Journal of Physics: Conference Series

PAPER • **OPEN ACCESS**

Performance and Advanced Data Placement Techniques with Ceph's Distributed Storage System

To cite this article: M D Poat and J Lauret 2016 *J. Phys.: Conf. Ser.* **762** 012025

View the article online for updates and enhancements.

Related content

- Current Status of the Ceph Based Storage Systems at the RACF
  A. Zaytsev, H. Ito, C. Hollowell et al.
- The deployment of a large scale object store at the RAL Tier-1
  A Dewhurst, I Johnson, J Adams et al.
- ERRATUM: 2004, AJ, 127, 861
  A. Z. Bonanos, K. Z. Stanek, A. H. Szentgyorgyi et al.

Recent citations

- Achieving cost/performance balance ratio using tiered storage caching techniques: A case study with CephFS
  M D Poat and J Lauret

This content was downloaded from IP address 131.169.4.70 on 17/01/2018 at 22:37

Performance and Advanced Data Placement Techniques with Ceph's Distributed Storage System

M D Poat[1], **J Lauret** [1]
[1] Brookhaven National Laboratory, P.O. Box 5000, Upton, New York 11973-5000, USA

E-mail: mpoat@bnl.gov

Abstract. The STAR online computing environment is a demanding concentrated multi-purpose compute system with the objective to obtain maximum throughput with process concurrency. Motivation for extending the STAR online compute farm from a simple job processing tool for data taking, into a multipurpose resource equipped with a large storage system would lead any dedicated resources to become an extremely efficient and an attractive multi-purpose facility. To achieve this goal, our compute farm is using the Ceph distributed storage system which has proven to be an agile solution due to its effective POSIX interface and excelling its object storage with IO concurrency. With this we have taken our cluster one step further in terms of IO performance by investigating and leveraging new technologies and key features of Ceph. With the acquisition of a 10Gb backbone network we have ensured to eliminate the network as a limitation. With further acquisition of large fast drives (1TB SSDs) we will show how one can customize the data placement options Ceph has to offer such as primary affinity, mounting OSD journals on SSDs, and Cache Tiering along with non-Ceph related local disk caching techniques. We will discuss the latest performance results along with the expected results by using each technique. We hope this paper will serve the community's interest for the Ceph distributed storage solution.

1. Introduction

The Solenoidal Tracker at RHIC (STAR) experiment at the Relativistic Heavy Ion Collider (RHIC) located at Brookhaven National Laboratory is an international collaboration that has been collecting vast amounts of data for the past ~16 years. The online infrastructure within STAR is made up of many heterogeneous compute systems and sub-systems. In previous work which was presented at the CHEP 2015 conference [1], we have shown how we took our online compute infrastructure and transformed it from a simple job processing farm into a multipurpose resource including a CephFS POSIX storage system offering users ~80TB of redundant storage. This storage system can be used for online QA, real-time calibration or a simple local all-purpose storage system. Since its inception, STAR has strived to further extend and leverage its local resources by pushing the limitations of the Ceph storage system to a new level in terms of IO performance. We have acquired a few different models of 1TB SSD drives intended for use within the Ceph cluster. Ceph has multiple data placement techniques such as primary affinity which ensures which OSD or drive receives the first write of a replicated object, Ceph also has a Cache Tiering method which allows you to create a Ceph pool of

[1] To whom any correspondence should be addressed.

Content from this work may be used under the terms of the Creative Commons Attribution 3.0 licence. Any further distribution of this work must maintain attribution to the author(s) and the title of the work, journal citation and DOI.
Published under licence by IOP Publishing Ltd

fast drives (SSD) and overlay the pool over a pool of slow drives (HDD). Lastly, since Ceph OSDs use journals when storing data and the journals placed on standard HDDs could result in slow writes, it is possible to put the journals on SSDs with the expectation of write a performance impact.

In this study we aim to compare previous IO results with each of the Ceph data placement techniques along with the comparison of non-Ceph related techniques all with the goal to increase the IO performance of our Ceph cluster while retaining redundancy and safe data.

2. Architecture

The Ceph distributed storage system was built with a specific architecture in mind to ensure the robustness, scalability, and reliability it promises. Every component of Ceph is infinitely scalable and by design contains no single point of failure. The three main components required for Ceph to run is the Monitor service, OSD service and the Metadata Service (needed for CephFS). The storage servers run the OSD service (Object Storage Daemon) which is used to store the actual data on disks. Typically, one OSD service is run per disk or per storage unit (RAID, etc.). The monitors are used to maintain the master copy of the cluster maps, the monitors distribute the cluster maps to all of the storage nodes and clients in order to keep track of where the data is being stored. Lastly, when adding the CephFS POSIX layer to Ceph, a metadata server is required. A Ceph metadata server (MDS) is used to store the POSIX semantics of the files stored in CephFS.

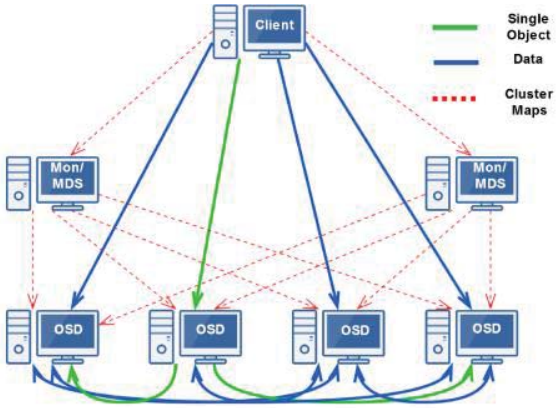

Figure 1: Architecture of the Ceph distributed storage system.

As mentioned previously, Ceph has no single point of failure. As seen in Figure 1, when clients read/write data into Ceph they contact the storage nodes directly opposed to going through a proxy or a middleman. If a storage node goes down, the data on the backend will be reshuffled to ensure consistent replication and the clients will just continue writing to the other storage nodes. Both the monitor servers which distribute the cluster maps, and the metadata servers which store the POSIX semantics, distribute their data to all storage nodes and clients. The monitor and metadata servers will work in parallel with redundancy, if one server goes down the other(s) will continue to work seamlessly.

3. Use of SSDs

SSDs have been known for their increase in single and multi-threaded IO performance. However, what we have learned is that not all SSDs perform at the same levels. We have obtained a few different models of SSD drives to give us an idea range of how a mixed set of SSDs perform compared to our 2TB Seagate ST2000NM0001 SAS HDDs in our Ceph cluster. The SSD drives obtained are all consumer grade drives; we have 4 - 1TB Crucial M550 models [2], 20 -1TB Mushkin

MKNSSDRE1TB models [3], and 4 - 480 GB Mushkin MKNSSDCR480GB-DX [4] models. While during its release, the 1TB Crucial drives cost ~$500 while the 1 TB Mushkin drives cost ~$250, both claim similar IO performance specifications. We tested the performance of each drive using multiple tests and the results were quite interesting. The performance impact of single thread writes to each drive (including the 2 TB HDD) was minimal. However, as the number of threads increase at a particular record size the performance of each drive is quite different. For example, at a 10 thread count with each thread composed of 1024k blocks the 1TB Crucial drives perform at 180 MB/s while the 1TB Mushkin drives only reach speed of 65 MB/s which is just slightly faster than the regular HDD at 55 MB/s. Lastly the 480 GB Mushkin drives hit 120 MB/s in this test. We summarize our results in Table 1. With these results in mind, it is clear that we need to be careful when trying to disentangle the IO performance results of Ceph and need to be sure we are using the correct hardware for the correct IO range. We are not expecting the IO performance within Ceph to be the same as the bare disks since the way Ceph writes to the disks creates wait time ensuring the writes have actually been written and synced properly.

Table 1. Write performance of 4 separate storage drives using Iozone. Performance per drive running 10 Iozone write threads at 1024 KB block sizes shown in MB/s.

Storage Drives	MB/s
1TB Crucial M550	180
480GB Mushkin MKNSSDCR480GB-DX	120
1TB Mushkin MKNSSDRE1TB	65
2TB Seagate ST2000NM0001 HDD	55

4. Data Placement Techniques

4.1. General

The Ceph community has an abundance of different data placement techniques and methods for storing data across specific drives, pools or nodes. In a cookie cutter stock Ceph cluster, all Ceph data pools are created with replication 3 and are spread across the storage nodes and OSDs thus distributing the data evenly across the cluster [5]. In this setup, if an OSD falls out of Ceph (due to a failing hard drive for example), Ceph will immediately rebalance the cluster to ensure consistent replication.

4.2. Primary Affinity

When a Ceph Client reads or writes data, the client always contacts the primary OSD in the acting set [6]. When writing, the client will write data to the first OSD where that OSD will then replicate the data out to two more OSDs fulfilling the replication 3 requirement. When reading, the client will only read from the primary OSDs. The primary affinity configuration has a range which can be set between 0 – 1. An OSD with the primary affinity setting of 0 means that the OSD will not be used as a primary while a setting of 1 means that it will be used as a primary. However, if you set half of your OSDs to 1 and the other half to 0.5, the first set of OSDs will be the primary 75% of the time while the others will be the primary 25% of the time.

Using the Primary Affinity setting we replaced 20 OSDS in an 80 OSD cluster composed of the 2TB Seagate SAS HDD with 20 of the 1TB Mushkin MKNSSDRE1TB SSD drives, we set the primary affinity of the SSD OSDs to 1 while all other OSDs were set to 0. Once the configuration was set we ran a series of single and multiple clients writes including single and concurrent writes per client using both RADOS object storage and the CephFS POSIX storage.

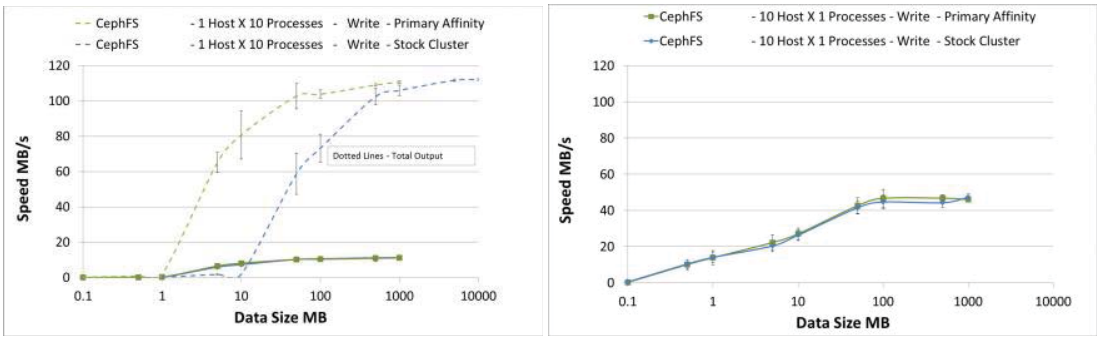

Figure 2: The left plot represents 1 client running 10 concurrent write processes into a CephFS cluster with 20 SSD OSDs with primary affinity set 1 while the other 60 HDD OSDs primary affinity is set to 0 vs. the same performance tests into a stock CephFS cluster composed of 80 HDD OSDs. The right plot is showing the performance of each client's performance when running 10 clients at 1 write process each into CephFS with primary affinity (20 SSDs & 60 HDDs) vs. a stock CephFS cluster of 80 HDDs. Both plots are in MB/s as a function of data file size. The dash curve on the left hand-side plot indicates aggregate IO per client while the solid line curves are normalized values per process.

Performance measurements were taken from tests run against both the RADOS object storage and CephFS storage using multiple tools, however all tests pertain to the CephFS approach. In Figure 2, it can be seen that the performance of CephFS using primary affinity settings with SSDs has been unchanged whether it is a single client writing or multiple clients writing. With further investigation, we found that Ceph uses "weights" for each individual OSD which determines the fill ratio. We currently are using 2 TB HDDs and 1 TB SSDs in our cluster. In our case, Ceph assigns the weight of 2 to the HDD OSDs and a weight of 1 to SSD OSDs (if, for example, we had 512 GB drives the weight would be 0.5). CephFS ensures that the cluster is filled evenly and this may actually cause Ceph to ignore the primary affinity settings by setting a priority on OSD weights and cluster fill ratio (this was seen when closely watching RADOS object store writes, although this was not a predictable behavior). Secondly, when writing into Ceph the SSD OSDs may be finished writing but still waiting for the HDD OSDs to sync for replication 3 virtually creating no performance gain. It is difficult to extrapolate the CephFS writing process as CephFS clients open many connections to many or nearly all storage nodes when writing into the cluster causing difficulty when trying to track down where the data is going. If the primary affinity setting is being respected, then the unchanged write performance would be due to the time to sync the HDDs for replication.

4.3. SSD Journals

In CephFS, when a client writes into the cluster the file is broken down into objects (chunks) and is written to the OSDs [7]. When Ceph writes to an OSD, there are two sequential IO operations, first writing to the journal and second writing to the filesystem itself. The journals will not release the data to the OSDs filesystem until it has acknowledged the write to itself; this can cause a slowdown in overall write performance. If there is heavy IO into the Ceph cluster, many journal operations would occur (Ceph writes small random IO into the journals). With this in mind, we created 3 OSDs per node each composed of the 2TB Seagate SAS HDD, and set each of the journals of the OSDs to run on 1

Mushkin MKNSSDRE1TB SDD. A 3OSD:1SSD per node configuration in hope for a write performance gain.

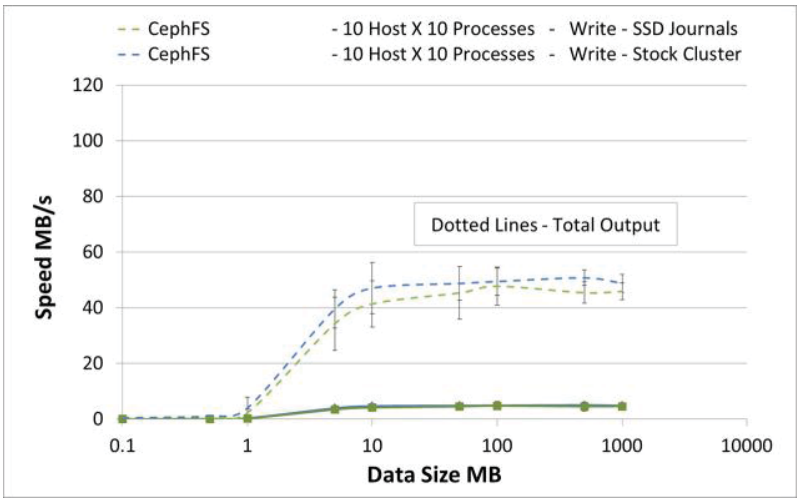

Figure 3: The plot represents 10 clients running 10 concurrent write processes into CephFS with the journal operations on SSDs vs. a stock CephFS cluster. The plot is showing the performance in MB/s as a function of data file size. The dash curves are representing the aggregate write speed per client of all 10 processes, the solid line curves are the speed per client of each write process.

When accounting for the error bar it can be seen that there is no write performance gain with this configuration. With further investigating, we found that when Ceph performs journal operations it uses the write options D_SYNC and O_DIRECT (D_SYNC does not allow any other writes until the initial write is finished and writing with O_DIRECT skips using the kernel page cache). In the test shown in Figure 3, 10 clients running 10 write processes into Ceph (100 concurrent writes) may not actually be enough concurrent writes to see a performance impact. Due to the small size of our cluster, when writing an excess of 100 concurrent writes, the Ceph monitors report blocked requests and heavy IO wait.

4.4. Cache Tiering

Cache Tiering involves creating a Ceph storage pool composed of fast drives (SSDs) that is overlaid on top of a backing storage pool composed of slower drives (cheap HDD) [8]. With this configuration the administrator can set a specific write back policy on how and when to flush data to the backing pool. The cache flush can be based on fill ratio, data popularity, cache age, etc. but the data can be primarily stored on the SSD pool and only flush to the HDD when needed. Unfortunately, at this time it does not appear that Cache Tiering is compatible (not implemented) within CephFS as Ceph is restricting the storage pools to be part of both a Cache Tier and CephFS simultaneously. With this restriction taken into account, we still wanted to simulate a Cache Tier by creating a CephFS storage system with a small SSD only pool and compare it to a small HDD only pool. The assumption being that in the deal Cache Tiering scenario, the IO would be directed to the SSD first in this case.

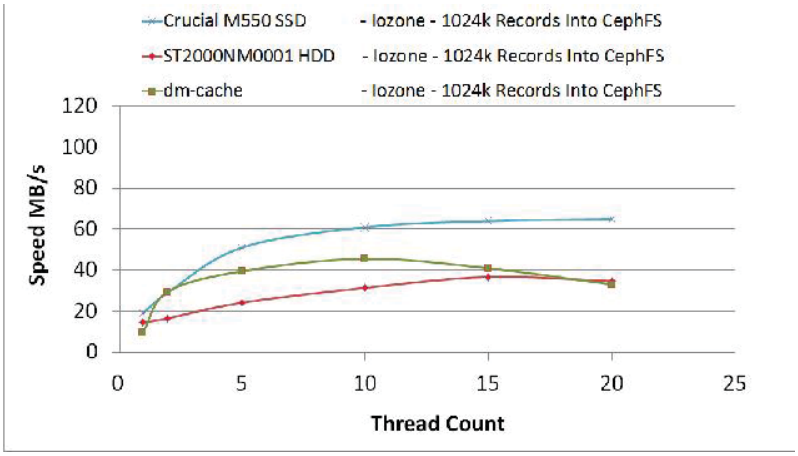

Figure 4: The plot represents one client using Iozone to test the write performance of CephFS composed of a 4 Crucial M550 SSD pool vs. CephFS composed of a 4 Seagate ST2000NM0001 HDD pool. We have further compared the performance with a CephFS composed of a dm-cache pool. The plot is shown in MB/s as a function of increased thread containing 1024k block sizes.

As it can be seen in Figure 4, there is indeed an expectation of a write performance gain for an SSD only CephFS (blue curve) as the thread count increases compared to the HDD only CephFS (red curve). This is partly due to using the Crucial M550 drives which have a better IO performance over the Mushkin MKNSSDRE1TB which have been used in many of the tests. The performance impact is present and this data is useful as we now know Ceph performs better when all the writes are written to faster performing drives.

5. dm-cache

We now know the use of SSDs can impact Ceph IO performance along with the understanding of the expected performance range with different SSD drives. The remaining question is how could we emulate Cache Tiering within a CephFS deployment without this feature being natively supported? One could achieve the same effect by using an external caching mechanism such as dm-cache [10], acting at a lower hardware level. While similar to a Ceph cache tier, dm-cache is local to each node and uses SSDs to act as a cache in front of HDDs, presenting to the higher level IO component (CephFS in this case) a logical device. In other words, this would circumvent the primary affinity and OSD weight mechanisms as Ceph would have no idea of the presence of SSDs. By using dm-cache, we could ensure the writes go to the SSDs first but note one downside: if we overlay one SSD over 3 HDD and create 3 OSDs on one machine and the SSD fails, we would lose all OSDs attached to that single machine from the Ceph cluster. In Figure 4, we are showing the performance of a ‘dm-cache only CephFS’ (green curve) compared to the ‘SSD only CephFS’ (blue curve) or ‘HDD only CephFS’ (red curve). The dm-cache policy in this case was set to a high write-promote setting which will allow streaming IO to be written to the SSD and immediately flush back to the HDD. While the performance of the ‘dm-cache only CephFS’ does show a performance gain over the ‘HDD only CephFS” setup, the performance is not at the same level as the ‘SSD only CephFS’, this is due to the write-back from the SSD to the HDDs. Note that at a high thread count (15 - 20 threads) the performance of the ‘dm-cache CephFS’ begins to degrade and is similar to the ‘HDD only CephFS’.

6. Conclusion

In this paper, we have tested multiple Ceph data placement techniques with the objective to increase

the IO write performance of our current Ceph cluster with the use of SSDs. While the data placement techniques appear intuitive and our expectations were high, we found that not all of these techniques align with the expected results. The primary affinity configuration did seem promising at first, however the OSD weights for balancing and the syncing with the HDD may be the cause of no performance impact. The OSD journals on SSD drives may only create a performance impact with better performing SSDs or only with an extremely large number of threads where our small Ceph cluster can't even keep up with. Cache Tiering was found to show a performance impact, but does not work with CephFS. Fortunately, we were able to revert to and prove that a local caching technique, such as using dm-cache, increases the IO performance of our system within caveats. In our preliminary testing, dm-cache does show positive performance impact, but may be risky if the SSDs were to fail. A less risky solution may be a 1:1 ratio of SSD:HDD but this would create a cost impact to STAR we were are trying to avoid.

Acknowledgements

This work was supported by the Office of Nuclear Physics within the U.S. Department of Energy's Office of Science.

References

[1] Poat M D, Lauret J and Betts W 2015 *POSIX and Object Distributed Storage systems – Performance Comparison Studies With Real-Life Scenarios in an Experimental Data Taking Context Leveraging OpenStack Swift & Ceph* J. Phys.: Conf. Ser 664 042031
[2] Crucial M550 SSD Storage http://www.crucial.com/usa/en/storage-ssd-m550
[3] Mushkin MKNSSDRE1TB SSD Storage http://www.poweredbymushkin.com/index.php/catalog/item/44-reactor/1076-reactor-1tb-7mm.html
[4] Mushkin MKNSSDCR480GB-DX SSD Storage http://www.poweredbymushkin.com/index.php/component/djcatalog2/item?id=665:7mm-chronos-deluxe-480gb&cid=9:chronos-deluxe
[5] Singh K 2015 *Learning Ceph* (Birmingham, UK)
[6] Ceph Primary Affinity http://docs.ceph.com/docs/master/rados/operations/crush-map/#primary-affinity
[7] Ceph OSD Journals on SSD http://docs.ceph.com/docs/master/start/hardware-recommendations/#solid-state-drives
[8] Ceph Cache Tiering http://docs.ceph.com/docs/master/rados/operations/cache-tiering/
[9] Iozone http://www.iozone.org/
[10] dm-cache https://en.wikipedia.org/wiki/Dm-cache