EPJ Web of Conferences **295**, 04028 (2024) https://doi.org/10.1051/epjconf/202429504028
CHEP 2023

Job CPU Performance comparison based on MINIAOD reading options: local versus remote

Justas Balcas[1,*], *Harvey* Newman[1,**], *Preeti* P. Bhat[1,***], *Sravya* Uppalapati[1,****], *Andres* Moya[1,†], *Catalin* Iordache[1,‡], and *Raimondas* Sirvinskas[1,§]

[1]George W. Downs Laboratory of Physics and Charles C. Lauritsen Laboratory of High Energy Physics 1200 E California Blvd Pasadena, California 91125

Abstract. A critical challenge of performing data transfers or remote reads is to be as fast and efficient as possible while, at the same time, keeping the usage of system resources as low as possible. Ideally, the software that manages these data transfers should be able to organize them so that one can have them run up to the hardware limits. Significant portions of LHC analysis use the same datasets, running over each file or dataset multiple times. By utilizing "on-demand" based regional caches, we can improve CPU Efficiency and reduce the wide area network usage. Speeding up user analysis and reducing network usage (and hiding latency from jobs by caching most essential files on demand) are significant challenges for HL-LHC, where the data volume increases to an exabyte level. In this paper, we will describe our journey and tests with the CMS XCache project (SoCal Cache), which will compare job performance and CPU efficiency using different storage solutions (Hadoop, Ceph, Local Disk, Named Data Networking). It will also provide insights into our tests over a wide area network and possible storage and network usage savings.

1 Introduction

The Worldwide LHC Computing Grid [1] (WLCG) is a global collaboration between hundreds of computer centers. They are geographically distributed over the world with various sizes of numbers of cores and data storage space in each. The WLCG provides data transfer services for the four main virtual organizations (CMS [3], Atlas [4], LHCb [5], Alice [6]) using two technologies: File Transfer Service (FTS3) [7] and XRootD [8]. XRootD is a distributed and scalable system for low-latency file access. It is the primary wide-area network data-access framework for the CMS experiment at the Large Hadron Collider (LHC) [2]. The motivation of this CPU performance testing in the present AAA (Any Data, Any Time, Anywhere) infrastructure [9] is to improve CMSSW [10] data access for eventual increased use of it as a transfer protocol. In this paper, we start by presenting current storage technologies and

*e-mail: jbalcas@caltech.edu
**e-mail: newman@hep.caltech.edu
***e-mail: preeti@caltech.edu
****e-mail: suppalap@caltech.edu
†e-mail: amoya@caltech.edu
‡e-mail: catalinn.iordache@gmail.com
§e-mail: raimis.sirvis@gmail.com

their implementation details within CMS. Then we continue with describing the implementation of data access in CMSSW and AAA XRootD Federation. In addition, we provide an overview of the currently used storage technologies and future directions for HL-LHC [11].

The aim of these tests is to provide information for the user community and site administrators with respect to what to expect in terms of CPU efficiency and network usage for jobs accessing data locally and or remotely over the wide area network.

2 AAA and CMSSW Infrastructure

Figure 1 shows schematics of the XRootD infrastructure that spans all of the CMS Tier-1 and Tier-2 sites in Europe and the United States. The hierarchical subscription of XRootD managers (cmsd process) provides resiliency in data access toward clients. Effectively this means that, if the accessed data are not available at the given site where the job lands, the client will be automatically redirected to the closest data available within the hierarchy of redirectors to the storage server where those data exist. Each site's XRootD server is interfaced with the local storage system, allowing it to export the CMS namespace current storage systems (dCache [12] and proxy dCache, HDFS[13], Lustre[14] and DPM[15], EOS[16], Ceph[17]). Site servers also subscribe to a local redirector. In their turn, the local redirectors from each site then subscribe to a redundant regional redirector. This creates a large tree-structured federated storage system: (i) United States: Fermilab and Nebraska (DNS round-robin alias cmsxrootd.fnal.gov) (ii) Europe: Bari, Pisa, Paris (DNS round-robin alias xrootd-cms.infn.it) (iii) Transitional: CERN (DNS round-robin alias cms-xrd-transit.cern.ch) Individual users or grid jobs can request a file from the regional redirector, which will then query the child nodes in the tree and redirect the user to a server that can serve the file. The entire AAA infrastructure overlays on top of existing storage systems, allowing users access to any on-disk data without explicit knowledge of the file location.

The overall collection of software of the CMS experiment referred to as CMSSW, is built around a Framework, an Event Data Model (EDM), and Services needed by the simulation, calibration, and alignment, as well as reconstruction modules that process event data used for physics analysis. The primary goal of the Framework and EDM is to facilitate the development and deployment of reconstruction and analysis software.

The CMSSW event processing model consists of one executable, called cmsRun, and many plug-in modules which are managed by the Framework. All the code needed for event processing (calibration, reconstruction algorithms, etc.) is contained in the modules. The same executable is used for both experimental data and Monte Carlo simulations. The CMSSW executable, cmsRun, is configured at run time by a job-specific configuration file. This file defines:

- which data to use;
- which modules to execute;
- which parameter settings to use for each module;
- the order or the executions of modules, called path;
- how the events are filtered within each path;
- how the paths are connected to the output files.

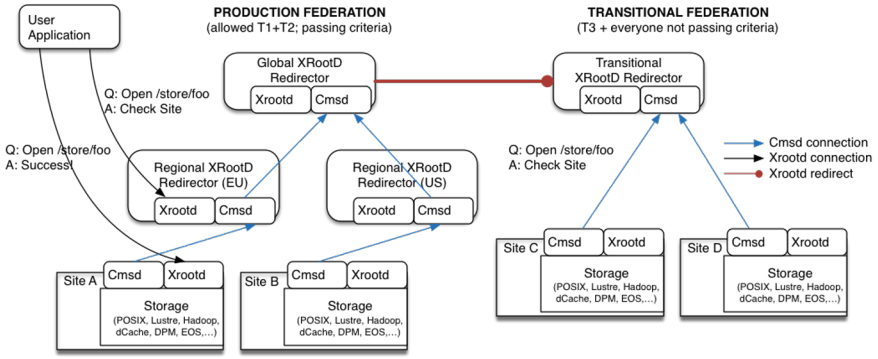

Figure 1. AAA, Any data, Anywhere, Anytime is built for the CMS experiment using the distributed CMS data cache. The resulting system, a hierarchy of redirectors, improves the scientific productivity of CMS physicists through better access to the data as well as the effectiveness of the storage infrastructure deployed among the CMS sites on the worldwide grid.

3 Distributed File Systems

We considered the following DFS (Distributed File Systems) - Ceph, Hadoop, XRootD, and XCache - which are currently used in Caltech [18] and UCSD [19] Tier2 data centers. More detailed descriptions are available below.

3.1 Ceph

Ceph is a powerful and versatile open-source distributed storage system that has gained significant popularity for its ability to provide scalable, high-performance, and fault-tolerant storage solutions. With Ceph, organizations can manage their data across clusters of computers, utilizing a unified storage pool accessible through various interfaces such as object storage, block storage, and file storage. One of Ceph's standout features is its use of a distributed architecture, where data is distributed across multiple nodes to ensure redundancy and prevent data loss in the event of hardware failures. Ceph's flexible design allows it to be deployed in various configurations, from small-scale setups to large data centers, and it offers seamless integration with popular virtualization platforms and cloud environments. This adaptability, combined with its cost-effectiveness and robustness, makes Ceph a compelling choice for running it as a Storage solution for CMS Experiment. For the following tests below - we installed a dedicated Ceph Filesystem of a total of 10 nodes, each with 3 HDDs (Hard Disk Drives) and 1G Uplink.

3.2 Hadoop Distributed File System (HDFS)

Hadoop Distributed File System (HDFS) is a fundamental component of the Apache Hadoop ecosystem, designed to efficiently store and manage vast amounts of data across distributed clusters of commodity hardware. HDFS employs a unique approach to data storage, breaking large files into smaller blocks and distributing these blocks across multiple nodes in the cluster. This enables parallel processing and high throughput for both reading and writing data. The system's fault tolerance mechanisms ensure data reliability by replicating each

block multiple times across different nodes, reducing the risk of data loss due to hardware failures. Its adaptability and scalability have made a huge impact on US-CMS Tier2 sites, where it is able to provide Petabytes of scalable storage for CMS. For the following tests below - we used production Hadoop installation of 8PB, deployed between many compute and data nodes (ranging from 3 disks to 60 disks per server).

3.3 XRootD

XRootD is an open-source software framework designed to facilitate efficient and scalable access to remote data storage and distribution. Originally developed for the high-energy physics community, XRootD has gained widespread adoption in various scientific and data-intensive fields. At its core, XRootD provides a client-server architecture that enables users to remotely access, manage, and transfer large datasets across geographically distributed sites. XRootD supports a range of data access protocols, making it versatile enough to integrate with different data storage solutions, such as local file systems, object stores, and distributed file systems. Researchers and institutions benefit from XRootD's ability to efficiently handle massive datasets, enabling collaborative research, analysis, and data sharing on a global scale. For the following tests - we tested multiple endpoints reading data remotely over a wide area network.

3.4 XRootD Cache

XRootD caching is a crucial feature within the XRootD framework that significantly enhances data access efficiency and reduces latency for distributed data storage and retrieval. The caching mechanism allows frequently accessed data to be stored temporarily on local storage nodes, closer to the applications that need it. This strategy effectively minimizes the need for repetitive data transfers over the network, resulting in faster data access times and reduced load on the remote storage systems. XRootD caching is particularly valuable in scenarios where large datasets are shared among multiple computational tasks.

3.5 Named Data Networking (NDN)

Named Data Networking (NDN) [21] represents a forward-thinking and innovative approach to internet communication and data distribution. Departing from the traditional host-centric model, NDN focuses on content retrieval by naming data itself rather than relying solely on host addresses. This paradigm shift introduces numerous advantages, including enhanced data security, reduced data redundancy, and improved content delivery. In NDN, data becomes the central entity, enabling efficient in-network caching and facilitating data sharing. When a user requests specific content, the NDN routers collaboratively locate and deliver the content based on its name, leveraging the inherent caching capabilities to expedite future access. This design inherently lends itself to scenarios where content dissemination, such as video streaming or software distribution, is paramount. NDN's potential to mitigate some of the challenges of today's internet architecture makes it an intriguing candidate for shaping the future of network communication and data sharing.

4 Testing environment

For the following tests, we choose one of the user analysis datasets, named: /DYJetsToLL_M-50_TuneCUETP8M1_13TeV-amcatnloFXFX-pythia8/RunIISummer16MiniAODv3-PUMoriond17_94X_mcRun2_asymptotic_v3_ext2-v1/MINIAODSIM. Given available

space at remote sites, All tests are running 269 jobs (that accounts of total 1TB data from the dataset). Only 1 test is run at a time and all these tests are running on eight dedicated machines without any external load. To note, data is removed from cache servers before repeated tests and the cache is empty before each run. Each test is described below:

- **Ceph** - dedicated Ceph installation for these tests between a total of 10 nodes (AMD Opteron 6378, 94GB RAM), each with 3 HDDs and 1G Uplink. Ceph configuration: 1 Metadata Server, 3 Monitors, Replication 2, Default Object/Stripe size, 100PGs/Disk, Ceph was mounted using Kernel drivers.
- **CephClient** - dedicated Ceph installation for these tests between a total of 10 nodes, each with 3 HDDs and 1G Uplink. This test used the same Ceph configuration as the Ceph Kernel mount test above. Ceph was mounted using the Fuse client.
- **Hadoop** - Production Hadoop distributed file system. These tests have been repeated multiple times as we find needed improvements. Repeated tests are highlighted as **Hadoop1**, **Hadoop2**.
- **Local** - Before job startup, data is downloaded to a local disk, and the CMSSW job reads data from the local disk. CMSSW is not making any remote open/read operations for job runtime.
- **RedirFnal** - Instruct job to read data via AAA US-CMS Redirector. At the time of run - data was available at 6 Disk Sites: Cern, Caltech, Nebraska, Griff, IFCA, and DESY. [1]
- **RedirCaltech** - Job reads data via Caltech Tier2 Redirector (9 XRootD Data Origins behind a single redirector). [1]
- **CERN** - Enforcing jobs to read data only from CERN XRootD Redirector. [1]
- **RedirCache** - Reading data over SoCal Cache [20] Redirector distributed between 2 Sites: Caltech and UCSD. Tests (**RedirCache1**) were repeated to highlight identified issues below. [1]
- **CacheDirect** - Reading directly over single cache servers. Tests were repeated (**CacheDirect1**) after a few identified issues highlighted below.[1]
- **CacheCern** - Reading data via Cache, while data is only available on CERN Storage, under /store/user/jbalcas. Tests were repeated multiple times and shown as: **CacheCern1**, **CacheCern2**. [1]
- **NDN** - Reading data over NDN Network, which is built on top of XRootD. NDN uses 1 Server to serve data: 2x Silver 4110, 94GB RAM, 10x8TB HDD Raid 0, 100Gbps Nic.

5 Results

In Figure 2, we show the total job runtime for all tests. Based on the results we identified several facts to highlight from Figure 2.

- Ceph Kernel mount is faster than Ceph FUSE Client. Fuse involves a context switch between user space and kernel space for every file system operation, which adds overhead. Also, FUSE requires data to be copied between user and kernel space, which results in additional memory overhead. FUSE itself is designed to provide isolation and security for applications.

[1]Data Reading is based on the Multisource algorithm: https://github.com/cms-sw/cmssw/blob/master/Utilities/XrdAdaptor/doc/multisource_algorithm_design.txt

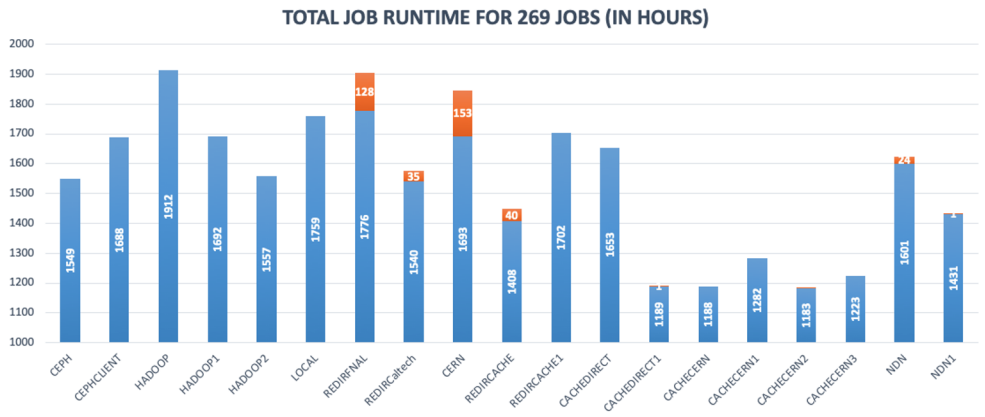

Figure 2. Total Job Runtime for 269 jobs in each of tests. Blue color- successful jobs. Orange - for failed jobs.

- Hadoop tests allowed us to identify several issues and improvements for the Production Filesystem. 1st bar - Identified a user overloading filesystem with reading RAW files over a wide area network; 2nd bar - Hadoop Balancer used in production heavily moves objects between Hadoop Cluster disks due to too high concurrent moves/dispatcherThread parameters - had to decrease those values. 3rd bar - Re-run of all tests after the first 2 fixes and it showed a huge improvement on the production cluster.

- Many remote reads, which involve XRootD protocol, showed failed jobs. Any time a job fails, all job runtime is wasted. Based on the log and job analysis, this was identified as a bug inside the CMSSW XRootD source selection algorithm. Multi-source-based selection should reconsider previously marked bad sources as possible data access points. In case data is available only at a few endpoints - remote connections can be lost, server rebooted.

- XRootD Caches, similarly to any distributed filesystem depend a lot on the healthiness of disks. Several tests have identified that a failing disk (soft I/O errors) can affect job performance.

- XRootD Caches depend on the data source providing data. In case of a bad data source and cache unable to prefetch the data, wasted time is low (90s as configured on CMSSW). In the case of a good data source, Caches allow to hide a huge latency and improve CPU Efficiency significantly over long distances. (Caltech to CERN is 170ms distance).

Figure 3 shows only equally successful jobs between different runs and it confirms the observation of caches. Caches are able to perform better than any other storage solution and give almost 20% CPU Performance boost in terms of event throughput compared to local reads (Ceph, Hadoop). The cache is not only able to improve performance for locally accessed data but also for any data over a wide area network. Some of the issues we encountered are: CMSSW Source re-selection on caches duplicates data and stores the same input file on multiple XRootD caches.

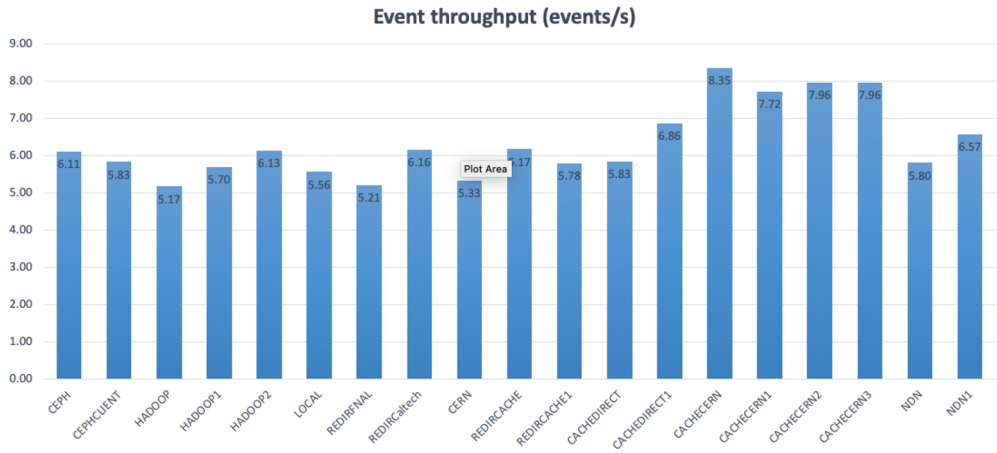

Figure 3. Event throughput for equally successful jobs (169) between different tests.

6 Summary

This study aimed at optimizing remote data reads to improve CPU efficiency. The study is motivated by the upcoming High-Luminosity LHC (HL-LHC) phase, which will generate significantly larger data volumes. The authors investigate different storage technologies including Ceph, Hadoop, XRootD, and Named Data Networking (NDN). Some key findings and observations from the study identified that hardware failures and disk health in distributed file systems can affect performance for a collaboration, but similarly, XRootD caching can significantly improve CPU performance, particularly over long distances, and minimize wasted time for data access. In conclusion, we received valuable insights and lessons into optimizing data access and storage solutions for the CMS experiment. These research outcomes are essential in addressing increasing data volume access in high-energy physics, particularly for HL-LHC.

References

[1] WLCG Project website: https://home.cern/science/computing/grid
[2] Bird I et al. "Computing for the Large Hadron Collider", 2011 Annu. Rev. Nucl. Part. S. 61: 99 doi: 10.1146/annurev-nucl-102010-130059
[3] CMS Collaboration, The CMS experiment at the CERN LHC, JINST 3 (2008) S08004, doi:10.1088/1748-0221/3/08/S08004
[4] ATLAS Collaboration, The ATLAS Experiment at the CERN LHC, JINST 3 (2008) S08003, doi: 10.1088/1748-0221/3/08/S08003
[5] LHCb Collaboration, The LHCb Detector at the LHC, JINST 3 (2008) S08005 doi: 10.1088/1748-0221/3/08/S08005
[6] ALICE Collaboration, The ALICE experiment at the CERN LHC, JINST 3 (2008) S08002 doi: 10.1088/1748-0221/3/08/S08002
[7] A A Ayllon et al. "FTS3: New Data Movement Service For WLCG", 2014 J. Phys.: Conf. Ser. 513 032081 doi: 10.1088/1742-6596/513/3/032081
[8] XRootD project page: http://www.xrootd.org/

[9] K. Bloom et al. "Any Data, Any Time, Anywhere: Global Data Access for Science", 2015 arXiv:1508.01443

[10] CMS Software framework, http://cms-sw.github.io/

[11] P Agostini et al. "The Large Hadron–Electron Collider at the HL-LHC", 2021 J. Phys. G: Nucl. Part. Phys. 48 110501 DOI: 10.1088/1361-6471/abf3ba

[12] A P Millar et al. "dCache, agile adoption of storage technology", 2012 J. Phys.: Conf. Ser. 396 032077 doi: 10.1088/1742-6596/396/3/032077

[13] Apache Hadoop Project page: https://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-hdfs/HdfsDesign.html

[14] Lustre Project page: https://www.lustre.org

[15] DPM Project page: https://lcgdm.web.cern.ch/dpm

[16] Peters A J and Janyst L. "Exabyte Scale Storage at CERN", 2011 J. Phys.: Conf. Series 331 052015 doi: 10.1088/1742-6596/331/5/052015

[17] Ceph Project page: https://ceph.io/

[18] Caltech Tier2 page: https://tier2.hep.caltech.edu

[19] UCSD page: https://ucsd.edu

[20] E. Fajardo et al, "Moving the California distributed CMS XCache from bare metal into containers using Kubernetes" EPJ Web of Conferences 245, 04042 (2020). DOI: 10.1051/epjconf/202024504042

[21] Susmit Shannigrahi et al. "Named Data Networking in Climate Research and HEP Applications" 2015 J. Phys.: Conf. Ser. 664 052033 doi 10.1088/1742-6596/664/5/052033