POSIX and Object Distributed Storage Systems Performance Comparison Studies With Real-Life Scenarios in an Experimental Data Taking Context Leveraging OpenStack Swift & Ceph

To cite this article: M D Poat *et al* 2015 *J. Phys.: Conf. Ser.* **664** 042031

View the article online for updates and enhancements.

POSIX and Object Distributed Storage Systems
Performance Comparison Studies With Real-Life Scenarios in an Experimental Data Taking Context Leveraging OpenStack Swift & Ceph

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

E-mail: mpoat@bnl.gov

Abstract. The STAR online computing infrastructure has become an intensive dynamic system used for first-hand data collection and analysis resulting in a dense collection of data output. As we have transitioned to our current state, inefficient, limited storage systems have become an impediment to fast feedback to online shift crews. Motivation for a centrally accessible, scalable and redundant distributed storage system had become a necessity in this environment. OpenStack Swift Object Storage and Ceph Object Storage are two eye-opening technologies as community use and development have led to success elsewhere. In this contribution, OpenStack Swift and Ceph have been put to the test with single and parallel I/O tests, emulating real world scenarios for data processing and workflows. The Ceph file system storage, offering a POSIX compliant file system mounted similarly to an NFS share was of particular interest as it aligned with our requirements and was retained as our solution. I/O performance tests were run against the Ceph POSIX file system and have presented surprising results indicating true potential for fast I/O and reliability. STAR's online compute farm historical use has been for job submission and first hand data analysis. The goal of reusing the online compute farm to maintain a storage cluster and job submission will be an efficient use of the current infrastructure.

1. Introduction
The Solenoidal Tracker at RHIC (STAR) experiment at the Relativistic Heavy Ion Collider (RHIC) located at Brookhaven National Laboratory is an international collaboration that has been collecting vast amounts of data for the past 15 years. The online compute farm at STAR is made up of hundreds of nodes, 30 of which are Scientific Linux nodes each containing six 2 TB drives, 8 cores, 16 GB of RAM, and two 1 Gb network interfaces. One of the two network interfaces is attached to a public network and the other a private backbone network. This cluster has been typically used for first hand data analysis at the experiment such as the processing and sampling of the data taken "live" for quality control, assurance and online fast calibration. In such workflows, a set of processes may need input from an event pool of cached data, process it and produce output that will then be consumed by other processes. The widely distributed nature of the tasks calls for storage available from many processing locations as far as the solution is deemed reliable. STAR recognized that the need for a large local storage system and the reuse of online hardware, to create a file storage system for users, would be an

efficient use of the current computing infrastructure. Investigation of many distributed storage aggregation solutions have taken place over recent years, analysis on the Hadoop File System is intuitive by nature however a simplistic POSIX interface is lacking [1]. A file storage system containing redundancy is a top necessity as hardware failure is inevitable. A storage system that can handle power outages and crashes with no data loss is desirable while a POSIX compliant interface offering simple, familiar user interaction would be preferred. OpenStack is an open source cloud system that controls large pools of compute, storage, and networking resources throughout a data center [2]. OpenStack supports multiple areas of computing infrastructure with its main goal offering infrastructure as a service. Ceph is a unified, distributed storage system designed to provide excellent performance, reliability and scalability [3]. Ceph is strictly storage driven devoting all efforts to its object store, block store and its file system.

In this study we aim to compare the architecture, performance, reliability and interaction of the two object storage systems OpenStack Swift and Ceph along with investigation of the POSIX file system Ceph has to offer.

2. Architecture and Configuration

Considering reliability a top priority for STAR's online distributed storage system, configuration for both OpenStack Swift and Ceph were set to data replication three whilst using long term stable release versions of both systems (OpenStack Havana & Ceph Firefly).

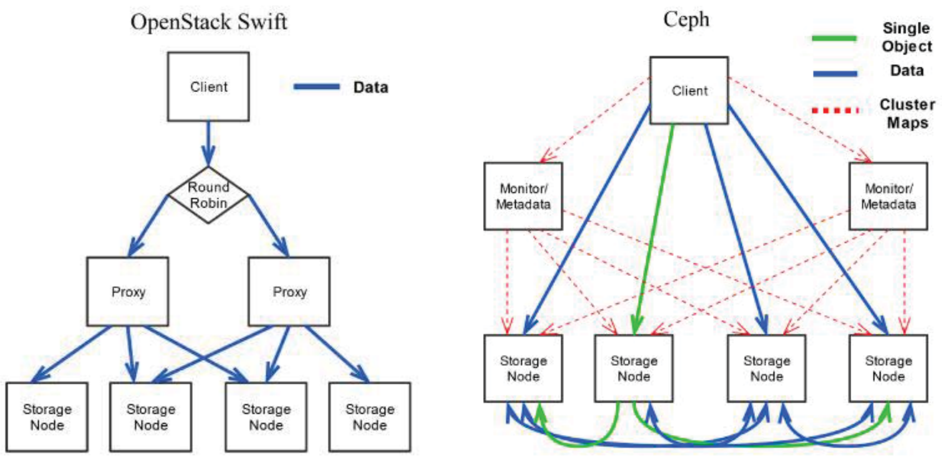


Figure 1: Architecture and data paths of OpenStack Swift and Ceph object storage systems. The left architecture map is showing one client writing to OpenStack Swift, the data (in blue) is transferred through the proxy servers using a round robin network connection. The proxy servers then distribute the data to three of the storage nodes. The right architecture map is showing one client writing data into the Ceph storage system. The blue/green lines show the path the data will take, the green line is meant to show the path of one object being stored. The red lines are showing the distribution of the cluster maps via the Monitor/Metadata servers.

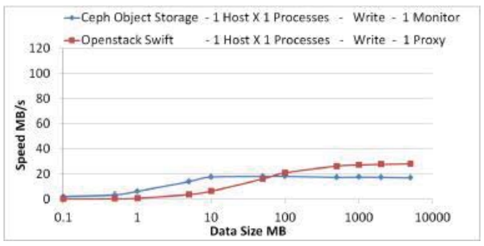
To run OpenStack Swift there are multiple services needed: the proxy service, container service, account service, and the object service. For our initial configuration, the container, account and object service were run on 10 storage nodes while an additional node was dedicated to the proxy service. As illustrated on Figure 1, when running file transfers using OpenStack Swift, the file is transferred through a proxy node then distributed to the storage nodes. With this model, the proxy node can become a bottleneck as all I/O is funnelled through one node. The proxy service can be very CPU & I/O intensive.

Ceph uses two services: the OSD service and the monitor service plus a metadata service if using CephFS POSIX FS. We did the same as with OpenStack Swift by dedicating 10 farm nodes as storage nodes (running the OSD service for each drive) and only ran 1 monitor service on a separate node. As shown on Figure 1 (right hand side), when a client uploads a file to the Ceph cluster instead of the file being transferred through a proxy, the clients connect directly to the storage nodes eliminating any bottleneck. The job of the monitor node is to distribute cluster maps to the clients and storage nodes which are essentially guidelines as to where to data should be placed.

3. Performance Measurements

3.1. Base performance
The drives available on our cluster were tested for their raw performance using a set of `dd` tests. We measured a maximum performance of 100 MB/sec in “sync” mode and 145 MB/sec on “direct” mode, consistent with the vendors maximum performance of 155 MB/sec for those drives.
A variety of tests were then run on each storage cluster to simulate actual uses while we captured the I/O performance of client transfers along with the overall load and network saturation on the storage nodes. Tests were made up of single client file transfers and multi-client file transfers consisting of single and parallel read and writes.

3.2. Single host, single stream test
Figure 2 shows the read and write performance as seen from a single host. The data size is varied while the speed in MB/sec is measured from a single process viewpoint.



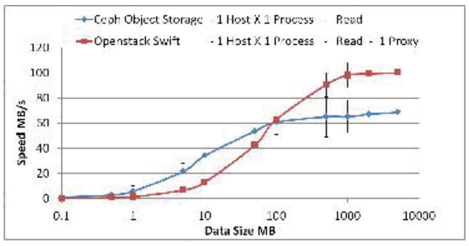


Figure 2: The left plot displays object storage write performance of 1 host running 1 write process while the right plot is displaying read performance of 1 host running 1 read process. Both plots are in MB/s as a function of data file size.

In our default configuration, the Swift configuration stores each replica as one file. Ceph write performance tapers off after ~10 MB file size due to objects being stored in 4 MB chunks, and performance is unchanged. When writing a single file using the Rados Gateway into Ceph object storage, the client will begin transferring the file to one storage node while simultaneously the storage node replicates the file to two other nodes thus completing replication three. While the write performance of the two systems are comparable, read performance with OpenStack Swift does outperform Ceph with larger files and reaches near the network limit.

3.3. Single host, multiple IO stream test
From a single client, we then proceeded in testing the scalability as a function of process concurrency. Essentially, we performed ten concurrent IO tests and measured both the average per process and the performance as an overall IO aggregate. The results are summarized on Figure 3.

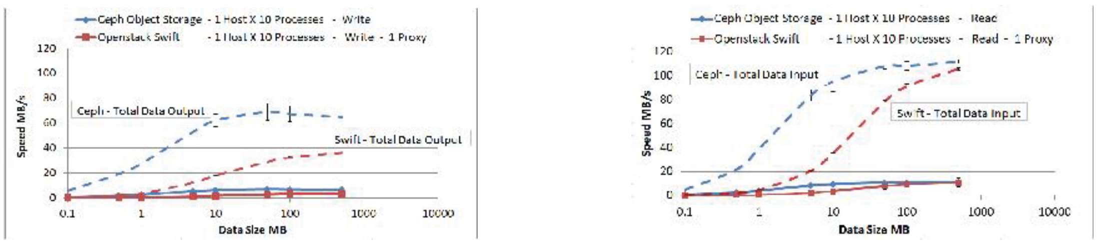


Figure 3: The left plot is object storage write performance of 1 host running 10 parallel write processes while the right plot is object storage read performance of 1 host running 10 parallel reads. Both plots are in MB/s as a function of data file size and the solid curves represent the speed per (read/write) process while the dotted lines represent the aggregate speed of all 10 processes.

As previously noted, the clients transferring files with OpenStack Swift are funnelling all data through the single proxy node. With Ceph the aggregate performance in this case scales much higher over OpenStack Swift because each individual client write process will create its own connection to the storage nodes thus spreading the I/O load across the storage cluster. In theory as you scale the number of OSDs in your Ceph cluster the faster it should perform as the load is distributed evenly. As a result, Ceph outperforms Swift in both read and write performance (albeit in read, the performance converges at high data chunk size to an asymptotic value equivalent to the network speed).

3.4. Multiple host tests

Another metric for scalability is to test performance as a function of node concurrency (one process per node, N nodes). The results of our tests are shown in Figure 4.

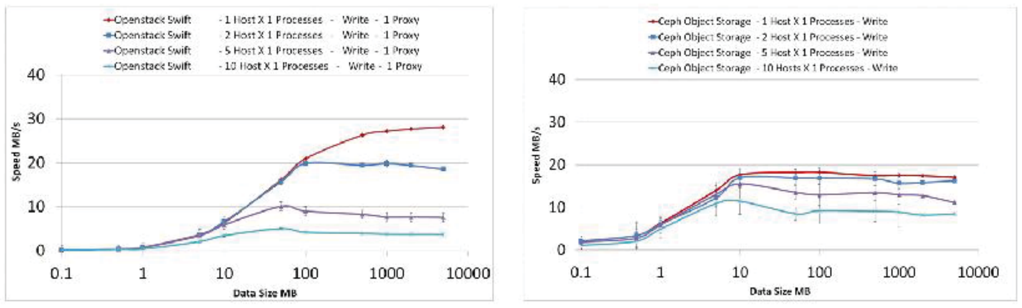


Figure 4: Shows OpenStack Swift object storage (left) and Ceph object storage (right) write performance while scaling the number of hosts writing single files in parallel. The curves are showing the write speed per node in MB/s as a function of data file size.

Consistent with the results indicated in section 3.2, Swift starts at a higher performance value for a single process than Ceph. However, the degradation of the IO performance per process is dramatic as concurrency increases (an order of magnitude from 1 to 10 host IO) while in the Ceph case, the performance decreases at a much lower rate (x2 at most from 1 to 10 hosts). While OpenStack Swift would be ideal in a single user, single host environment, as the number of clients increase the performance degradation per client is more drastic than the performance degradation of Ceph. With process concurrency, Ceph is a much more ideal solution as STAR is a multi-user environment and will have diverse use cases.

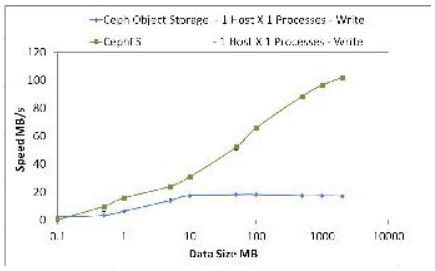
3.5. Swift, proxy node scaling

OpenStack Swift is not restricted to a single proxy node configuration and in most production cases multiple proxy nodes are set. With this, we scaled the number of proxy nodes from one to two to four, configured a DNS round robin for distributed client access and re-ran the performance measurements. The I/O performance per client scales as the number of proxy nodes scale as seen in Table 1. OpenStack recommends that proxy servers be dedicated to the service only. For STAR, dedicating multiple nodes to a particular service would be an inefficient use of our cluster as losing storage nodes would be detrimental to the overall storage capacity considering our limited size cluster. Ceph's monitor and metadata services are very lightweight and can be run on the same nodes running the storage service; such configuration ensures redundancy for the monitor service eliminating a single point of failure. At this time however, Ceph metadata servers cannot be run in parallel although a failover system is in place.

Table 1. OpenStack Swift peak write performance with scaling proxies servers. Performance per client while 10 clients write in parallel displaying average peak performance in MB/s at 100 MB file size.

Proxy Servers	Peak MB/s
1	3.7
2	7.6
4	12.5

3.6. Comparing Ceph Object storage and the POSIX compliant CephFS



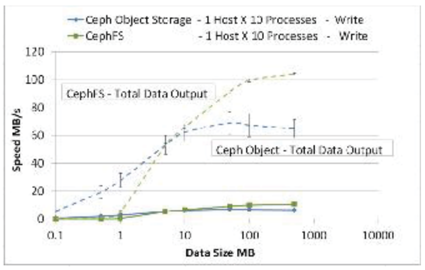


Figure 5: The left plot displays a single client running a single write process utilizing Ceph object storage vs. the Ceph POSIX file system storage. The right plot displays a single client running 10 parallel write processes in Ceph Object Storage vs. Ceph POSIX file system storage. Both plots are in MB/s speed as a function of data file size. The dash curve in the right hand-side plot indicates the aggregate IO while the solid curves are normalized values per process.

CephFS is a POSIX-compliant file system that uses the Ceph object storage to store its data [4]. The metadata service collectively manages the file system namespace and coordinates access to the shared OSD cluster. In Figure 5, we show our performance testing comparing Ceph Object and POSIX compliant implementations. As one can see, CephFS I/O performance scales significantly over the plain Ceph object storage due to CephFS breaking files down into smaller chunks and distributing them across the entire cluster instead of limiting to only 3 storage nodes per file. A client using CephFS that performs a simple copy will see connections and data transfers to many or all of the storage nodes containing OSD's. When a client writes multiple (in our case 10) parallel write processes the performance with CephFS goes unchanged as the workflow is the same for the entire data set with a single write process. The Ceph object storage appears to be catching up to the CephFS in parallel writes and that is due to each rados write process opening its own connections to the storage nodes. While writing in parallel with object storage has been seen to be more efficient, the performance will only scale depending on the number of parallel writes.

It is important to note that when writing to CephFS with multiple hosts in parallel, the performance per client is reasonable although network saturation and CPU I/O wait time is very heavy as there is a high load of cross communication. Since STAR's online cluster will be a multi-use cluster it will not only run jobs and be part of the storage cluster, but the cluster nodes will be acting as clients reading and writing from the storage system.

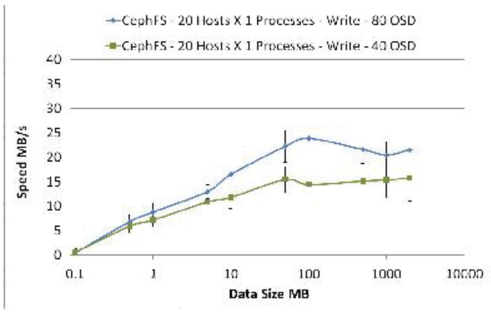


Figure 6: Twenty clients conducting one write process each shown in MB/s as a function of data file size comparison when scaling the number of OSD's from 40 to 80.

As noted earlier, it is advised that as you increase the number of storage nodes and OSDs the overall I/O performance of the cluster should increase as well. Significant performance gains were not noted until utilizing all 20 nodes in the storage cluster as both a storage node and a client writing to the cluster. This is shown on Figure 6 considering replication 3 with 40 OSDs, performance degradation may not be recognized until performing 14 parallel writes (14 writes * replication 3 > 40 OSDs). When reaching 14 parallel writes, this will queue parallel writes per operation to disk creating lots of CPU IO wait. STAR's production cluster will utilize all 30 nodes in the storage cluster earning us 120 OSDs, with this we should handle up to 40 parallel writes with no performance degradation.

4. Stability and Stress Tests

4.1. Small I/O Maximizing the IOPS

Small file sizes written over a long period can often cause problems for file systems as the metadata overhead can be abundant. In a cluster with 20 storage nodes (80 OSD) we attempted to utilize every core in the storage cluster as a client writing small parallel I/O into the CephFS storage cluster for a long period of time and examine the results. By running 1 KB and 1 B dd writes for 1 million counts on every core, the storage cluster and metadata server handled the load without any major problems. Overall we managed ~220 IOPS per node. The load on the metadata server was minimal as seen on Figure 7.

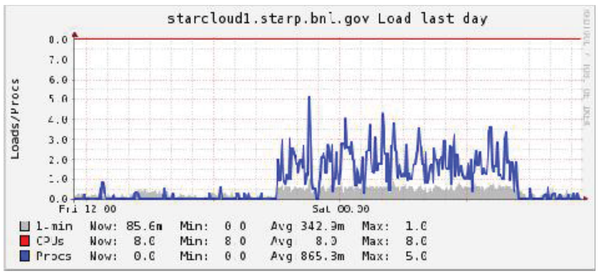


Figure 7: Plot displays the load on the monitor server while heavy parallel 1KB and 1B writes were taking place on the CephFS cluster.

4.2. Maximizing the Throughput
A CephFS cluster containing 20 storage nodes and 1 additional node running the monitor and metadata service was put to the test by utilizing half of the storage nodes as clients to perform 50 parallel `dd` write tests per client. We scaled the block sizes from 100 KB to 100 MB per process, the aggregate I/O reached ~850 MB/s although the CPU IO wait per node ~80%. From here we ran write tests requiring more RAM than the nodes had causing unresponsive machines. This test was done to simulate a user mistake or a problem where the cluster nodes would need to be rebooted by hand. Ceph salvaged the cluster in about one hour and no previously written data was lost. We found that if Ceph falls into a degraded state beyond automatic salvaging (half of cluster goes down) that you cannot interact (read/write) with both the CephFS and Ceph object storage, falling into a safe-mode like state.

5. Multi Use Cluster
The OSDs in Ceph sitting idle inflict about 1% - 2% load on the CPU's. Ceph ensures data integrity by performing scrubbing on placement groups. Ceph generates a catalogue of all objects and compares each primary object and its replicas to ensure no objects are missing or mismatched [5]. Light scrubbing is performed daily and deep scrubbing is performed weekly. The goal to reuse the STAR online compute cluster can be retained although Ceph does consume a lot of resource in heavy write scenarios. The main contextual use of CephFS for STAR is a write once read many scenarios. Single and parallel reads in CephFS does not consume much resource and can be considered fairly negligible, see Figure 8 for reference.

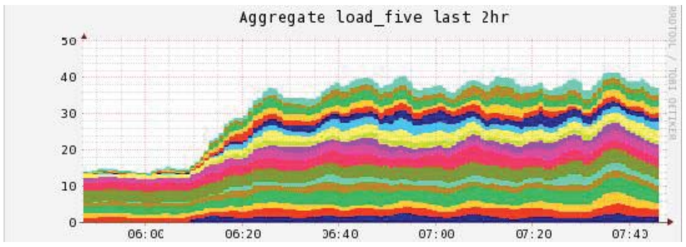


Figure 8: Plot displays the aggregate load on the entire storage cluster while a deep scrubbing is taking place. A total 25% load on each node (load average ~ 2 per node with 8 cores).

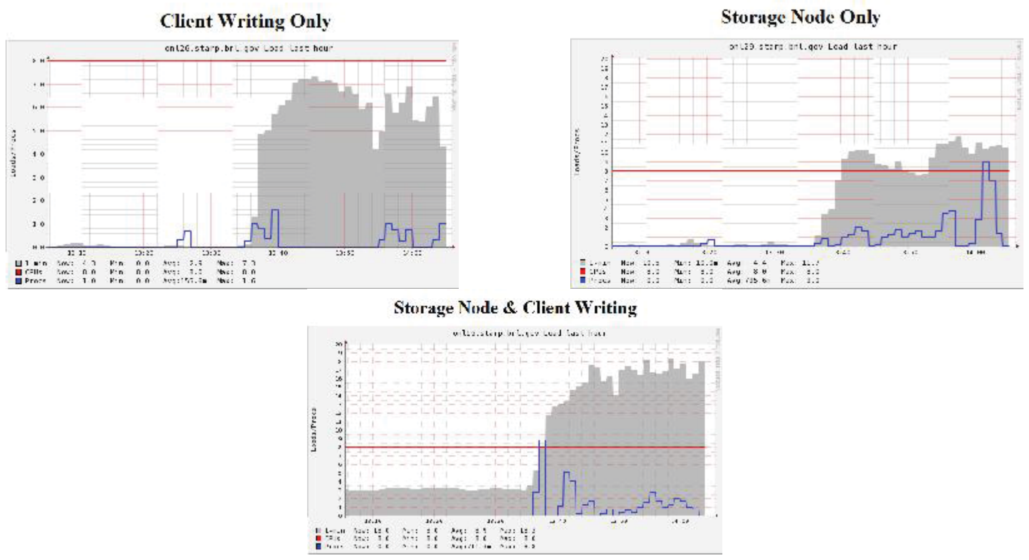


Figure 9: The top left plot displaying the load on 1 node performing writes into the CephFS only. The top right plot displays the load on 1 node running OSDs only. The bottom plot displayed the load on 1 node performing writes and running OSD simultaneously.

By utilizing every node in the storage cluster except for one, to perform write tests and utilizing all cores, we have been able to unravel the load of the OSDs and the writes on each node. For the top left plot in Figure 9, on this node we disabled the OSDs (we let Ceph rebalance before proceeding) then ran write tests on all the nodes belonging to the storage cluster (including the disabled one). We observed the load difference on the node just doing writes to the nodes performing writes and running OSDs. The load on the writing only node was ~1 per core, while the load on nodes writing and storing was ~2.25 per core (compare to bottom plot in Figure 9). For the top right plot in Figure 9, we then performed the opposite by choosing one node to run OSDs only, but not perform any write tests. The load peaked around ~1.375 per core on the OSD only node. Our concluding remark is that under full IO (write) load, mixing processing and storage using Ceph may be challenging within our test-bed as Ceph does put a fair load on the machines. However, heavy CPU IO wait has been observed along with complete saturation on the 1 Gb network links: it is likely that with a faster network links, the situation would ameliorate (investigation of a 10 Gb backbone network has been informally tested but does show positive signs of I/O performance increase.). Finally, the load impact due to reads alone is not significant, an encouraging observation for our usage (write once, read many).

6. Conclusion

In this paper, we have compared OpensStack Swift and Ceph. Both object storage systems have similar performance with single file writes although Ceph object storage seems to reach a limit with larger files. With I/O concurrency Ceph object storage scales over OpenStack Swift as there are no IO limiting bottlenecks and Ceph will ensure a proper load distribution. Furthermore with default parameters CephFS scales over Ceph object storage. CephFS performs striping on single file writes which will support immediate load balancing with small scale and large scale IO. OpenStack Swift would be an ideal solution in specific cases such as single client uses for maximum I/O and OpenStack's friendly usage (Virtual Machine caching). While offering no POSIX interface, OpenStack Swift is not ideal for STAR and its architecture can quickly create bottlenecks. CephFS in contrast offers versatility for STAR's uses and the ability to leverage the cluster for multi-use appears to be promising. For clients, beyond a few Ceph rpm packages and a later kernel version containing the Ceph kernel module, mounting the CephFS is simple. Ceph does have cache solutions for use of fast storage devices and hot data along with running OSD journals on SSDs for overall increased I/O performance [6]. Our future work and testing will evolve in this direction.

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

References

[1] Sangaline E and Lauret J 2014 *Experience, use and performance measurement of the Hadoop File System in a typical nuclear physics analysis workflow* 2014 J. Phys.: Conf. Ser 523 012006
[2] OpenStack http://www.openstack.org
[3] Ceph http://www.ceph.com/
[4] Weil S A 2007 *Ceph: Reliable, Scalable, and High-Performance Distributed Storage.* Doctoral dissertation, University of California, Santa Cruz, California
[5] Ceph data scrubbing http://docs.ceph.com/docs/master/rados/configuration/osd-config-ref/#scrubbing
[6] Ceph Cache Pool http://docs.ceph.com/docs/firefly/dev/cache-pool/