Ceph ~30PB Test Report

Dan van der Ster (CERN IT-DSS), Herve Rousseau (CERN IT-DSS)

Abstract

During March 2015 CERN IT-DSS provisioned nearly 30 petabytes of rotational disk storage for a 2 week Ceph test. For the past year, CERN IT has operated a 3 petabyte cluster in production for OpenStack and in test for various R&D projects. The present test therefore represents a 10-fold increase in scale versus known deployments[1]. This test consisted of adding 7200 4TB drives on 150 storage nodes to an existing (but much smaller) test cluster. The cluster was benchmarked and various failure conditions were simulated. In summary, the test showed that operating such a cluster is feasible, though we present some caveats and suggestions for improved scalability.

1. Motivation

CERN IT-DSS currently operates around 200PB of disk storage (usually with 2 copies, thus 100PB usable) and the LHC experiments continue to produce an estimated 1-2PB more data per month.

CERN's present disk storage system is EOS, a system build upon Xrootd and featuring an in-memory file location catalog. The large size of this catalog results in a few scalability limitations: each EOS cluster is limited to a few hundred million files (consuming hundreds of GB of memory); the catalog must be loaded from persistent storage after a restart, taking tens of minutes; the catalog is a single point of failure, though backup catalogs can take over after an outage.

For long term data archival, CERN developed and operates CASTOR, a hierarchical storage management system having its file location catalog in Oracle. The long term strategy for CERN data storage is to reduce the role of its disk layer to act only as a buffer between EOS/users and tapes. The size of this buffer is expected to be around 10PB usable.

Both the EOS and CASTOR use-cases present areas where new developments in consistent-hashing object stores can help solve our scalability limitations. Based on our successful experience with Ceph and OpenStack, we would like to understand if Ceph can help satisfy these growing storage requirements.

[1] We are not aware of public announcements of production Ceph deployments larger than our 3PB instance.

2. Hardware Description

This test used hardware which was procured for the EOS and CASTOR services -- the primary goal of EOS/CASTOR servers is to minimise cost per TB. We received 150 identical servers as described in the following table:

CPUs	2x Xeon E5-2650 v2 @ 2.6GHz (HT-enabled: 32 threads)
RAM	64GB
Network	Intel 82599 10 GbE NIC
System drives	2x 2TB HGST HUS724020AL
OSD drives	48x 4TB HGST HMS5C4040BL
SAS Controller	LSI 9207-8e HBA with 2x SFF-8088 external ports
SAS Chasses	2x 24-bay chassis (either Promise VTrak J830s or Xyratex)

From the start we acknowledge that these machines are not within the recommended Ceph hardware spec, notably the RAM/TB ratio is far off the suggested 1GB/TB. However, this hardware is demonstrated to work well in our existing scale-out storage solution EOS.

3. Installation and Deployment

These new servers were added to our existing pre-production cluster having 3 ceph-mons running Ceph firefly 0.80.8. All our machines run Scientific Linux 6.6.

3.1 Puppet Installation

Our installation procedure uses puppet to scan externally attached disks and call ceph-disk prepare on empty drives. Each drive is therefore prepared with a 20GB journal partition and 4TB-20GB data partition.

Following ceph-disk prepare, the drives are activated via udev. “Activation”, in this case, creates the OSD, adds the OSD keyring to the cluster, and then starts the OSD process. Our ceph.conf has *crush update location on start = false*, so the new OSDs are not assigned to a data pool at creation time.

3.1.1 Issues observed during Puppet Installation (attempt #1)

Our first attempt at deploying these 150 machines was to take a naive approach: we let Puppet start working at 18:00 on day 0 then let the ceph-disk prepare/activations happen over night. This approach pointed out several issues:

1. Our puppet exec for `ceph-disk prepare` had the “unless” condition as roughly `ceph-disk list | grep <device> | grep ceph`. This lead to many ceph-disk processes scanning the external drives simulateously, and eventually many of these processes hung. In general, `ceph-disk list` is quite slow on these 48-disk servers (compare with our 24-OSD machines, which never showed an issue).
2. The first couple thousand OSDs were created rather quickly, but once the number of OSDs exceeded a few thousand we started observing single OSD activations were taking a very long time.
3. Similarly, the first couple thousand OSD processes used little RAM, but once the number of OSDs was in the many thousands, OSD processes were consuming ~3GB.
4. During the initial installation, we observed ceph-mon LevelDBs exploding to >25GB and several monitor elections. One monitor which was manually compacted took more than 4 hours to synchronize.

3.2 *osdmap* Scalability

Our first attempt at deploying this hardware with our existing Puppet manifests and ceph configuration was a failure. We formed a few theories:

1. The `ceph-mons` were overloaded with too many OSD creation transactions. We added SSDs to these mons and upgraded the cluster to Giant, then the Hammer RC. Giant and later releases allow LevelDB reads during writes, so we expected these changes to help the cluster responsiveness.
2. The *osdmap* caching feature of `ceph-osd` processes was increasing memory consumption. Hints in this direction came as we saw the osdmap *dedup* function consuming lots of CPU with perf top.

When the cluster had 7200 OSDs, we downloaded the map and found that is measures 4MB in size. By default, the `ceph-osd` caches 500 previous osdmaps, it was clear that even with deduplication the map is consuming around 2GB of extra memory per `ceph-osd` daemon. After tuning this cache size, we concluded with the following configuration, needed on all `ceph-mon` and `ceph-osd` processes.

```
[global]
  osd map message max = 10

[osd]
  osd map cache size = 20
  osd map max advance = 10
  osd map share max epochs = 10
  osd pg epoch persisted max stale = 10
```

Having this configuration, ceph-osd daemons generally stay under 500 MB memory used, even with 7200 OSDs in the cluster.

3.4 Puppet installation (attempt #2)

Our second attempt to install the cluster used the aforementioned osdmap cache config settings and a couple of tricks to prevent too many rapid updates to the osdmap. These included:

1. We set the flags *noin* and *noup* in order to prevent every OSD boot from changing the osdmap. After all OSDs were installed, we unset those flags and then all OSDs were marked up and in within one or two changed osdmaps.
2. We set *crush update location = false* in order to prevent so many osdmap and crushmap changes. For running such a large cluster in production, we would develop a tool which manages the crushmap directly instead of relying on the *ceph update location* feature.

3.5 Pool Create/Delete Testing

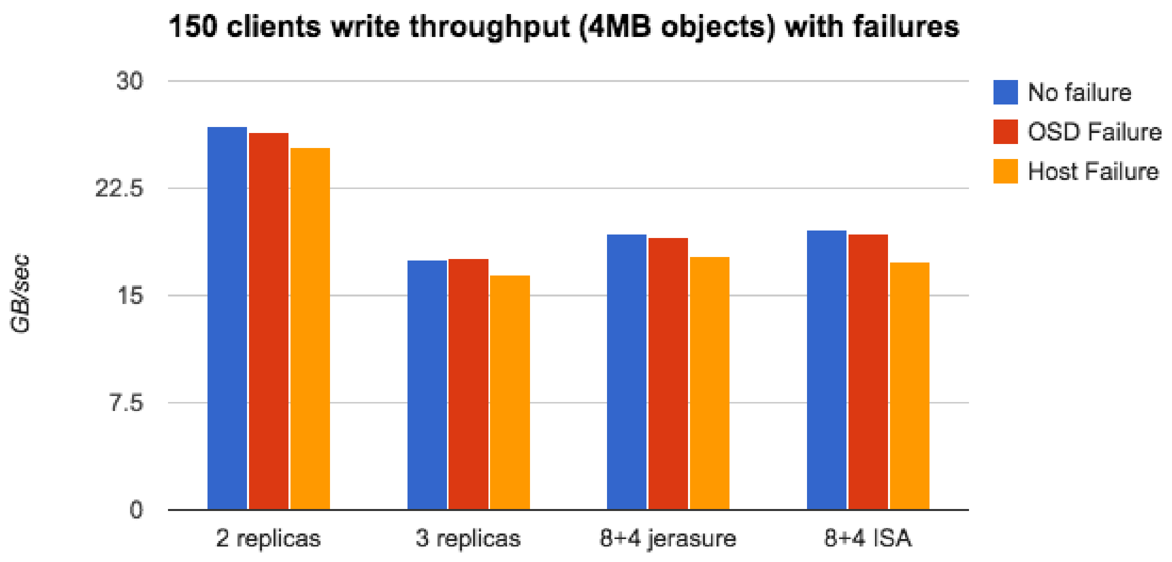
Following the successful installation of the cluster, we performed some initial basic pool create/delete testing. In these early tests we found that pool creations behaved normally, but pool deletions had problems.

On a cluster with 7200 OSDs, a data pool requires around 240,000 PGs to achieve the recommended 100 PGs per OSD. We tested with 65536 PGs. Deleting such a pool caused several minutes of monitor inaccessibility and elections. It seems that the pool deletion process puts the monitor in a tight loop and in our case caused a 10 minute outage. This was confirmed with repeated pool create/delete tests.

4. Performance Testing

We performed extensive bandwidth and IOPS testing to measure the performance of the cluster. Details are included in Appendix A.

The figure below shows a typical result. Bandwidth from many clients shows that write throughput scales linearly with the replication used. Also, performance during single OSD or single host failure does not dramatically decrease performance. It was not confirmed if the network switches were a bottleneck in this test.



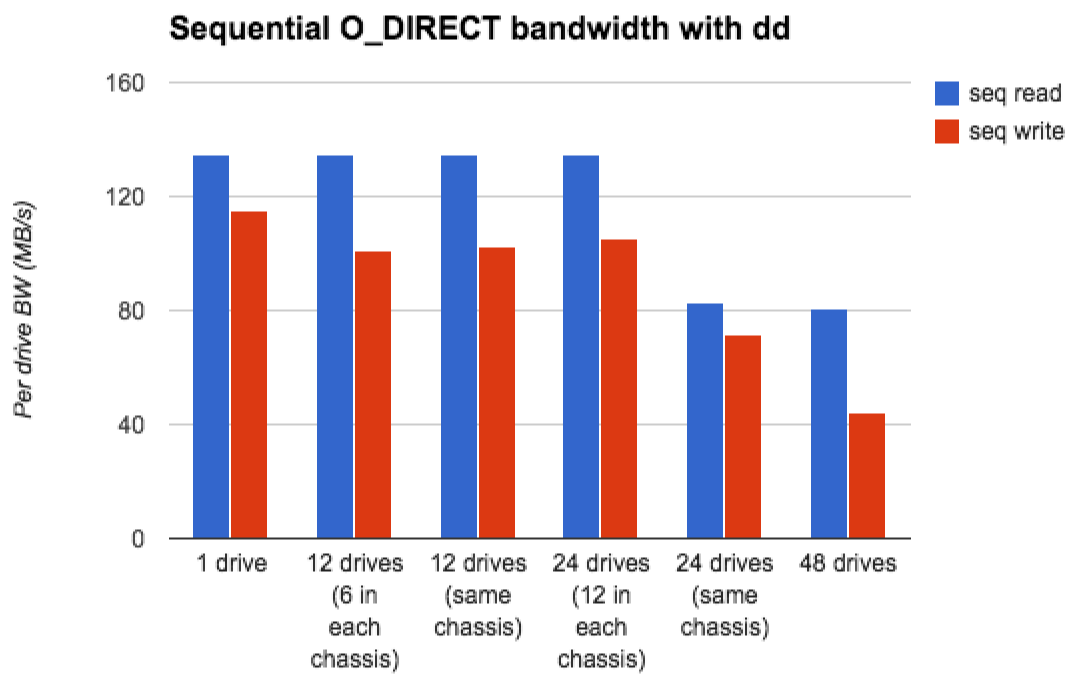
5. Summary

This report summarized a ~30PB test using 150 servers with 48x4TB drives each. In the test we succeeded to deploy the cluster and run performance tests, but several configuration changes were needed to fit the `ceph-osd` daemons within available memory. In conclusion we suggest the following points be followed up with the Ceph developers:

1. The osdmap is a scalability limitation. With 7200 OSDs it is 4MB in size, a size which causes various problems:
    a. While the osdmap is changing frequently, e.g. during cluster deployment, the `ceph-osd` spends a lot time in dedup function iterating over the ever-growing osdmap entries.
    b. The osdmap caching feature on the `ceph-osd` process consumes up to 4MB*500=2GB of memory, and osdmap dedup is not very effective in decreasing this requirement.
    c. Further to 1(b), we find that repeatedly caching 2GB of osdmaps on all OSDs of a local server is a waste of RAM (i.e. it would be better used as page cache). Some way to share osdmap cache between local OSDs would be helpful.
    d. As the osdmap becomes large, we had to employ tricks to prevent too many changes to the osdmap, including setting the *noup* and *noin* flags while deploying the cluster.
2. Pool creation uses an intermediate PG state “creating” which seems to keep the monitors responsive during creation of large pools (e.g. 65536). Pool deletion does not have an analogous deleting state, and in our tests `ceph-mon` processes were busy and unresponsive for up to 10 minutes while deleting a large pool.

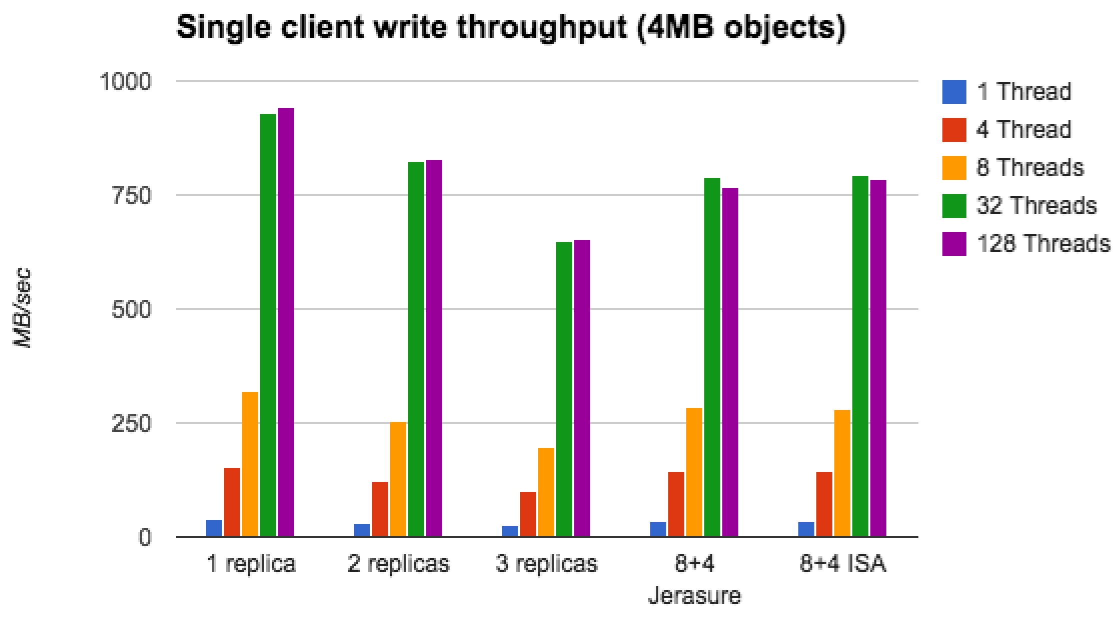
A. Appendix: Performance Plots

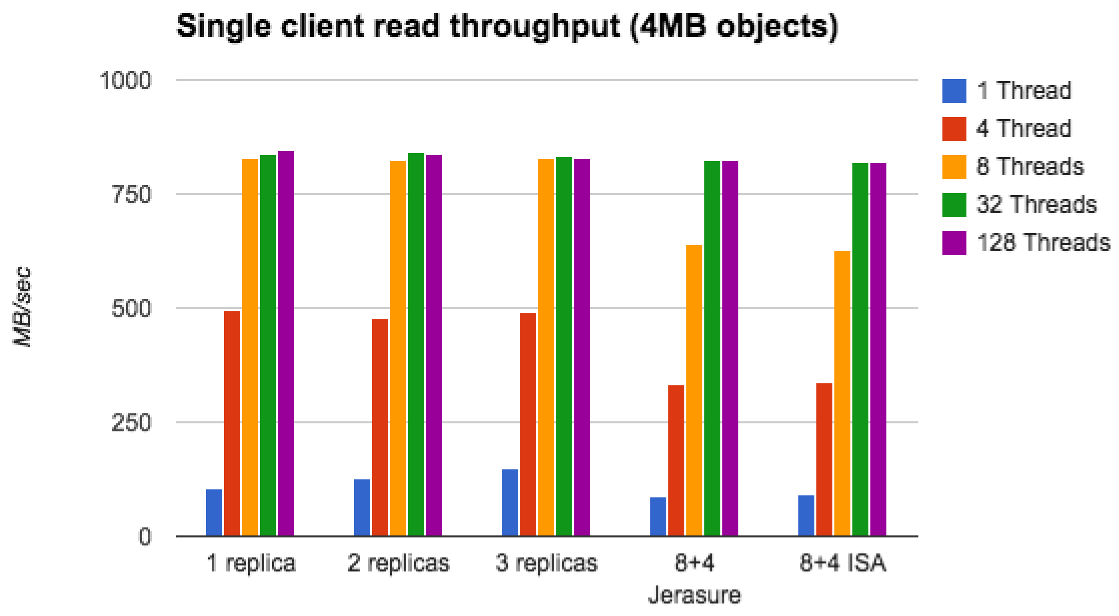
We first report the baseline sequential performance of this hardware. In the figure below we used dd with the O_DIRECT flag to measure sequential reading and writing to drives, with increasing concurrency. At the maximum, we observed a read throughput of 135MB/s and write throughput of 110MB/s (IO to a single drive). When all drives are used concurrently, we observe up to 80MB/s reads (per drive) and 43MB/s writes (per drive).



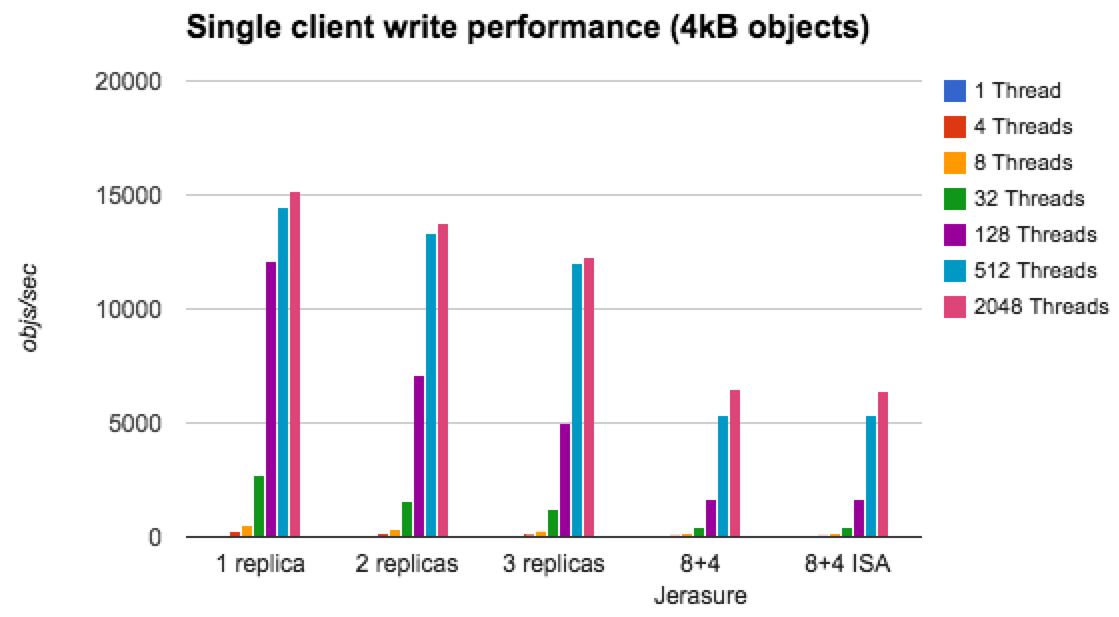
With this baseline we can calculate a maximum OSD write throughput of the entire cluster (assuming co-located OSD journals, single replica, and no other bottlenecks): 150 * 48 drives * 43MB/s / 2 =~ 150GB/s.

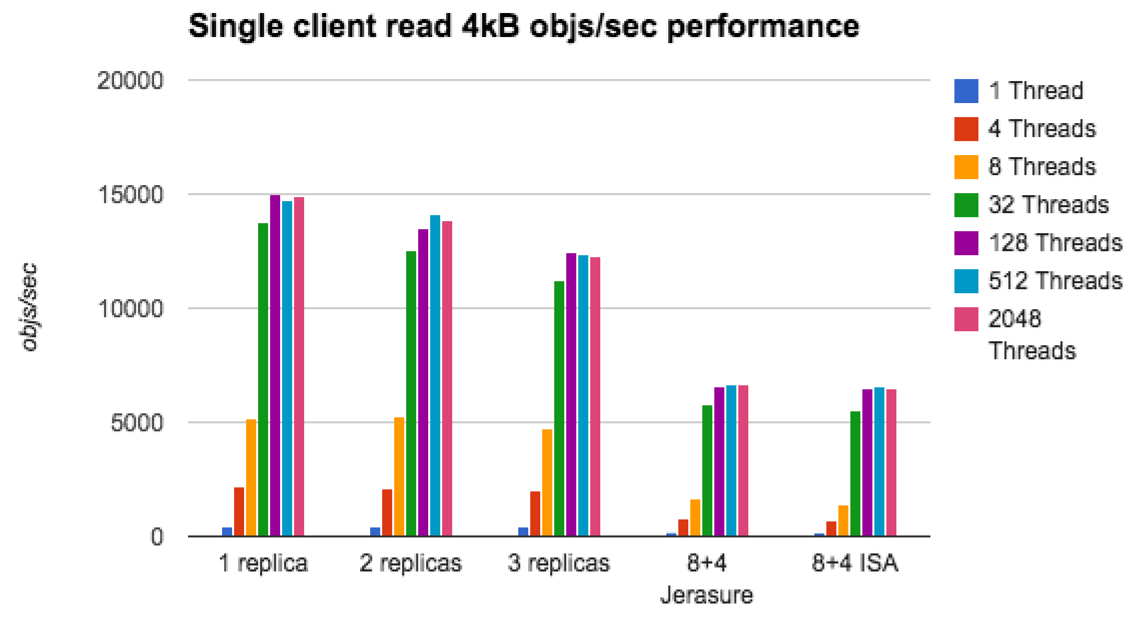
a. Single Client IO



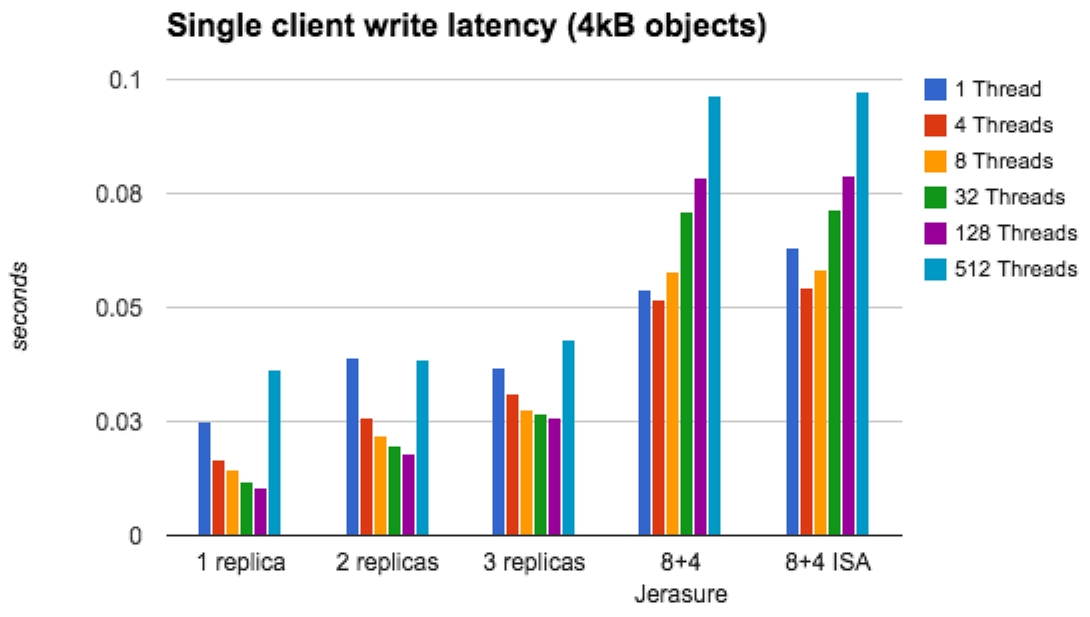


b. Single Client IOPS

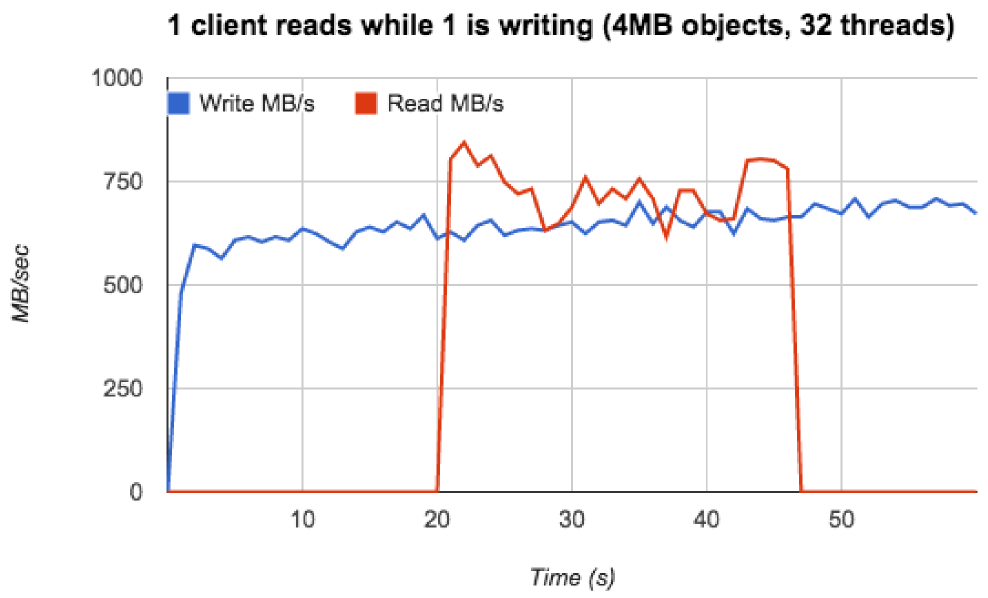




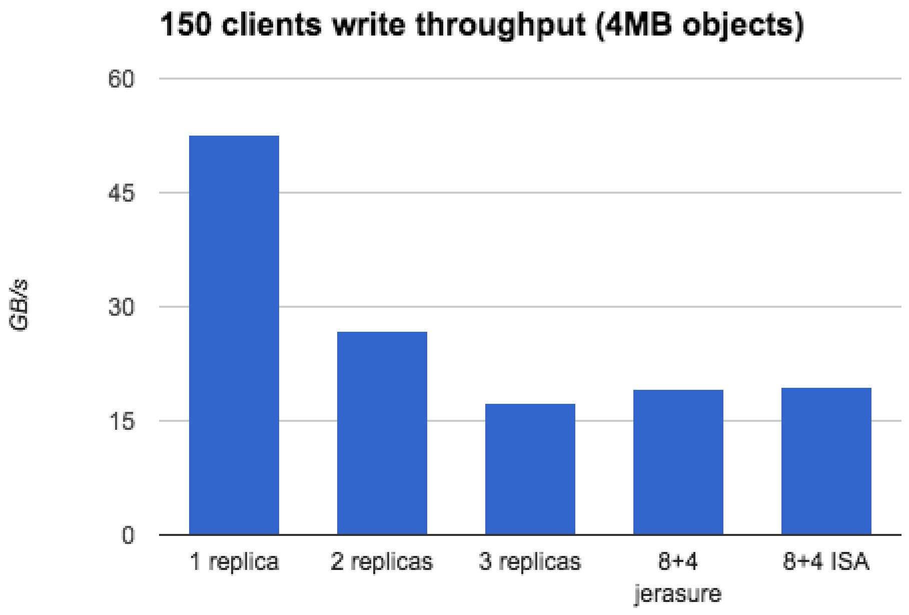
c. Single Client Latency



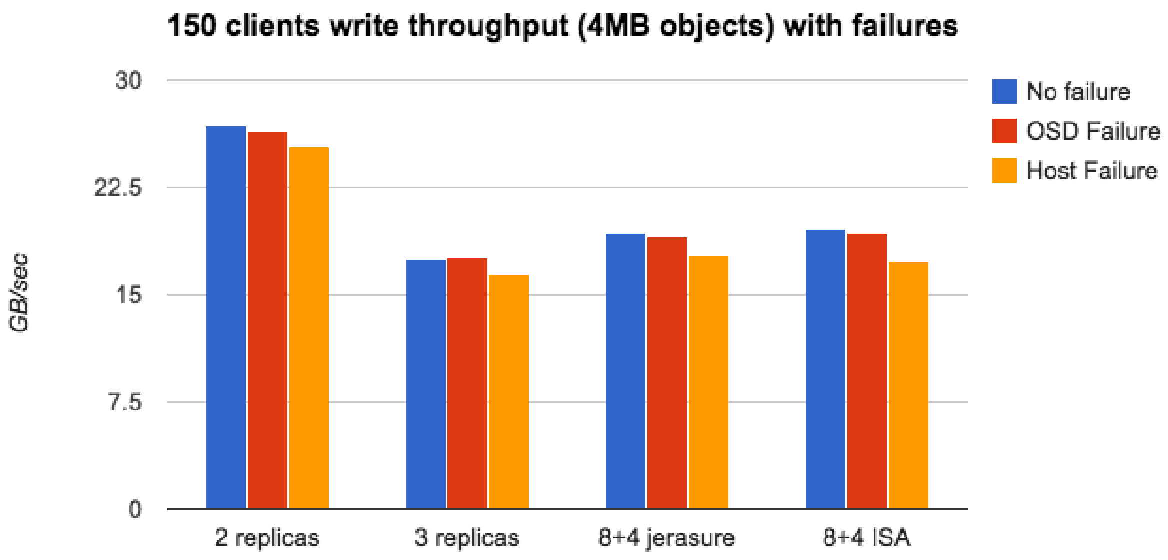
d. Mixed read/write



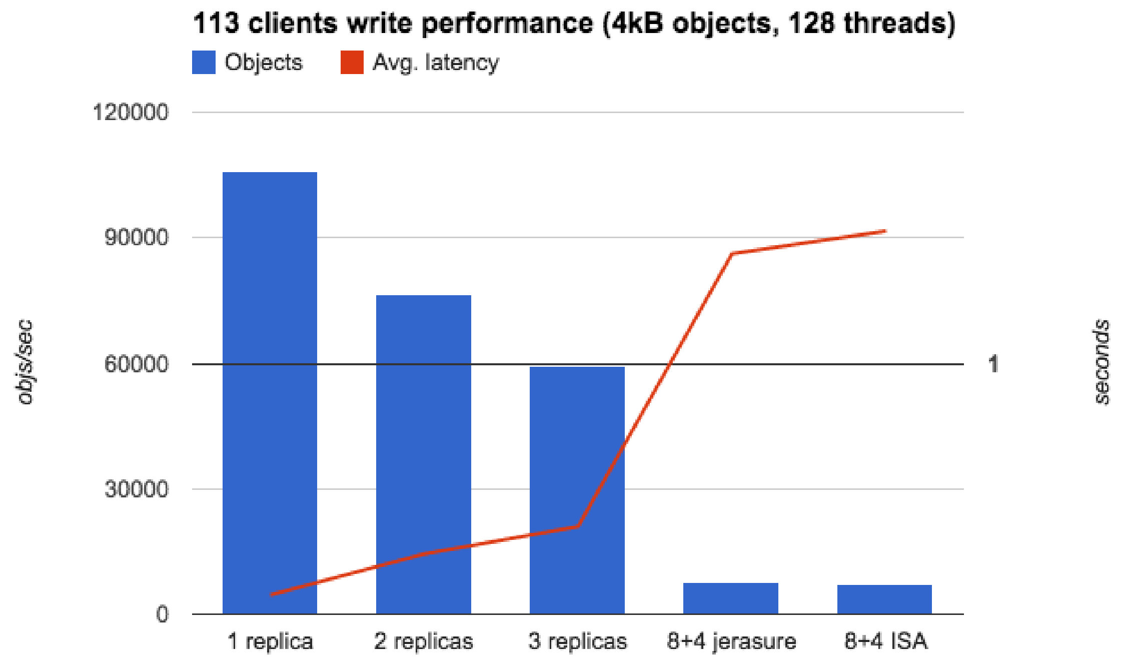
e. Many Clients Writing



f. Many Clients Failure



g. Many Clients IOPS


113 clients write performance (4kB objects, 128 threads)
Objects
Avg. latency
objs/sec
seconds
1 replica
2 replicas
3 replicas
8+4 jerasure
8+4 ISA