# Performance and Scalability Evaluation of the Ceph Parallel File System

Feiyi Wang[1], Mark Nelson[2], Sarp Oral[1], Scott Atchley[1], Sage Weil[2], Bradley W. Settlemyer[1], Blake Caldwell[1], and Jason Hill[1]

[1]Oak Ridge National Laboratory, Oak Ridge, Tennessee 37831
[2]Inktank Inc., Los Angeles, CA 90017

## ABSTRACT

Ceph is an emerging open-source parallel distributed file and storage system. By design, Ceph leverages unreliable commodity storage and network hardware, and provides reliability and fault-tolerance via controlled object placement and data replication. This paper presents our file and block I/O performance and scalability evaluation of Ceph for scientific high-performance computing (HPC) environments. Our work makes two unique contributions. First, our evaluation is performed under a realistic setup for a large-scale capability HPC environment using a commercial high-end storage system. Second, our path of investigation, tuning efforts, and findings made direct contributions to Ceph's development and improved code quality, scalability, and performance. These changes should benefit both Ceph and the HPC community at large.

## 1. INTRODUCTION

Oak Ridge Leadership Computing Facility (OLCF)[1] at Oak Ridge National Laboratory (ORNL) has a long history of deploying and running very-large-scale high-performance computing (HPC) systems. In order to satisfy the I/O demand of such supercomputers, OLCF also hosts large-scale file and storage systems. Lustre has been OLCF's choice as the distributed parallel file system for scratch I/O. However, we recognize that there are a wide variety of scientific workloads with different performance and data access requirements (e.g., RESTful interface, S3-like API, cloud solution integration) which might not be efficiently serviced by Lustre.

This paper presents our evaluation and findings on block I/O and file system performance of Ceph for scientific HPC environments. Through systematic experiments and tuning efforts, we observed that Ceph can perform close to 70% of raw hardware bandwidth at object level and about 62% of at file system level. We also identified that Ceph's metadata plus data journaling design is currently a weak spot as far as our particular storage infrastructure is concerned.

This rest of the paper is organized as follows. Section 2 provides an overview to Ceph and its architectural components; Section 3 gives an overview on our general test and evaluation methodology as well as testbed environment; Following it, Section 4 establishes the baseline performance and expectations for all critical components on the data path; Section 5 discusses our early runs, results, and issues bottom up: from middle tier RADOS object layer to the file system-level performance. In Section 6 we highlight the investigative and tuning effort, we compare results before and after, and how the process eventually bring the system performance to a respectable state. Finally, Section 7 summarizes our findings, observations, and future works.

## 2. BACKGROUND

Ceph [5] is a distributed storage system designed for scalability, reliability, and performance. The system is based on a distributed object storage service called RADOS (reliable autonomic distributed object store) that manages the distribution, replication, and migration of objects. On top of that reliable storage abstraction Ceph builds a range of services, including a block storage abstraction (RBD, or Rados Block Device) and a cache-coherent distributed file system (CephFS).

Data objects are distributed across Object Storage Devices (OSD), which refers to either physical or logical storage units, using CRUSH [6], a deterministic hashing function that allows administrators to define flexible placement policies over a hierarchical cluster structure (e.g., disks, hosts, racks, rows, datacenters). The location of objects can be calculated based on the object identifier and cluster layout (similar to consistent hashing [2]), thus there is no need for a metadata index or server for the RADOS object store. A small cluster of monitors (ceph-mon daemons) use Paxos to provide consensus on the current cluster layout, but do not need to explicitly coordinate migration or recovery activities.

CephFS builds a distributed cache-coherent file system on top of the object storage service provided by RADOS. Files are striped across replicated storage objects, while a separate cluster of metadata servers (ceph-mds daemons) manage the file system namespace and coordinate client access to files.

Ceph metadata servers store all metadata in RADOS objects, which provides a shared, highly-available, and reliable storage backend. Unlike many other distributed file system

[1]This research was supported by, and used the resources of, the Oak Ridge Leadership Computing Facility, located in the National Center for Computational Sciences at ORNL, which is managed by UT Battelle, LLC for the U.S. DOE (under the contract No. DE-AC05-00OR22725).


*PDSW'13* November 18, 2013, Denver, CO, USA

http://dx.doi.org/10.1145/2538542.2538562.

architectures, Ceph also embeds inodes inside directories in the common case, allowing entire directories to read from RADOS into the metadata server cache or prefetched into the client cache using a single request.

Client hosts that mount the file system communicate with metadata servers to traverse the namespace and perform file I/O by reading and writing directly to RADOS objects that contain the file data. The metadata server cluster periodically adjusts the distribution of the namespace across the MDS cluster by migrating responsibility for arbitrary subtrees of the hierarchy between a dynamic pool of active ceph-mds daemons. This dynamic subtree partitioning [7] strategy is both adaptive and highly scalable, allowing additional metadata server daemons to be added or removed at any time, making it ideally suited both for large-scale workloads with bursty workloads or general purpose clusters whose workloads grow or contract over time.

While we are interested in Ceph for its ability to support alternative workloads that are not easily accomplished with Lustre, in this study we are investigating the use of Ceph for future large-scale scientific HPC storage deployments.

# 3. TESTBED ENVIRONMENT AND TEST METHODOLOGY

We used a Data Direct Networks (DDN) SFA10K as the storage backend for this evaluation. SFA10K organizes disks into various RAID levels by two active-active RAID controllers. In our testbed, exported RAID groups (LUNs) are driven by four server hosts (oss-[1-4]). Each server host has two InfiniBand (IB) QDR connections (supported by a dual-port Mellanox ConnectX IB card) to SFA10K. By our calculation, this setup is adequate to drive the SFA10K at its maximum theoretical throughput (roughly 12 GB/s).

Our SFA10K system consists of 200 SAS drives and 280 SATA drives, hosted in a 10-tray configuration. The SATA and SAS disks in our testbed are distributed evenly over the SFA10K. Each SFA10K controller has two RAID processors and each RAID processor hosts a dual-port IB QDR card. The disks are organized into RAID groups by the RAID processors and then exported to connected server hosts.

All hosts (client and servers) were configured with Redhat 6.3 and kernel version 3.5.1 initially, and later upgraded to 3.9, Glibc 2.12 with syncfs support, locally patched. We used the Ceph 0.48 and 0.55 release in the initial tests, upgraded to 0.64 and then to 0.67RC for a final round of tests.

Our testing methodology is bottom to top. We start our evaluation from the block-level to establish a baseline performance and work our way up to the file system-level. We establish an expected theoretical performance of a given hardware or software layer and then compare that with the observed results of our experiments. Each added new layer to the stack introduces new bottlenecks and re-defines the expected theoretical performance of the overall system.

# 4. ESTABLISHING A PERFORMANCE BASELINE

## 4.1 Block I/O over Native IB

The lowest layer in the system are the exposed LUNs to the server hosts from the RAID controllers. These LUNs are configured as a RAID 6 8+2 array (8 data disks and 2 $P$ and $Q$ disks).

In prior tests we observed that a 7.2K RPM SAS disk can perform at 140 MB/s for 128 kB sequential I/O requests and a 7.2K RPM SATA disk can do 36 MB/s. For these tests, all disk caches were turned off. Therefore, in 8 data disk RAID group with 1 MB sequential I/O requests (each disks sees 128 kB chunks of the request) the aggregate performance is about 1.12 GB/s for a SAS RAID group and 288 MB/s for a SATA RAID group, if there are no caches along the I/O path. With the caches turned on, especially the write-back cache on the DDN RAID controller, we observe a significant performance improvement.

In our day-to-day HPC operations, we run our storage systems with write-back cache turned on on RAID controllers, if and only if, there is a method of cache-mirroring between the two active-active RAID controllers. With the write-back cache on we observe ˜1.4 GB/s per RAID 6 8+2 SAS group and ˜955 MB/s per RAID 6 8+2 SATA group, for 1 MB sequential I/O. These two performance numbers will be used as a baseline in our study and they are presented in Table 1. For these tests, the 8+2 RAID 6 groups are exported to the server hosts using SCSI RDMA Protocol (SRP) over IB protocol (which is the default for our normal HPC configurations). For a SATA disk group this translates into roughly 120 MB/s/disk and for a SAS disk group it is 175 MB/s/disk.

It is worth mentioning that we used our in-house developed synthetic benchmark, *fair-lio* for all our block-level testing. Fair-lio has better sequential I/O characteristics compared to commonly used XDD benchmark and it is asynchronous and based on the Linux `libaio` library.

Table 1: Baseline block I/O performance summary

| | |
|---|---|
| SAS single LUN sequential read | 1.4 GB/s |
| SATA single LUN sequential read | 955 MB/s |
| Single host with four SAS LUNs | 2.8 GB/s |
| Single host with seven SATA LUNs | 2.6 GB/s |

When exercising all 28 SATA LUNs from all four server hosts in parallel, we observed an 11 GB/s aggregate performance. The same level of performance was observed using all 20 SAS LUNs from four server hosts in parallel. Comparing these to the advertised peak performance of 12 GB/s of the DDN SFA10K, we concluded that our test setup is well configured to drive the backend disk system at close to full performance and we are limited by the RAID controller performance. Going forward in this study, 11 GB/s will be used the peak baseline for the entire test configuration.

## 4.2 Establishing an IP over IB Baseline Performance

Ceph uses the BSD Sockets interface in `SOCK_STREAM` mode (i.e., TCP). Because our entire testbed is built on IB QDR fabric (including client to server host connections over a 108-port Mellanox IB QDR switch), we used IP over IB (IPoIB) for networking[2]. Through simple `netperf` tests, we observed that a pair of hosts connected by IB QDR using IPoIB can transfer data at 2.7 GB/s (the hosts are tuned per recommendations from Mellanox). With all four server hosts (OSD servers), we expect the aggregate throughput to be in the neighborhood of 11 GB/s.

[2]As of writing of this paper, Inktank is investigating using rsockets to improve performance with IB fabric

# 5. CEPH EVALUATION: INITIAL RESULTS AND TUNINGS

## 5.1 Ceph RADOS Scaling

We used the open-source RADOS Bench tool, developed by Inktank, to perform our initial performance analysis of the underlying RADOS layer. RADOS Bench simply writes out objects to the underlying object storage as fast as possible, and then later reads those objects in the same order as they were written.

### 5.1.1 *Disable TCP autotuning*

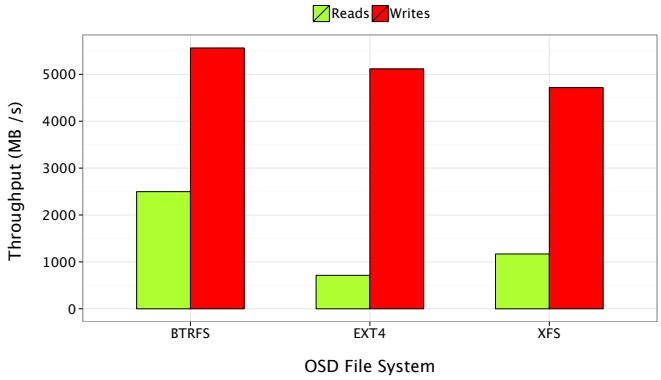



Figure 1: Evaluating RADOS bench **with** TCP auto tuning, 4 servers, 4 clients, 4 MB I/O

Our early testing indicated a read performace abberation. There were periods of high performance followed by periods of low performance or outright stalls that could last for up to 20 seconds at a time. RADOS bench recorded poor read performance across different backend file systems, as shown in Figure 1.[3]

We observed that outstanding operations on the clients were not being shown as outstanding on the OSDs. This appeared to be similar to a problem Jim Schutt at Sandia National labs encountered with TCP autotuning in the Linux kernel [3]. TCP auto tuning enables TCP window scaling by default and automatically adjusts the TCP receive window for each connection based on link conditions such as bandwidth delay product.

We disabled this behavior by setting `/proc/sys/net/ipv4/tcp_moderate_rcvbuf` to 0 on all nodes, improving the Ceph read performance as shown in Figure 2. Recent versions of Ceph work around this issue by manually controlling the TCP buffer size.

### 5.1.2 *Scaling OSDs per server*

We also observed that using two or more client processes and many concurrent operations are critical to achieve high performance. We tested eight client processes with 32 concurrent 4 MB objects in flight each. We created a pool for each RADOS Bench process to ensure that object reads come

[3]Although we conducted this particular experiment with three different flavors of local file system, with BTRFS the best performer, Inktank however doesn't think BTRFS is production-ready yet. So when not explicitly stated, the follow-on experiments use XFS by default.

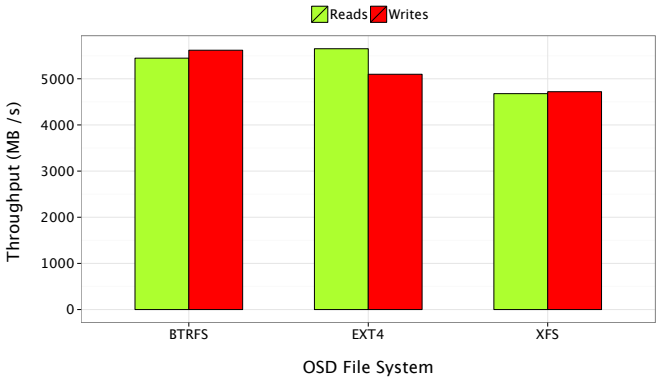



Figure 2: Evaluating RADOS bench with TCP auto tuning **disabled**, 4 servers, 4 clients, 4 MB I/O

from independent pools (RADOS Bench is not smart enough to ensure that objects are not read by multiple processes and thus possibly cached). A sync and flush is performed on every node before every test to ensure no data or metadata is in cache. All tests were run with replication set to one. Figure 3 shows that an OSD server exhibits near linear scalability up to nine OSDs, and is still trending upwards at eleven OSDs. This suggests that we have not reached the saturation point yet. Additional testing would require provisioning more OSDs on the SFA10K backend.

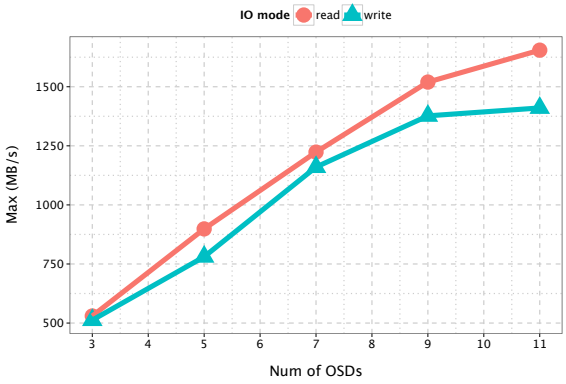



Figure 3: RADOS scaling on number of OSDs: single server single client

### 5.1.3 *Scaling OSD servers*

We measured performance at the client (using RADOS bench) for one to four OSD servers. Each additional OSD server adds eleven more OSDs into play. Based on Figure 3, the perfect scaling projects the aggregated read with 4 OSSs should be at 6616 MB/s and write at 5640 MB/s. Figure 4 demonstrates that both read and write *scales* with increasing number of OSSs, but at a loss of 13.6% and 16.0% respectively. We will discuss the improvement to this result through parameter probing.

## 5.2 Ceph File System Performance

We used the synthetic IOR benchmark suite [1] for file system level performance and scalability test. Each client

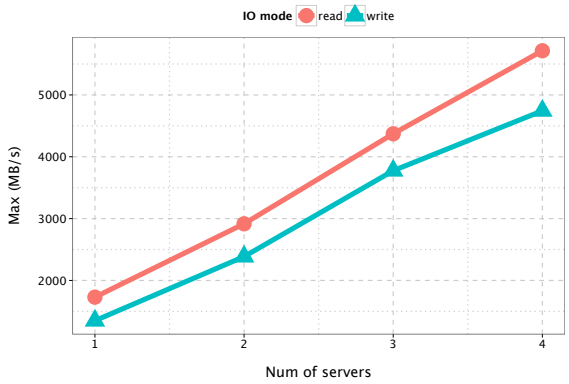



Figure 4: RADOS scaling on number of servers, with 8 clients


(a) 4 KB transfer size (b) 4 MB transfer size

Figure 5: CephFS scalability test with IOR

node has 6 GB of physical memory, the block size is set so as to mitigate cache effects. In addition, the test harness program issues commands at the beginning of each test to free pagecache, dentries and inodes.

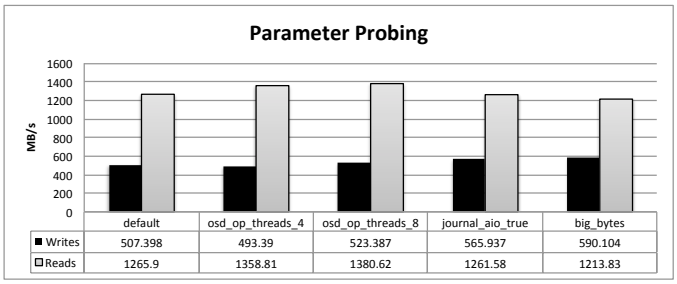
The full permutation of IOR parameters sweep were not explored due to I/O errors. We were able to record results in the I/O size boundary cases: 4 KB and 4 MB, using a fixed number of OSD servers (4) and fixed block size (8 GB), the results are illustrated in Figure 5a and 5b, we make the following observations:

- The small read (4 KB transfer size) performance is almost an anomaly
- The large read (4 MB transfer size) performance is almost half of the RADOS read performance.
- The write performance is also about half of what we can obtain from RADOS Bench. When number of clients reaches 8, there is a significant performance drop, which is a poor sign for scalability.

The bottom line is, though we have obtained decent performance number at RADOS level, but it did not translate into file system-level performance. We will investigate why it is so low compare to write performance and present improved results in Section 6.

# 6. FURTHER OPTIMIZATION AND IMPROVEMENTS

## 6.1 Improving RADOS Performance




| | default | osd_op_threads_4 | osd_op_threads_8 | journal_aio_true | big_bytes |
|---|---|---|---|---|---|
| Writes | 507.398 | 493.39 | 523.387 | 565.937 | 590.104 |
| Reads | 1265.9 | 1358.81 | 1380.62 | 1261.58 | 1213.83 |

Figure 6: Evaluating parameter impact through sweeping test

After the initial test results, we tried various combinations of tweaks including: changing the number of filestore op threads, putting all of the journals on the same disks as the data, doubling the OSD count, and upgrading Ceph to a development version which reduces the seek overhead caused by `pginfo` and `pglog` updates on XFS (these enhancements are now included as of the Ceph Cuttlefish release, v0.61).

We swept over Ceph configuration parameters to examine how different tuning options affect performance on the DDN platform. The result of this parameter probing is illustrated in Figure 6. Please refer to Appendix E of [4] for explanations of these probed parameters. In this figure, the default is where we started with. Reads benefit from increasing the `osd_op_threads` (4 and 8), with 7.3% and 9% improvement respectively. For writes, the substantial boost comes with `journal_aio` true and `big_bytes`, with 11.5% and 16.3% improvement respectively. Other probed parameters (not shown on this figure) doesn't bear tangible impacts.

## 6.2 Improving Ceph File System Performance

The initial stability issues mentioned in Section 5.2 are fixed by migrating from Ceph version 0.48/0.55 to 0.64, the latest stable version at the time of writing this report. Upgrading to the latest stable Ceph release allowed us to run a full IOR parameter sweep for the first time since we started evaluating the performance and scalability of the Ceph file system. This is another sign of how much Ceph development is currently in flux.

Even with these changes in place, less-than-ideal write performance and very poor read performances were observed during our tests. We also observed that by increasing the number of IOR processes per client node, the read performance degraded even further indicating some kind of contention either on the clients or on the OSD servers.

### 6.2.1 *Disabling Client CRC32*

At this point, we were able to both make more client nodes available for Ceph file system-level testing and also install a profiling tool called `perf` that is extremely useful for profiling both kernel and user space codes. Profiling with `perf` showed high CPU utilization on test clients due to CRC32c processing in the Ceph kernel client.

With client CRC32 disabled, we repeated the IOR tests. New results are shown in in Figure 7.

We observed that IOR write throughput increased dramatically and is now very close and comparable to the RADOS

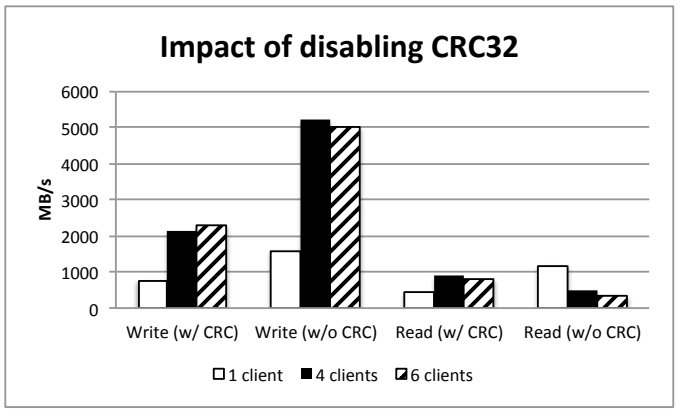



Figure 7: Impact of Disabling CRC32

Bench performance. Read performance continued to be poor and continued to scale inversely with the increasing number of client processes. Please note that, since these tests were performed, Inktank has implemented SSE4-based CRC32 code for Intel CPUs. While any kernel based CRC32 processing should have already been using SSE4 instructions on Intel CPUs, this update will allow any user-land Ceph processes to process CRC32 checksums with significantly less overhead.

### 6.2.2 *Improving IOR Read Performance*

Deeper analysis with `perf` showed that there was heavy lock contention during parallel compaction in the Linux kernel memory manager. This behavior was first observed roughly in the kernel 3.5 time frame which was the kernel installed on our test systems.[4] We upgraded our test systems with kernel version 3.9 and performed RADOS Bench test. The results were dramatic: read performance improved from 4678 MB/s to 7499 MB/s (a 60% improvement). Write performance was also fractionally improved from 4720 MB/s to 5284 MB/s (a 12% improvement).

Additionally, we increased the size of the CephFS kernel client's read-ahead cache.

By installing a newer kernel, increasing read-ahead cache size, and increasing the number of client IOR processes, we were able to achieve very satisfactory I/O performance at the Ceph file system-level.

### 6.2.3 *Repeating the IOR Scaling Test*

As before, we ran IOR scaling tests with two cases: transfer size 4 KB and 4 MB. These results are illustrated in Figure 8, respectively. As expected, we saw both improved read and write performance. These new read and write performance are in line with observed RADOS bench performance.

Writes seem to be topping out at around 5.2 GB/s (which is roughly what we would expect due to journaling effectively halving the write performance). Reads seem to be topping out anywhere from 5.6-7 GB/s, however it is unclear if read performance would continue scaling with more clients and get closer to 8 GB/s we observed with RADOS Bench.

## 7. OBSERVATIONS AND CONCLUSIONS

[4]For more information, please refer to http://lwn.net/Articles/517082/ and https://patchwork.kernel.org/patch/1338691/.


Figure 8: IOR Scaling Test: 4 KB and 4 MB transfer size

In our early tests, we experienced large performance swings during different runs, low read performance when transfer size is small, and I/O errors tend to happen when system is under stress (more clients and large transfer sizes). However, with systematic performance engineering and development efforts, we have seen a steady improvement through different releases. As of now, the Ceph system on our testbed is able to perform close to 70% of raw hardware capability at RADOS level and close to 62% at file system level (after accounting for journaling overheads) .

Ceph is built on the assumption that the underlying hardware components are unreliable, with little or no redundancy and failure detection capability. This assumption is not valid for high-end HPC centers like OLCF. Ceph performs **metadata + data** journaling, which is appropriate for host systems that have locally attached disks rather than DDN SFA10K-like hardware, where the backend LUNs are exposed as block devices through IB over SRP protocol. The journaling write requires twice the bandwidth compared to Lustre-like meta data-only journaling mechanism. For Ceph to be viable in large-scale capability HPC environments like OLCF, journaling operations need to support HPC storage hardware.

## Acknowledgments

The authors would like to thank Galen Shipman of ORNL for initiating the evaluation effort, and Greg Farnum of Inktank for providing early support.

# 8. REFERENCES


[1] IOR HPC benchmark. https://github.com/chaos/ior.

[2] D. Karger, E. Lehman, T. Leighton, R. Panigrahy, M. Levine, and D. Lewin. Consistent hashing and random trees: Distributed caching protocols for relieving hot spots on the world wide web. In *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*, pages 654–663. ACM, 1997.

[3] J. Schutt. Understanding delays due to throttling under very heavy write load. http://marc.info/?l=ceph-devel&m=133009796706284&w=2, 2012.

[4] F. Wang, M. Nelson, S. Oral, D. Fuller, S. Atchley, B. Caldwell, B. Settlemyer, J. Hill, and S. Weil. Ceph parallel file system evaluation report. Technical Report ORNL/TM-2013/151, Oak Ridge National Laboratory, 2013.

[5] S. A. Weil, S. A. Brandt, E. L. Miller, D. D. E. Long, and C. Maltzahn. Ceph: a scalable, high-performance distributed file system. In *Proceedings of the 7th symposium on Operating systems design and implementation*, OSDI '06, pages 307–320, Berkeley, CA, USA, 2006. USENIX Association.

[6] S. A. Weil, S. A. Brandt, E. L. Miller, and C. Maltzahn. Crush: controlled, scalable, decentralized placement of replicated data. In *Proceedings of the 2006 ACM/IEEE conference on Supercomputing*, SC '06, New York, NY, USA, 2006. ACM.

[7] S. A. Weil, K. T. Pollack, S. A. Brandt, and E. L. Miller. Dynamic metadata management for petabyte-scale file systems. In *Proceedings of the 2004 ACM/IEEE conference on Supercomputing*, page 4. IEEE Computer Society, 2004.