Evaluating the Performance and Scalability of the Ceph Distributed Storage System

Diana Gudu
Karlsruhe Institute of Technology
Karlsruhe, Germany
Email: diana.gudu@kit.edu

Marcus Hardt
Karlsruhe Institute of Technology
Karlsruhe, Germany
Email: marcus.hardt@kit.edu

Achim Streit
Karlsruhe Institute of Technology
Karlsruhe, Germany
Email: achim.streit@kit.edu

***Abstract*—As the data needs in every field continue to grow, storage systems have to grow and therefore need to adapt to the increasing demands of performance, reliability and fault tolerance. This also increases their complexity and costs. Improving the performance and scalability of storage systems while maintaining low costs is thus crucial. The evaluated open source storage system Ceph promises to reliably store data distributed across many nodes. Ceph is targeted at commodity hardware. This study investigates how Ceph performs in different setups and compares this with the theoretical maximum performance of the hardware. We used a bottom-up approach to benchmark Ceph at different architectural levels. We varied the amount of storage nodes and clients to test the scalability of the system. Our experiments revealed that Ceph delivers the promised scalability, and uncovered several points with improvement potential. We observed a significant increase of the write throughput by moving the Ceph journal to a faster location (in memory). Moreover, while the system scaled with the increasing number of clients operating the cluster, we noticed a slight performance degradation after the saturation point. We tested two optimisation strategies—increasing the available RAM or the object size—and noted a write throughput increase of up to 9% and 27%, respectively. Our findings improve the understanding of Ceph and should benefit future users through the presented strategies for tackling various performance limitations.**

I. Introduction

The advent of Big Data led many scientific communities to shift their focus towards this field, causing their needs for efficient data storage and management to constantly increase. Therefore, there is a growing interest in developing and evaluating new storage solutions. In this paper, we evaluate the performance of Ceph [1], an emerging distributed storage system designed for scalability, performance, cost efficiency and reliability. The larger context of this evaluation is supporting the needs of such a scientific community—the Human Brain Project (HBP) [2]—in which KIT will provide a Ceph-based object store. Use cases include: data publication/archival, content delivery network and Neuroinformatics data analytics.

The complexity of the distributed approach calls for careful configuration and tuning to obtain the maximum performance for given hardware and use cases. A comprehensive performance evaluation of the system is thus essential. In this paper we formulate and answer questions that will help fulfil our use cases, and at the same time improve our understanding of Ceph and distributed storage systems. Under which circumstances does the system work best? What are the limiting factors to its performance? How does the system behave when adding more storage units or more clients using the system? Does the performance degrade with too many clients (i.e. under extreme stress)? Ceph's layered architecture adds complexity to the system—what is the performance loss at each layer?

This paper is organised around these questions, as follows. Section II describes the architecture and main features of Ceph, as well as other storage technologies. In Section III we present related work that evaluates Ceph and other storage solutions. Section IV gives an overview of our testing environment, the hypotheses we set out to test and the benchmarking procedure. In Section V we discuss our performance results. Finally, in Section VI we summarise our work and findings.

II. Storage Solutions

Ceph is an open-source distributed storage system. Its core, RADOS, is a fully distributed, reliable and autonomous object store. Ceph's building blocks are called OSDs—Object Storage Daemons. OSDs are responsible for storing objects on local filesystems, as well as working together to replicate data, detect and recover from failures, or migrate data when OSDs join or leave the cluster. Ceph's design is ultimately rooted in the premise that failures are common in large-scale storage systems. Thus, Ceph aims to ensure reliability and scalability by leveraging the intelligence of the OSDs. Each OSD uses a journal to speed-up the write operations by coalescing small writes and flushing them asynchronously to the backing filesystem when the journal is full. The journal can be a different device, partition or file.

Ceph's most interesting feature is the CRUSH algorithm (Controlled Replication Under Scalable Hashing) [3]. CRUSH is a pseudo-random placement algorithm that allows OSDs and clients to compute object locations instead of looking them up in a centralised table. Then clients can directly interact with the OSDs for I/O. This promises, in theory, extreme scalability. The CRUSH map also allows for defining hierarchies of failure domains—for placing object replicas—at different levels, e.g. disk, host, rack and room. Based on these hierarchies and on so-called CRUSH rules, CRUSH maps objects to placement groups (logical aggregations of objects inside one pool) and then maps each placement group to one or more OSDs.

Ceph's system architecture is illustrated in Fig. 1. The monitors (MONs) are responsible for maintaining the cluster

978-1-4799-5666-1/14/$31.00 ©2014 IEEE

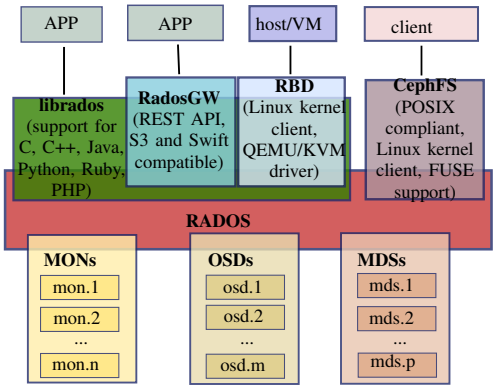


Fig. 1. The Ceph Architecture

map. Clients contact one of the monitors to obtain the most recent cluster map, which includes information on the cluster topology, a list of OSDs, pools, placement groups, mapping of placement groups to OSDs—all the information needed to further interact with the OSDs directly. The metadata servers (MDSs) store and manage Ceph filesystem's metadata.

Ceph exposes different interfaces to storage: a POSIX-compliant filesystem (CephFS), block storage (Rados Block Device/RBD) and a RESTful API (Rados Gateway). Clients can directly access RADOS through librados, with support for several programming languages, including: C/C++ and Python.

There is a set of alternative distributed storage systems and NoSQL databases that could fulfil our use cases.

The General Parallel File System (GPFS) [4] is widely used in HPC with good performance. However, GPFS is a proprietary filesystem, usually deployed on proprietary hardware. Thus, high costs and vendor lock-in were two of our arguments against GPFS.

The Hadoop Distributed File System (HDFS) [5] has been deployed on large production clusters with massive throughput capabilities. However, HDFS uses only one NameNode to manage the filesystem namespace (nevertheless, there is work in the direction of high availability by using a standby NameNode for automatic failover, feature released in Hadoop 2.0 [6]). This centralisation and single point of failure can limit the scalability and reliability of the system.

NoSQL (Not Only SQL) [7] approaches promise similar scalability to Ceph by using sharding—automatically partitioning and distributing the data across the cluster. However, they use a different data model—highly flexible and schema-less, but more appropriate for complex and specialized workloads. NoSQL databases can store data as key-value pairs, columns, documents, graphs, etc. The key-value approach that Ceph offers is sufficient for our use cases. Furthermore, NoSQL databases suffer from a lack of standardised interfaces, which often impedes their adoption, as opposed to Ceph, which exposes three most common interfaces: POSIX, block and S3.

As a result, despite being a young technology, we believe that Ceph fits our use cases best, due to the flexible access interfaces and low requirements for hardware.

III. Related Work

Performance of storage systems has been a topic of interest for many years, especially with the emergence of different parallel files systems, each designed to fill in certain gaps.

GPFS scalability and availability were evaluated by Schmuck et al. [8]. The authors investigated possible bottlenecks, such as distributed locking (used to guarantee POSIX semantics), which scaled surprisingly well. They also presented lessons learnt from the few hundred deployments at customer sites, related to load balancing, inode prefetching and GPFS replication as extra measure to RAID for fault tolerance.

Shvachko et al. evaluated the HDFS performance [9] on a 25 PB cluster with 3500 nodes at Yahoo, used for indexing the web with MapReduce as part of their search engine. The authors observed a linear scaling of bandwidth with the number of nodes, as well as an aggregated throughput of 34.2 GB/s during a large-scale sorting benchmark. They also discuss the scalability issues caused by the single NameNode and propose solutions to address them, some of which were later implemented in Hadoop [5] [6].

Wang et al. [10] evaluated the file and block I/O performance and scalability of Ceph, using a commercial high-end storage system. The authors measured the read and write throughput over different numbers of OSD servers and different number of clients. They found the best Ceph configuration for their system through a parameter sweep over a reduced parameter space, improving the RADOS performance by up to 16.3%. After repeated tuning, their Ceph system was able to reach 70% of the raw hardware capability at RADOS level, and 62% at the filesystem level. The paper's most important contributions, however, stem from the collaboration with the Ceph developers—their findings were incorporated to directly improve Ceph's performance and code quality.

Van der Ster et al. [11] from CERN IT describe a petabyte-scale deployment of Ceph, used as back-end block storage for the CERN Tier-0 OpenStack cluster. In this paper, the authors focus on their experience with deploying and running a large-scale cluster, best practices and their use cases for Ceph. They also report on various functionality and performance tests: they observed a decrease by 35% in the single client write throughput when increasing the replication from 1 to 3. Furthermore, they evaluated the scalability of Ceph with respect to the number of files written to the cluster and observed no load or memory consumption increase. However, they found that by adding a new OSD server in this case, the rebalancing took a long time (24 hours). In a subsequent, unpublished work [12], the authors report on IOPS limitations due to the co-location of the Ceph journal and the OSD data on spinning disks, and speculate that SSD journals could at least double the IOPS capacity.

Our work sets out to make two contributions over other Ceph evaluations—by using commodity hardware to quantify and explain performance overheads at different levels, and also by analysing Ceph's scalability over a larger, multidimensional parameter space.

IV. Testing Methodology

We devised a set of hypotheses that encompass the most important performance aspects of distributed systems, such as: scalability, overhead caused by complexity and specific hardware configuration. We then focused our benchmarking efforts towards exploring and validating the following hypotheses:

- H1 the system scales when adding more OSDs
- H2 the system scales with more clients
- H3 the RBD on top of RADOS incurs some overhead
- H4 the journal configuration is an important factor; placing the journal on disk limits scalability

To this end, we employed a bottom-up testing strategy in order to establish the theoretically expected maximum performance and the performance loss incurred at every additional layer of Ceph. We defined the following layers for testing:

1) raw disks and network
2) individual OSDs
3) distributed object store (RADOS)
4) block device (RBD)

As there is no unified tool for testing, we used different tools for each layer (tools cross-referenced with layers):

- *netcat* [13] and *iperf* [14] for the network (layer 1)
- *dd* [15] for the raw disk (layer 1)
- *osd tell* for the journaled write on each OSD (layer 2)
- *rados bench* for RADOS (layer 3)
- *fio* [16] for RBD (layer 4)

The tools *osd tell* and *rados bench* are shipped with Ceph and have a simple command line interface, where one can pass different parameters, such as: I/O operation, number of concurrent operations, duration of run or object size. Due to the low-level, custom interfaces to OSDs and RADOS, there are no alternative tools for benchmarking I/O at these layers.

fio is a widely used, highly flexible tool for benchmarking I/O. It supports several operating systems and I/O engines. It works on block devices, as well as files. Deutsche Telekom added support for a new I/O engine to *fio* [17] [18]—the *librbd* engine—that talks directly to Ceph RBD and the Ceph internal Filestore. We chose this modified version to eliminate the need for the RBD kernel client (not integrated in our chosen CentOS 6.4's kernel). The typical use case for RBD (back-end for IaaS) also uses the librbd API. An alternative tool for RBD is Ceph's *rbd bench*. Only using tools from the Ceph ecosystem has the advantage of a more consistent comparison, ensuring that the metrics are computed in the same way. However, tools like *fio* guarantee platform independence, reproducible results and make the comparison with other storage systems much easier.

Assembling the set of tools is challenging, especially with respect to comparing the performance reported by each tool. To alleviate this, we used the same synthetic, simple workloads for all tests—sequential write operations of fixed block/object size, followed by sequential reads. Moreover, we used a single performance metric at all layers—the bandwidth (throughput).

The scalability tests were performed by launching single client benchmarks in parallel on several clients. For the RADOS tests, we created a separate pool for each client, since *rados bench* cannot deal with multiple processes reading the objects written in a previous run. For the RBD tests, different RBD images in the same pool were sufficient. The metric for parallel tests was the aggregated throughput—the sum of the throughput measured at each client.

TABLE I
OSD server configuration

Processor	Intel Xeon E5430, 2.66 GHz, 8 cores
Network	2 × 1G Ethernet
Memory	32 GB
Disks	2 × SATA drives, 7200 RPM, 750 GB
Operating System	CentOS 6.4

TABLE II
Ceph cluster—with two different journal configurations

Ceph release	v0.72 (Emperor)
MONs | OSDs	3 | 8 (one per node)
OSD filesystem	XFS
Ceph journal	1 GB, *either* in memory (*tmpfs*)
	or a partition on the same disk as OSD data
Networks	1 public (for client access), 1 private (for OSD traffic)

The test environment comprised 32 commodity servers. The hardware specification of one server is summarised in Table I, and the Ceph cluster configuration in Table II. We tested two different journal configurations. First, the default installation of Ceph was used, which places the journal on the same disk as the OSD data, on a partition at the beginning of the disk; this means the expected write bandwidth is significantly lower. For our second test environment, we placed the journal in memory, thereby measuring the real overhead of the journal. We tested the system with cold cache by clearing the memory caches before every run (pagecache, dentries, inodes). We repeated the tests 3 times, plotted the average and standard deviation.

V. Performance Results

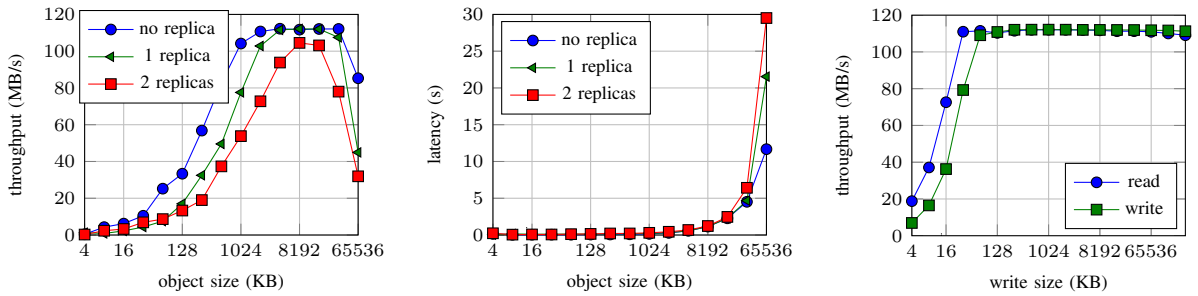
A. Performance Baseline

We first established the performance baseline imposed by the hardware. The limiting factors in this case are the network and disk. We measured a 111 MB/s network bandwidth and a 72 MB/s raw disk throughput (with *dd* writing 1 GB of zeroes). Furthermore, we measured the overhead of Ceph's journal on individual OSDs with *osd tell*. With Ceph, any write to an OSD translates to two write operations: one to the journal and another to the back-end filesystem. With the journal placed on the same disk as the OSD data, we observed a 30 MB/s throughput, compared to 62.5 MB/s for the in-memory journal.

These results entail theoretical limits for maximal read/write performance as functions of the number of OSDs (Table III).

TABLE III
Theoretical limits for maximal throughput with 8 OSDs

Per-client network throughput	111 MB/s
Cluster read throughput	8 × 72 = 576 MB/s
Cluster write throughput (journal on disk)	8 × 30 = 240 MB/s
Cluster write throughput (journal in memory)	8 × 62.5 = 500 MB/s



(a) RADOS: Write throughput (b) RADOS: Write latency (c) RBD: Throughput

Fig. 2. Single client tests: RADOS and RBD performance over different object sizes and different replication levels, 16 threads. Journal located in memory.

B. Single Client Tests

The first set of tests was performed using a single client. We observed that the journal configuration does not have a significant impact in this case, therefore it is sufficient to show the results of the in-memory journal configuration.

Using *rados bench*, we performed sequential write and read to the RADOS layer for 200 seconds and 100 seconds, respectively. The total size of data used by the benchmark was therefore variable. We measured the throughput with either a varying number of threads (or operations in-flight) at the client side and a fixed object size, or vice versa, with a varying object size and a fixed number of threads.

We observed that increasing the number of threads or the object size saturates the network bandwidth at the client side for both read and write. Replication can hinder the write performance, since it is done synchronously, with a performance penalty of up to 50% for 3-way replication. The read performance is not affected by replication because operations are always performed on the primary copy of an object.

Fig. 2(a) and Fig. 2(b) show the RADOS write throughput over varied object sizes. Note the low throughput for small object sizes—quite common for storage systems. The bandwidth reaches the network limit for medium-sized objects. An interesting result is the drop in performance for objects larger than 32 MB, doubled by an increase in latency. The reason for this high latency could be that large objects need to be split up into smaller sizes to be transmitted to the storage device.

The performance at the RBD layer over different block sizes is displayed in Fig. 2(c). No replication was used. A total amount of 1 GB of data was written to the RBD image. Though limited by the network bandwidth as well, RBD performs better than RADOS, because the RBD images are striped over objects of fixed size (by default, 4 MB).

To overcome the performance limit imposed by the client network, we had to increase the number of clients and OSDs to find the cluster's maximum performance.

C. Scaling OSDs

To verify hypothesis H1, we ran parallel tests (8 clients) on different deployments of the cluster—increasing the number of OSDs (8, 12, 16, 20). For each set-up, we used two journal configurations: on disk and in memory. We matched all other parameters to the ones that yielded the best results in the single client tests: 4 MB objects, 16 threads per client, no replication.

Fig. 3 shows that performance increases steadily with the number of OSDs, so H1 is valid, but it does not approach the hardware limit. The read performance, for example, scales almost linearly—the throughput for 16 OSDs is 1.91 times the one for 8 OSDs. However, compared to the hardware limits discussed in Subsection V-A, the throughput is at most 64% of the OSD-imposed limit (with 12 OSDs), and 93% of the network-imposed limit (with 20 OSDs). The write performance ranges from 33% to 70% of the theoretical limit with memory journal, and from 28% to 70% with disk journal.

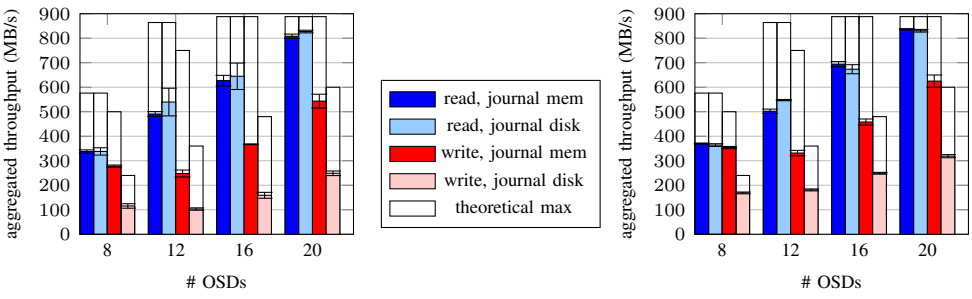
These results indicate that the current set-up does not use the disks efficiently. Possible explanations are: not enough clients to saturate any of the hardware resources; suboptimal Ceph or kernel configurations. We investigate these ideas in the next sections, as well as why RBD performs better than RADOS, despite RADOS being closer to the hardware.

D. Scaling Clients

We tested our system's performance with increasing number of clients (hypothesis H2) by running the parallel tests on all the different Ceph configurations. The results for RADOS, with 8 and 16 OSDs, are shown in Fig. 4(a) and 4(b).

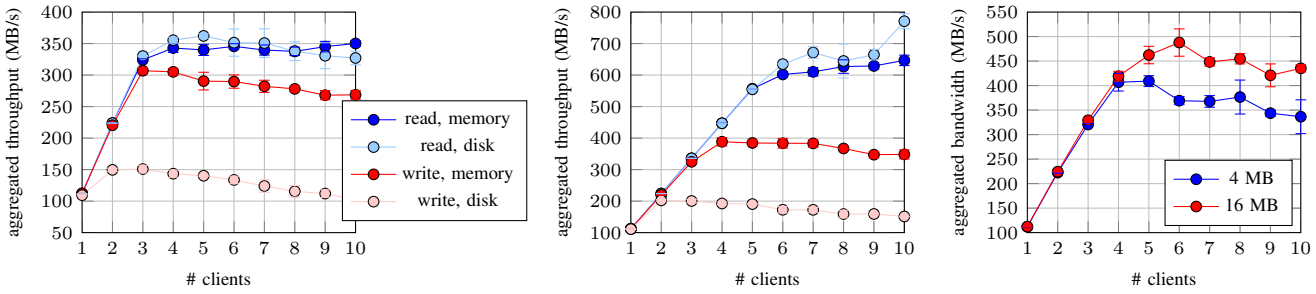
We note that hypothesis H2 was validated, up to a certain point, limited by disk throughput. With 16 OSDs, read throughput scales linearly up to 5 clients, while write throughput scales up to 4 and 2 clients, for memory and disk journal configuration, respectively. In the regions with linear scalability, the throughput reaches the theoretical network limit.

More interestingly, we observed a slight degradation in the RADOS write performance with increasing number of clients. Benchmarks performed at Intel [19] report a similar behaviour—with a different test set-up that involves virtual machines with attached volumes on top of RBD images. They believe the cause is an increase in disk seek: even though the I/O pattern at the client side is sequential, the I/O streams from each client get distributed across the OSDs and mixed together with streams from the other clients, resulting in a random access pattern on the OSD disks. Random disk access involves more disk seek, which has a strong impact on the disk performance. Two solutions were proposed:



(a) RADOS (b) RBD

Fig. 3. Scaling OSD nodes—throughput over increasing number of OSDs. 8 clients, 16 threads per client, 4 MB block size. The theoretical maximum performance for each case is depicted with an empty rectangle.



(a) 8 OSDs (b) 16 OSDs (c) 20 OSDs, changing the I/O pattern

Fig. 4. Scaling clients with different journal configurations (in memory and on disk): throughput at RADOS level over increasing number of clients, 16 threads per client, 4 MB object size, with 8 OSDs (a) and 16 OSDs (b). Improving this scalability by changing the I/O pattern (c): journal on disk, 20 OSDs, 16 threads per client, writing objects of different sizes.

1) **improve the random I/O performance** by speeding up the filestore, e.g. by increasing the RAM available or adding SSD for write-back/through cache in a tiered storage. The second option requires a radical hardware reconfiguration and Ceph's tiering feature (introduced in release v0.80). We examined the first option by tuning the Linux kernel: increasing the memory allocated for receive/send sockets, and the TCP read/write buffer size, to 12 MB. We observed an increase in write throughput as high as 9% (on average 5% after the saturation point). However, a side-effect was a decrease in the read throughput (on average 5% after the saturation point).
2) **change the I/O pattern**, by increasing the write size or enabling the RBD cache. Changing the object size from 4 MB to 16 MB boosted the write bandwidth by up to 27%, for 20 OSDs with the journal on disk (on average 21% after the saturation point), as shown in Fig. 4(c).

E. Different Journal Configurations

We measured the performance impact of Ceph's journal by running our parallel tests on different configurations of the Ceph cluster—placing the journal in memory (1 GB *tmpfs* partition) or co-located with the OSD data on a spinning disk (1 GB partition at the beginning of the disk, since it is faster).

Fig. 4 shows a doubling in write throughput by placing the journal in memory, which validates our hypothesis H4. The result is also a strong argument to test the placement of journals on SSDs. Even though SSDs and memory have different read/write performance, it is clear that a faster journal is required. In-memory journals are much faster, but extremely unreliable in the event of OSD crashes. SSDs are also fast, more reliable (except for the limited erase cycles), but orders of magnitude slower than memory. A more thorough analysis could shed more light on the best journal configuration.

F. RADOS versus RBD Performance

To test hypothesis H3, we quantified the impact of system complexity by measuring the performance loss at each layer. We postulated that there is a decrease in throughput from the raw disk, to the OSD level, to RADOS and finally, RBD; we then set out to measure this decrease. Fig. 5 illustrates this for writes to a Ceph set-up with 8 OSDs and journal in memory.

Note that, with enough clients (more than 5 in this case), the network capability does not count in establishing the maximum theoretical performance, but rather the disk capability. The penalty of writing objects first to the in-memory journal and then to the disk is around 13% of the disk capability. These are both theoretical limits measured on one OSD.

Moreover, the measured RBD write throughput has a penalty of 29% compared to the maximum journaled write on one OSD. A surprising finding was that the measured RADOS write performance is 6% to 23% less than the RBD

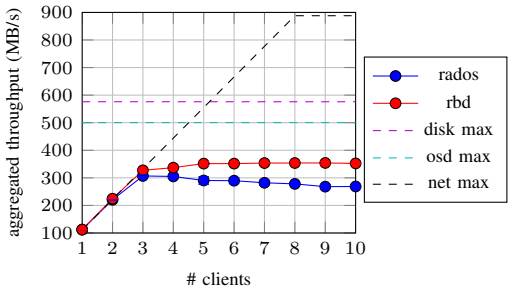


Fig. 5. Write performance of RADOS vs RBD over increasing number of clients. Journal in memory, 8 OSDs, 16 threads per client, 4 MB write size.

performance, worsening with more clients. Therefore, our hypothesis H3 was invalidated: the performance is better at the RBD layer than at the lower RADOS layer.

However surprising this may seem, we first must establish whether the comparison makes sense. Although all parameters are the same for the RADOS and RBD tests (write size, replication, number of clients, threads per client), the tools we used for testing are different, and they can behave differently, e.g. by applying implicit optimisations. We believe that one factor causing better performance of RBD is the client-side buffered I/O done by *fio*: by default, *fio* caches writes in memory and flushes/syncs them periodically, when the buffer cache is full. We then compared results from *fio* runs with buffered and direct I/O, and observed a difference of approximately 5% in throughput—which only amounts for less than half of the performance difference of 13% between RBD and RADOS.

VI. Conclusion

In this paper, we evaluated the Ceph distributed storage system. Our work makes two unique contributions. Firstly, our multidimensional scalability evaluation led to a better understanding of how each dimension (number of OSDs, number of clients, object size) affects performance, not necessarily in a liner fashion. Secondly, the tests were carried on commodity hardware, for which Ceph was initially designed, and therefore focused on unraveling and overcoming the limitations of the undelying platform; the results can benefit Ceph users that do not have access to high-end storage systems, but wish to maximise the performance of available hardware.

Our experiments provide several important insights:

- Ceph scales almost linearly when adding more storage nodes. Though an expected outcome—due to the Ceph design based on failure ubiquity and deterministic data distribution—it is now backed by empirical proof.
- The journal setting has a significant impact on the write performance: when the journal is placed in memory, the write throughput is double or higher than the case when the journal is co-located with the OSD data.
- Ceph scales well with many clients operating the cluster in parallel, up to the saturation of some resource—usually disk bandwidth. After saturation, the write performance degrades slightly with more clients, due to an increase in disk seek; this can be alleviated by improving the random I/O performance on OSDs or by changing the I/O pattern.
- RBD outperforms RADOS in most cases, which is unexpected, since RADOS is closer to the hardware than RBD. Partly explained by different optimisations included in the RBD client, this result raises the question of how to properly compare the performance of different interfaces to storage when there is no common testing tool available.

Acknowledgement

The research leading to these results has received funding from the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement no. 604102 (Human Brain Project).

References

[1] S. A. Weil, S. A. Brandt, E. L. Miller, D. D. Long, and C. Maltzahn, "Ceph: A scalable, high-performance distributed file system," in *Proceedings of the 7th symposium on Operating systems design and implementation*. USENIX Association, 2006, pp. 307–320.
[2] H. Markram, K. Meier, T. Lippert, S. Grillner, R. Frackowiak, S. Dehaene, A. Knoll, H. Sompolinsky, K. Verstreken, J. DeFelipe *et al.*, "Introducing the human brain project," *Procedia Computer Science*, vol. 7, pp. 39–42, 2011.
[3] 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*. ACM, 2006, p. 122.
[4] J. Barkes, M. R. Barrios, F. Cougard, P. G. Crumley, D. Marin, H. Reddy, and T. Thitayanun, "Gpfs: a parallel file system," *IBM International Technical Support Organization*, 1998.
[5] T. White, *Hadoop: The definitive guide*. " O'Reilly Media, Inc.", 2012.
[6] A. Oriani and I. C. Garcia, "From backup to hot standby: High availability for hdfs," in *Proceedings of the 2012 IEEE 31st Symposium on Reliable Distributed Systems*. IEEE Computer Society, 2012, pp. 131–140.
[7] R. Cattell, "Scalable sql and nosql data stores," *ACM SIGMOD Record*, vol. 39, no. 4, pp. 12–27, 2011.
[8] F. B. Schmuck and R. L. Haskin, "Gpfs: A shared-disk file system for large computing clusters." in *FAST*, vol. 2, 2002, p. 19.
[9] K. Shvachko, H. Kuang, S. Radia, and R. Chansler, "The hadoop distributed file system," in *Mass Storage Systems and Technologies (MSST), 2010 IEEE 26th Symposium on*. IEEE, 2010, pp. 1–10.
[10] F. Wang, M. Nelson, S. Oral, S. Atchley, S. Weil, B. W. Settlemyer, B. Caldwell, and J. Hill, "Performance and scalability evaluation of the ceph parallel file system," in *Proceedings of the 8th Parallel Data Storage Workshop*. ACM, 2013, pp. 14–19.
[11] D. van der Ster and A. Wiebalck, "Building an organic block storage service at cern with ceph," in *Journal of Physics: Conference Series*, vol. 513, no. 4. IOP Publishing, 2014, p. 042047.
[12] D. van der Ster. (2014, May) Ceph at cern: one year on. [Online]. Available: http://indico.cern.ch/event/274555/session/16/contribution/10
[13] *Hobbit*. (1996, Mar.) Netcat (version 1.10). [Online]. Available: http://nc110.sourceforge.net/
[14] (2010, Jul.) Iperf (version 2.0.5). [Online]. Available: http://iperf.fr
[15] P. Rubin, D. MacKenzie, and S. Kemp. (2014, Mar.) dd (coreutils) (version 8.4). [Online]. Available: http://www.gnu.org/software/coreutils/manual/html_node/dd-invocation.html
[16] J. Axboe. (2013, Jan.) Fio (version 2.0.13). [Online]. Available: http://freecode.com/projects/fio
[17] ——. (2014, May) Fio (version 2.1.9-17-gd9b1). [Online]. Available: http://git.kernel.dk/?p=fio.git;a=summary
[18] D. Al-Gaaf and D. Gollub. (2014, Feb.) Telekomcloud devops team – ceph performance analysis: fio and rbd. [Online]. Available: http://telekomcloud.github.io/ceph/2014/02/26/ceph-performance-analysis_fio_rbd.html
[19] J. Duang. (2013, Nov.) Measure ceph rbd performance in a quantitative way (part ii). [Online]. Available: https://software.intel.com/en-us/blogs/2013/11/20/measure-ceph-rbd-performance-in-a-quantitative-way-part-ii