CephFS for Media and Entertainment Workloads on SUSE Enterprise Storage

Publication Date: 2019-10-24

Contents

1 Introduction

The Media and Entertainment (M&E) industry has large and demanding storage requirements. These requirements vary by the production stage of the content. During early production, the requirement may be as simple as providing basic NAS services, while the rendering phase, likely requires relatively low latency and high throughput, and at the end of the process where delivery to the end users is the goal, the ability to provide massive content storage that operates using web native protocols is required.

While the focus of this paper is on the content creation portion of the lifecycle, SUSE Enterprise Storage also supports the latter part of the content lifecycle. Since large volumes of content are retained for long periods of time, consumption based pricing can make storage costs unsustainable. SUSE Enterprise Storage is ideally suited for such content because of its by-the-node pricing strategy, and its ability to store massive amounts of data, which scales well into the exabyte level, and distribute content via modern web-based protocols. Some examples of these use cases include content repositories for short-term retention or long-term archival, geo-distributed content repositories and origin servers for content delivery networks. An exhaustive discussion of the applicability of SUSE Enterprise Storage in these use cases is beyond the scope of this document, but will be covered in a future white paper.

The ever increasing resolution of digital imagery and the productions that are utilizing it create significant demands for performance and scalability of the storage systems utilized in the creation process. In an effort to provide a platform that can address the performance and scalability requirements, SUSE and Intel have undertaken an effort to build and tune an environment leveraging CephFS, a scalable, POSIX-compliant distributed file system that is part of the open-source object storage project, Ceph.

This white paper will provide an overview of the environment utilized, tests performed, the results, and most importantly, the specific tuning and configuration needed to replicate the results. Upon following the steps in this document, a working SUSE Enterprise Storage (v5.5) cluster will be operational and tuned for the high-performance requirements of the M&E industry.

2 Target Audience

This reference guide is targeted at architects and administrators who need to deploy a software defined storage solution to satisfy the needs of a M&E production environment. Familiarity with Linux and Ceph are assumed.

3 Business Value

SUSE Enterprise Storage

SUSE Enterprise Storage is a Ceph-based product that delivers a highly scalable, resilient, self-healing storage system designed for large-scale environments ranging from hundreds of Terabytes to Exabytes. This software defined storage product can reduce IT costs by leveraging industry-standard servers to present unified storage servicing block, file, and object protocols.

For M&E customers, the extreme scalability and flexibility of SUSE Enterprise Storage provide significant value by enabling the storage environment to grow across multi-generational hardware platforms and specify the transport protocol and data protection schemes that are most appropriate for a particular workload.

4 Hardware & Software

This work leveraged SUSE Enterprise Storage on two models of servers. The server and storage were selected to provide an appropriate mix of performance while managing costs.

STORAGE NODES:

- 10x 2U Server
  - 1x Intel Skylake 6142
  - 96GB RAM
  - Mellanox Dual Port ConnectX-4 100GbE
  - 12x Intel SSD D3-S4510 960GB
  - RAID-1 480GB M.2 Boot Device

ADMIN, MONITOR, AND PROTOCOL GATEWAYS:

- 6x 1U Server
  - 1x Intel Skylake 4112
  - 32GB RAM
  - Mellanox Dual Port ConnectX-4 100GbE
  - RAID-1 480GB M.2 Boot Device

SWITCHES:

- 2x 32-port 100GbE

SOFTWARE:

- SUSE Enterprise Storage (v5.5)
- SUSE Linux Enterprise Server 12 SP3

TIP

Please note that limited use subscriptions are provided with SUSE Enterprise Storage as part of the subscription entitlement

5 Requirements

Media and Entertainment workloads can be quite demanding. They require large amounts of storage with the ability to support both large, sequential I/O and small random I/O at the same time. Providing a POSIX-compliant access mechanism is critical for many of the applications to work correctly as they may be coordinating many writes to the same file.

In addition to capacity and performance, the system must provide data protection as production timelines are often quite tight and simply don't allow the time to rebuild and recover from tape media only to start the project over again.

5.1 Functional Requirements

The functional requirements of a Media and Entertainment environment can vary substantially based on the particular workflows and software being utilized. This paper attempts to call out the major items to be considered when planning and designing a proper architecture.

5.1.1 Performant Storage

Performant storage addresses two different areas of storage performance - latency and throughput.

Latency

Latency is the time from when the request was made to when it was completed. Two key aspects play into this measure when addressing rendering type workloads:

Metadata Performance

This refers to the speed of getting and setting attribute data for a file. This is especially important when working with large filesystems where an application may need to walk through multiple directory structures to find and assemble a working set of data. Slowness here can have a negative geometric impact on storage performance.

Time to First Byte

This measurement indicates the time-to-first data being fed to the process. Slowness here can dramatically impact small, random I/O.

Throughput

Single Thread

Depending on the rendering environment being utilized, single thread performance may or may not be a critical requirement. Some high-performance render and editing nodes use large local NVMe devices for their primary work space. Other scenarios may dictate a requirement for higher performance per-thread. It is important to understand the workflow of a particular environment before architecting a solution.

Aggregate

Being able to service many threads simoultaneously with each at a high rate is key to servicing a render farm. If the storage is performant at a single thread, but two threads are half the speed, and throughput is halved again at four threads and so on, then it is not a well-suited solution render farms.

5.1.2 Horizontal Scalability

Horizontal scalability refers to the ability of the cluster to scale capacity and aggregate performance by adding nodes. Ideally, this should be a linearly scaling model. In reality, this is often reduced by gateway nodes, choke points in the network infrastructure, and/or lack of scalable metadata services.

5.1.3 Multi-protocol Accessibility

Only rarely do render farms utilize the same type of operating system as the content creation workstation. It is often required to create content on a workstation utiling one OS while the render farm itself uses another. Providing the ability for multiple access methods to the same storage location helps to ensure an optimized workflow is possible.

6 Requirement to Feature/Functionality Mapping

The ability to map particular features and functionality of a solution to those provided by a product is critical to ensuring a proper solution fit. This section explores the matching of SUSE Enterprise Storage to those requirements noted above.

6.1 Performant Storage

SUSE Enterprise Storage is capable of providing acceptable latency and throughput to support rendering workloads. When pushed to 272 threads using a storage infrastructure built on 120 SATA SSDs, each process was able to realize over 66 MB/s throughput on a large sequential workload. From a latency perspective, the cluster maintained a 4k random read average latency of about 4ms while delivering over 2 million IOPS to the clients.

6.2 Horizontally Scalable

When it comes to scaleability, few storage architectures come close to what SUSE Enterprise Storage can provide. Because of the underlying Ceph technology, there are no choke points when using native protocols, thus allowing both cluster storage and aggregate performance capacity to grow with every node added.

6.3 Multi-Protocol accessibility

For media creation, there are multiple platforms that are utilized for creating the content. Some of these are used for designing the scene, while a different platform may be utilized for the render farm. While each application and render pipeline manager may be different, some require that the content be in the same place on both the workstation and the render node.

By providing SMB, NFS, and CephFS, all as viable connectivity protocols, SUSE Enterprise Storage makes it possible to create the work on one platform and have the render farm mount and use the same location without issue.

7 Overall View

SUSE Enterprise Storage is well suited to provide storage to these environments. To this end, during testing, some workflows were tested with commercial software for the render phase to understand the I/O patterns and requirements. The particular software being utilized appeared to require a moderate amount relatively low-latency operations but was more biased towards high-throughput. The result is that the efforts in this guide are focused on those two goals.

The remainder of this document is dedicated to tuning the storage to meet the performance requirements of the sampled I/O needs. The results of the tuning were significant increases in performance of the cluster.

8 Implementation and Tuning

This guide assumes a working SUSE Enterprise Storage cluster is in place as described in the documentation. (https://documentation.suse.com/) Specifically, the SUSE Enterprise Storage 5 Deployment Guide (https://documentation.suse.com/ses/5.5/html/ses-all/book-storage-deployment.html) as well as SUSE Linux Enterprise Server Administration Guide. (https://documentation.suse.com/ses/5.5/html/ses-all/book-storage-admin.html)

The emphasis of this section of the document is on specific design and tuning considerations.

8.1 Network Deployment Considerations

The following considerations for the network configuration should be attended to:

- Ensure that all nodes have at least two 25GbE or faster ethernet connections.
- Network bandwidth should be at least the total bandwidth of all storage devices present in the storage node, ideally, it would be a multiple (2x or 3x).

- Usage of VLANs on LACP bonded ethernet provides the best balance of bandwidth aggregation + fault tolerance.
- The network should support jumbo frame ethernet if all nodes connecting to the storage are able to do so, otherwise use the standard MTU.

8.2 Hardware Recommendations

The following considerations for the hardware platforms should be attended to:

- Set BIOS Power/Performance controls to the performance profile. This should eliminate frequency scaling and ensure that there is no added latency caused by it.
- Enable HT on Intel CPUs. This extra processing power is utilized effectively by Ceph.
- Ensure all add-in cards are in the optimal slots for performance.
- Utilize 12G SAS devices as they have higher throughput than SATA.
- Don’t mix SAS and SATA on the same controller channel as a SAS channel runs at the speed of the slowest device on the channel.
- Utilize NVMe or Intel Optane if possible. Considering the random read workload, the SATA SSD is delivering about 18,000 IOPS per device while the Intel Optane delivers at about 82,000 IOPS per device, meaning a smaller cluster could achieve the same performance quite easily.
- In cases where millions of files are present, there is value in utilizing high performance media, such as Intel Optane to host the CephFS metadata pool.

8.3 SLES Install and Base Performance

During the OS installation, do NOT select an install pattern that includes an X server. Doing so utilizes RAM resources that are better allocated to tuning storage related daemons.

It is also proper to evaluate the various individual components that are critical to the overall performance of the storage cluster.

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

- Perform iperf3 tests for network performance, use it and consider increasing the window size (-w) and running multiple streams to fully test the bandwidth capability. If at standard MTU, the NIC should be capable of running at approximately 70-80 percent of the advertised bandwidth. Move up to jumbo-frames, and the NIC should be able to saturate the line.
  This isn't necessarily true to faster topologies like 100Gb. In those topologies, saturating the NIC can take a substantial amount of OS and NIC 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 100Gb network. In the command line, the -N disables Nagle's buffering algorithm and the -l sets the buffer length to higher than the default 128k, 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

iperf Done.
```

- Use fio to test individual storage devices to understand the per-device performance maximums. Do this for all devices to ensure there are not any that are out of spec or are connected to expanders that 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.

It is suggested to test at least random 4k, random 64k, and sequential 64k and 1MB. This gives a reasonable overall view of the device's performance characteristics. When testing, it is important to use the raw device (/dev/sdX) and to use the directio engine with multiple jobs to maximize device performance under stress.

  - Next, run fio against all devices in an OSD 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 is simulating the node under heavy load.
    It is recommended to use 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 busses

- To maximize throughput of the cluster, it is necessary to tune the performance of both the cluster and the client nodes. Information on tuning comes from a variety of sources for this work. The primary source is the System Analysis and Tuning Guide (https://documentation.suse.com/sles/12-SP4/html/SLES-all/book-sle-tuning.html) for SUSE Linux Enterprise Server. Numerous other references were utilized, including documentation from hardware vendors.

8.4 Client and Cluster Tuning

There are several aspects of the Linux kernel that can be tuned on both the cluster and some clients. It is important to understand that most tuning is about gaining small incremental improvements that, in aggregate, represent a measureable and meaningful improvement in performance.

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, disabling write cache and forcing read_ahead to 2MB resulted in the best overall performance.
By placing the following file in /etc/udev/rules.d the device is detected by the model name shown in /sys/block/{devname}/device/model and assigned to disable write caching and setting the read_ahead_kb to 2MB.

```
/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", ATTR{queue/
nr_requests}="512"
SUBSYSTEM=="block", ATTRS{model}=="INTEL SSDS*", ACTION=="add|change", RUN+="/sbin/
hdparm -W 0 /dev/%k"
```

I/O tuning

Ensure that I/O is flowing in the most optimal pattern. For the cluster used in this test, that means enable multi-queue block I/O. This is done through adding a boot-time kernel parameter as found in the Section 12.4 of the System Analysis and Tuning Guide (https://documentation.suse.com/sles/12-SP4/html/SLES-all/book-sle-tuning.html) ↗. This is not an action that should be taken unilaterally on clusters that contain spinning media devices due to potential performance degradation for those devices. The general result is that there are multiple I/O queues assigned to the device, allowing more jobs to be handled simultaneously by those high-performance device such as NVMe and SSD that can service large numbers of requests.

Kernel Tuning

The second main aspect of the kernel boot-time tuning is to disable the side-channel attack mitigations present in the kernel. The bulk of the benefits from this tuning occur with smaller I/Os ranging in size from 4k to 64k. In particular, 64K 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-chanel attacks on some CPUs. 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 command similar to below:

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

The salt state and steps used in this testing can be found in *Section 11, “Appendix A: Salt State for Kernel Tuning”* (page 25).

Note

More recent CPUs may not experience a performance gain as mitigations are accounted for in the chip design, making this step potentially unnecessary.

Network Tuning

Proper tuning of the network stack can substantially improve latency and throughput for the cluster. This may even include tuning some deep-level items, such as the PCI max write requests assigned to the networking hardware, which is the case in this particular case. Be aware that this tuning is card- and slot-specific and needs to only be done in conjunction with the conditions and instructions supplied by the manufacturer. A full script for the testing we performed can be found in ???.

The first, and often most impactful tuning is to utilize jumbo frame packets. For this to be done, all interfaces utilizing the cluster must be set to the same setting with an MTU of 9000. Network switches will often have the ports set to 9100 or higher, which is fine, as they are only passing packets, not creating them.

The following salt command was used to ensure the bonded interfaces on all nodes under control (including the test load generation nodes) were utilizing a 9k MTU.

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

To set this persistently, YaST should be utilized to set the MTU for the bonded interface.

The PCIe write requests were set with the following salt commands:

Warning

This should only be done with guidance from the NIC manufacturer.

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

The reference for this setting can be found here Understanding PCIE Configuration for Maximum Performance - Mellanox (https://community.mellanox.com/s/article/understanding-pcie-configuration-for-maximum-performance)

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/{ifname}/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'
```

Many NIC drivers start with a defult value for the RX and TX buffers that is not optimal for high-throughput scenarios and doesn't 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
```

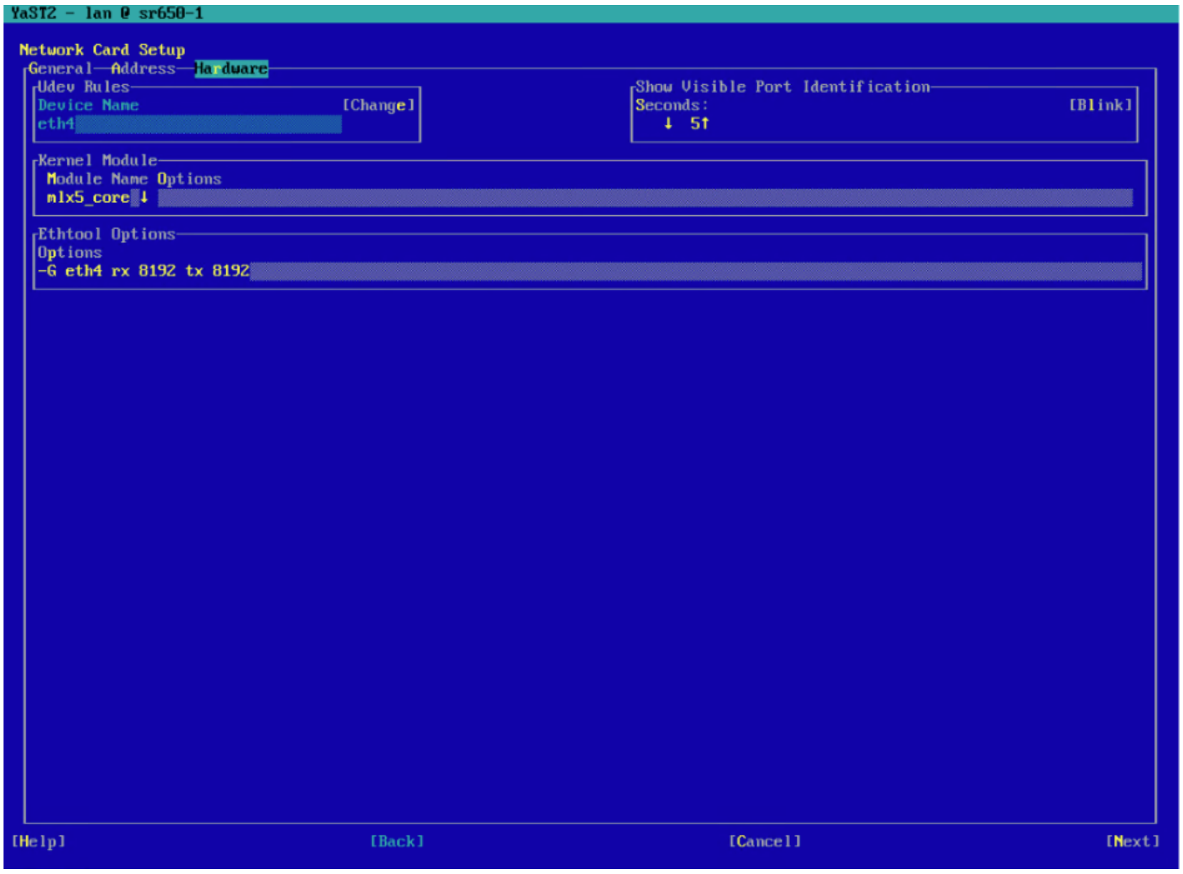
The output from this command should look similar to this:

```
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 up to 8k, but is only currently using 1k buffers. To adjust this for the cluster, the following command may be issued.

```
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.



```
YaST2 - lan @ sr650-1

Network Card Setup
General─Address─Hardware
Udev Rules
Device Name                                  [Change]
eth4
Show Visible Port Identification
Seconds:                                     [Blink]
  5
Kernel Module
Module Name Options
mlx5_core
Ethtool Options
Options
-G eth4 rx 8192 tx 8192

[Help]          [Back]          [Cancel]          [Next]
```

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

Setting the TCP low latency option disables IPV4 TCP prequeue processing, thus improving latency. It is recommended to experiment with this setting at both zero and one. In the lab testing that was performed, setting the value to one 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'
```

Next up is ensuring 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"'
```

In fast networks, TCP sequence numbers can be reused 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 better tracked.

```
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 may have negative effects in other ways.

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

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 application 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'
```

8.5 Ceph Specific Tuning

There are several places that Ceph itself has been tuned. After the following tuning sections are done, they are implemented by issuing the following commands replacing "master" with the hostname of your salt master/admin node:

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

After setting these files in place, ceph daemons need to be restarted. If combined with system level tuning, it may be desirable to apply all at once by rebooting the each node.

It is also possible to deploy these files before running stage.2 of the SUSE Enterprise Storage deployment process, and is especially desirable to do so if changing the bluestore_min_alloc_size as noted below.

Global Ceph Tuning

The global section of the configuration is tuned by modifying global.conf in the /srv/salt/ceph/configuration/files/ceph.conf.d/ directory.

Disable all unnecessary logging. Be aware, that if there is ever a need to work with support, they will likely need to re-enable some level of the logging.

```
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
```

OSD Tuning

The OSD section of the configuration is tuned by modifying osd.conf in the /srv/salt/ceph/configuration/files/ceph.conf.d/ directory.

The first set of tunings adjusts the OSD memory target. The settings below represent what was utilized in the testing, but loading more RAM in the storage nodes would allow for increasing both beyond 6GB and 4GB respectively, thus providing for additional performance benefits for some workloads.

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

The following settings have been shown to slightly improve 4k performance under mixed workload conditions. This change needs to be done before OSD deployment. If done after the fact, the OSDs will need to be redeployed for it to take effect.

```
bluestore_min_alloc_size = 4096
```

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
```

MDS Tuning

The MDS section of the configuration is tuned by modifying osd.conf in the /srv/salt/ceph/configuration/files/ceph.conf.d/ directory.

Tuning the metadata server cache allows for more metadata operations to come from RAM, resulting in improved performance. The limit utilized in testing was 16GB.

```
mds_cache_memory_limit=17179869184
```

CephFS Mount Options

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 options may be adjusted to improve performance, but it is recommended 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, thus 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.

9 Performance Results

The tests are comprised of a number of Flexible I/O (fio) job files run against multiple worker nodes. The job files and testing scripts may be found for review at: https://github.com/dm-byte/benchmaster ↗. This is a personal repository and no warranties are made in regard to the fitness and safety of the scripts found there.

The tests performed for this work significantly stressed the storage system to understand the maxium values that could be obtained for each test. A future update to this work will include latency-bounded values in addition to the maximum values.

The length of the test run, in combination with the ramp-up time specified in the job file, is intended to allow the system to overrun caches. This is a worst-case scenario for a system and would indicate that it is running at or near capacity.

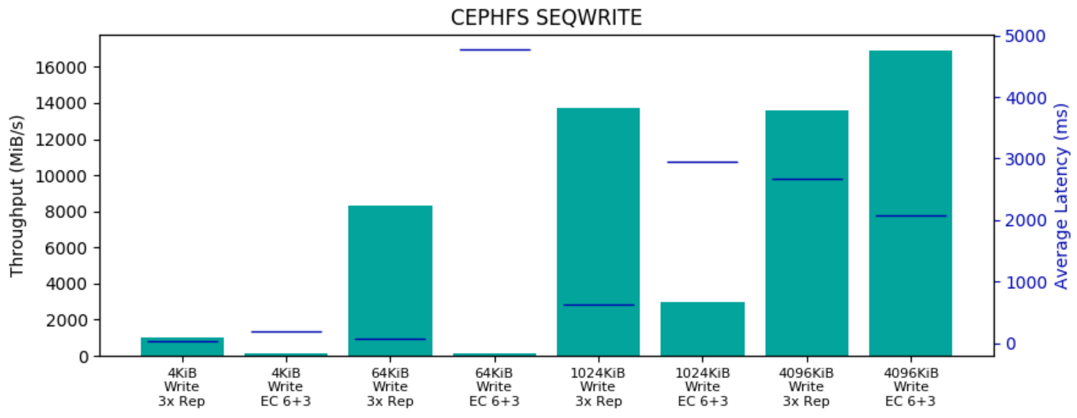
In the figures below, the x-axis labels indicate the block size in KiB on the top line and the data protection scheme on the bottom line. 3xrep is indicative of the Ceph standard 3 replica configuration for data protection while EC 6 + 3 is Erasure Coded using the ISA plugin with $k = 6$ and $m = 3$. The Erasure Coding settings were selected to fit within the cluster utilized for testing.

These settings, along with block size, max queue depth, jobs per node, and others, are all visible in the job files found at the repository link above.

Load testing was provided by an additional two nodes of the same configuration as the MDS on the same 100Gb network and 15 blade servers on a bonded 10Gb Network (aggregate 160Gb to blade chassis).

9.1 Sequential Writes

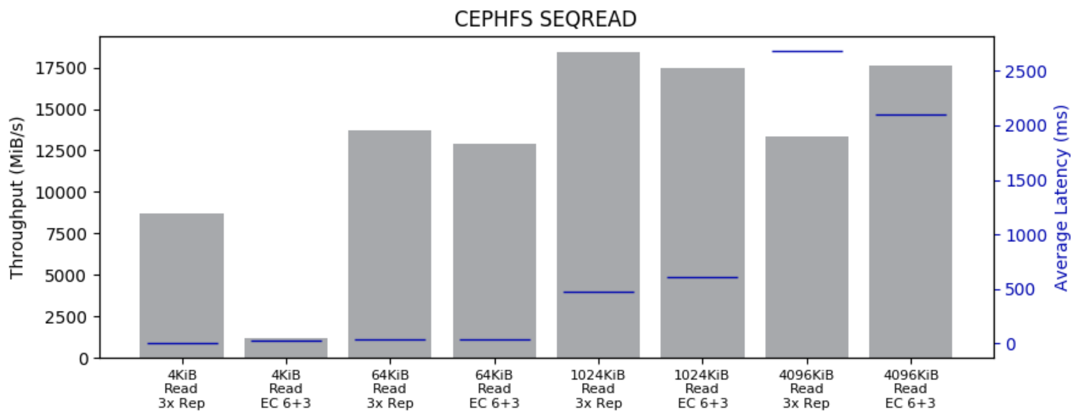
Sequential write I/O testing was performed across block sizes ranging from 4KiB to 4MiB.

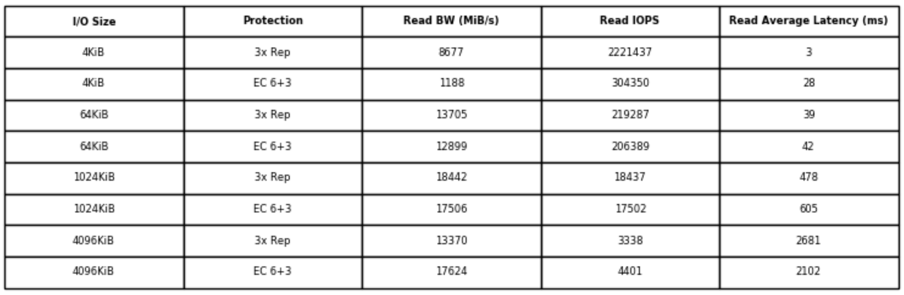


I/O Size	Protection	Write BW (MiB/s)	Write IOPS	Write Average Latency (ms)
4KiB	3x Rep	998	255550	34
4KiB	EC 6+3	169	43267	200
64KiB	3x Rep	8346	133537	65
64KiB	EC 6+3	116	1858	4768
1024KiB	3x Rep	13759	13754	636
1024KiB	EC 6+3	2960	2956	2954
4096KiB	3x Rep	13564	3386	2681
4096KiB	EC 6+3	16914	4224	2086

9.2 Sequential Reads

The sequential read tests were conducted across the same range of block sizes as the write testing.

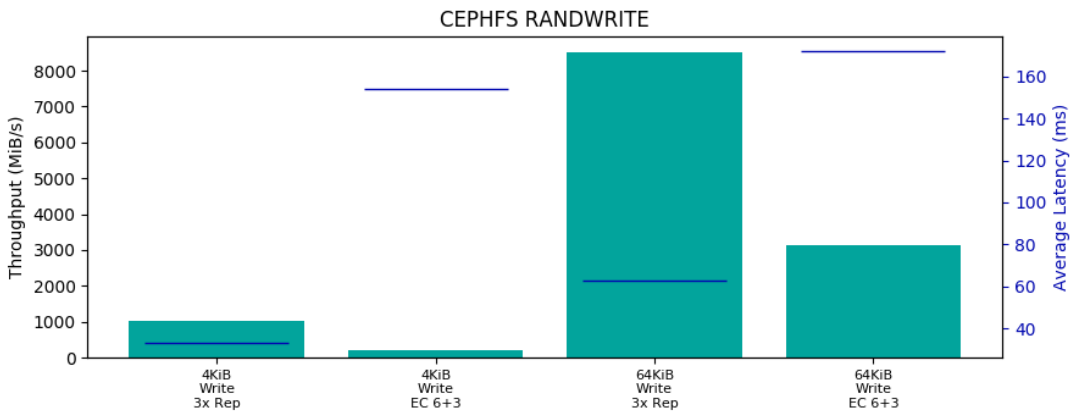




I/O Size	Protection	Read BW (MiB/s)	Read IOPS	Read Average Latency (ms)
4KiB	3x Rep	8677	2221437	3
4KiB	EC 6+3	1188	304350	28
64KiB	3x Rep	13705	219287	39
64KiB	EC 6+3	12899	206389	42
1024KiB	3x Rep	18442	18437	478
1024KiB	EC 6+3	17506	17502	605
4096KiB	3x Rep	13370	3338	2681
4096KiB	EC 6+3	17624	4401	2102

9.3 Random Writes

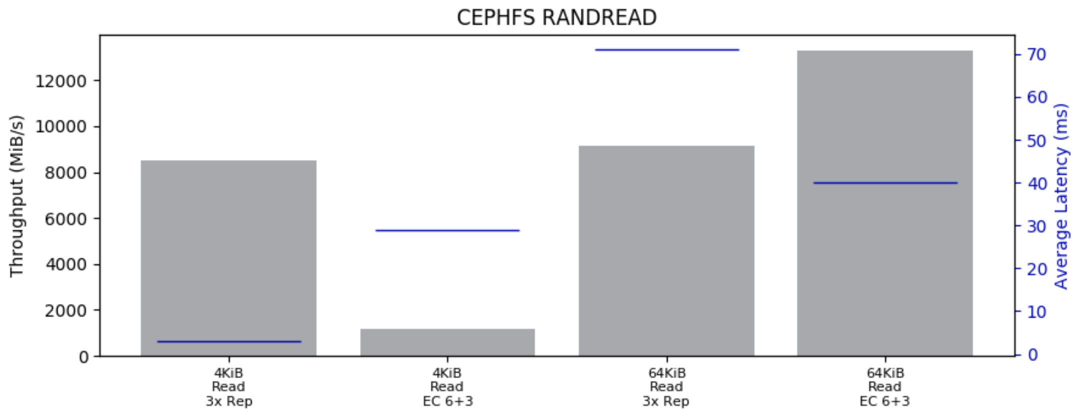
Random write tests were performed with the smaller I/O sizes of 4k and 64k.



I/O Size	Protection	Write BW (MiB/s)	Write IOPS	Write Average Latency (ms)
4KiB	3x Rep	1015	260082	33
4KiB	EC 6+3	220	56489	154
64KiB	3x Rep	8519	136306	63
64KiB	EC 6+3	3147	50362	172

9.4 Random Reads

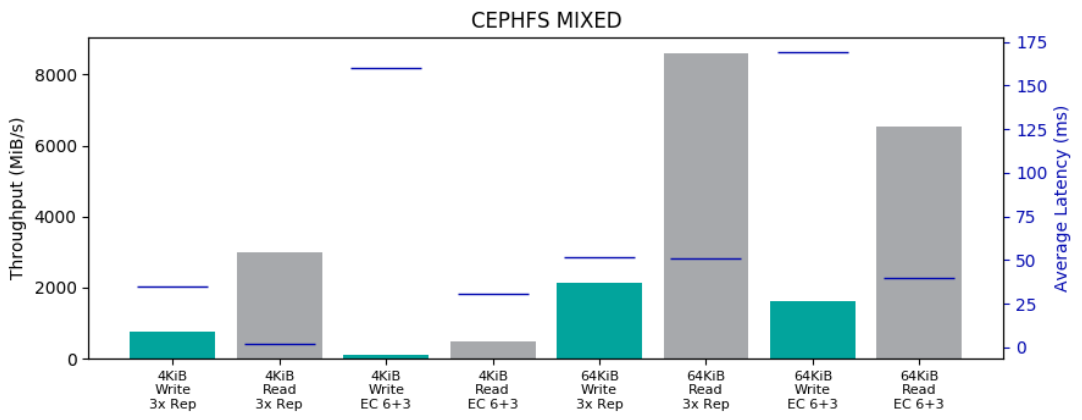
The random read tests were conducted on both 4k and 64k I/O sizes.



I/O Size	Protection	Read BW (MiB/s)	Read IOPS	Read Average Latency (ms)
4KiB	3x Rep	8508	2178222	3
4KiB	EC 6+3	1161	297458	29
64KiB	3x Rep	9127	146042	71
64KiB	EC 6+3	13298	212766	40

9.5 Mixed I/O

The mixed I/O tests were conducted on 4k and 64k I/O sizes. I/O is tested with 80% random reads and 20% random writes.



Write BW (MiB/s)	Write IOPS	Write Average Latency (ms)	Protection	Read Average Latency (ms)	Read IOPS	Read BW (MiB/s)
748	191726	35	3x Rep	2	768023	3000
118	30376	160	EC 6+3	31	121670	475
2144	34307	52	3x Rep	51	137631	8602
1633	26130	169	EC 6+3	40	104704	6544

10 Conclusion

Ceph continues to advance rapidly in the feature and performance categories and has now reached the point where it provides a viable solution for the Media and Entertainment segment of the market. To achieve the performance goals of most media and entertainment environments will require tuning and a small amount of testing of those tuning options to end up in an optimized state. However, once the tuning is performed, it is possible to have a distributed file system and object store with substantial performance capabilities.

In terms of sizing for a particular environment, these results above can be extrapolated to deliver per device performance metrics. This is done by dividing the particular measurement of IOPS or bandwidth by 120, the number of physical storage devices in this cluster. It is reasonable to use these per-device numbers to estimate performance when the system desired is larger than the system used in this work. If the desire is for higher density nodes, it is suggested that a proof of concept be run with the chosen density to determine the delivered performance metrics per device, and then utilize those numbers for scaling.

Finally, it is apparent that when the scalability of Ceph is coupled with media that provides strong performance characteristics, it becomes the ideal solution for a number of workloads that drive storage consumption with content creators.

11 Appendix A: Salt State for Kernel Tuning

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:

```
dmb_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: dmb_kern_tune
```

3. Issue the following command to set the state

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

4. Reboot the nodes

12 Appendix B: Network Tuning

```
#!/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'
```

13 Resources

SUSE Enterprise Storage Technical Overview https://www.suse.com/docrep/documents/1mdg7e-q2kz/suse_enterprise_storage_technical_overview_wp.pdf

SUSE Enterprise Storage v5 — Deployment Guide https://documentation.suse.com/ses/5.5/html/ses-all/book-storage-deployment.html

SUSE Linux Enterprise Server 12 — Administration Guide https://documentation.suse.com/ses/5.5/html/ses-all/book-storage-admin.html

Subscription Management Tool for SLES 12 https://documentation.suse.com/sles/12-SP4/html/SLES-all/book-smt.html