Exploring Future Storage Options for ATLAS at the BNL/SDCC facility

Qiulan Huang[1], *Vincent* Garonne[1], *Robert* Hancock[1], *Douglas* Benjamin[1], *Carlos* Gamboa[1], *Shigeki* Misawa[1], *Zhenping* Liu[1], on behalf of the ATLAS Collaboration

[1]Brookhaven National Laboratory, 2 Center St, Upton, 11973, US

Abstract. The ATLAS experiment is expected to deliver an unprecedented amount of scientific data in the High Luminosity(HL-LHC) era. As the demand for disk storage capacity in ATLAS continues to rise steadily, the BNL Scientific Data and Computing Center (SDCC) faces challenges in terms of cost implications for maintaining multiple disk copies and adapting to the coming ATLAS storage requirements. To address these challenges, the SDCC Storage team has undertaken a thorough analysis of the ATLAS experiment's requirements, matching them to suitable storage options and strategies, and has explored alternatives to enhance or replace the current storage solution.
This paper aims to present the main challenges encountered while supporting big data experiments such as ATLAS. We describe the experiment's specific requirements and priorities, particularly focusing on the critical storage system characteristics needed for the high-luminosity run and how the key storage components provided by the Storage team work together: the dCache disk storage system; its archival back-end, HPSS; and its OS-level backend Storage. Specifically, we investigate a novel approach to integrate Lustre and XRootD. In this setup, Lustre serves as backend storage and XRootD acts as an access layer frontend, supporting various grid access protocols. Additionally, we also describe the validation and commissioning tests, including a comparison between dCache and XRootd in performance. Furthermore, we provide a performance and cost analysis comparing OpenZFS and LINUX MD RAID, evaluate different storage software stacks, and showcase stress tests conducted to validate Third Party Copy (TPC) functionality.

1 Introduction

The BNL Scientific Data and Computing Center (SDCC) plays a crucial role in developing and operating storage services for various large scale scientific experiments, including ATLAS[1], RHIC[2], BelleII[3], DUNE[4], NSLS-II[5]. All of these experiments involve data intensive applications. Currently, BNL SDCC provides a total storage capacity of approximately 150 PB in disk storage and 220 PB in tape storage. The scientific data generated by supported programs is rapidly increasing, which will eventually push the

SDCC into the Exabyte scale in the coming years. Notably, the data produced by the ATLAS experiment will contribute significantly to this large volume of data. As the demand for disk storage capacity in ATLAS continues to grow steadily, the cost of supporting multiple disk copies and the future ATLAS storage requirements poses new challenges for SDCC. To address these challenges and ensure efficient and cost-effective data analysis, the storage team has undertaken a comprehensive analysis of the specific requirements of the ATLAS experiment, aligning them with suitable storage options and strategies, while exploring alternative solutions to complement or replace the existing storage setup.

This paper focuses on the main challenges presented by supporting multiple big data experiments such as ATLAS. We describe the requirements and priorities of ATLAS, highlighting the critical characteristics needed for the high-luminosity run. Furthermore, we explain how the key storage components work together, namely the dCache disk storage system and its OS-level backend Storage. In particular, we investigate a new solution that integrates Lustre[6] and XRootD[7]. In this setup, Lustre serves as backend storage and XRootD acts as the access layer frontend supporting various grid access protocols. Additionally, we discuss the validation and commissioning tests, including a comparison between dCache[8] and XRootD in performance. Moreover, we present a performance and cost comparison of OpenZFS[9] and Linux software RAID(MD RAID)[10], evaluate different storage software stacks, and provide details on stress tests conducted to validate Third Party Copy (TPC) [11] functionality.

2 ATLAS storage requirements

The ATLAS experiment at the LHC generates petabytes of data distributed among hundreds of computing sites worldwide. To establish a consistent and standardized data access interface, ATLAS specifies the requirements for SE (Storage Element) functions[12]. These requirements encompass aspects such as space tokens, checksums, storage technologies and protocols. Notably, ATLAS has transitioned to require two protocol supports, namely WebDAV and XRootD.

Regarding storage technologies, the recommended options include dCache, EOS[13] and XRootD. dCache is an advanced system that enables seamless access to files stored on disk or on magnetic tape drives within hierarchical storage managers (HSMs). The current storage solution at BNL is based on dCache. EOS, developed by CERN, serves as another distributed disk storage solution. CERN currently stores hundreds of Petabytes (PB) of data using EOS. XRootD is a high performance data system widely adopted by several science experiments on the Open Science Grid(OSG). XRootD operates both as software and a protocol, fulfilling various use cases:1) Exporting an existing file system through multiple protocols, 2) Enabling data federation, and 3) Providing caching services.

Irrespective of the storage solutions employed, it is imperative to ensure that they align with the present and future requirements of ATLAS. Additionally, any changes made to the storage services need to be transparent to ATLAS operations. These priorities are specifically summarized as follows.

- Proven performance and capacity scalability
- Protocol support(WebDAV/XRootD)
- High service availability and reliability (>99%)
- Cost efficient system architecture and implementation
- Sustainable operational costs

3 Storage components

The complete storage system may be implemented as one software package or a set of software packages working in concert. It can be divided into three distinct layers, as shown in Fig.1. These layers, from bottom to top, include: 1) Backend Storage Layer: create the storage "blocks" used by the storage system to store data; 2) Unified Storage System Layer: Organises the storage blocks provided by the backend into a coherent and unified storage space for storing data; 3)Access Layer: Provide access protocol support for clients.

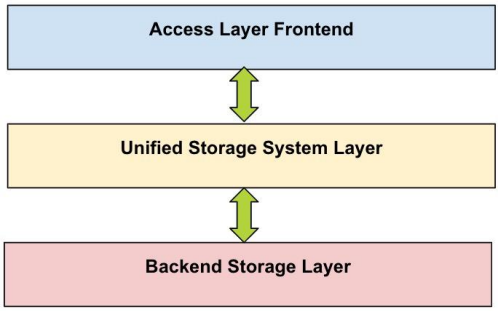

Fig. 1. Storage components

4 Storage evaluation

In order to assess storage software stacks, our evaluation focuses on the components situated within the three layers outlined in Figure 1. Regarding the backend storage layer, we consider two approaches. The first involves solutions at the OS level like Linux software Raid (MD RAID) and OpenZFS. The second approach pertains to software-defined solutions like Lustre and Ceph. Therefore, the targets for evaluation encompass MD RAID, OpenZFS, Lustre and Ceph.

As dCache and XRootD are recommended storage technologies that meet the ATLAS storage requirements, we also evaluated them. Figure 2 illustrates the evaluated components across different layers.

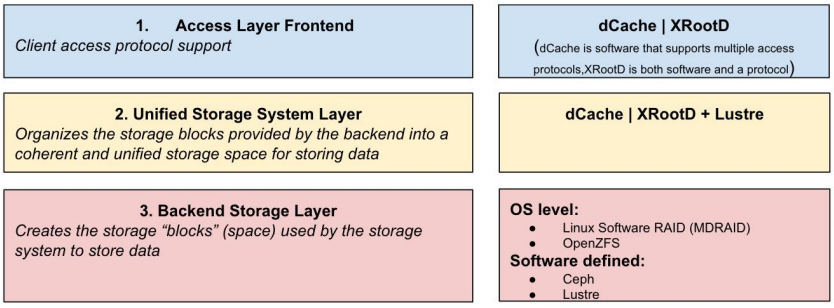

Fig. 2. Evaluated storage components

Ceph is well-known for providing an erasure-code function, which allows Ceph to achieve either greater usable storage capacity or increased resilience to disk failure for the same number of disks versus the standard replica method. However, early studies showed that Ceph was not considered for long-term objectives and needs at the US Tier-1. The main reason (at that time) was the certain limitations became apparent. Hardware requirements for storage (DB/WAL), memory (RAM) and CPU resources were significantly higher than

for other storage solutions, resulting in increased costs and resource allocation. In the case of a usable capacity of 20PB disk storage, OpenZFS presented a more cost-effective option. Furthermore, performance evaluations revealed that streaming sequential performance in Ceph only reached 40% of the hardware capabilities. This performance gap raised concerns about meeting ATLAS data processing and analysis requirements. Ease of management was another aspect that played a role in the decision. Troubleshooting and managing the Ceph service in a containerized environment introduced complexities that could potentially hinder efficient operations.
Considering these factors, the evaluation will focus on the following alternative storage solutions, which were pursued instead, and could better address the performance, cost, and ease of management requirements.

4.1 Alternative storage solutions and evaluation

Lustre is an open source, POSIX-compliant distributed parallel file system, commonly used in large-scale cluster computing environments, meeting the requirements of HPC. A novel approach involves integrating Lustre and XRootD, where Lustre serves as the backend storage, while XRootD acts as the access layer frontend.
This integration enables the exporting of data from an existing network storage solution, such as Lustre and providing both the XRootD and WebDAV protocols. Fig. 3 illustrates the workflow of this innovative solution. The I/O requests come from users' jobs or the experiment data management system, Rucio[14]. Thanks to Lustre's full POSIX compatibility, this new solution can provide both POSIX access via Lustre and grid job access interface via XRootD standalone servers.

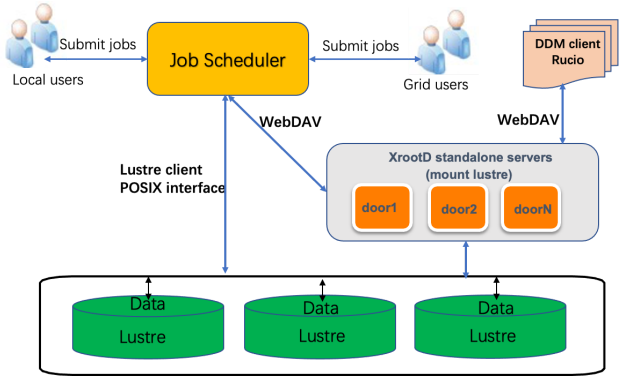

Fig.3. The workflow of XRootd+Lustre

This new storage solution successfully satisfies all the specified ATLAS storage requirements. To validate and evaluate the new storage solution and software stack, we have set up a testbed consisting of 10 servers with identical hardware specifications: 36 CPU cores (Intel(R) Xeon(R) Gold 6254 CPU @ 3.10GHz), 384GB RAM, 50Gbps network bandwidth, and one JBOD with 102x14TB disk drives. Two large-scale distinct storage systems have been established, with each system offering a storage capacity of 5PB. 5 servers are configured as Lustre OSS servers (with MD RAID as the backend storage layer) and the other 5 servers are configured as dCache pool servers (with OpenZFS/Lustre storage as the backend storage layer).
Fig.4 and Fig.5 show the architecture of Lustre and dCache Deployment. Figure 5 demonstrates that multiple combinations of the unified storage layer and backend storage

are possible. Therefore, we discuss three alternative configurations to evaluate the full storage stack as outlined in Fig.2.

- dCache with OpenZFS pools
- dCache with Lustre storage pools
- XRootD with Lustre storage.

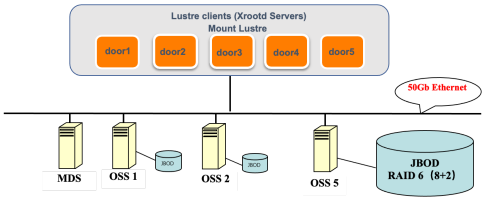

Fig.4. Lustre testbed deployment

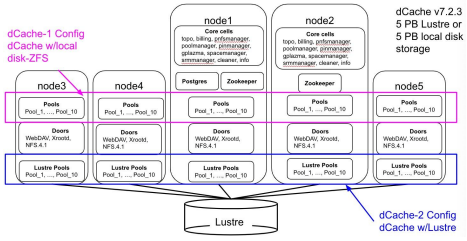

Fig.5. dCache testbed deployment

4.2 Performance Evaluation

A series of stress tests were conducted to evaluate the performance of the alternative storage solutions. The objective of these stress tests was to saturate the IO throughput and maintain a sustained peak rate across different storage configurations. The tests were orchestrated and controlled using the File Transfer Service (FTS), by submitting a substantial number of Third Party Copy (TPC) transfers to simulate a realistic load with production data. During these tests, a total of 500,000 files were utilized, with a maximum limit of 1200 concurrent transfers.

4.2.1 Stress test for Third Party Copy (TPC) validation

In order to simulate a realistic ATLAS load in our testbed, we submitted the TPC read/write tests with production data. Fig.6 shows the data flow of the tests.

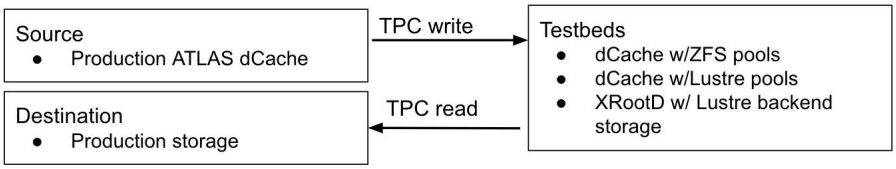

Fig.6. dCache testbed deployment

The result of the TPC write test is presented in table 1. From table 1, dCache configured with a Lustre backend (dCache w/Lustre) got the best IO performance and the IO traffic of XRootD configured with the Lustre backend (XRootD w/Lustre) is ~1.5 times that of dCache configured with a ZFS backend (dCache w/ZFS).

Table 1. TPC write test result

Davs TPC	XRootD w/Lustre	dCache w/ZFS	dCache w/ Lustre
IO traffic	**3.1GB/s per door**	**+2.0GB/s per door**	**+3.8 GB/s per door**

Davs TPC	XRootD w/Lustre	dCache w/ZFS	dCache w/ Lustre
CPU usage	**<10% per door**	**~40% per door**	**~68%**
Success rate	**>98.5%**	**>99.4%**	**>98%**

During the TPC write testing, various adjustments and configurations were implemented to optimize the performance. This included tuning the FTS limit value, which determines the maximum number of active requests. Additionally, the encrypted transfer feature on dCache was disabled to minimise any potential performance bottlenecks. During the FTS limit value tuning tests, we observed that when the FTS limit value was set to less than 150, the performance of dCache with ZFS improved as the value increased. The peak IO traffic reached approximately 2 GB/s per door. Similarly, when the FTS limit value was set to less than 600, the performance of XRootD with Lustre improved as the FTS limit value increased, with a peak IO traffic of around 3.1 GB/s per door. Regarding the tests conducted to disable encrypted transfers, the results indicated that no significant improvement was observed with dCache version 8.2.15. Thus, disabling the encrypted-transfers did not yield substantial performance gains in this particular configuration.
The TPC read test result is shown in Table 2. The read performance of XRootD with Lustre is higher than those of dCache with ZFS and dCache with Lustre

Table 2. TPC read test result

Davs TPC	XRootD w/Lustre	dCache w/ZFS	dCache w/ Lustre
IO traffic	**~2.3GB/s**	**~1.15GB/s**	**~1.2GB/s**
CPU usage	**<3% per door**	**<3% per door**	**<3% per door**
Comments	**1)XRootD+Lustre gets best read performance, about 50% higher than dCache+ZFS and dCache+Lustre pools. 2) dCache with ZFS and Lustre pools perform about the same.**		

4.2.2 Backend Storage evaluation: OS Level

As illustrated in Section 3, backend storage evaluation at the OS level includes MD RAID and OpenZFS. In order to compare MD RAID and OpenZFS, a series of tests were conducted. Fig.7 and Fig.8 shows MD RAID performs better in random read and write, while OpenZFS performs better in sequential read and write except when the MD RAID is configured with 10x10. For capacity overhead, MD RAID gets less capacity overhead for similar configuration, which is shown in Table 3.

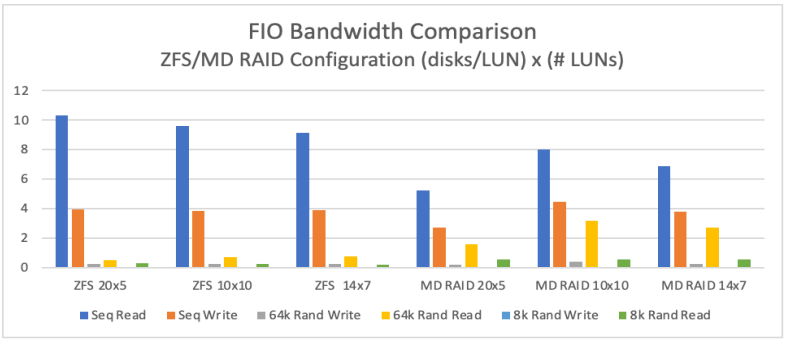

Fig.7. IO Bandwidth comparison between MD RAID and ZFS

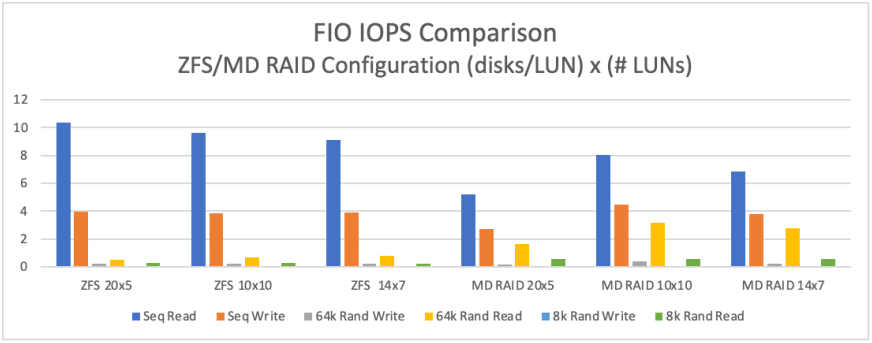

Fig.8. IOPS comparison between MD RAID and ZFS

Table 3. Capacity comparison (Configurations for one JBOD with 104 disk drivers)

Test Name	ZFS 20x5	ZFS 10x10	ZFS 14x7	MD RAID 20x5	MD RAID 10x10	MD RAID 14x7
Full Capacity (TiB)	**1132**	**970**	**1024**	**1150**	**1020**	**1071**
Overhead Factor	**1.148**	**1.339**	**1.269**	**1.133**	**1.286**	**1.214**

Based on the test results, both backends have advantages and disadvantages. For example, MD RAID rebuilds faster on very full disk LUNs, with no performance penalty while capacity usage is above 85%, and less capacity overhead for a similar configuration. On the other hand, OpenZFS has better data integrity: block checksums, auto-healing corrupt data, better IO performance in sequential read/write, separate file systems in the same pool that can be tuned to different IO access patterns and a dRAID feature that can significantly lower rebuild times to reduce disk failures.

In view of all the above, we have chosen OpenZFSas the backend storage for the new hardware of the ATLAS dCache storage system at the SDCC.

4.2.3 Checksum calculation in dCache and XRootD

One of the purposes of the tests was to compare the checksum calculation between dCache and XRootd. dCache calculates checksums dynamically as the file is received or written to disk, while XRootD calculates the checksum after the file has been written to disk. In this way, the XRootD checksum calculation introduces additional IO traffic, resulting in an increased load on the network and backend storage servers in terms of CPU utilization, disk usage, and other resources, and needs more gateway and tunings, which can saturate the storage backend IO bandwidth.
By analyzing the log files of FTS jobs, we observed errors during TPC write tests both in XRootD with Lustre and dCache with the ZFS configuration, most of which are checksum related issues such as checksum timeout and HTTP 500 errors. Tests show checksum timeout errors happen when there are a large number of active requests on FTS (above 1000). The HTTP 500 error can be fixed by increasing the maximum number of checksum calculations that may run at the same time for XRootD.

5 Conclusions

This paper focuses on reassessing the current storage implementation and exploring future storage solutions to meet the current and upcoming HL-LHC requirements for ATLAS.
We discuss the various alternative storage solutions and evaluate the performance. The evaluation tests show some findings:

- XRootD + Lustre gets better IO performance than dCache+ZFS for TPC transfer. TPC write performance of XRootD with Lustre is ~1.5 times dCache with ZFS, and TPC read performance of XRootD with Lustre is ~2 times dCache with ZFS.
- ZFS brings more advantages compared to MD RAID.
- dCache dynamic checksum calculation behaves better compared to XRootD.

While performance is not everything, there are some other important factors to be considered in terms of stability and operational experience. Based on our findings, we have chosen the dCache+ZFS configuration as our preferred storage option in the medium term. However, it is important to note that while Ceph may not be suitable for every use case at the US Tier-1, it can still be a viable solution for specific scenarios and offer benefits (e.g., HL-LHC R&D project). XRootD with Lustre exhibits better performance, although further validation is required for various production workflows.

References

1. ATLAS Collaboration, The ATLAS Experiment at the CERN Large Hadron Collider, JINST 3 (2008) S08003
2. RHIC experiment, https://www.bnl.gov/rhic/
3. Belle II experiment, https://www.belle2.org/
4. DUNE experiment, https://www.dunescience.org/
5. M. Rakitin, S. Campbell, D. Allan, T. Caswell, D. Gavrilov, M. Hanwell et al. 2022 J. Phys.: Conf. Ser. 2380 012100
6. Lustre web site, http://www.lustre.org
7. A. Dorigo , P. Elmer, F. Furano and A. Hanushevsky. XROOTD/TXNetFile: a highly scalable architecture for data access in the ROOT environment. WSEAS Transactions on Computers (2005)

8. M. Tigran, A. Olufemi, P. Fuhrmann, V. Garonne, D. Litvintsev, P. Millar, A. Rossi, M. Sahakyan, et al. dCache-Storage for advanced scientific use cases and beyond. EPJ Web of Conferences 214, 04042 (2019)
9. ZFS on Linux, Online: http://zfsonlinux.org/
10. MD RAID: Linux software RAID, https://www.thomas-krenn.com/en/wiki/Linux_Software_RAID_Information
11. B. Bockelman, A. Ceccanti, F. Furano, P. Millar, D. Litvintsev, A. Forti. Third-party transfers in WLCG using HTTP. EPJ Web of Conferences 245, 04031 (2020)
12. ATLAS requirements on SE functions: https://twiki.cern.ch/twiki/bin/viewauth/AtlasComputing/StorageSetUp#Protocols
13. A.J. Peters, E.A. Sindrilaru and G. Adde. J. Phys.: Conf. Ser. 664 042042(2015)
14. M. Barisits et al. Rucio - Scientific data management, Comput. Softw. Big Sci. 3 (2019) 11