The LHCb HLT2 Storage System: A 40-GB/s System Made of Commercial Off-the-Shelf Components and Open-Source Software

Pierfrancesco Cifra, Francesco Sborzacchi, Niko Neufeld, *Member, IEEE*, and Frédéric Hemmer, *Member, IEEE*

***Abstract*— The Large Hadron Collider beauty (LHCb) experiment is designed to study differences between particles and antiparticles as well as very rare decays in the charm and beauty sector of the standard model at the Large Hadron Collider (LHC). With the major upgrade done in view of Run 3, the detector will read out all events at the full LHC bunch-crossing frequency of 40 MHz. The LHCb data acquisition (DAQ) system will be subject to a considerably increased data rate, reaching a peak of 40 Tb/s. The second stage of the two-stage filtering consists of more than 10 000 multithreaded processes, which simultaneously write output files at an aggregated bandwidth of 100 Gb/s. At the same time, a small number of file-moving processes will read files from the same storage to copy them over to tape storage. This whole mechanism must run reliably over months and be able to cope with significant fluctuations. Moreover, for cost reasons, it must be built from off-the-shelf components. In this article, we describe LHCb's solution to this challenge. We show the design, present reasons for the design choices, the configuration and tuning of the adopted software solution, and present performance figures.**

***Index Terms*— Ceph, data acquisition (DAQ), distributed file system, storage.**

I. Introduction

AFTER the major upgrade [1] done in view of Run 3, the Large Hadron Collider beauty (LHCb) experiment [2] will read out the entire detector at a frequency of 30 MHz, resulting in a data rate of 40 Tb/s. One of the main goals of the LHCb upgrade is the transition to a full software event trigger [3]. This requires a new readout system, a new event builder, and a substantially upgraded purely software-based event filter system to acquire, assemble, and select events at the full rate of 30 MHz. There are clear advantages to having a purely software-based trigger. The software can be easily modified, allowing unprecedented flexibility as and when the LHCb physics program changes. It also has the advantage that computing power is readily upgradeable and benefits from the ability to purchase more CPU power for the same price at a later stage. A first filter level, called HLT1, will perform a fast reconstruction and selection of the events to reduce the event rate to 1 MHz, and the second filter level, called HLT2, will asynchronously read events from HLT1 buffer 100% of the time and perform a full reconstruction and selection, resulting in a final reduced and constant event rate of ∼100 kHz, producing mast data files (MDF) [4] files containing raw data. At the same time, a dedicated cluster will simultaneously read those files from the same buffer area to move and store them permanently over EOS open storage (EOS) tape storage [5]. The size of the HLT2 buffer is determined by the need to cover several runs and days. It needs to be large enough to give the experts time to restore normal running conditions in the case of failure of one component of the data acquisition (DAQ) chain. The bandwidth is most important: 4000 producers will write files to the HLT2 buffer at an individually low rate, while a small number of consumers will read, merge, perform check operations over those files, and push them permanently over EOS storage. The HLT2 storage infrastructure needs to be able to sustain an input rate of 10 GB/s and an output rate of 10 GB/s simultaneously. Fig. 1 shows the LHCb trigger layout. Nevertheless, the reliability aspect needs to be taken into account since a failure in the buffer will directly have a negative impact on the data taking. The systems have to be capable of sustaining the loss of several disks (in general of a full node) to give us time enough to intervene and recover from the failure. In this article, we describe LHCb's solution to this challenge, using technologies available today and taking into account the economic aspect. Before the spread of the SARS-CoV-2 pandemic and the global hardware shortage, the cost of a commercial out-of-the-box solution with the same disk space, performances, and reliability was around U.S. $3M (based on informal quotes we got from

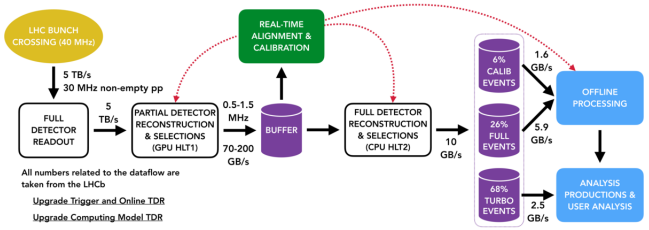

Fig. 1. LHCb trigger layout. The HLT2 storage area is placed immediately after the HLT2 full detector reconstruction and selection.

Manuscript received 2 November 2022; revised 25 January 2023; accepted 26 January 2023. Date of publication 30 January 2023; date of current version 16 June 2023.

Pierfrancesco Cifra is with the Nikhef National Institute for Subatomic Physics, 1098 XG Amsterdam, The Netherlands (e-mail: pierfrancesco.cifra@cern.ch).

Francesco Sborzacchi is with INFN-LNF, 00044 Frascati, Italy.

Niko Neufeld and Frédéric Hemmer are with CERN, Meyrin, 1211 Geneva, Switzerland.

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TNS.2023.3240882.

Digital Object Identifier 10.1109/TNS.2023.3240882

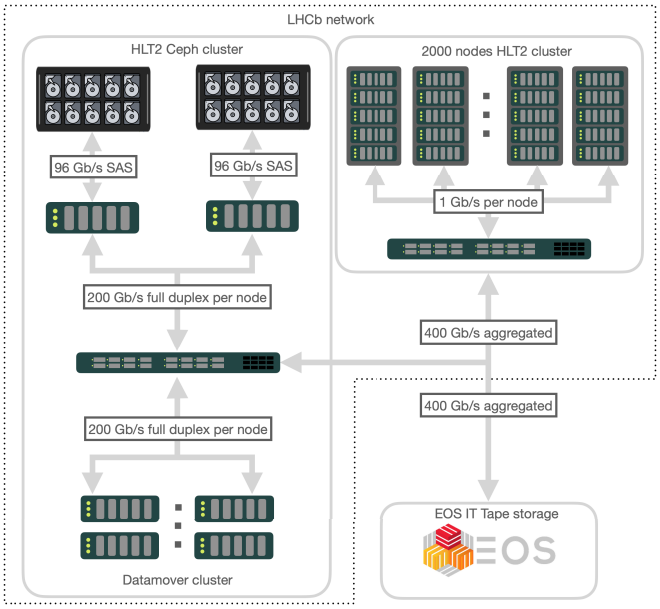

Fig. 2. System architecture of the HLT2 storage system. The 2000 nodes in the HLT2 cluster will write to the buffer area, while the datamover cluster will read the same files and move them to the EOS IT tape storage.

several vendors). It is likely that this could have been lower in a competitive call for tender, but our budget was an order of magnitude smaller, which seemed to preclude any commercial solution at the time. The rest of this article is organized as follows. In Section II, we give a description of present similar storage systems. In Section III, we present the system architecture of the storage infrastructure and the configuration and tuning of the storage backend software adopted. In Section IV, we present the performance and usage results of the system. Finally, in Section V, we summarize our conclusions.

II. Storage Systems for DAQ in High-Energy Physics

For large experiments in High-Energy Physics, the buffering of data between the experimental area and the permanent storage site is essential to avoid dataloss in the case of an interrupted connection with Tier0 storage. Such systems have been built for previous Runs; different technical possibilities have been explored, and open-source or commercial solutions have been adopted in the past by the various experiments at CERN. At the Compact Muon Solenoid (CMS) experiment [6], the Storage and Transfer System was in charge of moving the data to tape storage. The storage system was based on Lustre cluster file system [7], which was capable of delivering an aggregated throughput of 7 GB/s [8]. For Run 3, the same system will be replaced by a more powerful similar one, which will likely meet 31–61 GB/s throughput and total storage of 2–4 PB [9]. Alice experiment [10], opted for a commercial solution [11], based on the Fiber Channel and storage arrays capable of delivering a throughput of 4.5 GB/s in writing and 2.5 GB/s in reading. The cost of its underlying hardware makes it no longer feasible to scale with the number of file system clients needed in Run 3 [12], and likely candidate solutions to replace this system are based on Lustre or IBM GPFS [13]. In LHCb, for Run 2, there was no such storage system: the files produced by HLT2 were locally stored on the local disks of the servers of the farm and moved over to permanent storage [14]. With this approach, it was not possible to keep the data taking alive in the case of disconnections with the permanent storage systems, since the disk space on the servers was limited. As the data at the output stage are valuable physics data that must be retained, a failing disk must not lead to dataloss. The disk buffer is an important element to decouple the data-taking of the experiment from possible issues in the downstream data distribution. We investigated the possibility of a solution based on commercial off-the-shelf hardware and a POSIX-compliant file system built on top of a distributed object store. Ceph is not new in High-Energy Physics; however, we decided to optimize cost and use an erasure-coded configuration, as opposed to a more conventional, replica-based one. The concern with replication is its consumption of storage, generally 3× the core capacity, while erasure coding requires a smaller overhead to ensure reliability at the cost of CPU power. To the best of our knowledge, this has not been attempted at the scale and throughput we describe in this article.

III. HLT2 Storage System Architecture

Given the requirements of a storage area capable of achieving a throughput of 10 GB/s in input and output simultaneously, we decided to set our goals at 20 GB/s for both reading and writing to have a factor two error margin. In our use case, it is pointless to set a higher margin since the network connectivity from the HLT2 farm to the storage area is limited to 40 GB/s by design. It also needs to expose a POSIX interface to the clients to allow the writing and reading of MDF files, and it has to be sufficiently large to act as a buffer between the HLT2 computing cluster, and the small set of nodes in charge of the file-moving over the EOS tape storage. For the backend software, our choice was oriented toward an open-source solution. In commercial storage software, it is common to have term-capacity licensing, incurring huge costs for a system of this size. We were also mainly interested in the available monitoring capabilities and the support is given by the community for the chosen software. Among the various open-source solutions for such a system, we have found the Ceph [15] software-defined storage as best suited to our purpose. The version of Ceph used is 17.2.0 (Quincy). Some preliminary tests were done on old infrastructure to have an overview of Ceph, and once convinced, we bought the new dedicated hardware described later. In this section, we describe the system architecture of the storage buffer for HLT2. We give an overview of the backend software adopted, describing the configuration we used, its tuning, and the hardware specifications.

A. Ceph

Ceph is an open-source object-based parallel distributed storage system and bases its storage capabilities on three main components.

1) A cluster of object storage daemons (OSDs) that collect and store all data and metadata on the physical devices.

This object storage cluster composed by the OSDs is seen by Ceph clients and metadata servers (MDS) as a single logical object store and namespace called reliable autonomic distributed object store (RADOS) [16].

2) A set of MDS which manages the namespace and ensures security, consistency, and coherence.
3) Controlled replication under scalable hashing (CRUSH) [17], a pseudorandom data distribution function used to store object replicas according to a configured replication policy and failure domain. Ceph first maps objects into placement groups (PGs) using a simple hash function and then determines the OSDs to which the object is mapped.

Clients can access the storage through the Ceph file system (CephFS), a POSIX-compliant file system. In CephFS, the MDS daemon manages the directory map and filename information for the file system. Every file is striped into fixed units at the RADOS level and stored on several OSD servers so that CephFS achieves high performance and reliability.

B. Data and Metadata Pool

Ceph mostly bases its storage capability on pools, which are logical groups of objects. Pools can be of two different types: replica, where a fixed number of replicas for each object are stored; and erasure code pools [18], where each object is stored as a set of chunks divided between actual data and erasure codes. Between the two possibilities, our data pool-type choice is oriented toward an $8 + 2$ (eight data chunks, two parity chunks) erasure code pool using the Reed Solomon error correction codes [19]. With this configuration, the object is first divided into eight chunks and then two coding chunks are computed, for recovery. The encoding is performed on packets of a predefined size at a time. This solution allows to save space against a replicated pool, at the cost of computational power. The overhead given by the parity chunks will be a small percentage of the size of the objects, while with a replicated pool, the space needed to store the replicas drastically reduces the real capacity of the system. On the other hand, more CPU power is needed to perform all the encoding calculations. This type of data pool also exploits the high number of spindles by spreading objects across multiple OSDs that can be accessed concurrently, resulting in a faster write for large files. In general, with this configuration, up to two faults simultaneously are tolerated. In practice, the recovery capabilities and a large number of servers allow us to quickly recalculate the lost data chunks and place them again on a healthy node, restoring the normal running conditions of the system. The usable capacity of HLT2 decreases from a raw size of 11 PB to a size of 8.8 PB, which with an input rate of 10 GB/s is enough to cover eight continuous days of data production. We deployed OSDs on a combination of slow and fast devices (HDD and NVMe drives, respectively), placing data on the first ones, and the metadata on the second ones using RocksDB [20]. Each NVMe drive stores the metadata for several OSDs. The metadata pool is used for managing file metadata in a Ceph File System, including inodes, dentries, and the file system hierarchy. We configured this pool (order of magnitude smaller in size than the data pool) with a replication policy (x3) and placed it on the SSD drives present on the servers.

C. Tuning

Particular effort has been made to understand Ceph internals to tune the system and identify possible bottlenecks. On top of the operating system and network, each Ceph component, erasure code profile, OSD, and MDS have been fine-tuned and optimized specifically for our use case: writing and reading large files at the same time. We evaluated the system's behavior with different combinations of parameters and chose the best ones from the measurements.

1) Erasure Code Profile: We tried several erasure code profiles, changing the size of the packet on which the encoding is performed, and we opted for a size of 16 kB. With this value, the encoding will be done on packets of 16 kB size at a time. The $8 + 2$ configuration was chosen because it guarantees a good compromise between redundancy and disk space overhead. The size of the packet on which the coding is done largely impacts the overall performance.

2) Object Storage Daemon: We fine-tuned a few OSD parameters. The number of OSD threads will be adjusted by the system according to the load, with a minimum of six and a maximum of 12 threads. We set the maximum write size at 512 MB and the maximum number of concurrent operations per OSD at 8192.

3) Placement Groups: We tuned the number of PGs. It is recommended to have between 50 and 100 PGs per OSD to balance out resource usage, data durability, and distribution. We set a number of 4096 PGs for the data pool, which, with 720 drives, resulted in 56 PGs per OSD.

4) CephFS: We fine-tuned CephFS's striping unit and object size. We opted for a 64-MB object size and a striping count of 1. In general, every file smaller than the object size will be zero-filled. This will not be the case since MDF files are typically GB large.

5) Client: The CephFS kernel driver is used by clients to mount and access the buffer. It allows mounting CephFS as a regular file system with native kernel performance. The only parameters we tuned were read and write size, set to 64 MB, and the readahead set to 512 MB.

D. Hardware Infrastructure

The system is composed of 12 servers and 12 disk shelves. A disk shelf is a physical enclosure containing multiple disk drives capable of monitoring the status of each individual disk, power supply, temperature, and fans. The hardware specifications for OSD and MDS servers and disk shelves are, respectively, summarized in Tables I and II. On each server, the two network interfaces are configured with IEEE 802.3ad link aggregation [21], ensuring a throughput of 200 Gb/s and removing the single point of failure at the network level. Memory provides a single-bit Error Correcting Code (ECC). Storage systems are sensitive to memory errors; data will always reside in memory first before being written to disk and requires ECC to ensure data integrity. Each node is connected

TABLE I
OSD AND MDS SERVER

Server	Supermicro SYS-220BT-HNTR
CPU	2 x Intel(R) Xeon(R) Gold 6326 CPU @ 2.90GHz
Memory	16 x 32 GB DDR4-3200 ECC
Disk	4 x 1.9 TB Samsung NVMe + 1 TB Samsung EVO 860
OS	RHEL 8, Linux 4.18
Network	Mellanox MT28800 ConnectX-5 Ex, PCIeGen4 x16
Storage	Broadcom LSI SAS38xx Fusion-MPT 12GSAS

TABLE II
DISK SHELF CONFIGURATION

Enclosure	Western Digital Ultrastar Data 102, 2 Disk controllers
Disks	60 x Ultrastar DC HC550 18 TB

to the disk shelf through serial attached SCSI (SAS) cables for an aggregated bandwidth of 96 Gb/s. Disk connections are redundant as well, due to the multiple SAS cables per node and the double disk controller of the shelves. Drives are equally distributed among the servers, exploiting the zoning capability of the enclosures. Fig. 2 shows the architecture of the whole system, highlighting the network topology and disk connections. This configuration has been designed to ensure maximum performance and reliability.

IV. RESULTS

The whole system has been tested under different running conditions, trying to reproduce the data-taking scenario and simulating failures. I/O operations were performed both from a small set of nodes and from a large cluster composed of 2000 nodes accessing the storage area simultaneously. We tested the system in three different scenarios. In 100% writing, 100% reading, and in a mixed mode with 50% writing and 50% reading. Tests have been carried out by writing to CephFS using the *dd* [22] OS utility with 16-MB buffer size and *pv* [23] for limiting the writing rate in some specific cases.

A. Small Cluster Client Test

The first set of tests was performed using a small cluster of 12 clients connected to the same switch as the Ceph cluster. The goal was to fine-tune the system and to have an estimation of the upper-bound limit for the maximum performance that the system is able to achieve, which, with an average speed of 225 MB/s per disk, the theoretical aggregated maximum bandwidth is 162 GB/s. The most significant parameter was the erasure code data pool size. Fig. 3 reports the maximum throughput according to the size of the packet on which the erasure coding is performed. The 12 clients were used to write or read several streams in parallel. The system achieved a throughput of 120 GB/s when writing and 100 GB/s when reading. Fig. 4 shows the aggregated throughput over time in the two scenarios. We then tested the system in the mixed scenario, observing a throughput of 45 GB/s writing and 25 GB/s reading. Fig. 5 reports results for this test. During this time, the storage area has been filled with more than 1 million files (85 million objects). Results show that this solution is capable of carrying massive loads for a long time; even with disk space usage of 75%, there is no important degradation from the performance point of view. Fig. 6 reports

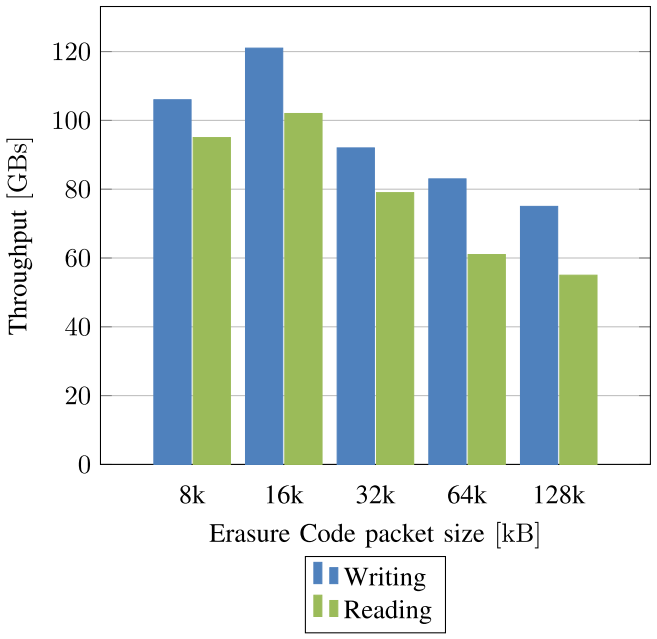

Fig. 3. Maximum throughput observed in the function of the packet size on which the erasure coding is performed. Based on the measurements, we observed that a packet size of 16k provides the best performance.

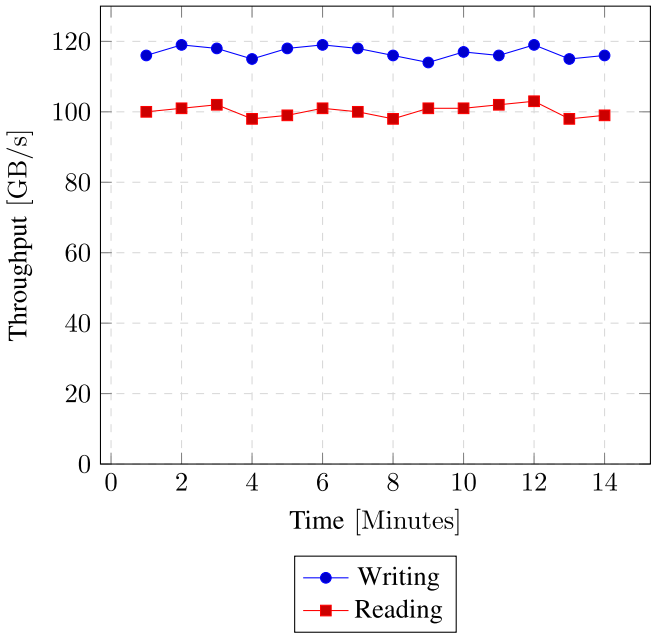

Fig. 4. Throughput observed during the different tests. In the chart, the results are reported for the 100% writing scenario and the 100% reading scenario. The chart presents some oscillations: this is a complex system; hard disk access and seek time, hard disk rotational latency, and network and system load must be considered. In addition, the measurements are the results of samples averaged over a time frame of 1 min.

the aggregated throughput over disk space occupancy. Performance starts to drop after the buffer exceeds 60%, with a significant degradation being observed only after 80%. This is still enough to satisfy our requirements. Many file systems are affected by performance degradation after their occupancy reaches a certain value. Nevertheless, this is a threshold that will unlikely be reached due to the basic idea behind this buffer area: files will be immediately moved to tape storage after they

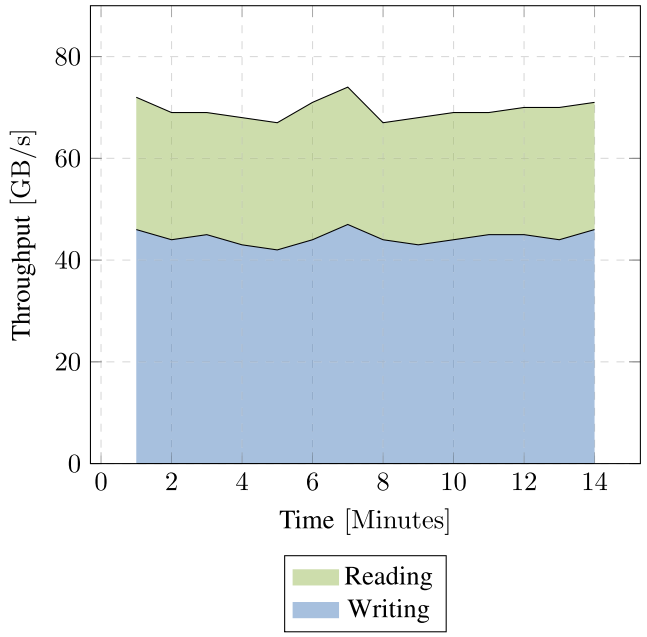

Fig. 5. Aggregated throughput in the mixed scenario with 50% writing, and 50% reading at the same time.

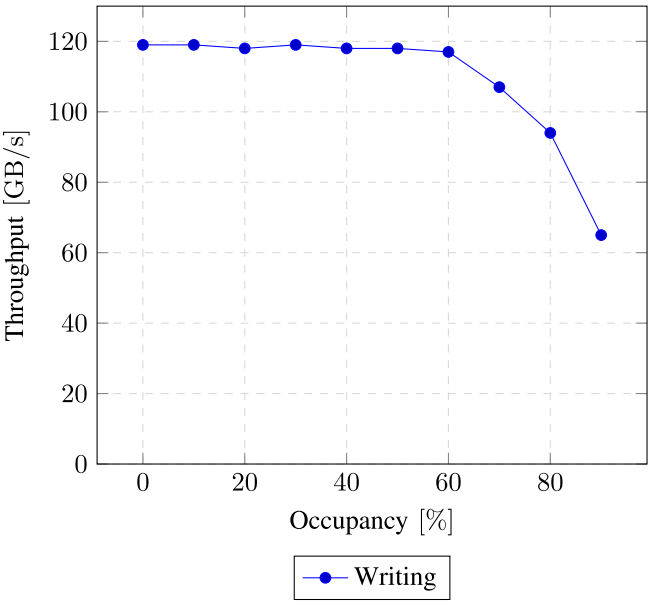

Fig. 6. Chart summarizing how the throughput varies according to occupancy. Performance starts degrading only after 60% of occupancy is reached. Nevertheless, the initial requirements are still widely satisfied.

have been written to it, keeping the occupancy under this limit most of the time.

B. Large-Scale Test

The large-scale test aimed to reproduce the HLT2 trigger system writing to the Ceph storage using the same HLT2 computing infrastructure. The 2000 nodes were used to write a small amount of data while reading from another cluster at the same time. We initially limited both writing and reading to 20 GB/s each. This was our goal to have a factor two error margin against the original goal of 10 GB/s. The system performed correctly, and the results show that it widely satisfies

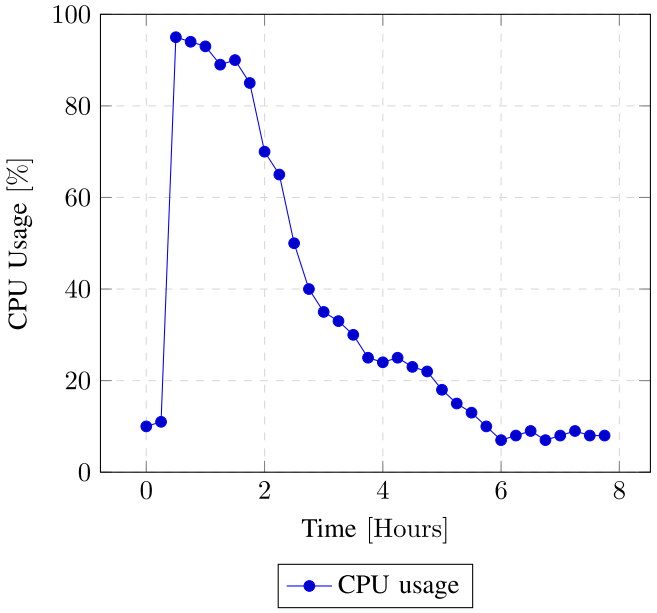

Fig. 7. Chart reports the CPU usage of one node after a failure.

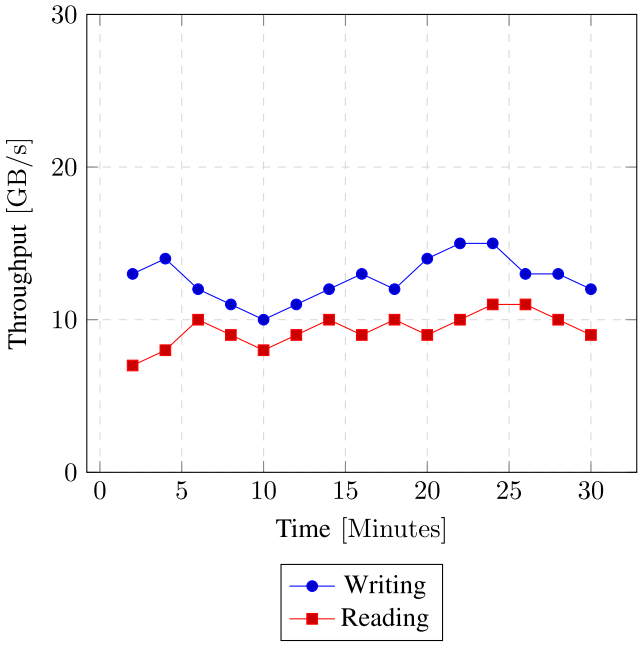

Fig. 8. Performance of the system observed during commissioning.

our requirements, keeping performance uniform over time. The same test has been repeated without limiting the rate. Again, we obtained the same results as in the small cluster test, showing that this solution scales very well according to the number of clients.

C. Node Failure

We tested the system by simulating a failure on one and two nodes simultaneously to evaluate if performance is impacted significantly by the loss of 8% and 16% of OSD, respectively. Failure is a CPU and disk-consuming condition since Ceph needs to recompute all the lost data and parity chunks. CPU usage reaches almost 100% just after the failure, slowly decreasing to normal values after the failure is recovered. The whole process took 6 h, with an occupancy of 50%.

In Fig. 7, we report the CPU usage of one node during recovery from failure. Nevertheless, in the unlikely case of a double failure at the node level, the system is able to provide a minimum performance of 10 GB/s in both writing and reading. The high peak of load during failure justifies the purchase of a considerable amount of CPU power.

D. Commissioning

The system has been successfully commissioned for the start of Run 3 and we had the chance to see how it behaves in a normal running conditions of data taking. We can claim that the implementation of the system exceeds the requirements since, in the mixed scenario, the maximum writing and reading speeds are, respectively, 2.25 and 1.25 times the requirements. In Fig. 8, we report the I/O rate during 30 min of data taking. The input and output rates during this time frame are mainly driven by the current conditions of the DAQ.

V. Conclusion

In this article, we have presented our solution for the LHCb HLT2 storage. The system as it is designed widely satisfies the initial performance and reliability requirements. The maximum achievable aggregated throughput is 70 GB/s and it can sustain the loss of up to two nodes while keeping the minimum performance needed. We can summarize our conclusions in five points.

- This solution scales well with many clients operating the cluster in parallel. This is indeed the optimal running condition for such a system. Ceph is designed to handle several I/O streams in parallel, exploiting the number of available spindles. This perfectly meets our needs since the HLT2 buffer will be accessed simultaneously by thousands of clients.
- The system is able to keep the performance uniform even with high occupancy. This means that if the data transfer to tape storage fails, it is possible to accumulate data for several days and send it later.
- Performances are not significantly impacted when a failure occurs. During data taking, this gives us enough time to recover from the failure without the risk of losing data.
- In normal running conditions, the system is not bounded by the CPU but by the number of spindles. Full CPU power is mainly needed to recover from a failure.
- The proposed solution has a total cost of U.S. $400k for the hardware, with a final price of U.S. $45 per usable TB. The market conditions at the time of the purchase (late 2021) need to be taken into account since the price and availability were strongly affected by the global computer hardware shortage.

Furthermore, its structure, the POSIX compliantness, the use of open-source software, and general-purpose hardware make the proposed solution very versatile, making it very easy to adapt it for other use cases beyond the LHCb one.

References

[1] T. Colombo et al., "The LHCb DAQ upgrade for LHC run3," *IEEE Trans. Nucl. Sci.*, vol. 66, no. 7, pp. 982–985, Jul. 2019. [Online]. Available: https://cds.cern.ch/record/2757329

[2] The LHCb Collaboration, "The LHCb detector at the LHC," *J. Instrum.*, vol. 3, Aug. 2008, Art. no. S08005, [Online]. Available: https://cds.cern.ch/record/1129809

[3] T. Colombo et al., "The LHCb online system in 2020: Trigger-free read-out with (almost exclusively) off-the-shelf hardware," *J. Phys., Conf.*, vol. 1085, Feb. 2018, Art. no. 032041, [Online]. Available: http://cds.cern.ch/record/2665498

[4] M. Frank (Oct. 2006). *Online RAW Data Format*. Accessed: Aug. 20 2022. [Online]. Available: https://edms.cern.ch/ui/file/784588/1/Online_Raw_Data_Format.pdf

[5] A. J. Peters, E. A. Sindrilaru, and G. Adde, "EOS as the present and future solution for data storage at CERN," *J. Phys., Conf.*, vol. 664, Aug. 2015, Art. no. 042042, [Online]. Available: https://cds.cern.ch/record/2134573

[6] The CMS Collaboration et al., "The CMS experiment at the CERN LHC. The compact muon solenoid experiment," *J. Instrum.*, vol. 3, May 2008, Art. no. S08004, [Online]. Available: https://cds.cern.ch/record/1129810

[7] *Lustre*. Accessed: Aug. 20, 2022. [Online]. Available: https://www.opensfs.org/lustre/

[8] J. M. Andre, "Online data handling and storage at the CMS experiment," in *Proc. 21st Int. Conf. Comput. High Energy Nucl. Phys.*, 2015, Art. no. 082009. [Online]. Available: https://cds.cern.ch/record/2016893

[9] The CMS Collaboration et al., (2017). *The Phase-2 Upgrade of the CMS Tracker*. CERN. [Online]. Available: https://cds.cern.ch/record/2272264

[10] The ALICE collaboration et al., "The ALICE experiment at the CERN LHC," *JINST*, vol. 3, May 2008, Art. no. S08002. [Online]. Available: https://cds.cern.ch/record/1129812

[11] *Quantum StorNext Helps CERN Accelerate Research and Discovery*. Accessed: Aug. 20, 2022. [Online]. Available: https://www.quantum.com/globalassets/customer-success/case-studies-pdf/cern-quantum-stornext-helps-cern-accelerate-research-and-discovery-cs00105a.pdf.

[12] P. Buncic, M. Krzewicki, and P. V. Vyvre. (2015). *Technical Design Report for the Upgrade of the Online-Offline Computing System*. Accessed: Aug. 20, 2022. [Online]. Available: https://cds.cern.ch/record/2011297

[13] F. Schmuck and R. Haskin, "GPFS: A shared-disk file system for large computing clusters," in *Proc. 1st USENIX Conf. File Storage Technol. (FAST)*. Monterey, CA, USA: USENIX Association, 2002, pp. 1–15.

[14] M. Frank, C. Gaspar, V. E. Herwijnen, B. Jost, and N. Neufeld "Deferred high level trigger in LHCb: Boost to CPU resource utilization, *J. Phys., Conf.*, vol. 513, Aug. 2014, Art. no. 012006. [Online]. Available: https://cds.cern.ch/record/2055698

[15] A. S. Weil, A. S. Brandt, L. E. Miller, D. E. D. Long, and C. Maltzahn, "Ceph: A scalable, high-performance distributed file system," in *Proc. 7th Symp. Operating Syst. Design Implement. (OSDI)*, Seattle, WA, USA, Nov. 2006, pp. 307–320.

[16] S. Weil et al., "RADOS: Fast scalable reliable storage service for petabyte-scale storage clusters," in *Proc. ACM Petascale Data Storage Workshop (PDSW)*, Nov. 2007, pp. 35–44.

[17] S. A. Weil, S. A. Brandt, E. L. Miller, and C. Maltzahn, "CRUSH: Controlled, scalable, decentralized placement of replicated data," in *Proc. ACM/IEEE Conf. Supercomputing*, Nov. 2006, p. 122.

[18] S. B. Balaji, M. N. Krishnan, and M. Vajha, "Erasure coding for distributed storage: An overview," *Sci. China Inf. Sci.*, vol. 61, Sep. 2018, Art. no. 100301.

[19] B. Sklar, *Reed Solomon Codes*. Englewood Cliffs, NJ, USA: Prentice-Hall, 2001.

[20] S. Dong, "RocksDB: Evolution of development priorities in a key-value store serving large-scale applications," *ACM Trans. Storage*, vol. 17, no. 4, pp. 1–32, 2021.

[21] *IEEE Standard for Information Technology—Local and Metropolitan Area Networks—Part 3: Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications-Aggregation of Multiple Link Segments*, in IEEE Std. 802.3ad-2000, Jun. 2000, doi: 10.1109/IEEESTD.2000.91610.

[22] P. Rubin, D. MacKenzie, and S. Kemp. (Apr. 2022). *GNU Coreutils*. Version 9.1. Accessed: Aug. 20, 2022. [Online]. Available: https://www.gnu.org/software/coreutils/manual/coreutils.html#dd-invocation

[23] *Linux Pipe Viewer*. PV 1.6.6. Accessed: Aug. 20, 2022. [Online]. Available: https://linux.die.net/man/1/pv