Experience, use, and performance measurement of the Hadoop File System in a typical nuclear physics analysis workflow

E Sangaline[1] and J Lauret[2]
[1]Physics Department, University of California Davis, Davis, CA 95616-5270, USA
[2]Physics Department, Brookhaven National Laboratory, Upton, NY 11973-5000, USA

E-mail: sangaline@nuclear.ucdavis.edu

Abstract. The quantity of information produced in Nuclear and Particle Physics (NPP) experiments necessitates the transmission and storage of data across diverse collections of computing resources. Robust solutions such as XRootD have been used in NPP, but as the usage of cloud resources grows, the difficulties in the dynamic configuration of these systems become a concern. Hadoop File System (HDFS) exists as a possible cloud storage solution with a proven track record in dynamic environments. Though currently not extensively used in NPP, HDFS is an attractive solution offering both elastic storage and rapid deployment. We will present the performance of HDFS in both canonical I/O tests and for a typical data analysis pattern within the RHIC/STAR experimental framework. These tests explore the scaling with different levels of redundancy and numbers of clients. Additionally, the performance of FUSE and NFS interfaces to HDFS were evaluated as a way to allow existing software to function without modification. Unfortunately, the complicated data structures in NPP are non-trivial to integrate with Hadoop and so many of the benefits of the MapReduce paradigm could not be directly realized. Despite this, our results indicate that using HDFS as a distributed filesystem offers reasonable performance and scalability and that it excels in its ease of configuration and deployment in a cloud environment.

1. Introduction

Most modern NPP experiments currently utilize the ROOT framework for data analysis and binary ROOT files for storage [1]. The ROOT distribution includes XRootD which is a highly scalable and fault-tolerant distributed storage solution designed as an extension of ROOT's built in rootd file server [2]. XRootD is commonly used in conjunction with ROOT and has reliably handled the storage and analysis requirements of a number of experiments over the years. The configuration of XRootD does not, however, lend itself to applications where resources are being dynamically added and removed on short time scales. While exploring options for more flexible scaling of computing resources with virtualization we have found that XRootD is best suited for relatively static allocations of resources. This has motivated us to investigate other possibilities for distributed filesystems that are better suited for dynamic scaling.

Hadoop is a software framework for the distributed storage and processing of large data sets [3]. It is maintained as an open source project by the Apache Software Foundation and implements many of the ideas behind Google's MapReduce and Google File System (GFS) [4]. The underlying technology was designed to achieve robust scalability on commodity hardware and thus has redundancy, fault tolerance, and flexibility in deployment built in as key features. These aspects have played a significant role in Hadoop's adoption in industry where it is used to store and analyse datasets that are large even by NPP standards. For example, the Large Hadron Collider (LHC) produces 15 PB of data per year while Facebook's Hadoop clusters grow by over 150 PB of data per year [5], [6]. The success of Hadoop in handling storage and cluster growth of this magnitude makes it a possible candidate for consideration within NPP applications.

We aim to evaluate the performance of the Hadoop File System (HDFS) with respect to the needs of NPP and our experiences with dynamic resource allocation [7]. This work serves to complement prior investigations of HDFS's applicability in NPP by evaluating additional access methods and further investigating I/O scaling [8], [9].

2. Testing Environment

For the purposes of testing, a cluster of 25 nodes was constructed. A Xen kernel under Scientific Linux 5.9 was installed as the host system and a guest virtual machine was configured on each node and allocated one core of a dual core 1.8 GHz AMD Opteron Processor, 4 GB of RAM, four 2 TB drives, and a 1 Gb/s network interface attached to a 1 Gb/s switch [10]. The virtual nodes were configured to run CentOS 5.9 with the full STAR software environment. Apache Hadoop 1.1.2 was deployed on the virtual cluster using a 64 MB block size as well as MapR's commercial distribution of Hadoop with an M5 evaluation license. MapR required direct access to the four storage drives while for Apache Hadoop the drives were instead configured as a RAID 5 array formatted as ext3.

Using the Hadoop client and accessing the local filesystem directly were used as baselines to assess the relative overheads of Hadoop itself and of the various access methods. It should be noted that a libhdfs plugin is now available in ROOT but that compatibility issues prevent the STAR environment from being used with newer versions of ROOT. Instead, several different methods for mounting HDFS and accessing it through standard POSIX libraries were configured:

- Fuse-DFS. A mountable interface to HDFS based on the Filesystem in Userspace project (FUSE) [11].
- HDFS NFS Proxy. A project which allows HDFS access via an NFS server [12].
- MapR. A proprietary alternative which also allows the filesystem to be mounted via NFS [13].

3. Tests and Results

A variety of tests were run to get a basic comparison of the reliability and access rates for the different methods. The systems were configured to drop caches before and after each test and to include sync times in the performance measurements. The I/O speeds were investigated as a function of file size, replication, and the number of active clients. Unless otherwise noted, all rates refer to sequential file access.

For single client file access it was found that there was little change in I/O rates with the replication, although this does not include the time for full replication to be reached. Local disk write rates saturated at 30 MB/s while the Hadoop client saturated at 24 MB/s and both HDFS NFS Proxy and Fuse-DFS saturated at around 12 MB/s. Write operations would mainly be used for importing data files into a cluster and with the Hadoop client this would bring relatively low overhead. The read rates can be seen in figure 1 where we notice considerable overhead relative to local disk access. It was found that HDFS NFS Proxy had serious stability issues with file sizes above 200 MB.

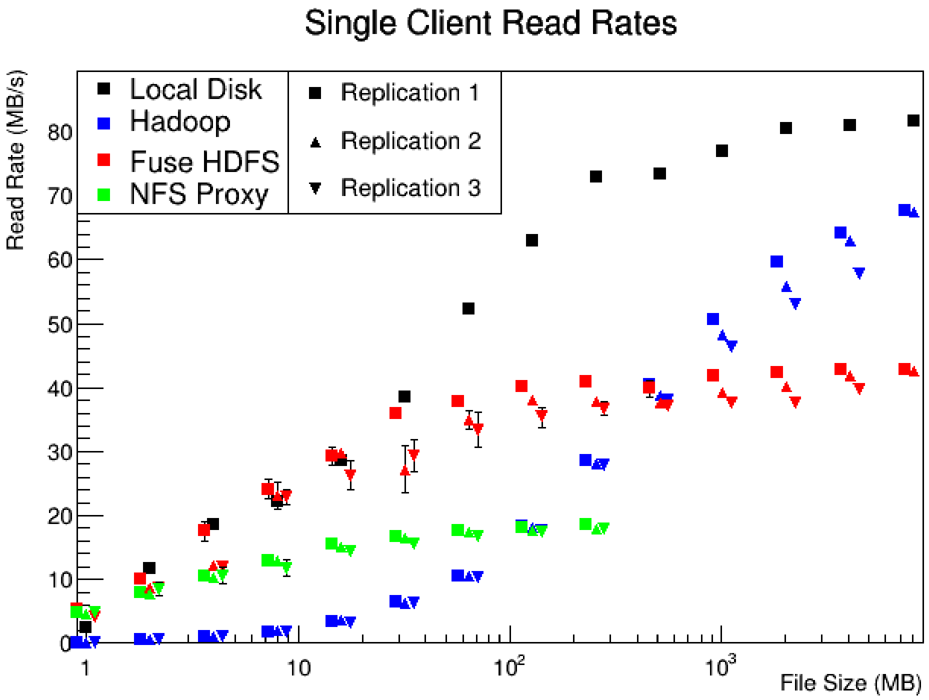

Figure 1. The read rates for various access methods as a function of file size.

The multiple client read rates offerinsight into the scalability in the situation where multiple analyses might need to access the same files. From figure 2 we find that for all interfaces the total read rate scales linearly with the number of clients, even as the number of clients grows considerably larger than the replication. We also find that MapR significantly outperforms the other interfaces for reading.

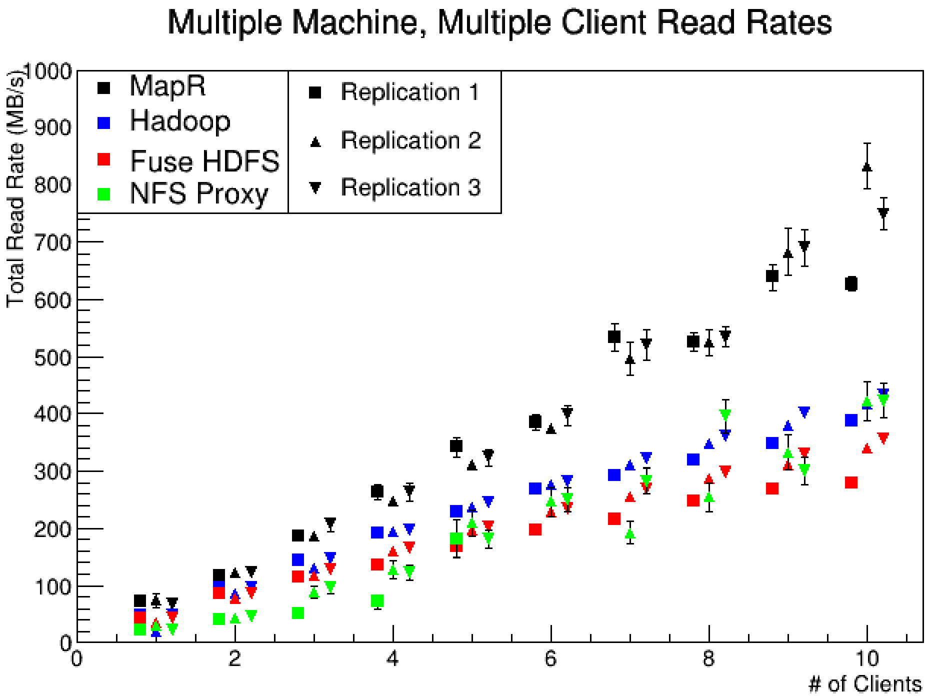

Figure 2. The scaling of read rates as a function of number of clients for a single 1 GB file.

HDFS was designed for sequential access and, because of this, the previously discussed rates are larger than what we would expect for non-sequential reads. To get a more realistic view of what sort of performance we could expect we created 1 GB ROOT files containing Au+Au events and wrote analysis code within the STAR framework to run over them. The code applies basic event quality cuts and fills several histograms, tasks that would be typical of a simple physics analysis. Running these analysis on the Hadoop virtual cluster yielded interesting results which can be found in table 1. Fuse-DFS is comparable in performance to local disk access and significantly outperforms MapR, contrary to the tests with sequential reads.

Table 1. STAR Analysis Rates

Access Method	Read Rate (MB/s)
Fuse-DFS	9.9+/-0.1
MapR	3.1+/-0.1
Local Disk	9.7+/-0.6

4. Conclusions and Outlook

We have performed basic performance testing for various HDFS access methods in the context of a virtualized cluster for NPP analysis. There is further work to be done in terms of understanding performance under more realistic conditions but we can already form some important conclusions.

We were able to quickly and easily configure and deploy Hadoop images on new machines. When switching between MapR and Hadoop we were required to frequently make changes in disk and server configurations, to format HDFS, and to repopulate the cluster with files. Nodes were trivial to deploy and large data files could be transferred into the cluster with little overhead and with replication handled automatically. Qualitatively speaking, this was all relatively straightforward and satisfies the properties we were looking for in something that could be easily deployed on short timescales.

Our multiple client testing demonstrated linear scaling beyond the replication, and in the future we will run more extensive tests in order to understand how saturation will occur. Tests run which performed actual analysis tasks found the analysis rate for Fuse-DFS to be consistent with local disk access and to outperform the proprietary MapR. This result is very promising and offers hope that performance in actual applications might be better than in the sequential tests. Overall, we find that HDFS would likely make a viable distributed filesystem for quick deployment of NPP analysis clusters and hope to continue our studies to better understand performance under saturation conditions.

5. Acknowledgements

We would like to acknowledge and thank the DOE and NSF for funding that supported this work as well as the University of California Davis and Brookhaven National Laboratory for resources that they made available.

References

[1] Moneta L et al. 2008 Recent developments of the ROOT mathematical and statistical software *J. Phys.: Conf. Ser.* **119** 042023
[2] XRootD http://xrootd.slac.stanford.edu/
[3] Apache Hadoop http://hadoop.apache.org/
[4] Ghemawat S, Gobio H and Leung S-T 2003 The Google File System *Proc. 19th ACM Symposium on Operating Systems Principles*

[5] Kanti P 2009 Black Holes at the *LHC Lecture Notes in Physics* **769** 387-423

[6] Under the Hood: Scheduling MapReduce jobs more efficiently with Corona https://www.facebook.com/notes/facebook-engineering/under-the-hood-scheduling-mapreduce-jobs-more-efficiently-with-corona/10151142560538920

[7] Borthakur D 2013 Hdfs architecture http://hadoop.apache.org/docs/r1.2.1/hdfs_design.html

[8] Bockelman B 2009 Using Hadoop as a grid storage element *J. Phys.: Conf. Ser.* **180** 012047

[9] Riahi H *et al.* 2012 Using Hadoop File System and MapReduce in a small/medium Grid site *J. Phys.: Conf. Ser.* **396** 042050

[10] The Xen Project http://www.xenproject.org/

[11] Filesystem in userspace http://fuse.sourceforge.net/

[12] HDFS NFS Proxy https://github.com/cloudera/hdfs-nfs-proxy

[13] The MapR Distribution for Apache Hadoop http://www.mapr.com/Download-document/7-MapR-White-Paper