University of Amsterdam

System & Network Engineering

Linux Open Source Distributed Filesystem

July, 2013

Authors:
Remco van Vugt remco.vanvugt@os3.nl

Abstract

Ceph is a clustered, distributed open source storage system for the Linux platform. In this research an assessment is made on the scalability, reliability and manageability of Ceph in a large multi-Petabyte environment such as the environment at SURFsara. Focus lies on the filesystem part of Ceph, CephFS. Performance experiments are used as a basis for the assessment of scalability and to determine the handling of various failure conditions.

Results show that although Ceph delivers promising performance and resiliance against various failure conditions, it is not yet ready for production use in the environment at SURFsara. Especially the Meta Data Server (MDS) lacked reliability and did not scale very well.

Contents

1 Introduction

SURFsara supports researchers and the academic community in The Netherlands. SURFsara provides various computing services to support research, including storage, grid computing and cloud computing.
Storage needs at SURFsara are in the order of Petabytes and constantly growing. SURFsara is looking for a distributed, POSIX compliant filesystem with the following key requirements:

- Scalability: the filesystem needs to able to expand (virtually) unlimited as long as storage is added to the cluster. Scalability is required in both size and performance.
- Reliability: the filesystem needs to be able to handle node and disk failures without affecting availability.
- Manageability: the filesystem needs to be easy manageble in terms of expanding, decommisioning failed and replaced nodes and other hardware related changes. This should be handled by the filesystem in an automated manner and without affecting availability.https://www.sharelatex.com/templates

In this research we try to establish whether the open source Ceph storage system offers an alternative to traditional proprietary appliance based SAN storage, for running a large on-disk data archive containing multiple petabytes of data. Ceph is a open source distributed storage system supporting block, object and file-based storage.
Ceph was created by Sage Weil in 2007 [16]. Ceph is under active development, and is both supported by the community as well as commercial. Ceph architecture as shown in figure 1 consists of a distributed object store (RADOS), application library (LIBRADOS), a block device implementation (RBD) and a filesystem component (CephFS).

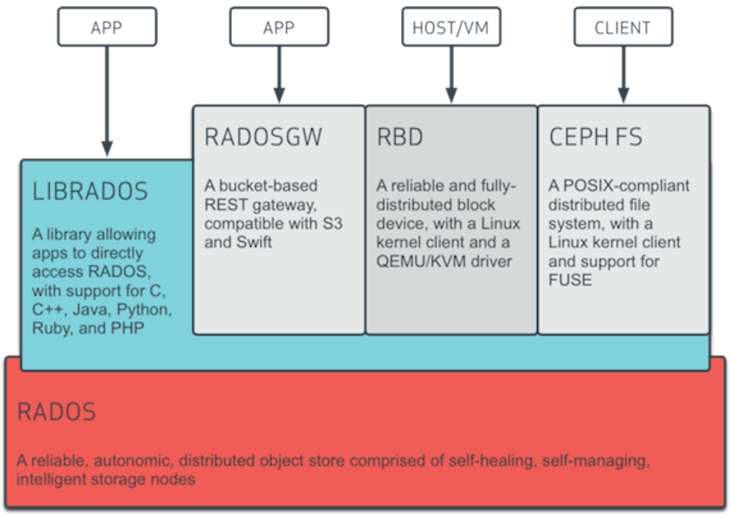

APP
APP
HOST/VM
CLIENT
RADOSGW
A bucket-based REST gateway, compatible with S3 and Swift
RBD
A reliable and fully-distributed block device, with a Linux kernel client and a QEMU/KVM driver
CEPH FS
A POSIX-compliant distributed file system, with a Linux kernel client and support for FUSE
LIBRADOS
A library allowing apps to directly access RADOS, with support for C, C++, Java, Python, Ruby, and PHP
RADOS
A reliable, autonomic, distributed object store comprised of self-healing, self-managing, intelligent storage nodes

Figure 1: Ceph architecture. Note: adapted from Ceph documentation [2]

CephFS could be a possible candidate for the distributed filesystem that is needed at SURFsara. CephFS is the latest addition to the Ceph project. According to the developers, it is suitable for production use with a limited feature set [3]

1.1 Related work

A lot has been written on distributed filesystems in general, however no recent work was found on Ceph in particular at the time of writing. Ceph is included in a theoretical assessment on performance of distributed storage system by Christopher Olson and Ethan Miller [9].

The question whether CephFS is *production-ready* has come up on mailing lists on several occasions, and exact answers are not given. The development team behind CephFS recently made a blog post on the topic [3] stating that it is production ready for a limited feature set. The research by Sage Weil [16] is already more than five years old. Ceph has been under substantial development over the last five years. Therefore a re-assessment of Ceph and CephFS as a distributed storage system will contribute to the open source community and will be useful for SURFsara and organizations facing comparable data storage challenges.

1.2 Research question

Is the current version of CephFS (0.61.3) production-ready for use as a distributed filesystem in a multi-petabyte environment, in terms of stability, scalability, performance and manageability?

To answer the research question, three sub questions have been defined concentrating on the main metrics (stability, scalability and performance, manageability) used for assessing the *production-readiness* of CephFS.

1.3 Subquestions

- *Is Ceph, and an in particular the CephFS component, stable enough for production use at SURFsara?*
  The CephFS component of Ceph is under development and not all features are stable at the moment of writing. We need find out whether the technical limitations that exist in the current implementation are acceptable for production use at SURFsara. Also, the underlying RADOS component of Ceph needs to be tested in order see if it is ready for use in a large production environment.

- *What are the scaling limits in CephFS, in terms of capacity and performance?*
  Performance will be tested on the relatively limited test setup. Tests on Petabyte level cannot be conducted. A theoretical assessment will be made on scaling and possible issues.

- *Does Ceph(FS) meet the maintenance requirements for the environment at SURFsara?*
  As the projected environment is expected to grow rapidly to a scale of multiple Petabytes of data, hardware failures are to be expected rather than to be considered an exception. The system must handle failures in an automated manner, making the system as low-maintenance as possible. Expanding capacity, and decomissioning old hardware must be straightforward and low-risk to data integrity. Where possible, maintenance tasks must be integrated into existing tooling.

1.4 Research Method

The research will consist of a series of experiments on stability, performance and scalability. The results of these experiments will be used to make an assement on the *production-readiness* of Ceph in a multi-petabyte environment such as the enviroment at SURFsara.

The details of the experiments are explained further in chapter three of this report.

1.5 Scope

The main focus of this research is on CephFS and the underlying RADOS object store, as this is the main use case of the Ceph project at SURFsara and the latest addition to the Ceph project. The object storage gateway and the application library (RADOSGW and LIBRADOS) of the Ceph project are not included in the scope of this research. The RADOS Block Device (RBD) is not extensively tested, however a quick comparison is made between CephFS and RBD on performance to assist in drawing conclusions on possible performance bottlenecks.

2 Ceph architecture

Ceph consists of three node types: Monitor (MON) node, Meta Data Store (MDS) node and Object Storage Device (OSD). Ceph provides three storage options: Object storage, Filesystem and Block storage. Object storage is simular to Amazon S3 [1] and can be a replacement for S3 as the interface is compatible. The filesystem storage can be used by mounting the filesystem using either a kernel driver or a FUSE implementation. The filesystem can be mounted by multiple client systems concurrently. The block storage component provides a kernel block device which can only be mapped to a single client system at a time.

Monitor nodes establish the cluster quorum by using node majority. This implies an odd number of MON nodes must be used. As one node does not provide redundancy, a minimum of three MON nodes must be deployed.

Ceph handles data as objects, which are stored as flat files on the filesystem of an OSD daemon as shown in figure 2. The underlying filesystem could be ext4, XFS or BTRFS. XFS is currently advised for production usage.

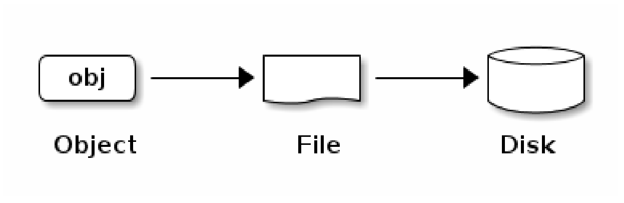


Figure 2: Ceph object based storage. Adapted from Ceph documentation [2]

OSD daemons run on OSD nodes, typically one daemon per physical storage device (e.g harddisk). Ceph uses the CRUSH algorithm to store data in a pseudo random manner. CRUSH is an acronym for *Controlled Replication Under Scalable Hashing.* The algorithm calculates the location for storing objects in the cluster, which is reversible by the client. This eliminates the need for a central database to obtain the location of objects. Therefore, in the CRUSH algorithm lies the key to the scalability of Ceph.

In order to use the CRUSH algorithm, the clients need information on the topology of the cluster. This is implemented in the CRUSH-map, which is part of the cluster map. The CRUSH-map holds a list of OSD's, grouped into buckets as shown in figure 3. Buckets are aggregations of OSD devices, grouped by possible sources of correlated failure. For example, OSD devices relying on the same machine, power supply, network or physical location. The CRUSH map can be edited in order to customize the failure domain behaviour.

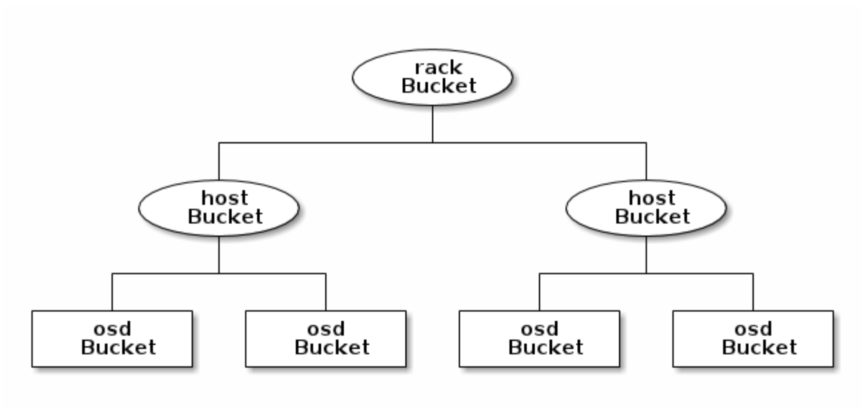


Figure 3: Crush buckets. Note: adapted from Ceph Documentation [2]

Data objects are placed in Placement Groups (PGs). PG's are placed on OSDs, with one primary OSD and one or more replica's on other OSDs. The primary OSD for the given placement group is responsible for accepting write operations from the client for the objects in that particular PG. The other OSDs in the PG accept only read operations from clients for objects in that particular PG. When data is written to the primary OSD for a given PG, data is replicated by the primary OSD to the other OSDs in the placement group. As soon as this is done, the write operation is acknowledged to the client. This implies synchronous replication. By offloading the replication from the client to the OSD, the client implementation is lightweight and relatively simple.

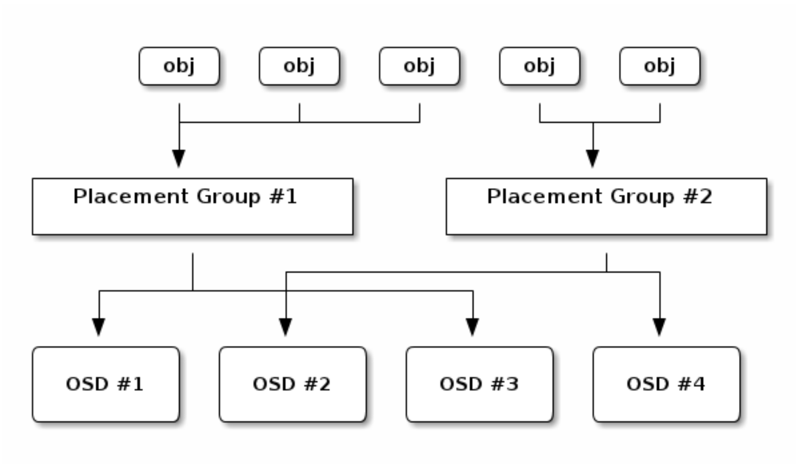


Figure 4: Crush Placement Groups. Note: adapted from Ceph Documentation [2]

When an OSD fails, a configurable timer is started to allow the OSD to come back up. If the OSD stays down, and the timer runs out (300 seconds by default) Ceph starts to restore the degraded PGs by remapping the objects on the failed OSD to another OSD. Proper balancing of PGs over OSDs is taken into account when rebuilding. If needed

PGs are rebalanced to adapt to achieve a balanced state.

Clients interact directly with the OSD daemons, avoiding the single point of failure and the performance bottleneck introduced by a central broker in the I/O path. By interacting directly with the OSDs, clients and OSDs form a full mesh, maximizing the utilization of available bandwidth on the client.

3 Experiments

3.1 Test setup

In order to gain hands-on experience with Ceph configuration and operation, a test setup is constructed. The test setup consists of a total of eight nodes with the following roles and specifications:

3.1.1 Hardware

2 Monitor and MDS nodes *Dell Poweredge R720xd*

- Dual Intel Xeon E5-2609 CPU
- 128 GB DDR3 RAM
- 12 x 600GB 15k SAS disk (Seagate ST3600057SS) (not used)
- 2 x 300GB 10k RPM SAS disk (Seagate ST9300605SS) (Operating system)

1 monitor node, 4 OSD nodes, 1 client node *Dell Poweredge R720xd*

- Dual Intel Xeon E5-2609 CPU
- 64 GB DDR3 RAM
- 12 x 3TB 7.2k RPM SATA disk (Seagate ST33000650NS) (OSD storage)
- 2 x 300GB 10k RPM SAS disk (Seagate ST9300605SS) (Operating system)

  The nodes are connected via 10 gigabit Ethernet, using a Juniper EX-4550 switch. Another Juniper EX-2200 switch is used to connect the management network and console interfaces. There are 2 seperate 10GbE segments. One segment is used for client access traffic, whilst the other is exclusively used for heartbeat, replication and other internal cluster traffic. The topology is shown in 5

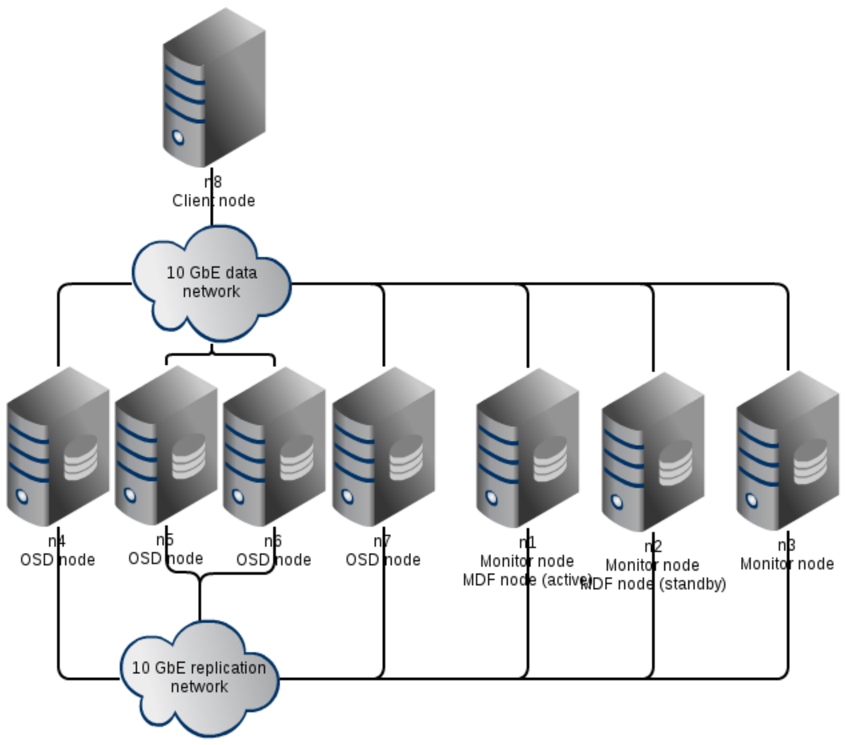


Figure 5: Test setup

3.1.2 Configuration

The nodes are installed with 64 bits CentOS 6.4 Linux. All nodes are running the stock kernel (2.6.32-358) except for the client node which is running a custom compiled kernel (3.9.4) to support the CephFS native kernel driver.
The Dell PERC H710P controller that came with the Poweredge R720xd machines did not support disabling RAID and running in JBOD mode. As a workaround, 12 RAID-0 sets containing one harddisk each were created. Two partitions were created on each disk, with a GPT disk layout. A small 10GB parition to hold the OSD journal, and the rest of the drive to hold the OSD data.

3.2 Ceph stability experiments

3.2.1 CephFS and MDS stability

To test the stability of the CephFS implementation, and the MDS the following tests will be executed:

Test	Description
MDS failure	Causing the MDS to fail during heavy I/O operation to test failover
Large file test	Create a file of 1TB to test if large files can be handled by the filesystem
Many file test	Create an increasing amount of files in a single directory.
Concurrent I/O	Perform file operations in a single directory from multiple clients.

3.2.2 Ceph general stability

To assess the stability of Ceph and RADOS in general the following tests will be executed. All tests will be executed during random I/O operation.

Test	Description
Monitor failure	Cause a monitor node to fail (power-off)
Object Storage Device (OSD) failure	Cause one OSD node to fail (power-off)
Object Storage Device disk failure	Pull a disk on one OSD.
Object Storage Device data corruption	Overwrite part of the files containing stored objects with random data.
Monitor daemon failure	Stop the monitor daemon non-graceful
Object Storage Device (OSD) daemon failure	Stop the monitor daemon non-graceful

3.3 Ceph scaleability

To assess the scalability of Ceph, the replication and heartbeat protocol will be studied to determine if there is substantial networking overhead when the number of OSDs increases.
Also a calculation will be made on scaling in terms of maximum number of OSD daemons per node, factoring in the load on CPU and RAM.

The following tests serve to gather performance data, used in the theoretical scalability assessment.

Test	**Description**
Test CephFS metadata operations	Test various POSIX metadata operations on CephFS
Test RBD (RADOS Block Device) metadata operations	Test various POSIX metadata operations on XFS formatted RBD
Test OSD troughput using OSD bench	Test individual OSD troughput using RADOS OSD benchmark tool
Test OSD harddisk throughput using hdparm	Test OSD underlying I/O subsystem using hdparm

3.4 Ceph performance

A number of tests will be performed to test the performance of Ceph and CephFS under various conditions (normal, degraded and rebalancing). For the benchmarks, the following tools will be used:

- Bonnie++ [4]
- RADOS bench [15]
- dd [6]
- hdparm [13]
- dstat [17]

Both IO operations per second as well as troughput in both read and write operations will be tested.

Test	**Description**
Test CephFS IOPS using Bonnie++	Test both random and sequential I/O operations using Bonnie++
Test CephFS IOPS while degraded using Bonnie++	Test both random and sequential I/O operations using Bonnie++ with Ceph cluster degraded
Test CephFS IOPS while rebalancing using Bonnie++	Test both random and sequential I/O operations using Bonnie++ with Ceph cluster rebalancing
Test CephFS troughput using RADOS bench	Test read and write troughput using RADOS bench
Test CephFS troughput while degraded using RADOS bench	Test read and write troughput using RADOS bench with Ceph cluster degraded
Test CephFS troughput while rebalancing using RADOS bench	Test read and write troughput using RADOS bench with Ceph cluster rebalancing

4 Results

4.1 Ceph Stability

4.1.1 CephFS and MDS stability

– **MDS failure test** Using the fs-metadata test [14] tool from the Linux kernel source tree, heavy I/O is generated on CephFS. The fs-metadata tool creates a k-tree data structure. A k-tree structure is a transformation of the binary-tree which has k child nodes instead of 2 in the binary-tree.
fs-metadata does not only test directory's but also files. Besides k child directory's, there are also k files created under a given node.
The fs-metadata test tool was executed with a tree depth of 5, and *k=10.* It also has the option to multithread, it was executed using 4 threads.
When the fs-metadata tool runs for 5 minutes, power was cut to the Ceph node running the MDS using the remote management interface.
Results During the first 5 minutes of the test, metadata operations function normally. Directory listing and directory traversing in the shell gives adequate response. As soon as the machine running the MDS is powered off, the cluster responds by promoting the standby MDS to active MDS. The switchover completes within 5 seconds, making the total failover less than 10 seconds.
Traversing directory's, or trying to open files from the mounted Ceph volume on the client-node is not possible. The respective *ls* and cat commands hang indefinitely or time-out with an I/O error. In the Ceph realtime monitor I/O operations drop to 0 operations per second. After a while, the kernel on the client node crashes (kernel panic) on a Non Maskable Interrupt (NMI).
This behaviour was reproducable in a second attempt. In a third attempt without the fs-metadata test running the kernel did not crash but I/O operations still did not succeed. The fourth attempt however showed a smooth failover with normal I/O behaviour.

– **Large file test** Using the Linux *dd* utility 1 TB of data from /dev/zero is copied to a file on the mounted CephFS with a block size of 4mb.
Results The file was created without problems.

– **Many files test** Up to 64.000 files are created in a single directory on the CephFS.
Results This was tested during the metadata performance tests, up to 64000 files in a single directory. No problem was encountered regarding stability.

– **Concurrent I/O test** Two threads simultaneously read and write on the CephFS.
Results No problems were encountered.

4.1.2 Ceph general stability

- **Monitor failure test** While writing to CephFS using *dd* a monitor node (not running MDS) is powered down.
  Results When the machine is powered down, this is logged on the other monitor nodes. No impact is noticeable on the I/O operations. The cluster shows it is running in degraded mode. When the monitor node is restarted with the *dd* still running, the cluster calls a new quorum election. The cluster resumes normal operation without any impact on I/O operations.
- **OSD failure test** While writing to CephFS using dd a OSD node is powered down.
  Results When the machine is powered down, within 20 seconds the cluster log shows the reports from neighbours coming in on the node being down. The placement groups (PG's) hosted on the OSD daemons running on the node which is powered off are showing active+degraded status. After 300 seconds, the cluster marks the OSD instances running on the node as *out*. The Ceph cluster starts remapping the PG's affected to restore redundancy. The CephFS automatically shrinks to account for the missing storage space. When the machine is rebooted, it automatically rejoins the cluster. The percentage degraded PG's drops from around 11 percent to around 1 percent. The CephFS automatically expands back to 131 Terabyte.

- **OSD disk failure test** While writing to CephFS using *dd* a disk was pulled from an OSD node, without unmounting cleanly.
  Results The removal of the disk caused the OSD daemon using the particular disk to crash. This was detected by the other OSD's and reported to the monitor nodes. The cluster initated a rebuild and returned to a healthy state.
- **OSD data corruption test** Some of the flat files of the OSD are corrupted by appending random data to the files
  Results The corruption causes the OSD daemon to crash. The OSD is marked out of the cluster, and objects are relocated.

- **OSD daemon crash test** A daemon is killed without allowing it to shutdown cleanly.
  Results The crashed daemon is handled by the cluster without service interruption.

4.1.3 Conclusion on stability

Failures that were triggered on OSD and MON nodes within the designed fault-tolerance limits and respective failure domains are all handled without any undesired side-effects. The limits for fault-tolerance in Ceph are configurable by the user by increasing the number of replica's. The failure domain is also user-configurable which allows the user to adapt the placement of the replica's in such way to minimize the risk of all replica's becoming unavailable.

Failures that were triggered on MDS nodes were not in all cases handled correctly by the cluster. There were a number of issues regarding MDS failover and cluster management in general:

- MDS failure, during high I/O load on CephFS.
  When a MDS daemon crashes when the CephFS is under heavy I/O stress, the failure is handled by the cluster. However, the client using the kernel driver either experiences a kernel crash (kernel panic) or a lockup up I/O subsystem. The client could only remount the CephFS after a reboot.
- Multi-MDS, random crashes.
  As described in the CephFS documentation, the use of multiple active MDS servers is not considered to be stable at the moment of writing. Several crashes of MDS daemons were observed during testing
- Scaling back from multiple active MDS to a single MDS.
  After testing multiple active MDS, it was not possible to scale back to single MDS. The cluster keeps indicating an unhealthy state. The only way to revert to a single MDS was deleting the CephFS data and metadata pools from RADOS. This means all data on the CephFS is lost.

4.2 Ceph performance

CephFS IOPS using Bonnie++ Using the Bonnie++ tool on the CephFS mount point with a 125 GB filesize to avoid cache, a sequential write troughput of **534 megabyte / sec** and a sequential read troughput of **451 megabytes /sec** were recorded. Random I/O operations were handled at **2908 IOPS**.

CephFS IOPS while degraded using Bonnie++ To avoid Ceph to start rebuilding and rebalancing automatically, we set the *noout* flag on the cluster. This way, the OSD's we take down are not marked out of the cluster and therefore no rebuild will be triggered automatically. We then shutdown node 5, and start testing with Bonnie++.

Using the Bonnie++ tool on the CephFS mount point with a 125 GB filesize to avoid cache, a sequential write trougput of **555 megabyte / sec** and a sequential read troughput of **443 megabyte / sec** were recorded. Random I/O operations were handled at **1224 IOPS**.

CephFS IOPS while rebalancing using Bonnie++ To force Ceph to start rebalancing, three OSD's are marked out of the cluster. The cluster starts rebuilding using the OSD's still in the cluster. Once done, the three OSD's are marked in the cluster. The cluster now starts rebalancing to spread the PG's and objects equal over the OSD's.

Using the Bonnie++ tool on the CephFS mount point with a 125 GB filesize to avoid cache, a sequential write troughput of **396 megabyte / sec** and a sequential read troughput of **275 megabyte / sec** were recorded. Random I/O operations were handled at **2180 IOPS**.

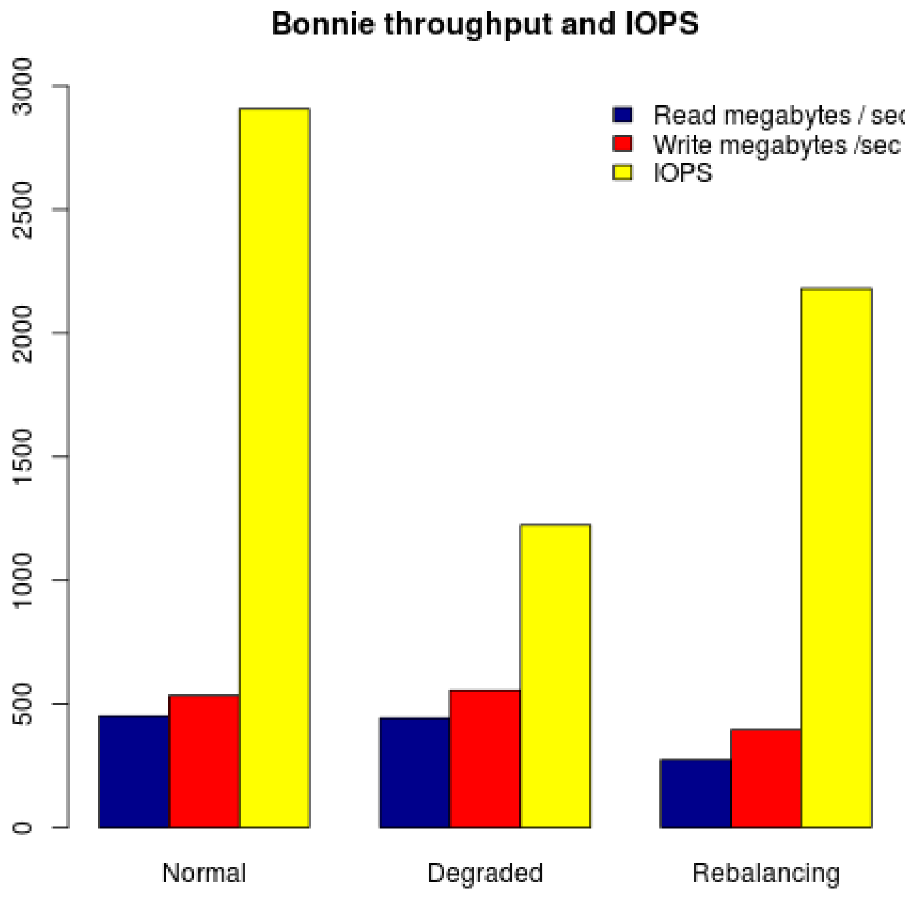


Figure 6: Bonnie++ benchmark troughput and IOPS test results

CephFS troughput using dd
Using the Linux dd tool, a total of 125 GB was written from /dev/zero to a file on the CephFS. A block size of 4 megabyte was used. A write throughput of **640 megabyte /sec** was recorded. The test was repeated to test read throughput, reading the contents of the created file to */dev/null*. The cache was cleaned before running the test. A read throughput o **358 megabyte / sec** was recorded.

RADOS troughput using RADOS bench
Using the RADOS benchmark tool, 16 concurrent writes of 4 megabyte in size are being executed for 900 seconds. This resulted in an average troughput of **840 megabytes / sec**. This figure was consistent with a standard deviation of 37 megabytes / sec over 189005 write operations.

Sequential read troughput was tested using the data written in the RADOS benchmark write test, executed for 900 seconds. This resulted in an average throughput of **541 megabytes / sec** over 121992 read operations.

RADOS troughput using RADOS bench when degraded
Using the RADOS benchmark tool, 16 concurrent writes of 4 megabytes in size are being executed for 900 seconds. This resulted in an average throughput of **951 megabytes / sec**. This figure was consistent with a standard deviation of 54 megabytes /sec over 214200 write operations.

Sequential read throughput was tested using the data written in the RADOS benchmark write test, executed for 900 seconds. This resulted in an average troughput of **514 megabytes / sec** over 115777 read operations.

RADOS troughput using RADOS bench when rebalancing
Using the RADOS benchmark tool, 16 concurrent writes of 4 megabytes in size are being executed for 900 seconds. This resulted in an average throughput of **489 megabytes / sec**. This figure was consistent with a standard deviation of 76 megabytes /sec over 110152 write operations.

Sequential read throughput was tested using the data written in the RADOS benchmark write test, executed for 900 seconds. This resulted in an average troughput of **410 megabytes / sec** over 92376 read operations.

RADOS troughput using RADOS bench when rebuilding
A rebuild was triggered by removing an OSD with the *no-out* flag set to avoid rebalancing. The OSD was formatted and reinserted into the cluster.
Using the RADOS benchmark tool, 16 concurrent writes of 4 megabytes in size are being executed for 900 seconds. This resulted in an average throughput of **478 megabytes / sec**. This figure was consistent with a standard deviation of 82 megabytes /sec over 107501 write operations.

Sequential read throughput was tested using the data written in the RADOS benchmark write test, executed for 900 seconds. This resulted in an average troughput of **460 megabytes / sec** over 103755 read operations.

Normal	Read	Write	IOPS
Bonnie++	451	534	1224
RADOS bench	541	840	
Degraded	Read	Write	IOPS
Bonnie++	443	555	2908
RADOS bench	541	840	
Rebalancing	Read	Write	IOPS
Bonnie++	275	396	2180
RADOS bench	410	489	

Table 1: Benchmark troughput in megabytes / sec and IOPS using Bonnie++ and RADOS-bench

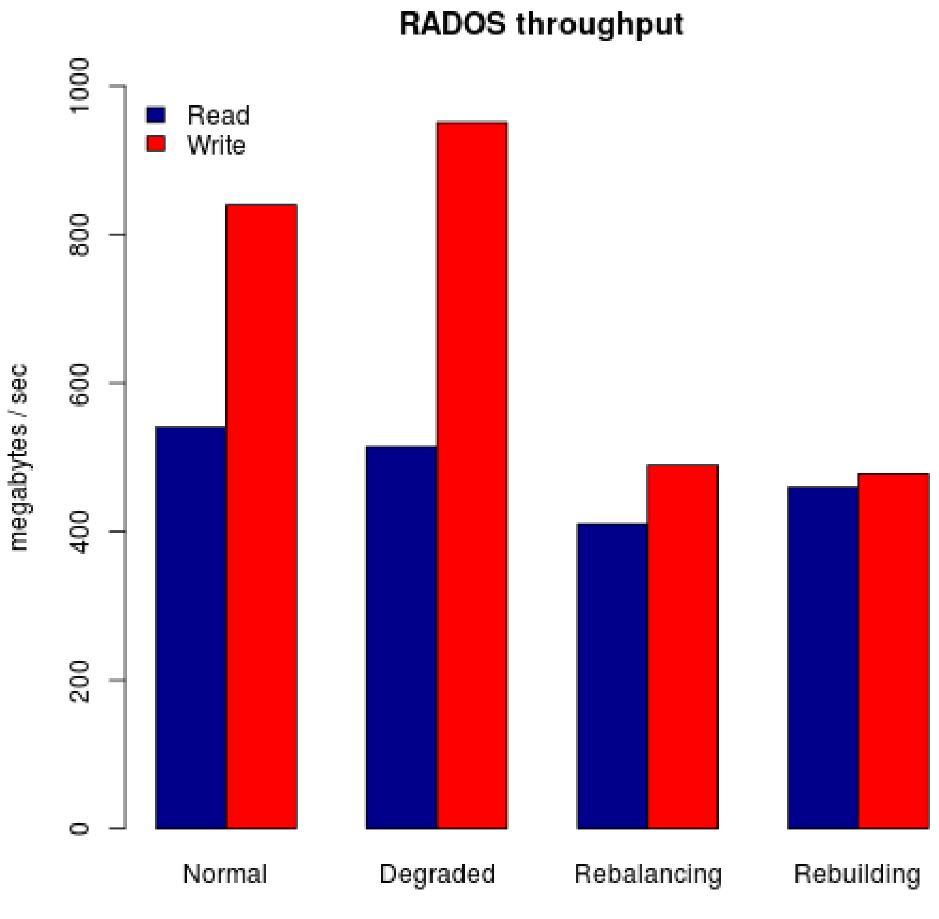


Figure 7: RADOS bench troughput test results

4.2.1 Conclusion on performance

As shown in figure 7 and table the overall write performance is better than read performance. This in contradiction to the fact that read can be distributed over multiple OSD nodes to avoid hardware limits. It is assumed that the difference is caused by controller write cache on the OSD nodes, and by the fact that the OSDs cache in memory. Once written to the OSD journal, the write is acknowledged.

The overall throughput on CephFS is lower than the throughput measured with RADOS bench. This can be contributed to the fact that CephFS comes with some overhead. Moreover, Bonnie++ measures both sequential as well as random I/O. Another limitation on RADOS bench is the fact that it only appends new data to a pool, in 4 megabyte blocks. 4 megabyte is the default object size for storing objects on the OSD filesystem. Therefore, RADOS bench is a more theoretical benchmark whereas Bonnie++ is a more real world scenario.

From figure 7 it is clear that when the Ceph cluster is under some sort of maintenance (rebalancing or rebuilding) there is an impact on performance. Write performance is affected the most. One possible explanation is memory usage on the OSD nodes. The Ceph developer documentation [2] states that memory usage on OSD nodes increases when rebuilding or rebalancing. As a result of this, there might be less memory available for caching on the OSD nodes. Another factor is disk usage: random seek behaviour during the rebuild operation decreases performance. Overall, performance is in line with hardware capabilities. Because of time constraints, this research is limited to fairly default configuration parameters. With additional tuning of parameters, performance can possibly be improved.

4.3 Ceph scaleability

Based on the results of the performance tests, a theoretical assesment of the scaleability of a Ceph setup will be made.

4.3.1 MON scaleability

The MON nodes in a Ceph cluster are lightweight daemons. No significant load on the systems running the monitor deamons was observed during any of the experiments. No foreseeable scaling issues are identified with the MON component.

4.3.2 OSD scaleability

OSD nodes can be scaled both vertical and horizontal.

Vertical scaling OSD nodes can be scaled vertical by adding more storage to the node and therefore increasing the capacity of the cluster. Each disk runs an OSD daemon, which comes with a certain overhead. In the benchmarks, two factors have been identified which can limit scalability:

– Host CPU usage: CPU usage on the hosts increases with the number of OSDs active. As seen in figure 9 already 70 % of the CPU capacity is used when the OSD node is under heavy I/O load.

– Host I/O subsystem: As identified in Figure 8 the performance decreases as the number of disks increases.

Horizontal scaling

Using RADOS bench, the individual OSDs running on one node can be benchmarked. This benchmark measures the **local** troughput to each disk.

A benchmark was executed writing 1024 megabytes in 4 megabyte blocks to the individual OSDs. This was executed in parallel over all 48 OSDs in the cluster. This resulted in an average write throughput of **64 megabytes / second** per OSD (harddisk) with a standard deviation of 8 megabytes / second.

With the write maximum of 64 megabytes / second per OSD, this comes down to a theoretical maximum of 3072 megabytes / second over all OSD daemons in the cluster. With a replication level of 2, the theoretical maximum is 1500 megabytes /second as replicas are written synchronously.

Number of OSDs	PGs	Throughput (MB /sec)	Theoretical max (MB /sec)	Overhead %
24	1200	586	768	24
36	1800	908	1152	22
48	2400	1267	1500	16

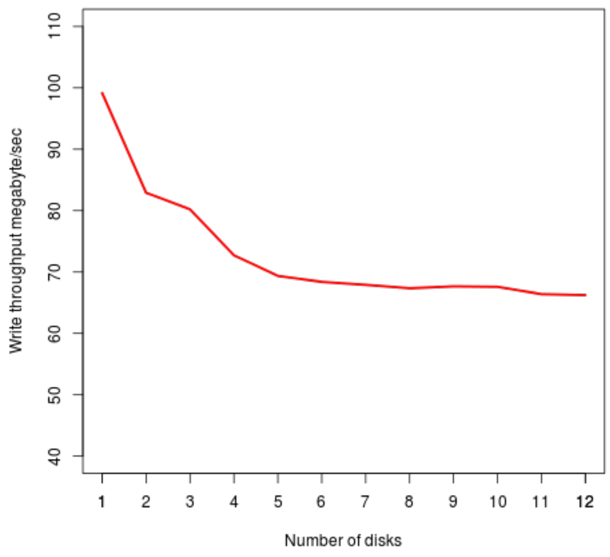


Figure 8: Troughput (1024M writes, 4M blocks) per disk on one node, in MB/sec

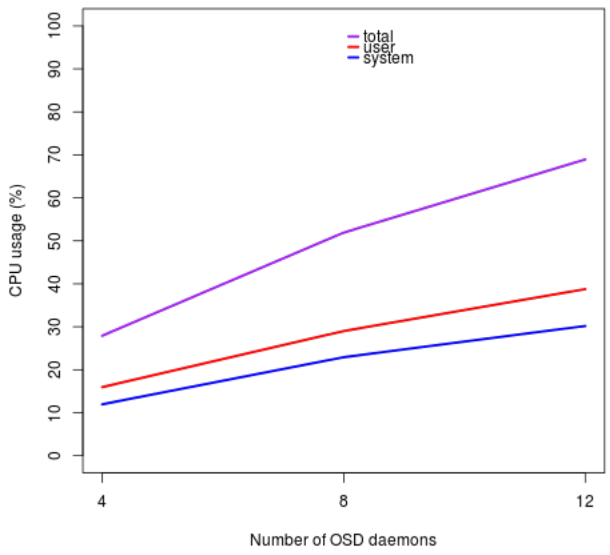


Figure 9: CPU usage over 8 CPU cores, against number of OSD daemons on one node

4.3.3 MDS scaleability

Because of the current limitations in the MDS design, the scaleability of the MDS is critical factor in the overall scaleability of CephFS. To test the metadata aspect of the filesystem, we use mdtest and Metarates as a benchmark tool. Mdstat carries out the following metadata operations using posix filesystem calls:

- Create directory
- Stat directory
- Remove directory
- Create file
- Stat file
- Remove file
- Create directory tree
- Remove directory tree

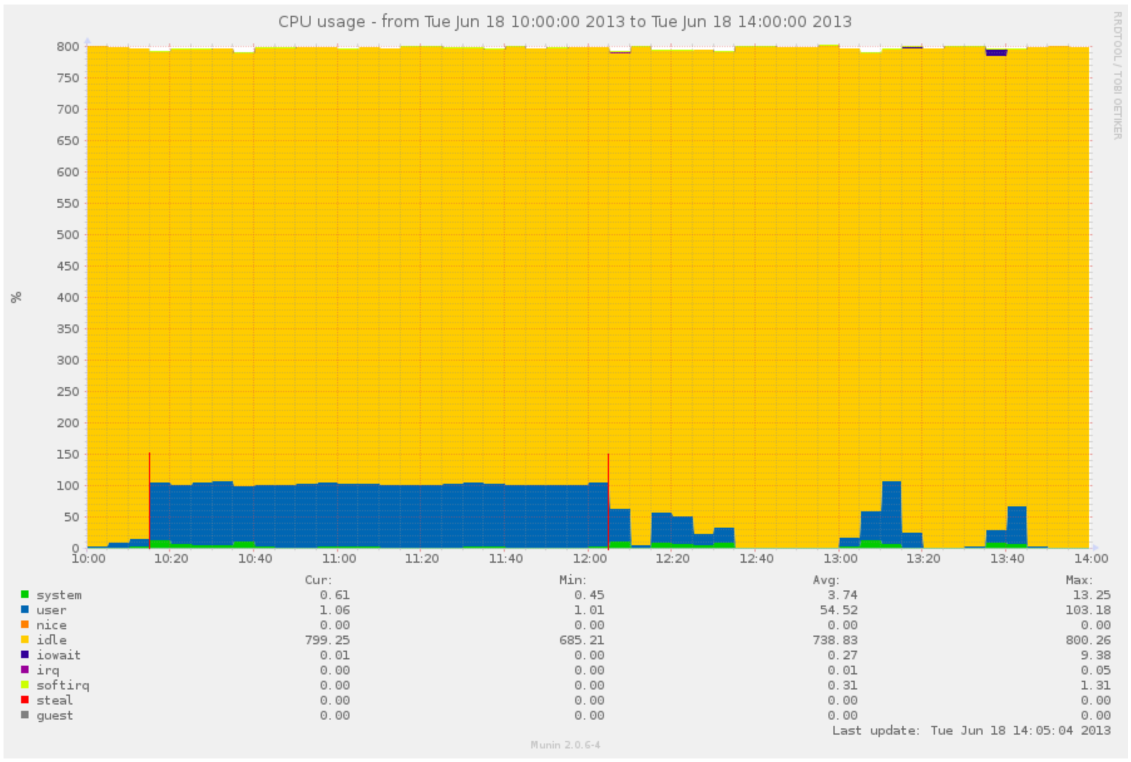
The benchmark was carried out using 1000,2000,4000,8000,16000 and 32000 files in a single directory. In some cases the benchmark runs did not finish, in that case 16000 or 32000 files in a single directory is omitted from the results. The test was executed 10 times for each number of files. The mean was calculated from the 10 samples. Both the single MDS, as well as the non-supported three active MDS setup was tested. The code for this feature is not yet stable, but still included in the distribution.

4.3.3.1 Single active MDS setup **Single threaded benchmark, single directory**

Files (qty)	Dir. create	Dir. stat	Dir. remove	File create	File stat	File read	File remove	Tree create	Tree remove
1000	1205	166733	1398	1546	168602	109516	1244	549	11
2000	982	159981	1090	578	162621	105630	954	616	3
4000	503	158215	1053	228	160224	105375	872	759	13
8000	248	159412	955	335	160820	102233	761	670	6
16000	487	159996	783	174	160346	100277	288	598	6
32000	360	162566	222	76	160684	103171	93	475	65

Table 2: POSIX filesystem operations per second with 1 active MDS in a single directory, single threaded

During the benchmarks, the test setup was monitored using Munin. While running the mdtest benchmark on the single MDS setup, the CPU usage shown in Figure 10 was observed.



	Cur:	Min:	Avg:	Max:
🟩 system	0.61	0.45	3.74	13.25
🟦 user	1.06	1.01	54.52	103.18
🟧 nice	0.00	0.00	0.00	0.00
🟨 idle	799.25	685.21	738.83	800.26
🟪 iowait	0.01	0.00	0.27	9.38
🟪 irq	0.00	0.00	0.01	0.05
🟩 softirq	0.00	0.00	0.31	1.31
🟥 steal	0.00	0.00	0.00	0.00
⬛ guest	0.00	0.00	0.00	0.00

Last update: Tue Jun 18 14:05:04 2013

Figure 10: MDS aggrated CPU usage

The CPU usage in Figure 10 is aggregated over 8 cores, with 800 % being the maximum. Analysis showed the Ceph MDS daemon process besides using multi-threading was not able to use more than one processor core at a time. The scaling limit observed in the benchmark results in table seems to be imposed by the maximum CPU usage of one processor core in the system.

Multithreaded benchmark

In order to simulate multiple clients writing to the CephFS simultaneously, 8 concurrent threads of the mdstat benchmark are executed simultaneously using mpirun from Open MPI [7].

Files (qty)	Dir. create	Dir. stat	Dir. remove	File create	File stat	File read	File remove	Tree create	Tree remove
1000	1037	540450	1399	1339	317928	90025	1257	492	10
2000	824	180429	1123	610	97405	89590	959	534	3
4000	678	158984	996	387	138067	134101	776	570	4
8000	433	145529	772	467	150105	130510	707	446	5
16000	588	141363	699	224	139629	124187	291	590	9

Table 3: POSIX filesystem operations per second with 1 active MDS in a single directory, multi- threaded

Multithreaded benchmark, multiple directory

Files (qty)	Dir. create	Dir. stat	Dir. remove	File create	File stat	File read	File remove	Tree create	Tree remove
1000	2526	94663	1990	3704	95429	84994	2638	25	4
2000	2235	93266	2185	2003	104394	86049	1886	31	3
4000	1798	92626	2265	1253	85862	79982	2272	17	2
8000	1095	90800	2557	1621	85670	77292	1801	20	4
16000	1237	87280	2407	406	85731	78112	1211	17	2
32000	1594	87428	971	483	93549	89105	1019	17	5

Table 4: POSIX filesystem operations per second with 1 active MDS in multiple directories, multi- threaded

4.3.3.2 Three active MDS setup **Single threaded benchmark, single directory**

This benchmark measures the basic scenario of one client using the CephFS in a single directory. One thread of the mdstat benchmark is executed.

Files (qty)	Dir. create	Dir. stat	Dir. remove	File create	File stat	File read	File remove	Tree create	Tree remove
1000	1221	170222	1404	1504	173180	111041	1137	590	29
2000	1061	170559	1122	1010	171485	109975	897	624	40
4000	961	168690	905	879	170006	109600	785	500	12
8000	696	165873	714	744	166670	103785	518	721	73
16000	549	163780	400	375	148981	103626	175	779	166

Table 5: POSIX filesystem operations per second with 3 active MDS in a single directory, single threaded

Multithreaded benchmark, single directory

In order to simulate multiple clients using the CephFS simultaneously in a single directory, 8 concurrent threads of the mdstat benchmark are executed simultaneously using mpirun from the C multiprocessing framework.

Files (qty)	Dir. create	Dir. stat	Dir. remove	File create	File stat	File read	File remove	Tree create	Tree remove
1000	1136	315469	229	238	328894	26341	365	144	84
2000	1017	408722	1167	1067	172327	123935	857	728	51
4000	812	462008	767	838	242341	199597	802	658	41
8000	728	284743	675	607	244787	189827	639	788	39
16000	554	250655	553	397	264596	206027	193	842	120

Table 6: POSIX filesystem operations per second with 3 active MDS in a single directory, multi-threaded

Multithreaded benchmark, multiple directory

The mdstat benchmark was also executed using a separate directory for each thread. This tests a more realistic scenario in which several users are performing I/O operations (causing metadata operations) in different directory's at the same time.

Files (qty)	Dir. create	Dir. stat	Dir. remove	File create	File stat	File read	File remove	Tree create	Tree remove
1000	3470	140241	2755	3527	124413	111687	2410	27	4
2000	2692	429770	2879	2474	201078	120170	2807	37	6
4000	2456	134531	2417	2143	169385	128842	2337	26	3
8000	2116	144180	2475	1507	145683	120112	2712	19	4
16000	1336	91855	2094	1316	214681	160614	1503	19	6
32000	1248	128293	1110	646	88410	141731	796	18	6

Table 7: POSIX filesystem operations per second with 3 active MDS in multiple directory's, multi-threaded

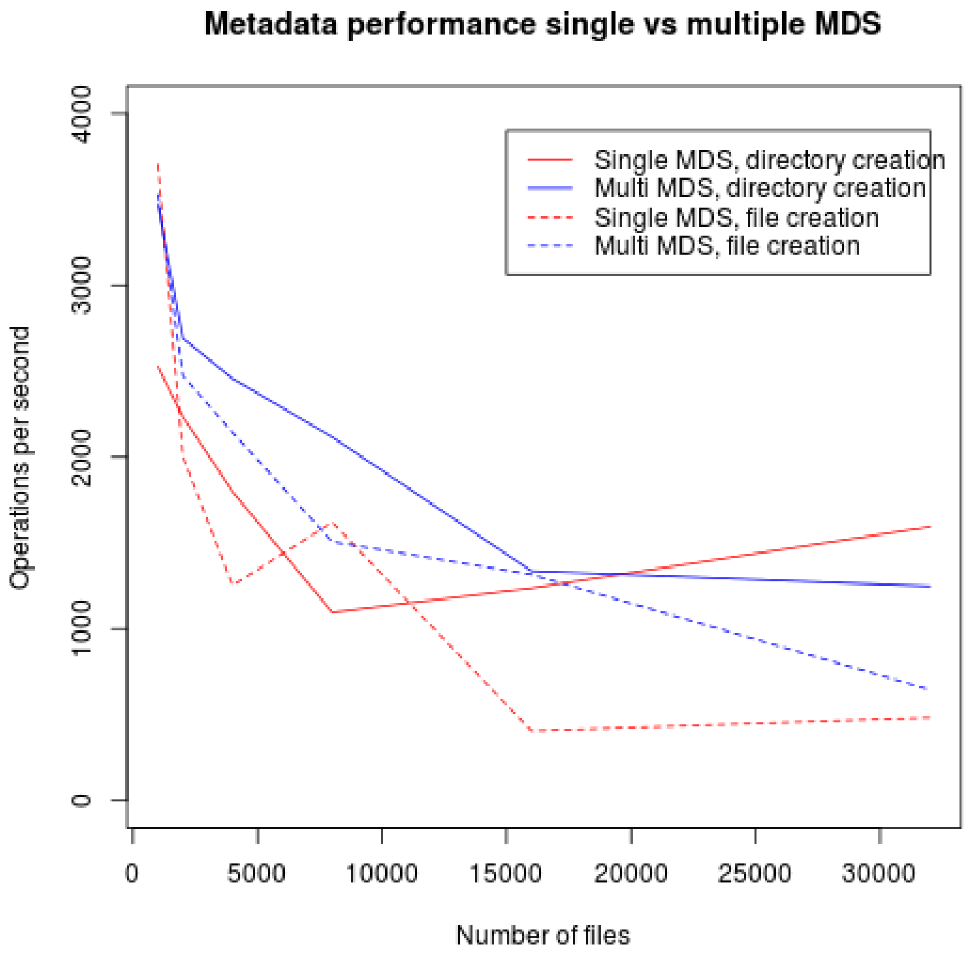


Figure 11: MDS performance comparison, single vs multiple active MDS

4.3.3.3 Comparison RBD with XFS To establish which component is causing the metadata bottleneck (the MDS or underlying OSD's) the same benchmark is executed on a XFS formatted volume on top of a Rados Block Device (RBD).

Files (qty)	Dir. create	Dir. remove	File create	File read	File remove	Tree create	Tree remove
1000	26420	25499	18210	1236646	25511	13559	924
2000	25632	25225	18440	1252662	25522	13752	482
4000	24116	24785	18932	1021544	26331	12768	279
8000	23925	24501	20565	1236924	25604	11054	133
16000	23851	24756	20811	1278445	23892	11569	72
32000	23190	24490	20671	1253322	23963	12775	36

Table 8: Metadata benchmark XFS on RBD

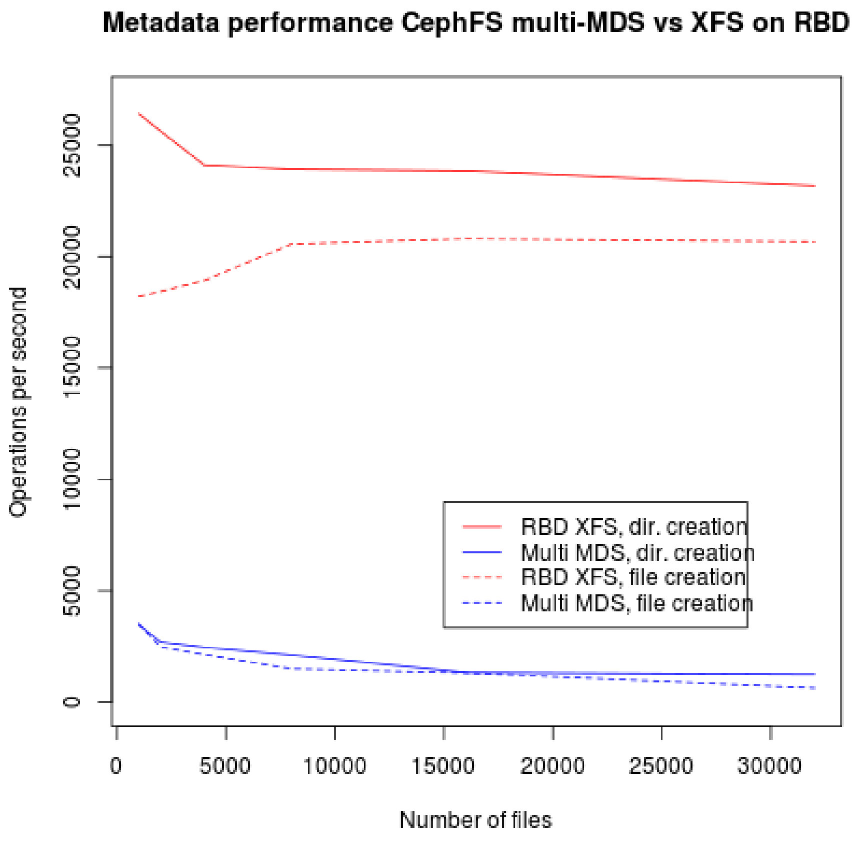


Figure 12: CepFS multi-MDS vs XFS on RBD metadata performance comparison, single directory

4.3.4 Conclusion on scalability

The OSD benchmark results show a near linear horizontal scaling of the OSDs. The troughput measured in benchmarks is close to the theoretical troughput, with a 15% to 20% overhead. This overhead is in the network stack and the replica processing.

Vertical scaling is more limited by the OSD host cpu power, and I/O subsystem. It is outside the scope of this research to investigate tuning options to improve vertical scalability. Possible options are using jumbo frames to lower the number of network interrupt requests and therefore lower the CPU usage The Dell PERC-raid controller used in this test setup, did not support attaching the harddisks without using the RAID capabilities (JBOD-mode). Therefore, 12 RAID-0 arrays were used to simulate this behaviour. This might have added some overhead which caused the effect demonstrated in figure 8. Disk, and disk controller throughput is dependent on controller model, file system and RAID mode as concluded by Mark Nelson [8].

The scalability of the MDS is limited by the fact that threads are not balanced across CPU cores as shown in figure 10, and by the fact that scaling horizontally is not yet considered stable. With todays multicore CPUs true multithreading would have given the single MDS setup more room to scale and make it more usable considering it's current single-MDS limitation. The CPU effect as shown in figure 10 is in fact confirmed in a mailing list discussion by Greg Farnum [5], one of the lead developers of Ceph.
The metadata tests with the multiple active MDS show that it scales horizontally. However, the performance is much lower than XFS on RBD as shown in figure 12. This finding contradicts the results found by Sage Weil [16] in his research which was the start of the Ceph project. One possible explanation is scale, as this research was limited in the amount of available hardware and time. Weil tested up to 128 MDS, while in this research no more than 3 MDS were tested. Also, the tests were conducted using the default settings. There may have been some settings which enable scaling options that were disabled by default for data safety reasons. Documentation on the MDS is still limited.

5 Conclusion

Overall, the experiments proved that the RADOS object store is stable and ready for production. CephFS can be used in production, however at the time of writing running multiple MDS is not advisable. Stability problems encountered during the experiments, with multiple MDS and with MDS failover, exaggerated this point. Apart from the MDS component, Ceph meets the stability requirements at SURF-sara. However, because of the fact that the filesystem functionality is such a large part in the overall case at SURFsara the first subquestion must be concluded negative. At the moment of writing, CephFS is not yet stable enough to be used in a large scale production environment at SURFsara.

The scalability experiments showed in accordance with the theory that the RADOS object store scales near lineair with the number of OSD daemons added to the cluster. Up to 48 OSD daemons, the cluster matched the performance that can be expected from the hardware taking the replication level into account. The full-mesh principle between client and OSDs worked as expected, as shown by the fact that by using two clients simultaneously the 10GbE bandwidth limit was exceeded.
The MDS scalability showed a different picture in the scaling experiments. With the Lisa system at SURFsara [12] currently holding 136 million inodes, and the Dutch national supercomputer holding 50 million inodes large amounts of files are to be expected in the environment at SURFsara. Whether CephFS will be suitable for use in production depends largely on the number of metadata operations that will be performed on the filesystem. This leads to the answer to the second subquestion: the number of metadata operations is limited at the figures shown in table 7. The scale of this research was not large enough to reach scalability limits in terms of capacity. However, there is no indication of such a limit observed in the experiments conducted.

The stability experiments showed that the RADOS object store is highly resilient against component failures. No problems should be expected by harddisk failures, or even entire nodes failing. Expanding capacity is handled automatically by the rebalancing of PGs amongst OSDs. OSDs can be taken out of the cluster daemon by daemon, minimizing the amount of data that needs rebuilding. This gives granular control over the process. The same mechanism can be used to replace disks in batch. Failed disks do not need to be replaced right away as long as spare disks are available to be introduced to the cluster once a disk fails.
In this particular test setup, replacing disks imposed the need for a reboot of the entire node because the RAID-0 array needs to be created. This can be circumvented by installing management tools for the RAID controller, so the RAID-0 array can be created without rebooting.
The management tools included in the Ceph distribution are suitable for integrat-

ing into various tooling. The included management tools support output in both structured (JSON) as well as unstructured human-readable (plain text) format. The answer to the third subquestion is that CephFS meets the maintenance requirements at SURFsara in terms of low-maintenance and integration into tooling.

Is the current version of CephFS (0.61.3) production-ready for use as a distributed filesystem in a multi-petabyte environment, in terms of stability, scalability, performance and manageability?

Summarizing the answers to the individual subquestions, the answer to the research question must be concluded that CephFS 0.61.3 is not yet suitable for production use at SURFsara. The main arguments for this conclusion ar the stability problems found in the MDS, and the scalabiilty problems found with the MDS. With a dedicated development team now focussing on the development of CephFS [3], this however may change in the near future. In general Ceph is a promising concept, with a large number of possible uses cases due to it's modular design offering object, file and block storage in a single solution.

6 Limitations and futher work

The scale of this research is limited by both time (4 weeks) as well as availability of equipment. Further research on a larger scale could draw a better picture on capacity scaling.
Due to the limited timeframe, Ceph has been tested using a standard configuration. Tweaking various configuration parameters of both the operating system and Ceph could possibly improve performance and/or scalability. Furthermore is is advisable to repeat this research in the future, as soon as a new major version with improvements to the MDS is released.

7 References

[1] Amazon, Amazon S3, http://aws.amazon.com/s3/

[2] Ceph development team, Ceph documentation, http://ceph.com/docs/master/

[3] Ceph development team, CephFS MDS Status Discussion, http://ceph.com/dev-notes/cephfs-mds-status-discussion/

[4] Coker,R., Bonnie++, http://www.coker.com.au/bonnie++/

[5] Farnum, G., Ceph MDS threading discussion, http://www.mail-archive.com/ceph-devel@vger.kernel.org/msg09076.html

[6] GNU coreutils, dd, http://www.gnu.org/software/coreutils/manual/coreutils.html

[7] Open MPI, Open MPI: Open Source High Performance Computing, http://www.open-mpi.org/

[8] Nelson, M., Ceph performance part 1: disk controller throughput, http://ceph.com/community/ceph-performance-part-1-disk-controller-write-throughput/

[9] Olson, C. and Miller, E., Secure Capabilities for a Petabyte-Scale Object-Based Distributed File System, 2005, University of California, Santa-Cruz, 2005

[10] SURFsara, Cartesius, https://www.surfsara.nl/systems/cartesius

[11] SURFsara, HPC Cloud, https://www.surfsara.nl/systems/hpc-cloud

[12] SURFsara, The Lisa system, https://www.surfsara.nl/systems/lisa

[13] Unknown author, hdparm, http://sourceforge.net/projects/hdparm/

[14] Torvalds, L., fs-metadata, https://kernel.googlesource.com/pub/scm/utils/cpu/mce/mce-test/+/master/cases/stress/hwpoison/tools/fs-metadata/fs-metadata.sh

[15] Weil, S., RADOS bench, http://ceph.com/w/index.php?title=Benchmark

[16] Weil, S., Ceph: Reliable, Scalable and High-Performance Distributed Storage, University of California, Santa-Cruz, 2007

[17] Wiers, D., dstat, http://dag.wieers.com/home-made/dstat/