OSD daemon will allocate memory up to its memory target (as controlled by the `osd_memory_target` parameter), and this determines the potential size of the BlueStore cache. The BlueStore cache is a read cache that by default is populated when objects are read. By setting the cache's minimum size higher than the default, it is guaranteed that the value specified will be the minimum cache available for each OSD. The idea is that more low-probability cache hits may occur.

The default `osd_memory_target` value is 4 GB: For example, each OSD daemon running on a node can be expected to consume that much memory. If a node's total RAM is significantly higher than `number of OSDs × 4 GB` and there are no other daemons running on the node,

performance can be increased by increasing the value of `osd_memory_target`. This should be done with care to ensure that the operating system will still have enough memory for its needs, while leaving a safety margin.

If you want to ensure that the BlueStore cache will not fall below a certain minimum, use the `osd_memory_cache_min` parameter. Here is an example (the values are expressed in bytes):

```
osd_memory_target = 6442450944
osd_memory_cache_min = 4294967296
```

Tip

As a best practice, start with the full memory of the node. Deduct 16 GB or 32 GB for the OS and then deduct appropriate amounts for any other workloads running on the node. For example, MDS cache if the MDS is colocated. Divide the remainder by the number of OSDs on that host. Ensure you leave room for improvement. For example:

```
(256 GB - 32 GB ) / 20 OSDs = 11,2 GB/OSD (max)
```

Using this example, configure approximately 8 or 10 GB per OSD.

By default, BlueStore automatically tunes cache ratios between data and key-value data. In some cases it may be helpful to manually tune the ratios or even increase the cache size. There are several relevant counters for the cache:

- `bluestore_onode_hits`
- `bluestore_onode_misses`
- `bluestore_onode_shard_hits`
- `bluestore_onode_shard_misses`

If the misses are high, it is worth experimenting with increasing the cache settings or adjusting the ratios.

Adjusting the BlueStore cache size above default has the potential to improve performance of small-block workloads. This can be done globally by adjusting the `_cache_size` value. By default, the cluster utilizes different values for HDD and SSD/NVMe devices. The best practice would be to increase the specific media cache you are interested in tuning:

- `bluestore_cache_size_hdd` (default 1073741824 - 1 GB)
- `bluestore_cache_size_ssd` (default 3221225472 - 3 GB)

Note

If the cache size parameters are adjusted and auto mode is utilized, `osd_memory_target` should be adjusted to accomodate the OSD base RAM and cache allocation.

In some cases, manually tuning the cache allocation percentage may improve performance. This is achieved by modifying the disabling autotuning of the cache with this configuration line:

```
bluestore_cache_autotune=0
```

Changing this value invalidates tuning of the `osd_memory_cache_min` value.

The cache allocations are modified by adjusting the following:

- `bluestore_cache_kv_ratio` (default .4)
- `bluestore_cache_meta_ratio values` (default .4)

Any unspecified portion is used for caching the objects themselves.

6.4 RBD

6.4.1 RBD Cluster

As RBD is a native protocol, the tuning is directly related to OSD or general Ceph core options that are covered in previous sections.

6.4.2 RBD Client

Read ahead cache defaults to 512 kB; test by tuning up and down on the client nodes.

```
echo {bytes} > /sys/block/rbd0/queue/read_ahead_kb
```

If your workload performs large sequential reads such as backup and restore, then this can make a significant difference in restore performance.

6.5 CephFS

Most of the performance tuning covered in this section pertains to the CephFS Metadata Servers. Because CephFS is a native protocol, much of the performance tuning is handled at the operating system, OSD and BlueStore layers. Being a file system that is mounted by a client, there are some client options that are covered in the client section.

6.5.1 MDS Tuning

In filesystems with millions of files, there is some advantage to utilizing very low-latency media, such as NVMe, for the CephFS metadata pool.

Utlizing the **`ceph-daemon perf dump`** command, there is a significant amount of data that can be examined for the Ceph Metadata Servers. It should be noted that the MDS perf counters only apply to metadata operations. The actual IO path is from clients straight to OSDs.

CephFS supports multiple metadata servers. These servers can operate in a multiple-active mode to provide load balancing of the metadata operation requests. To identify whether the MDS infrastructure is under-performing, one would examine the MDS data for request count and reply latencies. This should be done during an idle period on the cluster to form a baseline and then compared when under load. If the average time for reply latencies climbs too high, the MDS server needs to be examined further to identify whether the number of active metadata servers should be augmented, or whether simply increasing the metadata server cache may be sufficient. A sample of the output from the general MDS data for count and reply latency are below:

```
"mds": {
  # request count, interesting to get a sense of MDS load
         "request": 0,
         "reply": 0,
  # reply and the latencies of replies can point to load issues
         "reply_latency": {
             "avgcount": 0,
             "sum": 0.000000000,
             "avgtime": 0.000000000
         }
```

```
        }
```

Examining the `mds_mem` section of the output can help with understanding how the cache is utilized. High inode counters can indicate that a large number of files are open concurrently. This generally indicates that more memory may need to be provided to the MDS. If MDS memory cannot be increased, additional active MDS daemons should be deployed.

```
  "mds_mem": {

          "ino": 13,
          "ino+": 13,
          "ino-": 0,
          "dir": 12,
          "dir+": 12,
          "dir-": 0,
          "dn": 10,
          "dn+": 10,
          "dn-": 0,
```

A high `cap` count can indicate misbehaving clients. For example, clients that do not hand back caps. This may indicate that some clients need to be upgraded to a more recent version, or that the client needs to be investigated for possible issues.

```
  "cap": 0,
  "cap+": 0,
  "cap-": 0,
```

This final section shows memory utilization. The RSS value is the current memory size used. If this is roughly equal to the `mds_cache_memory_limit`, the MDS could probably use more memory.

```
  "rss": 41524,
  "heap": 314072
},
```

Another important aspect of tuning a distributed file system is recognizing problematic workloads. The output values below provide some insight to what the MDS daemon is spending its time on. Each heading has the same three attributes as the `req_create_latency`. With this information, it may be possible to better tune the workloads.

```
  "mds_server": {
          "dispatch_client_request": 0,
          "dispatch_server_request": 0,
          "handle_client_request": 0,
          "handle_client_session": 0,
```

```
            "handle_slave_request": 0,
            "req_create_latency": {
                "avgcount": 0,
                "sum": 0.000000000,
                "avgtime": 0.000000000
            },
            "req_getattr_latency": {},
            "req_getfilelock_latency": {},
            "req_link_latency": {},
            "req_lookup_latency": {},
            "req_lookuphash_latency": {},
            "req_lookupino_latency": {},
            "req_lookupname_latency": {},
            "req_lookupparent_latency": {},
            "req_lookupsnap_latency": {},
            "req_lssnap_latency": {},
            "req_mkdir_latency": {},
            "req_mknod_latency": {},
            "req_mksnap_latency": {},
            "req_open_latency": {},
            "req_readdir_latency": {},
            "req_rename_latency": {},
            "req_renamesnap_latency": {},
            "req_rmdir_latency": {},
            "req_rmsnap_latency": {},
            "req_rmxattr_latency": {},
            "req_setattr_latency": {},
            "req_setdirlayout_latency": {},
            "req_setfilelock_latency": {},
            "req_setlayout_latency": {},
            "req_setxattr_latency": {},
            "req_symlink_latency": {},
            "req_unlink_latency": {},
        }
```

Tuning the metadata server cache allows for more metadata operations to come from RAM, resulting in improved performance. The example below sets the cache to 16 GB.

```
mds_cache_memory_limit=17179869184
```

6.5.2 CephFS - Client

From the client side, there are a number of performance affecting mount options that can be employed. It is important to understand the potential impact on the applications being utilized before employing these options.

The following mount options may be adjusted to improve performance, but we recommend that their impact is clearly understood prior to implementation in a production environment.

noacl

Setting this mount option disables POSIX Access Control Lists for the CephFS mount, lowering potential metadata overhead.

noatime

This option prevents the access time metadata for files from being updated.

nodiratime

Setting this option prevents the metadata for access time of a directory from being updated.

nocrc

This disables CephFS CRCs, thus relying on TCP Checksums for the correctness of the data to be verified.

rasize

Setting a larger read-ahead for the mount may increase performance for large, sequential operations. Default is 8 MiB.

6.6 RGW

There are a large number of tunables for the Rados GateWay (RGW). These may be specific to the types of workloads being handled by the gateway and it may make sense to have different gateways handling distictly different workloads.

6.6.1 Sharding

The ideal situation is to understand how many total objects a bucket will host as this allows a bucket to be created with an appropriate number of shards at the outset. To gather information on bucket sharding, issue:

```
radosgw-admin bucket limit check
```

The output of this command appears like the following format:

```
  "user_id": "myusername",
          "buckets": [
```

```
        {
            "bucket": "mybucketname",
            "tenant": "",
            "num_objects": 611493,
            "num_shards": 50,
            "objects_per_shard": 12229,
            "fill_status": "OK"
        }
    ]
```

By default, Ceph reshards buckets to try and maintain reasonable performance. If it is known ahead of time how many shards a bucket may need, based on a ratio of 1 shard per 100 000 objects, it may be pre-sharded. This reduces contention and potential latency issues when resharding will occur. To pre-shard the bucket, it should be created and then submitted for sharding with the **`rgw-admin`** command. For example:

```
radosgw-admin bucket reshard --bucket={bucket name} --num-shards={prime number}
```

Where the `num-shards` is a prime number. Each shard should represent about 100 000 objects.

6.6.2 Limiting Bucket Listing Results

If a process relies on listing the buckets on a frequent basis to iterate through results, yet only uses a small number of results for each iteration it is useful to set the `rgw_max_listing_results` parameter.

6.6.3 Parallel I/O Requests

By default, the Object Gateway process is limited to eight simultaneous I/O operations for the index. This can be adjusted with the `rgw_bucket_index_max_aio` parameter.

6.6.4 Window Size

When working with larger objects, increasing the size of the Object Gateway windows for `put` and `get` can help with performance. Modify the following values in the Object Gateway section of the configuration:

```
rgw put obj min window size = [size in bytes, 16MiB default]
rgw get obj min window size = [size in bytes, 16MiB default]
```

6.6.5 Nagle's Algorithm

Nagle's algorithm was introduced to maximize the use of buffers and attempt to reduce the number of small packets transmitted over the network. While this is helpful in lower bandwdith environments, it can represent a performance degredation in high-bandwidth environments. Disabling it from RGW nodes can improve performance. Including the following in the Ceph configuation RGW section:

```
tcp_nodelay=1
```

6.7 Administrative and Usage Choices

6.7.1 Data Protection Schemes

The default replication setting keeps three total copies of every object written. The provides a high level of data protection by allowing up to two devices or nodes to fail while still protecting the data.

There are use cases where protecting the data is not important, but where performance is. In these cases, such as HPC scratch storage, it may be worthwhile to lower the replication count. This can be achieved by issuing a command such as:

```
ceph osd pool set rbd size 2
```

6.7.2 Erasure Coding

When using erasure coding, it is best to utilize optimized coding pool sizes. Experimental data suggests that the optimial pool sizes have either four or eight data chunks. It is also important to map this in relation to your failure domain model. If your cluster failure domain is at the node level, you will need at least `k+m` number of nodes. Similarly, if your failure domain it at the rack level, then your cluster needs to be spread over `k+m` racks. The key consideration is that distribution of the data in relation to the failure domain should be taken into consideration.

When using erasure coding schemes with failure domains larger than a single node, the use of Local Reconstruction Codes (LRC) may be beneficial due to lowered utilization of the network backbone, especially during failure and recovery scenatios.

There are particular use cases where erasure coding may even increase performance. These are mostly limited to large block (1 MB+) sequential read/write workloads. This is due to the parallelization of I/O requests that occurs when splitting objects into chunks to write to multiple OSDs.

7 Cache Tiering

A *cache tier* is an additional storage layer implemented between the client and the standard storage. It is designed to speed up access to pools stored on slow hard disks and erasure coded pools.

Typically, cache tiering involves creating a pool of relatively fast storage devices (for example, SSD drives) configured to act as a cache tier, and a backing pool of slower and cheaper devices configured to act as a storage tier. The size of the cache pool is usually 10-20% of the storage pool.

7.1 Tiered Storage Terminology

Cache tiering recognizes two types of pools: a *cache pool* and a *storage pool*.

Tip

For general information on pools, see *Book "Administration Guide", Chapter 22 "Managing Storage Pools"*.

storage pool

Either a standard replicated pool that stores several copies of an object in the Ceph storage cluster, or an erasure coded pool (see *Book "Administration Guide", Chapter 24 "Erasure Coded Pools"*).

The storage pool is sometimes referred to as 'backing' or 'cold' storage.

cache pool

A standard replicated pool stored on a relatively small but fast storage device with their own ruleset in a CRUSH Map.

The cache pool is also referred to as 'hot' storage.

7.2 Points to Consider

Cache tiering may *degrade* the cluster performance for specific workloads. The following points show some of its aspects that you need to consider:

- *Workload-dependent*: Whether a cache will improve performance is dependent on the workload. Because there is a cost associated with moving objects into or out of the cache, it can be more effective when most of the requests touch a small number of objects. The cache pool should be large enough to capture the working set for your workload to avoid thrashing.
- *Difficult to benchmark*: Most performance benchmarks may show low performance with cache tiering. The reason is that they request a big set of objects, and it takes a long time for the cache to 'warm up'.
- *Possibly low performance*: For workloads that are not suitable for cache tiering, performance is often slower than a normal replicated pool without cache tiering enabled.
- `librados` *object enumeration*: If your application is using `librados` directly and relies on object enumeration, cache tiering may not work as expected. (This is not a problem for Object Gateway, RBD, or CephFS.)

7.3 When to Use Cache Tiering

Consider using cache tiering in the following cases:

- Your erasure coded pools are stored on FileStore and you need to access them via RADOS Block Device. For more information on RBD, see *Book "Administration Guide", Chapter 23 "RADOS Block Device"*.
- Your erasure coded pools are stored on FileStore and you need to access them via iSCSI. For more information on iSCSI, refer to *Book "Administration Guide", Chapter 27 "Ceph iSCSI Gateway"*.
- You have a limited number of high-performance storage and a large collection of low-performance storage, and need to access the stored data faster.

7.4 Cache Modes

The cache tiering agent handles the migration of data between the cache tier and the backing storage tier. Administrators have the ability to configure how this migration takes place. There are two main scenarios:

write-back mode

In write-back mode, Ceph clients write data to the cache tier and receive an ACK from the cache tier. In time, the data written to the cache tier migrates to the storage tier and gets flushed from the cache tier. Conceptually, the cache tier is overlaid 'in front' of the backing storage tier. When a Ceph client needs data that resides in the storage tier, the cache tiering agent migrates the data to the cache tier on read, then it is sent to the Ceph client. Thereafter, the Ceph client can perform I/O using the cache tier, until the data becomes inactive. This is ideal for mutable, data such as photo or video editing, or transactional data.

read-only mode

In read-only mode, Ceph clients write data directly to the backing tier. On read, Ceph copies the requested objects from the backing tier to the cache tier. Stale objects get removed from the cache tier based on the defined policy. This approach is ideal for immutable data such as presenting pictures or videos on a social network, DNA data, or X-ray imaging, because reading data from a cache pool that might contain out-of-date data provides weak consistency. Do not use read-only mode for mutable data.

7.5 Erasure Coded Pool and Cache Tiering

Erasure coded pools require more resources than replicated pools. To overcome these limitations, we recommend to set a cache tier before the erasure coded pool. This is a requirement when using FileStore.

For example, if the “hot-storage” pool is made of fast storage, the “ecpool” created in *Book “Administration Guide”, Chapter 24 “Erasure Coded Pools”, Section 24.3 “Erasure Code Profiles”* can be speeded up with:

```
cephadm@adm > ceph osd tier add ecpool hot-storage
cephadm@adm > ceph osd tier cache-mode hot-storage writeback
cephadm@adm > ceph osd tier set-overlay ecpool hot-storage
```

This will place the “hot-storage” pool as a tier of ecpool in write-back mode so that every write and read to the ecpool is actually using the hot storage and benefits from its flexibility and speed.

```
cephadm@adm > rbd --pool ecpool create --size 10 myvolume
```

For more information about cache tiering, see *Chapter 7, Cache Tiering*.

7.6 Setting Up an Example Tiered Storage

This section illustrates how to set up a fast SSD cache tier (hot storage) in front of a standard hard disk (cold storage).

Tip

The following example is for illustration purposes only and includes a setup with one root and one rule for the SSD part residing on a single Ceph node.

In the production environment, cluster setups typically include more root and rule entries for the hot storage, and also mixed nodes, with both SSDs and SATA disks.

1. Create two additional CRUSH rules, 'replicated_ssd' for the fast SSD caching device class and 'replicated_hdd' for the slower HDD device class:

   ```
   cephadm@adm > ceph osd crush rule create-replicated replicated_ssd default host ssd
   cephadm@adm > ceph osd crush rule create-replicated replicated_hdd default host hdd
   ```

2. Switch all existing pools to the 'replicated_hdd' rule. This prevents Ceph from storing data to the newly added SSD devices:

   ```
   cephadm@adm > ceph osd pool set POOL_NAME crush_rule replicated_hdd
   ```

3. Turn the machine into a Ceph node using `ceph-salt`. Install the software and configure the host machine as described in *Book “Administration Guide”, Chapter 2 “Salt Cluster Administration”, Section 2.1 “Adding New Cluster Nodes”*. Let us assume that its name is `node-4`. This node needs to have 4 OSD disks.

   ```
   [...]
   host node-4 {
      id -5  # do not change unnecessarily
      # weight 0.012
      alg straw
   ```

```
    hash 0  # rjenkins1
    item osd.6 weight 0.003
    item osd.7 weight 0.003
    item osd.8 weight 0.003
    item osd.9 weight 0.003
}
[...]
```

4. Edit the CRUSH map for the hot storage pool mapped to the OSDs backed by the fast SSD drives. Define a second hierarchy with a root node for the SSDs (as 'root ssd'). Additionally, change the weight and add a CRUSH rule for the SSDs. For more information on CRUSH Map, see *Book "Administration Guide", Chapter 20 "Stored Data Management", Section 20.5 "CRUSH Map Manipulation"*.
   Edit the CRUSH Map directly with command line tools such as **`getcrushmap`** and **`crushtool`**:

```
cephadm@adm > ceph osd crush rm-device-class osd.6 osd.7 osd.8 osd.9
cephadm@adm > ceph osd crush set-device-class ssd osd.6 osd.7 osd.8 osd.9
```

5. Create the hot storage pool to be used for cache tiering. Use the new 'ssd' rule for it:

```
cephadm@adm > ceph osd pool create hot-storage 100 100 replicated ssd
```

6. Create the cold storage pool using the default 'replicated_ruleset' rule:

```
cephadm@adm > ceph osd pool create cold-storage 100 100 replicated
 replicated_ruleset
```

7. Then, setting up a cache tier involves associating a backing storage pool with a cache pool, in this case, cold storage (= storage pool) with hot storage (= cache pool):

```
cephadm@adm > ceph osd tier add cold-storage hot-storage
```

8. To set the cache mode to 'writeback', execute the following:

```
cephadm@adm > ceph osd tier cache-mode hot-storage writeback
```

   For more information about cache modes, see *Section 7.4, "Cache Modes"*.
   Writeback cache tiers overlay the backing storage tier, so they require one additional step: you must direct all client traffic from the storage pool to the cache pool. To direct client traffic directly to the cache pool, execute the following, for example:

```
cephadm@adm > ceph osd tier set-overlay cold-storage hot-storage
```

7.7 Configuring a Cache Tier

There are several options you can use to configure cache tiers. Use the following syntax:

```
cephadm@adm > ceph osd pool set cachepool key value
```

7.7.1 Hit Set

Hit set parameters allow for tuning of *cache pools*. Hit sets in Ceph are usually bloom filters and provide a memory-efficient way of tracking objects that are already in the cache pool.

The hit set is a bit array that is used to store the result of a set of hashing functions applied on object names. Initially, all bits are set to `0`. When an object is added to the hit set, its name is hashed and the result is mapped on different positions in the hit set, where the value of the bit is then set to `1`.

To find out whether an object exists in the cache, the object name is hashed again. If any bit is `0`, the object is definitely not in the cache and needs to be retrieved from cold storage.

It is possible that the results of different objects are stored in the same location of the hit set. By chance, all bits can be `1` without the object being in the cache. Therefore, hit sets working with a bloom filter can only tell whether an object is definitely not in the cache and needs to be retrieved from cold storage.

A cache pool can have more than one hit set tracking file access over time. The setting `hit_set_count` defines how many hit sets are being used, and `hit_set_period` defines for how long each hit set has been used. After the period has expired, the next hit set is used. If the number of hit sets is exhausted, the memory from the oldest hit set is freed and a new hit set is created. The values of `hit_set_count` and `hit_set_period` multiplied by each other define the overall time frame in which access to objects has been tracked.

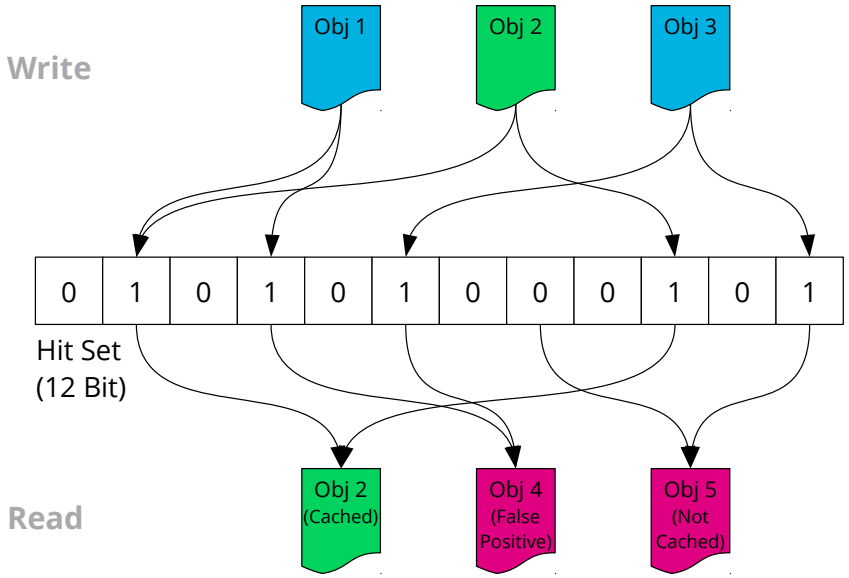

FIGURE 7.1: BLOOM FILTER WITH 3 STORED OBJECTS

Compared to the number of hashed objects, a hit set based on a bloom filter is very memory-efficient. Less than 10 bits are required to reduce the false positive probability below 1%. The false positive probability can be defined with `hit_set_fpp`. Based on the number of objects in a placement group and the false positive probability Ceph automatically calculates the size of the hit set.

The required storage on the cache pool can be limited with `min_write_recency_for_promote` and `min_read_recency_for_promote`. If the value is set to `0`, all objects are promoted to the cache pool as soon as they are read or written and this persists until they are evicted. Any value greater than `0` defines the number of hit sets ordered by age that are searched for the object. If the object is found in a hit set, it will be promoted to the cache pool. Keep in mind that backing up objects may also cause them to be promoted to the cache. A full backup with the value of '0' can cause all data to be promoted to the cache tier while active data gets removed from the cache tier. Therefore, changing this setting based on the backup strategy may be useful.

Note

The longer the period and the higher the `min_read_recency_for_promote` and `min_write_recency_for_promote` values, the more RAM the `ceph-osd` daemon consumes. In particular, when the agent is active to flush or evict cache objects, all `hit_set_count` hit sets are loaded into RAM.

7.7.1.1 Use GMT for Hit Set

Cache tier setups have a bloom filter called *hit set*. The filter tests whether an object belongs to a set of either hot or cold objects. The objects are added to the hit set using time stamps appended to their names.

If cluster machines are placed in different time zones and the time stamps are derived from the local time, objects in a hit set can have misleading names consisting of future or past time stamps. In the worst case, objects may not exist in the hit set at all.

To prevent this, the `use_gmt_hitset` defaults to '1' on a newly created cache tier setups. This way, you force OSDs to use GMT (Greenwich Mean Time) time stamps when creating the object names for the hit set.

Warning: Leave the Default Value

Do not touch the default value '1' of `use_gmt_hitset`. If errors related to this option are not caused by your cluster setup, never change it manually. Otherwise, the cluster behavior may become unpredictable.

7.7.2 Cache Sizing

The cache tiering agent performs two main functions:

Flushing

The agent identifies modified (dirty) objects and forwards them to the storage pool for long-term storage.

Evicting

The agent identifies objects that have not been modified (clean) and evicts the least recently used among them from the cache.

7.7.2.1 Absolute Sizing

The cache tiering agent can flush or evict objects based on the total number of bytes or the total number of objects. To specify a maximum number of bytes, execute the following:

```
cephadm@adm > ceph osd pool set cachepool target_max_bytes num_of_bytes
```

To specify the maximum number of objects, execute the following:

```
cephadm@adm > ceph osd pool set cachepool target_max_objects num_of_objects
```

Note

Ceph is not able to determine the size of a cache pool automatically, therefore configuration of the absolute size is required here. Otherwise, flush and evict will not work. If you specify both limits, the cache tiering agent will begin flushing or evicting when either threshold is triggered.

Note

All client requests will be blocked only when `target_max_bytes` or `target_max_objects` is reached.

7.7.2.2 Relative Sizing

The cache tiering agent can flush or evict objects relative to the size of the cache pool (specified by `target_max_bytes` or `target_max_objects` in *Section 7.7.2.1, "Absolute Sizing"*). When the cache pool consists of a certain percentage of modified (dirty) objects, the cache tiering agent will flush them to the storage pool. To set the `cache_target_dirty_ratio`, execute the following:

```
cephadm@adm > ceph osd pool set cachepool cache_target_dirty_ratio 0.0...1.0
```

For example, setting the value to 0.4 will begin flushing modified (dirty) objects when they reach 40% of the cache pool's capacity:

```
cephadm@adm > ceph osd pool set hot-storage cache_target_dirty_ratio 0.4
```

When the dirty objects reach a certain percentage of the capacity, flush them at a higher speed. Use `cache_target_dirty_high_ratio`:

```
cephadm@adm > ceph osd pool set cachepool cache_target_dirty_high_ratio 0.0..1.0
```

When the cache pool reaches a certain percentage of its capacity, the cache tiering agent will evict objects to maintain free capacity. To set the `cache_target_full_ratio`, execute the following:

```
cephadm@adm > ceph osd pool set cachepool cache_target_full_ratio 0.0..1.0
```

7.7.3 Cache Age

You can specify the minimum age of a recently modified (dirty) object before the cache tiering agent flushes it to the backing storage pool. Note that this will only apply if the cache actually needs to flush/evict objects:

```
cephadm@adm > ceph osd pool set cachepool cache_min_flush_age num_of_seconds
```

You can specify the minimum age of an object before it will be evicted from the cache tier:

```
cephadm@adm > ceph osd pool set cachepool cache_min_evict_age num_of_seconds
```

7.7.4 Examples

7.7.4.1 Large Cache Pool and Small Memory

If lots of storage and only a small amount of RAM is available, all objects can be promoted to the cache pool as soon as they are accessed. The hit set is kept small. The following is a set of example configuration values:

```
hit_set_count = 1
hit_set_period = 3600
hit_set_fpp = 0.05
min_write_recency_for_promote = 0
min_read_recency_for_promote = 0
```

7.7.4.2 Small Cache Pool and Large Memory

If a small amount of storage but a comparably large amount of memory is available, the cache tier can be configured to promote a limited number of objects into the cache pool. Twelve hit sets, of which each is used over a period of 14,400 seconds, provide tracking for a total of 48 hours. If an object has been accessed in the last 8 hours, it is promoted to the cache pool. The set of example configuration values then is:

```
hit_set_count = 12
hit_set_period = 14400
hit_set_fpp = 0.01
min_write_recency_for_promote = 2
min_read_recency_for_promote = 2
```

8 Improving Performance with LVM cache

Warning: Technology Preview

LVM cache is currently a technology preview.

LVM cache is a caching mechanism used to improve the performance of a logical volume (LV). Typically, a smaller and faster device is used to improve I/O performance of a larger and slower LV. Refer to its manual page (`man 7 lvmcache`) to find more details about LVM cache.

In SUSE Enterprise Storage, LVM cache can improve the performance of OSDs. Support for LVM cache is provided via a `ceph-volume` plugin. You can find detailed information about its usage by running `ceph-volume lvmcache`.

8.1 Prerequisites

To use LVM cache features to improve the performance of a Ceph cluster, you need to have:

- A running Ceph cluster in a stable state ('HEALTH_OK').
- OSDs deployed with BlueStore and LVM. This is the default if the OSDs were deployed using SUSE Enterprise Storage 6 or later.
- Empty disks or partitions that will be used for caching.

8.2 Points to Consider

Consider the following points before configuring your OSDs to use LVM cache:

- **Verify that LVM cache is suitable for your use case.** If you have only a few fast drives available that are **not** used for OSDs, the general recommendation is to use them as WAL/DB devices for the OSDs. In such a case, WAL and DB operations (small and rare operations) are applied on the fast drive while data operations are applied on the slower OSD drive.

Tip

If latency is more important for your deployment than IOPS or throughput, you can use the fast drives as LVM cache rather than WAL/DB partitions.

- If you plan to use a fast drive as an LVM cache for multiple OSDs, be aware that **all OSD operations (including replication) will go through the caching device**. All reads will be queried from the caching device, and are only served from the slow device in case of a cache miss. Writes are always applied to the caching device first, and are flushed to the slow device at a later time ('writeback' is the default caching mode).
  When deciding whether to utilize an LVM cache, verify whether the fast drive can serve as a front for multiple OSDs while still providing an acceptable amount of IOPS. You can test it by measuring the maximum amount of IOPS that the fast device can serve, and then dividing the result by the number of OSDs behind the fast device. If the result is lower or close to the maximum amount of IOPS that the OSD can provide without the cache, LVM cache is probably not suited for this setup.

- **The interaction of the LVM cache device with OSDs is important.** Writes are periodically flushed from the caching device to the slow device. If the incoming traffic is sustained and significant, the caching device will struggle to keep up with incoming requests as well as the flushing process, resulting in performance drop. Unless the fast device can provide much more IOPS with better latency than the slow device, do not use LVM cache with a sustained high volume workload. Traffic in a burst pattern is more suited for LVM cache as it gives the cache time to flush its dirty data without interfering with client traffic. For a sustained low traffic workload, it is difficult to guess in advance whether using LVM cache will improve performance. The best test is to benchmark and compare the LVM cache setup against the WAL/DB setup. Moreover, as small writes are heavy on the WAL partition, it is suggested to use the fast device for the DB and/or WAL instead of an LVM cache.

- If you are not sure whether to use LVM cache, use the fast device as a WAL and/or DB device.

8.3 Preparation

You need to split the fast device into multiple partitions. Each OSD needs two partitions for its cache: one for the cache data, and one for the cache metadata. The minimum size of either partition is 2 GB. You can use a single fast device to cache multiple OSDs. It simply needs to be partitioned accordingly.

8.4 Configuring LVM cache

You can find detailed information about adding, removing, and configuring LVM cache by running the `ceph-volume lvmcache` command.

8.4.1 Adding LVM cache

To add LVM cache to an existing OSD, use the following command:

```
cephadm@osd > ceph-volume lvmcache add
 --cachemetadata METADATA-PARTITION
 --cachedata DATA-PARTITION
 --osd-id OSD-ID
```

The optional `--data`, `--db` or `--wal` specifies which partition to cache. Default is `--data`.

Tip: Specify Logical Volume (LV)

Alternatively, you can use the `--origin` instead of the `--osd-id` option to specify which LV to cache:

```
[...]
--origin VOLUME-GROUP/LOGICAL-VOLUME
```

8.4.2 Removing LVM cache

To remove existing LVM cache from an OSD, use the following command:

```
cephadm@osd > ceph-volume lvmcache rm --osd-id OSD-ID
```

8.4.3 Setting LVM cache Mode

To specify caching mode, use the following command:

```
cephadm@osd > ceph-volume lvmcache mode --set CACHING-MODE --osd-id OSD-ID
```

CACHING-MODE is either 'writeback' (default) or 'writethrough'

8.5 Handling Failures

If the caching device fails, all OSDs behind the caching device need to be removed from the cluster (see *Book "Administration Guide", Chapter 2 "Salt Cluster Administration", Section 2.7 "Removing an OSD"*), purged, and redeployed. If the OSD drive fails, the OSD's LV as well as its cache's LV will be active but not functioning. Use **pvremove** *PARTITION* to purge the partitions (physical volumes) used for the OSD's cache data and metadata partitions. You can use **pvs** to list all physical volumes.

8.6 Frequently Asked Questions

Q: 1. *What happens if an OSD is removed?*

When removing the OSD's LV using **lvremove**, the cache LVs will be removed as well. However, you will still need to call **pvremove** on the partitions to make sure all labels have been wiped out.

Q: 2. *What happens if the OSD is zapped using* **ceph-volume zap***?*

The same answer applies as to the question **What happens if an OSD is removed?**

Q: 3. *What happens if the origin drive fails?*

The cache LVs still exist and **cache info** still shows them as being available. You will not be able to uncache because LVM will fail to flush the cache as the origin LV's device is gone. The situation now is that the origin LV exists, but its backing device does not. You can fix it by using the **pvs** command and locating the devices that are associated with the origin LV. You can then remove them using

```
cephadm@osd > sudo pvremove /dev/DEVICE or PARTITION
```

You can do the same for the cache partitions. This procedure will make the origin LV as well as the cache LVs disappear. You can also use

```
cephadm@osd > sudo dd if=/dev/zero of=/dev/DEVICE or PARTITION
```

to wipe them out before using **pvremove**.

A Salt State for Kernel Tuning

There are a significant number of items that can be tuned to provide improved performance of a Ceph cluster. While the material provided in previous sections attempts to showcase important items, it should not be considered exhaustive. It is highly recommended that use of the information contained herein be done so with a scientific approach to understanding the performance needs and outcomes of specific tuning in comparison with baseline performance.

To utilize this Salt state follow these steps:

1. Create directory named `my_kerntune` in `/srv/salt/`.
2. Create `/srv/salt/my_kerntune/init.sls` with the following contents:

```
 my_kern_tune:
   file.replace:
     - name: /etc/default/grub
     - pattern: showopts.*
     - repl: showopts intel_idle.max_cstate=0 processor.max_cstate=0 idle=poll
scsi_mod.use_blk_mq=1 nospec spectre_v2=off pti=off spec_store_bypass_disable=off
l1tf=off"

 grub2_mkconfig:
   cmd.run:
     - runas: root
     - name: grub2-mkconfig -o /boot/grub2/grub.cfg
     - onchanges:
       - file: my_kern_tune
```

3. Issue the following command to set the state:

```
salt '*' state.apply my_kerntune
```

4. Reboot the nodes.

Warning

Verify the **`grub`** kernel command line and ensure the pattern match specified by the pattern: parameter is appropriate. As is, this will overwrite anything after the `showopts` kernel command line argument.

B Ring Buffer Max Value Script

This script is intended to be run on a single host at a time. It will set all interfaces on the host to the maximum ring buffer values. It may be used with some sort of orchestration, for example Salt, to apply to all hosts in a cluster.

```
 for i in `ls /etc/sysconfig/network/ifcfg-*`;do
   nicname=`echo $i|cut -f2 -d"-"`
   echo nicname=$nicname
   if [ `ethtool -g $nicname 2>/dev/null |wc -l ` -gt 6 ]; then
     ethtoolcmd="-G $nicname rx `ethtool -g $nicname|head -6|grep RX:|cut -f2 -d":"|
xargs` tx `ethtool -g $nicname|head -6|grep TX:|cut -f2 -d":"|xargs`"
     echo ethtoolcmd=$ethtoolcmd
     sed -i "s/ETHTOOL_OPTIONS=''/ETHTOOL_OPTIONS='$ethtoolcmd'/g" $i
   fi
   done
```

C Network Tuning

This script uses Salt to apply a number of settings to a cluster. This was used during testing so that parameters could be rapidly adjusted. It is intended to be an example only and is not intended for use without modification.

```
#!/bin/bash
#These commands tune the write size of the pcie interface for the NIC cards.
salt '*' cmd.run 'setpci -s 5b:00.0 68.w=5936'
salt '*' cmd.run 'setpci -s 5b:00.1 68.w=5936'

#Set Jumbo Frames
salt '*' cmd.run 'ip link set bond0 mtu 9000'

#Set the rx queue for rx-0 to use all CPU cores local to the device.
salt '*' cmd.run 'for j in `cat /sys/class/net/bond0/bonding/slaves`;do
LOCAL_CPUS=`cat /sys/class/net/$j/device/local_cpus`;echo $LOCAL_CPUS > /sys/class/net/
$j/queues/rx-0/rps_cpus;done'

#Set send and recieve buffers for both network interfaces involved in the bond
salt '*' cmd.run 'ethtool -G eth4 rx 8192 tx 8192'
salt '*' cmd.run 'ethtool -G eth5 rx 8192 tx 8192'

#Ensure SACK is on
salt '*' cmd.run 'sysctl -w net.ipv4.tcp_sack=1'
#Ensure timestamps are on to prevent possible drops
salt '*' cmd.run 'sysctl -w net.ipv4.tcp_timestamps=1'

#Set the max conections to 2048
salt '*' cmd.run 'sysctl -w net.core.somaxconn=2048'

#Set TCP to low latency
salt '*' cmd.run 'sysctl -w net.ipv4.tcp_low_latency=1'
salt '*' cmd.run 'sysctl -w net.ipv4.tcp_fastopen=1'

#Set min, default, and max send and receive buffers
salt '*' cmd.run 'sysctl -w net.ipv4.tcp_rmem="10240 87380 2147483647"'
salt '*' cmd.run 'sysctl -w net.ipv4.tcp_wmem="10240 87380 2147483647"'

salt '*' cmd.run 'sysctl -w net.core.netdev_max_backlog=250000'
```

D Ceph Maintenance Updates Based on Upstream 'Nautilus' Point Releases

Several key packages in SUSE Enterprise Storage 6 are based on the Nautilus release series of Ceph. When the Ceph project (https://github.com/ceph/ceph) publishes new point releases in the Nautilus series, SUSE Enterprise Storage 6 is updated to ensure that the product benefits from the latest upstream bugfixes and feature backports.

This chapter contains summaries of notable changes contained in each upstream point release that has been—or is planned to be—included in the product.

Nautilus 14.2.20 Point Release

This release includes a security fix that ensures the `global_id` value (a numeric value that should be unique for every authenticated client or daemon in the cluster) is reclaimed after a network disconnect or ticket renewal in a secure fashion. Two new health alerts may appear during the upgrade indicating that there are clients or daemons that are not yet patched with the appropriate fix.

To temporarily mute the health alerts around insecure clients for the duration of the upgrade, you may want to run:

```
cephadm@adm > ceph health mute AUTH_INSECURE_GLOBAL_ID_RECLAIM 1h
cephadm@adm > ceph health mute AUTH_INSECURE_GLOBAL_ID_RECLAIM_ALLOWED 1h
```

When all clients are updated, enable the new secure behavior, not allowing old insecure clients to join the cluster:

```
cephadm@adm > ceph config set mon auth_allow_insecure_global_id_reclaim false
```

For more details, refer ro https://docs.ceph.com/en/latest/security/CVE-2021-20288/.

Nautilus 14.2.18 Point Release

This release fixes a regression introduced in 14.2.17 in which the manager module tries to use a couple of Python modules that do not exist in some environments.

- This release fixes issues loading the dashboard and volumes manager modules in some environments.

Nautilus 14.2.17 Point Release

This release includes the following fixes:

- `$pid` expansion in configuration paths such as `admin_socket` will now properly expand to the daemon PID for commands like **`ceph-mds`** or **`ceph-osd`**. Previously, only **`ceph-fuse`** and **`rbd-nbd`** expanded `$pid` with the actual daemon PID.
- RADOS: PG removal has been optimized.
- RADOS: Memory allocations are tracked in finer detail in BlueStore and displayed as a part of the **`dump_mempools`** command.
- CephFS: clients which acquire capabilities too quickly are throttled to prevent instability. See new config option `mds_session_cap_acquisition_throttle` to control this behavior.

Nautilus 14.2.16 Point Release

This release fixes a security flaw in CephFS.

- CVE-2020-27781 : OpenStack Manila use of **`ceph_volume_client.py`** library allowed tenant access to any Ceph credentials' secret.

Nautilus 14.2.15 Point Release

This release fixes a ceph-volume regression introduced in v14.2.13 and includes few other fixes.

- ceph-volume: Fixes **`lvm batch –auto`**, which breaks backward compatibility when using non rotational devices only (SSD and/or NVMe).
- BlueStore: Fixes a bug in `collection_list_legacy` which makes PGs inconsistent during scrub when running OSDs older than 14.2.12 with newer ones.
- MGR: progress module can now be turned on or off, using the commands **`ceph progress on`** and **`ceph progress off`**.

Nautilus 14.2.14 Point Release

This releases fixes a security flaw affecting Messenger V2 for Octopus and Nautilus, among other fixes across components.

- CVE 2020-25660: Fix a regression in Messenger V2 replay attacks.

Nautilus 14.2.13 Point Release

This release fixes a regression introduced in v14.2.12, and a few ceph-volume amd RGW fixes.

- Fixed a regression that caused breakage in clusters that referred to ceph-mon hosts using dns names instead of IP addresses in the `mon_host` parameter in `ceph.conf`.
- ceph-volume: the **`lvm batch`** subcommand received a major rewrite.

Nautilus 14.2.12 Point Release

In addition to bug fixes, this major upstream release brought a number of notable changes:

- The **`ceph df command`** now lists the number of PGs in each pool.
- MONs now have a config option `mon_osd_warn_num_repaired`, 10 by default. If any OSD has repaired more than this many I/O errors in stored data, a `OSD_TOO_MANY_REPAIRS` health warning is generated. In order to allow clearing of the warning, a new command **`ceph tell osd.`***`SERVICE_ID`* **`clear_shards_repaired`** *`COUNT`* has been added. By default, it will set the repair count to 0. If you want to be warned again if additional repairs are performed, you can provide a value to the command and specify the value of `mon_osd_warn_num_repaired`. This command will be replaced in future releases by the health mute/unmute feature.
- It is now possible to specify the initial MON to contact for Ceph tools and daemons using the `mon_host_override config` option or `--mon-host-override` *`IP`* command-line switch. This generally should only be used for debugging and only affects initial communication with Ceph's MON cluster.
- Fix an issue with osdmaps not being trimmed in a healthy cluster.

Nautilus 14.2.11 Point Release

In addition to bug fixes, this major upstream release brought a number of notable changes:

- RGW: The **`radosgw-admin`** sub-commands dealing with orphans – **`radosgw-admin orphans find`**, **`radosgw-admin orphans finish`**, **`radosgw-admin orphans list-jobs`** – have been deprecated. They have not been actively maintained and they store intermediate results on the cluster, which could fill a nearly-full cluster. They have been replaced by a tool, currently considered experimental, **`rgw-orphan-list`**.
- Now, when `noscrub` and/or `nodeep-scrub` flags are set globally or per pool, scheduled scrubs of the type disabled will be aborted. All user initiated scrubs are *not* interrupted.
- Fixed a ceph-osd crash in committed OSD maps when there is a failure to encode the first incremental map.

Nautilus 14.2.10 Point Release

This upstream release patched one security flaw:

- CVE-2020-10753: rgw: sanitize newlines in s3 CORSConfiguration's ExposeHeader

In addition to security flaws, this major upstream release brought a number of notable changes:

- The pool parameter `target_size_ratio`, used by the PG autoscaler, has changed meaning. It is now normalized across pools, rather than specifying an absolute ratio. If you have set target size ratios on any pools, you may want to set these pools to autoscale `warn` mode to avoid data movement during the upgrade:

  ```
  ceph osd pool set POOL_NAME pg_autoscale_mode warn
  ```

- The behaviour of the `-o` argument to the RADOS tool has been reverted to its original behaviour of indicating an output file. This reverts it to a more consistent behaviour when compared to other tools. Specifying object size is now accomplished by using an upper case O `-O`.
- The format of MDSs in **`ceph fs dump`** has changed.

- Ceph will issue a health warning if a RADOS pool's `size` is set to 1 or, in other words, the pool is configured with no redundancy. This can be fixed by setting the pool size to the minimum recommended value with:

  ```
  cephadm@adm > ceph osd pool set pool-name size num-replicas
  ```

  The warning can be silenced with:

  ```
  cephadm@adm > ceph config set global mon_warn_on_pool_no_redundancy false
  ```

- RGW: bucket listing performance on sharded bucket indexes has been notably improved by heuristically – and significantly, in many cases – reducing the number of entries requested from each bucket index shard.

Nautilus 14.2.9 Point Release

This upstream release patched two security flaws:

- CVE-2020-1759: Fixed nonce reuse in msgr V2 secure mode
- CVE-2020-1760: Fixed XSS due to RGW GetObject header-splitting

In SES 6, these flaws were patched in Ceph version 14.2.5.389+gb0f23ac248.

Nautilus 14.2.8 Point Release

In addition to bug fixes, this major upstream release brought a number of notable changes:

- The default value of `bluestore_min_alloc_size_ssd` has been changed to 4K to improve performance across all workloads.
- The following OSD memory config options related to BlueStore cache autotuning can now be configured during runtime:

  ```
  osd_memory_base (default: 768 MB)
  osd_memory_cache_min (default: 128 MB)
  osd_memory_expected_fragmentation (default: 0.15)
  osd_memory_target (default: 4 GB)
  ```

You can set the above options by running:

```
cephadm@adm > ceph config set osd OPTION VALUE
```

- The Ceph Manager now accepts `profile rbd` and `profile rbd-read-only` user capabilities. You can use these capabilities to provide users access to MGR-based RBD functionality such as `rbd perf image iostat` and `rbd perf image iotop`.

- The configuration value `osd_calc_pg_upmaps_max_stddev` used for upmap balancing has been removed. Instead, use the Ceph Manager balancer configuration option `upmap_max_deviation` which now is an integer number of PGs of deviation from the target PGs per OSD. You can set it with a following command:

  ```
  cephadm@adm > ceph config set mgr mgr/balancer/upmap_max_deviation 2
  ```

  The default `upmap_max_deviation` is 5. There are situations where crush rules would not allow a pool to ever have completely balanced PGs. For example, if crush requires 1 replica on each of 3 racks, but there are fewer OSDs in 1 of the racks. In those cases, the configuration value can be increased.

- CephFS: multiple active Metadata Server forward scrub is now rejected. Scrub is currently only permitted on a file system with a single rank. Reduce the ranks to one via **`ceph fs set`** *`FS_NAME`* **`max_mds 1`**.

- Ceph will now issue a health warning if a RADOS pool has a `pg_num` value that is not a power of two. This can be fixed by adjusting the pool to an adjacent power of two:

  ```
  cephadm@adm > ceph osd pool set POOL_NAME pg_num NEW_PG_NUM
  ```

  Alternatively, you can silence the warning with:

  ```
  cephadm@adm > ceph config set global mon_warn_on_pool_pg_num_not_power_of_two false
  ```

Nautilus 14.2.7 Point Release

This upstream release patched two security flaws:

- CVE-2020-1699: a path traversal flaw in Ceph Dashboard that could allow for potential information disclosure.
- CVE-2020-1700: a flaw in the RGW beast front-end that could lead to denial of service from an unauthenticated client.

In SES 6, these flaws were patched in Ceph version 14.2.5.382+g8881d33957b.

Nautilus 14.2.6 Point Release

This release fixed a Ceph Manager bug that caused MGRs becoming unresponsive on larger clusters. SES users were never exposed to the bug.

Nautilus 14.2.5 Point Release

- **Health warnings are now issued if daemons have recently crashed.** Ceph will now issue health warnings if daemons have recently crashed. Ceph has been collecting crash reports since the initial Nautilus release, but the health alerts are new. To view new crashes (or all crashes, if you have just upgraded), run:

  ```
  cephadm@adm > ceph crash ls-new
  ```

  To acknowledge a particular crash (or all crashes) and silence the health warning, run:

  ```
  cephadm@adm > ceph crash archive CRASH-ID
  cephadm@adm > ceph crash archive-all
  ```

- `pg_num` **must be a power of two, otherwise** `HEALTH_WARN` **is reported.** Ceph will now issue a health warning if a RADOS pool has a `pg_num` value that is not a power of two. You can fix this by adjusting the pool to a nearby power of two:

  ```
  cephadm@adm > ceph osd pool set POOL-NAME pg_num NEW-PG-NUM
  ```

Alternatively, you can silence the warning with:

```
cephadm@adm > ceph config set global mon_warn_on_pool_pg_num_not_power_of_two false
```

- **Pool size needs to be greater than 1 otherwise `HEALTH_WARN` is reported.** Ceph will issue a health warning if a RADOS pool's size is set to 1 or if the pool is configured with no redundancy. Ceph will stop issuing the warning if the pool size is set to the minimum recommended value:

  ```
  cephadm@adm > ceph osd pool set POOL-NAME size NUM-REPLICAS
  ```

  You can silence the warning with:

  ```
  cephadm@adm > ceph config set global mon_warn_on_pool_no_redundancy false
  ```

- **Health warning is reported if average OSD heartbeat ping time exceeds the threshold.** A health warning is now generated if the average OSD heartbeat ping time exceeds a configurable threshold for any of the intervals computed. The OSD computes 1 minute, 5 minute and 15 minute intervals with average, minimum, and maximum values.
  A new configuration option, `mon_warn_on_slow_ping_ratio`, specifies a percentage of `osd_heartbeat_grace` to determine the threshold. A value of zero disables the warning.
  A new configuration option, `mon_warn_on_slow_ping_time`, specified in milliseconds, overrides the computed value and causes a warning when OSD heartbeat pings take longer than the specified amount.
  A new command **`ceph daemon mgr.`***`MGR-NUMBER`* **`dump_osd_network`** *`THRESHOLD`* lists all connections with a ping time longer than the specified threshold or value determined by the configuration options, for the average for any of the 3 intervals.
  A new command **`ceph daemon osd.# dump_osd_network`** *`THRESHOLD`* will do the same as the previous one but only including heartbeats initiated by the specified OSD.

- **Changes in the telemetry MGR module.**
  A new 'device' channel (enabled by default) will report anonymized hard disk and SSD health metrics to `telemetry.ceph.com` in order to build and improve device failure prediction algorithms.
  Telemetry reports information about CephFS file systems, including:

  - How many MDS daemons (in total and per file system).
  - Which features are (or have been) enabled.

- How many data pools.
- Approximate file system age (year and the month of creation).
- How many files, bytes, and snapshots.
- How much metadata is being cached.

Other miscellaneous information:

- Which Ceph release the monitors are running.
- Whether msgr v1 or v2 addresses are used for the monitors.
- Whether IPv4 or IPv6 addresses are used for the monitors.
- Whether RADOS cache tiering is enabled (and the mode).
- Whether pools are replicated or erasure coded, and which erasure code profile plug-in and parameters are in use.
- How many hosts are in the cluster, and how many hosts have each type of daemon.
- Whether a separate OSD cluster network is being used.
- How many RBD pools and images are in the cluster, and how many pools have RBD mirroring enabled.
- How many RGW daemons, zones, and zonegroups are present and which RGW frontends are in use.
- Aggregate stats about the CRUSH Map, such as which algorithms are used, how big buckets are, how many rules are defined, and what tunables are in use.

If you had telemetry enabled before 14.2.5, you will need to re-opt-in with:

```
cephadm@adm > ceph telemetry on
```

If you are not comfortable sharing device metrics, you can disable that channel first before re-opting-in:

```
cephadm@adm > ceph config set mgr mgr/telemetry/channel_device false
cephadm@adm > ceph telemetry on
```

You can view exactly what information will be reported first with:

```
cephadm@adm > ceph telemetry show        # see everything
cephadm@adm > ceph telemetry show device # just the device info
cephadm@adm > ceph telemetry show basic  # basic cluster info
```

- **New OSD daemon command `dump_recovery_reservations`.** It reveals the recovery locks held (`in_progress`) and waiting in priority queues. Usage:

```
cephadm@adm > ceph daemon osd.ID dump_recovery_reservations
```

- **New OSD daemon command `dump_scrub_reservations`.** It reveals the scrub reservations that are held for local (primary) and remote (replica) PGs. Usage:

```
cephadm@adm > ceph daemon osd.ID dump_scrub_reservations
```

- **RGW now supports S3 Object Lock set of APIs.** RGW now supports S3 Object Lock set of APIs allowing for a WORM model for storing objects. 6 new APIs have been added PUT/GET bucket object lock, PUT/GET object retention, PUT/GET object legal hold.

- **RGW now supports List Objects V2.** RGW now supports List Objects V2 as specified at https://docs.aws.amazon.com/AmazonS3/latest/API/API_ListObjectsV2.html.

Nautilus 14.2.4 Point Release

This point release fixes a serious regression that found its way into the 14.2.3 point release. This regression did not affect SUSE Enterprise Storage customers because we did not ship a version based on 14.2.3.

Nautilus 14.2.3 Point Release

- Fixed a denial of service vulnerability where an unauthenticated client of Ceph Object Gateway could trigger a crash from an uncaught exception.

- Nautilus-based librbd clients can now open images on Jewel clusters.

- The Object Gateway `num_rados_handles` has been removed. If you were using a value of `num_rados_handles` greater than 1, multiply your current `objecter_inflight_ops` and `objecter_inflight_op_bytes` parameters by the old `num_rados_handles` to get the same throttle behavior.
- The secure mode of Messenger v2 protocol is no longer experimental with this release. This mode is now the preferred mode of connection for monitors.
- `osd_deep_scrub_large_omap_object_key_threshold` has been lowered to detect an object with a large number of omap keys more easily.
- The Ceph Dashboard now supports silencing Prometheus notifications.

Nautilus 14.2.2 Point Release

- The `no{up,down,in,out}` related commands have been revamped. There are now two ways to set the `no{up,down,in,out}` flags: the old command

  ```
  ceph osd [un]set FLAG
  ```

  which sets cluster-wide flags; and the new command

  ```
  ceph osd [un]set-group FLAGS WHO
  ```

  which sets flags in batch at the granularity of any crush node or device class.
- **`radosgw-admin`** introduces two subcommands that allow the managing of expire-stale objects that might be left behind after a bucket reshard in earlier versions of Object Gateway. Expire-stale objects are expired objects that should have been automatically erased but still exist and need to be listed and removed manually. One subcommand lists such objects and the other deletes them.
- Earlier Nautilus releases (14.2.1 and 14.2.0) have an issue where deploying a single new Nautilus BlueStore OSD on an upgraded cluster (i.e. one that was originally deployed pre-Nautilus) breaks the pool utilization statistics reported by **`ceph df`**. Until all OSDs have been reprovisioned or updated (via **`ceph-bluestore-tool repair`**), the pool statistics will show values that are lower than the true value. This is resolved in 14.2.2, such that

the cluster only switches to using the more accurate per-pool stats after *all* OSDs are 14.2.2 or later, are Block Storage, and have been updated via the repair function if they were created prior to Nautilus.

- The default value for `mon_crush_min_required_version` has been changed from `firefly` to `hammer`, which means the cluster will issue a health warning if your CRUSH tunables are older than Hammer. There is generally a small (but non-zero) amount of data that will be re-balanced after making the switch to Hammer tunables.
  If possible, we recommend that you set the oldest allowed client to `hammer` or later. To display what the current oldest allowed client is, run:

  ```
  cephadm@adm > ceph osd dump | grep min_compat_client
  ```

  If the current value is older than `hammer`, run the following command to determine whether it is safe to make this change by verifying that there are no clients older than Hammer currently connected to the cluster:

  ```
  cephadm@adm > ceph features
  ```

  The newer `straw2` CRUSH bucket type was introduced in Hammer. If you verify that all clients are Hammer or newer, it allows new features only supported for `straw2` buckets to be used, including the `crush-compat` mode for the Balancer (*Book “Administration Guide”, Chapter 21 “Ceph Manager Modules”, Section 21.1 “Balancer”*).

Find detailed information about the patch at https://download.suse.com/Download?buildid=D38A7mekBz4~

Nautilus 14.2.1 Point Release

This was the first point release following the original Nautilus release (14.2.0). The original ('General Availability' or 'GA') version of SUSE Enterprise Storage 6 was based on this point release.

Glossary

General

Admin node

The node from which you run the `ceph-deploy` utility to deploy Ceph on OSD nodes.

Bucket

A point that aggregates other nodes into a hierarchy of physical locations.

> **Important: Do Not Mix with S3 Buckets**
>
> S3 *buckets* or *containers* represent different terms meaning *folders* for storing objects.

CRUSH, CRUSH Map

Controlled Replication Under Scalable Hashing: An algorithm that determines how to store and retrieve data by computing data storage locations. CRUSH requires a map of the cluster to pseudo-randomly store and retrieve data in OSDs with a uniform distribution of data across the cluster.

Monitor node, MON

A cluster node that maintains maps of cluster state, including the monitor map, or the OSD map.

Node

Any single machine or server in a Ceph cluster.

OSD

Depending on context, *Object Storage Device* or *Object Storage Daemon*. The `ceph-osd` daemon is the component of Ceph that is responsible for storing objects on a local file system and providing access to them over the network.

OSD node

A cluster node that stores data, handles data replication, recovery, backfilling, rebalancing, and provides some monitoring information to Ceph monitors by checking other Ceph OSD daemons.

PG

Placement Group: a sub-division of a *pool*, used for performance tuning.

Pool

Logical partitions for storing objects such as disk images.

Routing tree

A term given to any diagram that shows the various routes a receiver can run.

Rule Set

Rules to determine data placement for a pool.

Ceph Specific Terms

Alertmanager

A single binary which handles alerts sent by the Prometheus server and notifies end user.

Ceph Storage Cluster

The core set of storage software which stores the user's data. Such a set consists of Ceph monitors and OSDs.

AKA "Ceph Object Store".

Grafana

Database analytics and monitoring solution.

Prometheus

Systems monitoring and alerting toolkit.

Object Gateway Specific Terms

archive sync module

Module that enables creating an Object Gateway zone for keeping the history of S3 object versions.

Object Gateway

The S3/Swift gateway component for Ceph Object Store.