Journal of Grid Computing (2025) 23:20
https://doi.org/10.1007/s10723-025-09801-w

RESEARCH

High Throughput Data Transfer System for ALICE Experiment in Run 3

Alice-Florenţa Şuiu · Costin Grigoraş · Nicolae Ţăpuş · Latchezar Betev

Received: 13 February 2025 / Accepted: 13 March 2025

Abstract The European Organization for Nuclear Research (CERN), headquartered in Geneva, Switzerland, operates the Large Hadron Collider (LHC), the world's most powerful particle accelerator. One of the four main LHC experiments is A Large Ion Collider Experiment (ALICE), which is dedicated to studying the properties of quark–gluon plasma, a state of matter that existed shortly after the Big Bang. To enhance its data collection capabilities, ALICE underwent a significant upgrade during the LHC's Long Shutdown 2 (2019–2021). This upgrade focused on modernizing the experiment's detector and data acquisition (DAQ) systems. The result is a vastly improved ability to capture and analyze data from heavy-ion collisions. This article provides an overview of the software and processes used to transfer and manage the experiment's massive datasets and highlights the key achievements and insights gained since the upgraded system became operational.

Keywords Queueing network models · Data acquisition system · Multithreading · Priority queues · Distributed data management system · Monitoring and alerts system

Mathematics Subject Classification (2010) 68M20

The authors would like to thank the ALICE Grid Computing team at CERN for their valuable feedback and discussions, as well as the members of the ALICE Editorial Board for their thorough review and constructive suggestions that helped improve this manuscript.

A.-F. Şuiu · N. Ţăpuş
Faculty of Automatic Control and Computer Science, National University of Science and Technology POLITEHNICA Bucharest, Bucharest, Romania
e-mail: nicolae.tapus@upb.ro

A.-F. Şuiu (✉) · C. Grigoraş · L. Betev
CERN, Geneva, Switzerland
e-mail: alice-florenta.suiu@cern.ch; alice_florenta.suiu@upb.ro; asuiu@cern.ch

C. Grigoraş
e-mail: costin.grigoras@cern.ch

L. Betev
e-mail: latchezar.betev@cern.ch

1 Introduction

The Large Hadron Collider (LHC), the largest and most powerful particle accelerator in the world, is run by the European Organization for Nuclear Research (CERN) [1], one of Europe's top research institutions. CERN, which is situated in Geneva, Switzerland, studies subatomic particles and their interactions through heavy-ion and proton–proton collisions in order to conduct fundamental research into the structure of matter. With detectors that enable physicists to test particle physics theories and observe new phenomena, the LHC is home to four large experiments. A Large Ion Collider Experiment (ALICE) [2] is one of the experiments, which was created especially to investigate heavy-ion collisions and the characteristics of quark–gluon plasma, which was present soon after the Big Bang.

During the Long Shutdown 2 (2019–2021), ALICE upgraded its detector and data acquisition (DAQ) systems, allowing it to collect data at a ∼100 times higher rate than in Run 1 & 2 [3]. Thus, during LHC Run 3 [3], which began in 2022, ALICE processes around 3.5 TB/s of data from its detectors, which is compressed to 200 GB/s and stored on a 150 PB disk buffer for offline processing. The data is processed in real time by a computing farm consisting of two types of nodes: First Level Processors (FLPs), which receive data directly from the detectors and perform initial processing, and Event Processing Nodes (EPNs), which are a second layer of nodes that are equipped with CPUs and GPUs. The EPNs reconstruct and compress events, and produce data needed for detector calibration. Then, they temporarily store the data on high-capacity SSDs before transferring it to remote storage using a tool called EPN2EOS. This system operates under strict constraints [4], being optimized for parallel data transfer and monitoring to ensure smooth operation and continuous data collection. It runs on the EPNs and optimizes the use of their computing resources, allowing them to focus on data collection, compression, and storage [4–6].

This paper provides an overview of the EPN2EOS role in the ALICE data processing workflow, detailing its implementation and features. It also describes the EPN2EOS state machine and its modeling using the Jackson network model [7], followed by the results observed after its deployment into production.

The structure of this article is as follows: Section 2 discusses DAQ systems, highlighting their characteristics and usage at CERN, detailing the current design employed by the ALICE experiment. This section also explains the role of EPN2EOS in the data processing workflow.

Section 3 offers an overview of the key features related to the implementation of EPN2EOS.

Section 4 introduces the concept of queueing theory and provides a characterization of several queueing network models, with one of these models being used to simulate the EPN2EOS.

Section 5 goes further by describing the EPN2EOS in detail, presenting its state machine and modeling it using the Jackson network model.

After giving a technical overview of EPN2EOS, Section 6 analyzes the results obtained by EPN2EOS since its deployment to production.

2 Data Acquisition Systems

DAQ systems are used for collecting, processing, and analyzing data from real-time systems by measuring various physical properties. These systems are composed of both hardware and software components, which can be managed either through direct human interaction or remotely via remote access. A typical DAQ system can be seen in Fig. 1 and includes electrical signal equipment that provides precise data measurement and acquisition, sensors that convert the data into a format understandable by the software, computational nodes that handle the processing and analysis of the collected data, and hardware such as switches, routers, Ethernet links, optical links, or InfiniBand that facilitates data transfer between system components. Additionally, DAQ systems incorporate temporary storage to hold processed results before they are transferred to a persistent storage system located remotely [8].

DAQ systems are specifically designed to continuously and automatically collect data from a physical environment, process it quickly and efficiently, provide accurate real-time measurements, enable fast transfer of processed data to a persistent storage location, and temporarily store the results for a period of time, depending on the data transfer capacity to the persistent storage [8].

At CERN, DAQ systems respect the general architecture and are employed in the major experiments

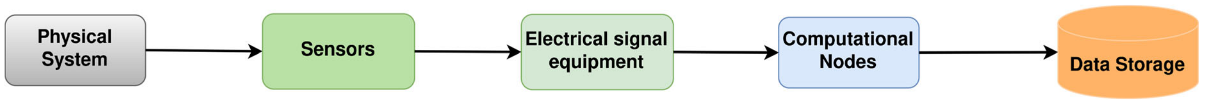


Fig. 1 General architecture of a DAQ system - illustrates the main components, including data acquisition modules, processing units, and data storage elements

along the LHC to ensure accurate and reliable data collection and processing from the detectors.

The following subsection describes the current implementation of the DAQ system used by the ALICE experiment during Run 3, as well as the role of EPN2EOS in the data processing workflow.

2.1 ALICE DAQ System in Run 3

In the Run 3 operation mode, the detector signals are read out continuously. This means that the data readout does not consist of distinct, isolated events, but rather multiple collisions that may overlap in time. Consequently, the fundamental processing unit for the ALICE experiment is not an individual event, but the Time Frame (TF), which typically captures around 10 milliseconds of continuous data-taking.

In order to optimize the architecture, an initial real-time compression step is applied to handle the large volume of data generated by the detectors. This compression is carried out by the First Level Processors (FLPs), which are the closest computational nodes to the experiment and thus receive data directly from it. The FLP cluster is composed of 200 Dell PowerEdge R740 nodes, which are tasked with reading, aggregating, and transmitting data from the detectors to the EPNs [9]. The FLP farm receives data from the detectors at a total rate of 3.5 TB/s across 8000 optical links and performs a preliminary data compression, reducing the rate to 900 GB/s through zero suppression. The FLPs then organize this data into Sub-Time Frames (STFs), which are sent to an available EPN to assemble the complete TF [4].

The EPN farm consists of 275 computing nodes equipped with AMD Rome 32-core CPUs and AMD MI50 GPUs with 32 GB VRAM, and 70 computing nodes equipped with AMD Rome 48-core CPUs and AMD MI100 GPUs with 32 GB VRAM. An InfiniBand physical network supports the transfer of STF data, enabling real-time communication between the FLPs and EPNs [9]. Once an EPN receives and assembles a complete TF, it has 30 seconds to process it. This process involves interpreting the data through various algorithms, ultimately resulting in the creation of Compressed Time Frames (CTFs) [10]. The compressed data is first stored on the EPN's local storage (SSD) before being transferred to the main storage [4].

The storage solution, referred to as EOS [11], is designed with low latency and high capacity, efficiently managing the storage of large volumes of data in file format while also supporting interactive analysis. EOS comprises two primary components: the client-side, which offers a command-line interface and access to a mounted file system, and the server-side, which encompasses components for storing file metadata, the actual data, and a message queue for asynchronous message transmission [12]. The main storage system has a total capacity of 150 PB [4].

The 'Computational Nodes' item from Fig. 1 consists of FLP and EPN nodes shown in Fig. 2. The 'Data Storage' entity is represented by the EOS component.

The EPN2EOS tool operates on each EPN, handling the essential task of managing compressed experimen-

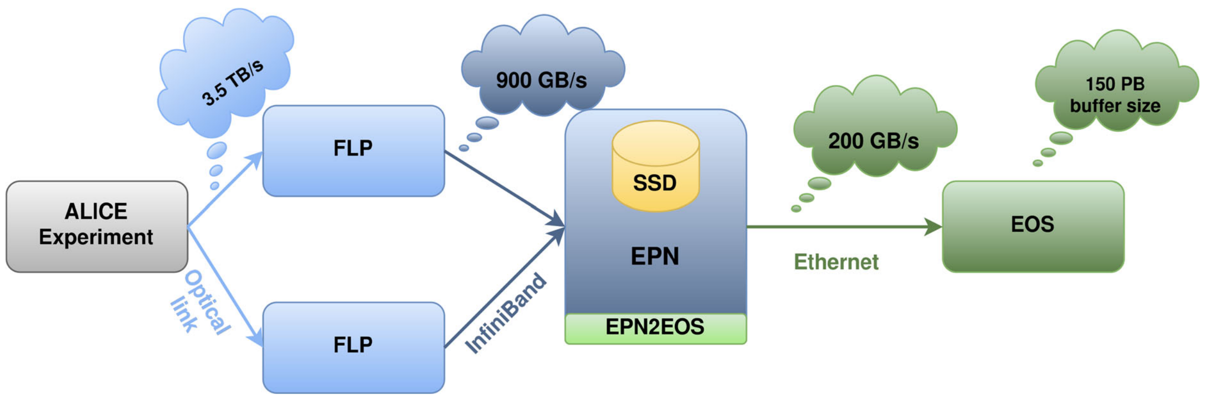


Fig. 2 DAQ architecture for ALICE in Run 3 [4] - represents the DAQ architecture used in the ALICE experiment during Run 3, highlighting the placement of the EPN2EOS component within the data processing workflow

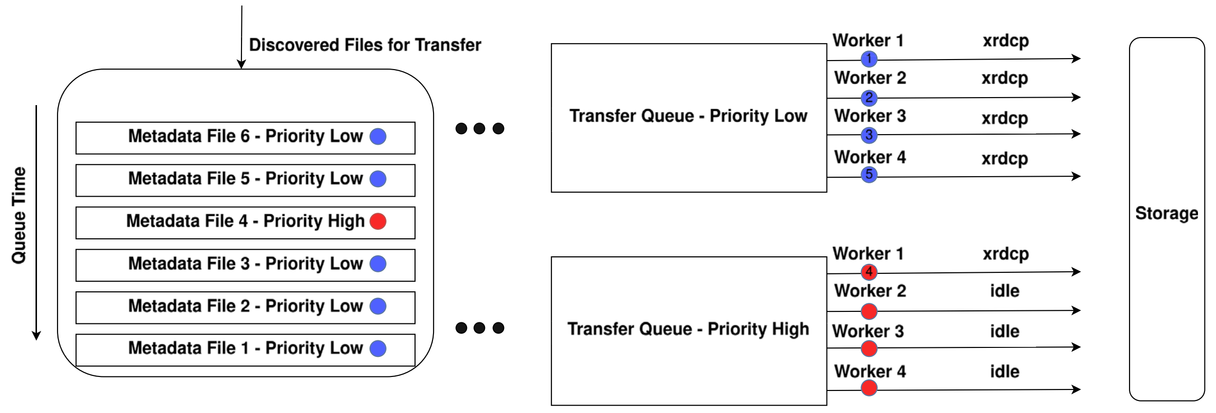

Discovered Files for Transfer
Queue Time
Metadata File 6 - Priority Low
Metadata File 5 - Priority Low
Metadata File 4 - Priority High
Metadata File 3 - Priority Low
Metadata File 2 - Priority Low
Metadata File 1 - Priority Low
Transfer Queue - Priority Low
Transfer Queue - Priority High
Worker 1
Worker 2
Worker 3
Worker 4
xrdcp
idle
Storage

Fig. 3 Transfer watcher diagram [4] - illustrates the flow a data file goes through to be transferred to storage, from the discovery of its associated metadata file by a file event watcher to its placement in the appropriate transfer queue and the actual transfer of the corresponding data file

tal data and facilitating its transfer from the EPNs' local storage to EOS.

3 EPN2EOS Implementation Details

This section presents an overview of the features related to the implementation of the EPN2EOS. More implementation details can be found in the article 'EPN2EOS Data Transfer System' [4].

EPN2EOS is a system written in Java that runs as a daemon service on EPN nodes. It has two main tasks: transferring files containing data collected from the ALICE experiment and registering the metadata associated with each data file in the ALICE catalog [5, 13], a database that keeps track of all data files that have been transferred to the disk buffer. Each file containing collected data has a unique associated metadata file. These metadata files are placed in a specific directory defined in a configuration file used during the service startup. The activity of this directory is monitored by a file event watcher created by EPN2EOS, which, for simplicity, we will refer to as the transfer watcher. The primary function of the transfer watcher is to notify EPN2EOS whenever a new metadata file is added to the directory. Each file is assigned a specific priority, and for each priority, EPN2EOS creates a separate queue where the corresponding files are placed. Additionally, EPN2EOS spawns a separate set of threads to manage the transfer of files from each priority queue [4]. The data transfer process is illustrated in Fig. 3.

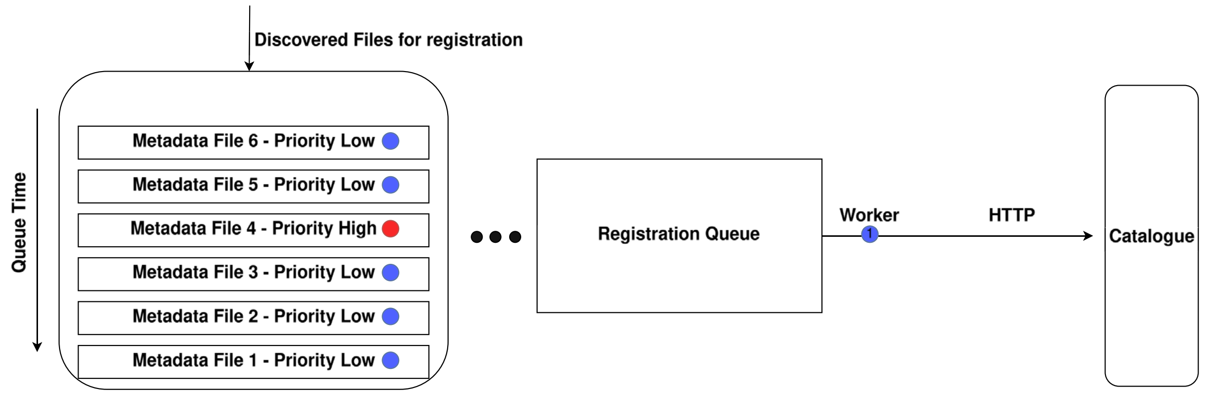

Discovered Files for registration
Queue Time
Metadata File 6 - Priority Low
Metadata File 5 - Priority Low
Metadata File 4 - Priority High
Metadata File 3 - Priority Low
Metadata File 2 - Priority Low
Metadata File 1 - Priority Low
Registration Queue
Worker
HTTP
Catalogue

Fig. 4 Registration watcher diagram - outlines the flow through which the metadata associated with a successfully transferred file is registered in the catalogue

Table 1 Global monitoring metrics [4]

Metric name	Metric description
Services reporting	number of EPN2EOS instances that report metrics to the monitoring system
Queued	number of files in the transfer queues / registration queue
Ongoing	number of active transfers / active registrations
Queued size	total size of files waiting to be transferred
Slots	maximum number of transfers / registrations that can run in parallel per priority
Copy rate	data transmission rate
Success rate	success rate in files per second
Failure rate	error rate in files per second for both the transfer and registration processes

In this example, 6 metadata files have been detected in the directory, 5 with low priority and 1 with high priority. The files with low priority are placed in the transfer queue associated with this priority. The same happens for the high-priority file. Additionally, each transfer queue has 4 workers assigned to handle the file transfers. Since the low-priority queue contains 5 files but only 4 available workers, this means that 4 data files will be transferred in parallel, while the fifth will remain in the queue, waiting for a worker to become available. On the other hand, the high-priority transfer queue has only one file and 4 associated workers. Thus, one of the workers will transfer the data file to storage, while the remaining three workers will stay idle.

After the file transfer is successfully completed, the data file is removed from the local storage of the EPN node, and the metadata file is moved to another directory specified in the previously mentioned configuration file. The activity of this directory is monitored by another file event watcher created by EPN2EOS, which we will refer to as the registration watcher. The primary function of the registration watcher is to notify EPN2EOS whenever a new metadata file that is ready for registration is added to the directory. Then, EPN2EOS queues each metadata file in a registration queue and spawns separate threads for this operation, each thread being responsible for one of the metadata files. The registration path is illustrated in Fig. 4. Here, 6 files have been detected in the directory. These are placed in the registration queue, which has a worker assigned to handle the metadata registration in the catalog. Since there is only one thread, only one file is processed at a time, with the others waiting in the registration queue. After the file registration is successfully completed, the metadata file is removed from the local storage of the EPN node.

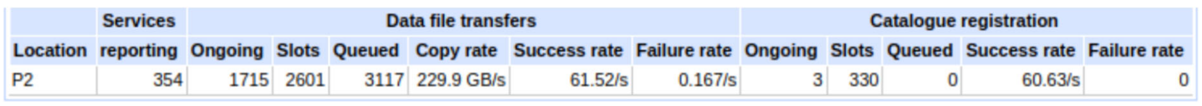
If the data transfer fails, the copy operation is retried after a delay calculated using the Exponential Backoff [4] strategy. Once this delay is determined, the metadata file is re-added to the appropriate transfer queue. Similarly, if the registration process fails, the same retry method is applied.

EPN2EOS is a vital component of the ALICE experiment, making real-time monitoring and detailed logging of its activities essential. The EPN2EOS metrics and system parameters are transmitted to the MonALISA [4, 14] monitoring framework, where the data is aggregated for accounting purposes and presented to operators.

Figure 5 provides a global view extracted from the monitoring page during a rate performance test. This view includes the information detailed in Table 1.

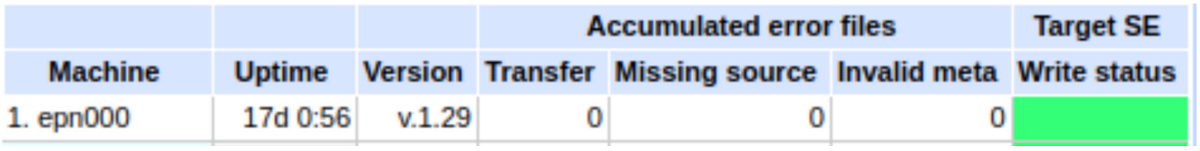
The failure rate represents the number of files per second for which the transfer or registration has failed but remains in the retry cycle, ensuring that no data files are lost and no metadata files remain unregistered.

In addition to the aggregated information reported by all EPN2EOS instances, we also monitor these met-



	Services	Data file transfers						Catalogue registration				
Location	reporting	Ongoing	Slots	Queued	Copy rate	Success rate	Failure rate	Ongoing	Slots	Queued	Success rate	Failure rate
P2	354	1715	2601	3117	229.9 GB/s	61.52/s	0.167/s	3	330	0	60.63/s	0

Fig. 5 Global view during a rate performance test - shows a snapshot from the monitoring page and highlights a global overview of EPN2EOS's operation, providing insights into its performance and functionality



			Accumulated error files			Target SE
Machine	Uptime	Version	Transfer	Missing source	Invalid meta	Write status
1. epn000	17d 0:56	v.1.29	0	0	0	🟩

Fig. 6 Entry in the monitoring page during a rate performance test - illustrates a row from the monitoring page that highlights the metrics reported per EPN node

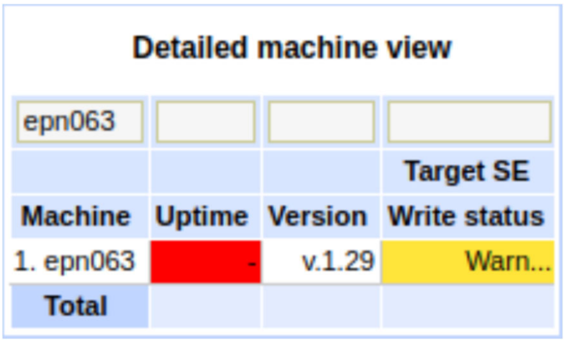
rics individually for each node (Fig. 6). Thus, besides the metrics mentioned previously, there are also metrics such as: the time that has passed since the last restart of the EPN2EOS (Uptime), the version of EPN2EOS that runs on the EPN node (Version), metrics for different types of errors like transfer or registration errors, and also a metric that represents the status of the local storage of each EPN node (Write status).

The write status metric is very important because depending on the status of the disk, alert messages are sent as follows:

If the disk usage reaches 50% of its capacity, a corresponding message is shown on the monitoring page reporting the status of the disk space (Fig. 7).

If the disk usage reaches 90% of its capacity, the EPN2EOS starts transferring the files to the fallback storage (ALICE::CERN::EOSP2) and sends an email alert with a corresponding message. Additionally, a corresponding message is displayed on the monitoring page, as shown in Fig. 8.

If the disk usage reaches 95% of its capacity, the EPN2EOS stops running and sends an email alert. The tool will repeatedly attempt to restart until the disk space issue is resolved.



Detailed machine view

epn063			
			Target SE
Machine	Uptime	Version	Write status
1. epn063	🟥 -	v.1.29	Warn...
Total			

Fig. 7 Warning storage threshold alert - indicates a notification displayed on the monitoring page when the disk usage per EPN node reaches 50% of its capacity

Finally, an alert will be sent if EPN2EOS does not respond for longer than 15 minutes on one or more nodes in production (Fig. 9). The list of nodes considered to be in production is obtained by querying the EPN node info service.

4 Queueing Network Models

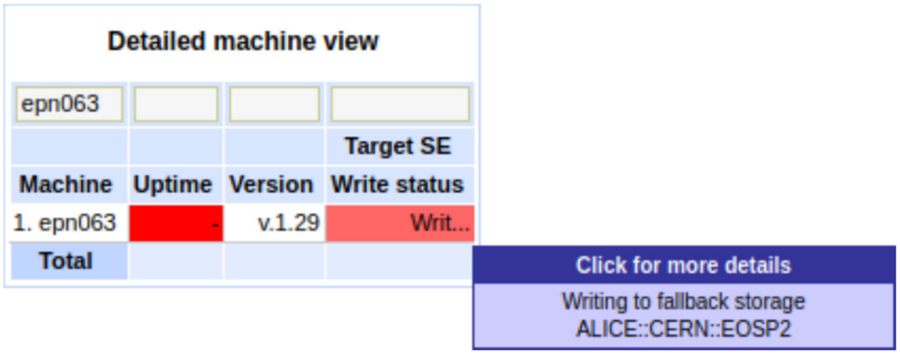
Queueing theory examines the behavior of systems where jobs wait in a queue to be served because the demand for a resource exceeds the system's processing capacity at a given period. These types of systems are prevalent in many areas today, such as in healthcare systems when patients call to schedule an appointment, in supermarkets where customers line up to pay for their groceries, in web servers handling requests from connected users, or in systems managing the flow of cars, airplanes, network packets, and more [7, 15].

Queueing theory is useful for evaluating system performance and, on a more detailed level, helps determine key metrics like the average number of jobs in the system at any given time, the waiting time for a job before it is processed and the average time a job spends within the system. A typical queueing system (Fig. 10) involves a job entering the system, waiting in a queue, being processed, and then leaving the system once completed [7, 15].

Before implementing the EPN2EOS tool, we analyzed three queueing network models to determine which one best fits our case. Therefore, the following subsection outlines the characteristics of three related queueing network models used to estimate job flow and server utilization within a system, based on the network's type and structure.

4.1 Jackson Network Model

The Jackson network model [7] is a general model of queueing networks that allows the existence of cycles



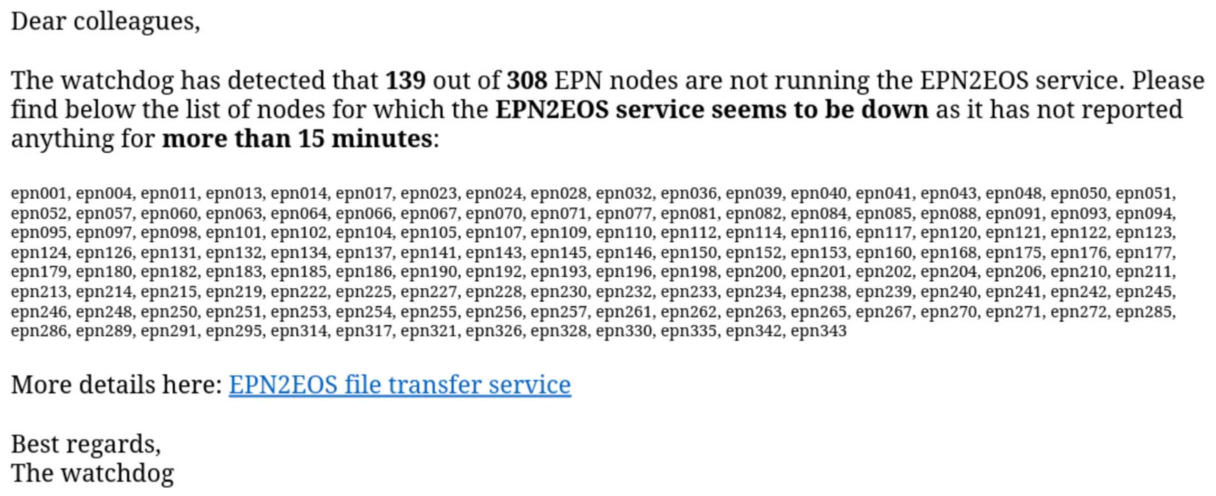
Detailed machine view

epn063			
			Target SE
Machine	Uptime	Version	Write status
1. epn063	-	v.1.29	Writ...
Total			

Click for more details

Writing to fallback storage ALICE::CERN::EOSP2

Fig. 8 Fallback storage threshold alert - indicates a notification displayed on the monitoring page when the disk usage per EPN node reaches 90% of its capacity, triggering the transfer of files to the fallback storage (ALICE::CERN::EOSP2)



Dear colleagues,

The watchdog has detected that **139** out of **308** EPN nodes are not running the EPN2EOS service. Please find below the list of nodes for which the **EPN2EOS service seems to be down** as it has not reported anything for **more than 15 minutes**:

epn001, epn004, epn011, epn013, epn014, epn017, epn023, epn024, epn028, epn032, epn036, epn039, epn040, epn041, epn043, epn048, epn050, epn051, epn052, epn057, epn060, epn063, epn064, epn066, epn067, epn070, epn071, epn077, epn081, epn082, epn084, epn085, epn088, epn091, epn093, epn094, epn095, epn097, epn098, epn101, epn102, epn104, epn105, epn107, epn109, epn110, epn112, epn114, epn116, epn117, epn120, epn121, epn122, epn123, epn124, epn126, epn131, epn132, epn134, epn137, epn141, epn143, epn145, epn146, epn150, epn152, epn153, epn160, epn168, epn175, epn176, epn177, epn179, epn180, epn182, epn183, epn185, epn186, epn190, epn192, epn193, epn196, epn198, epn200, epn201, epn202, epn204, epn206, epn210, epn211, epn213, epn214, epn215, epn219, epn222, epn225, epn227, epn228, epn230, epn232, epn233, epn234, epn238, epn239, epn240, epn241, epn242, epn245, epn246, epn248, epn250, epn251, epn253, epn254, epn255, epn256, epn257, epn261, epn262, epn263, epn265, epn267, epn270, epn271, epn272, epn285, epn286, epn289, epn291, epn295, epn314, epn317, epn321, epn326, epn328, epn330, epn335, epn342, epn343

More details here: EPN2EOS file transfer service

Best regards,
The watchdog

Fig. 9 EPN2EOS running status - errors detected - shows an email notification sent when EPN2EOS does not respond for more than 15 minutes on one or more production nodes, helping to detect potential failures

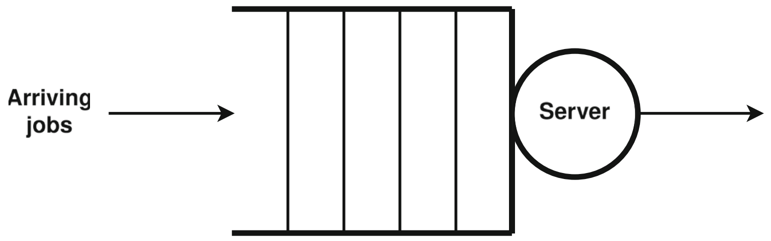


Fig. 10 A typical queueing system [7] - illustrates the process where a job enters the system, waits in a queue, undergoes processing, and exits once completed

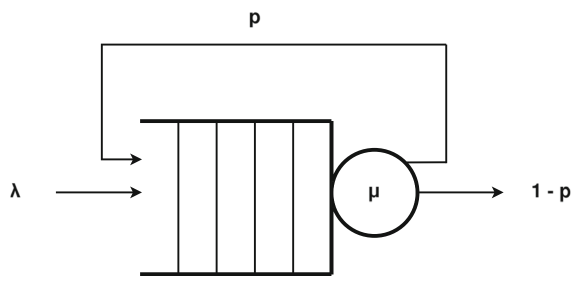


Fig. 11 A server as modeled by a Jackson network [7] - outlines the general characteristics of the model, where a server has an unbounded queue that holds incoming jobs

within the network. Figure 11 illustrates the general characteristics of the model. It presents a server that has an unbounded queue where jobs are placed as they enter the system. Each job is ordered in the queue according to a timestamp representing the time it entered the system, and processed according to First Come First Serve (FCFS) policy. For example, if we consider a job i that entered the system at time t_i and a job j that entered the system at time t_j, where $t_i < t_j$, then job i will be processed before job j [7].

The routing of jobs in this model is probabilistic. The parameter p denotes the probability that a job will fail, which implies its return to the system, and added back into the processing queue, consequently $1 - p$ is the probability that a job will successfully execute and leave the system. The system works with two types of jobs: those that come from outside and those that have failed and are re-queued for processing. The jobs that come from outside appear in the system with rate λ, and the service rate of the server is denoted by μ [7].

4.2 Gordon–Newell Network Model

The Gordon–Newell model [16] is a network model generally applied to closed networks, where no new jobs can enter from outside, and the existing jobs do not leave the system. As a result, the number of jobs in the system remains constant. The job processing policy follows FCFS, and routing within the system is probabilistic [16].

Consider the example shown in Fig. 12, where there are $n = 4$ servers, each with its own processing queue and service rate denoted by μ_i, where i is a number between 1 and 4. The routing policy for this example works as follows: a job processed by server i is moved to server j with probability p_{ij}, where [16]

$$\sum_{j=1}^{n} p_{ij} = 1 \ \forall i = 1, 2, ..., n\,. \tag{1}$$

For instance, the probability that a job processed by server 2 is moved to server 4 is denoted by p_{24}. This probabilistic routing policy between servers cre-

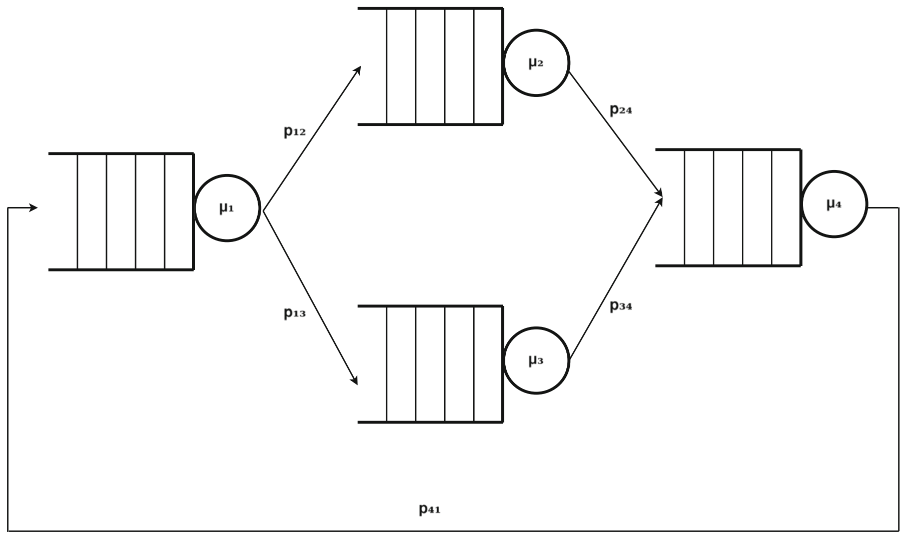


Fig. 12 Example Gordon–Newell network - a system with four servers, each with its own processing queue and service rate μ_i, where jobs are routed probabilistically, moving from server i to server j with probability p_{ij}

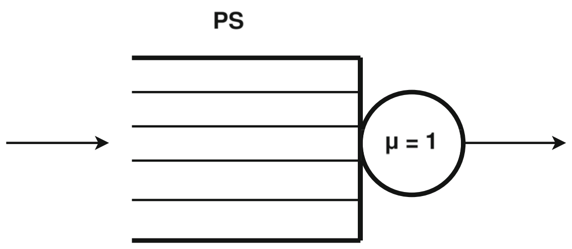


Fig. 13 Example BCMP network [7] - a server operating under a Processor Sharing policy, where n jobs enter the system simultaneously, and each job is processed at a rate of $\frac{\mu}{n}$ with a total service rate of 1

ates interdependence between their processing queues because each job can be added to another server's queue with a certain probability.

A computer system with several CPUs sharing various resources (memory units and I/O devices) can serve as a real-world illustration of this model. Because processes in this type of system are neither generated or destroyed but instead travel between various service units (CPUs, memory units, and I/O devices), the number of processes stays constant.

4.3 BCMP Network Model

BCMP is a network model named after its four developers: Baskett, Chandy, Muntz, and Palacios-Gomez [17]. The model is more general than the Jackson network model, allowing for multiple job processing policies within the system, such as FCFS, Processor-Sharing (PS), or Infinite Server Stations. Additionally, it can be applied to both open and closed networks. However, this flexibility introduces greater complexity, as it supports multiple job processing methods and classifications. Essentially, the model allows the servers in the system to have different properties, meaning they can handle jobs in different ways. This makes the model more suitable for systems where job flows vary based on job type or the processing rules applied [17].

Consider a scenario where n jobs enter the system simultaneously, and the server has a service rate of 1 (Fig. 13). The processing policy is PS, meaning that each job is processed at a rate of $\frac{\mu}{n}$. Each job requires one unit of time to be fully processed, meaning the server must allocate one unit of processing time to complete each job. Since all n jobs arrive at the same time, the server splits its processing capacity equally among them. A common practical example of this is a CPU using a round-robin scheduling policy to process its tasks [7].

4.4 Summary

Using the Jackson network model, we can efficiently and easily determine the flow of jobs within the system and the server utilization under similar conditions. In this model, jobs are treated the same regardless of which server processes them, making the server queues independent of each other.

The Gordon–Newell network model can be viewed as a Jackson network model, but applied to closed networks, where no new jobs from outside are allowed. In other words, this network model is more suited for systems with a fixed number of jobs, where server queues are interdependent, and the network is effectively closed.

The Jackson network model can be seen as a particular case of the BCMP network model, where job processing follows an FCFS policy, and the network is open, allowing new jobs to enter into the system. In other words, for systems where each job is treated the same regardless of which server processes it, the Jackson network model is simpler and more efficient to apply.

A summary of the characteristics of the three analyzed models is presented in Table 2.

In conclusion, these chosen models from queueing network theory are related, but only the Jackson network model is perfectly suited for modeling the file

Table 2 Key findings from the analysis of the three queueing newtork models

Model	Network	Queue policy	Queue dependency	Number of jobs	Job routing
Jackson network	Open, Closed	FCFS	independent queues	arbitrary	probabilistic
Gordon–Newell network	Closed	FCFS	interdependent queues	fixed	probabilistic
BCMP network	Open, Closed	FCFS, PS	independent queues	arbitrary	probabilistic

transfer scheduler used in the ALICE experiment. The reasoning behind this choice is explained in Section 5.

5 EPN2EOS Design

This section describes the EPN2EOS state machine as well as its modeling using the Jackson network model [7].

Figure 14 illustrates the steps of the data file transfer process within the EPN2EOS tool. In the initial state (S0), a metadata file is placed in a specific directory monitored by the transfer watcher. The watcher queues the metadata file for transfer (S1). Next, this metadata file is dequeued, and the transfer of the associated data file to remote storage is initiated (S2). If the transfer is successful (S3), the associated data file is deleted (S3') from the local storage of the EPN node. At the same time, the metadata file is moved to a specific directory defined by the registration watcher (S3"). At this point, the watcher queues the metadata file for registration (S5), the metadata file is dequeued and its attributes are registered in the ALICE catalog (S6) [13]. If the registration is successful, the metadata file is deleted from the local storage of the EPN node (S7). Otherwise, a delay time is calculated, which determines the time at which the registration of the attributes mentioned in the metadata file will be retried (S8) and the metadata file is placed back in the registration queue (S5).

If the transfer of the data file fails, another delay time is computed, which determines the time at which the transfer of the file will be retried (S4), and the metadata file is placed back in the transfer queue (S1).

If the local storage of the EPN node reaches 95% of its capacity, EPN2EOS stops running and sends the appropriate email alert.

As shown in Fig. 14, in the case of a failed transfer or registration, a cycle is created within the system (S1

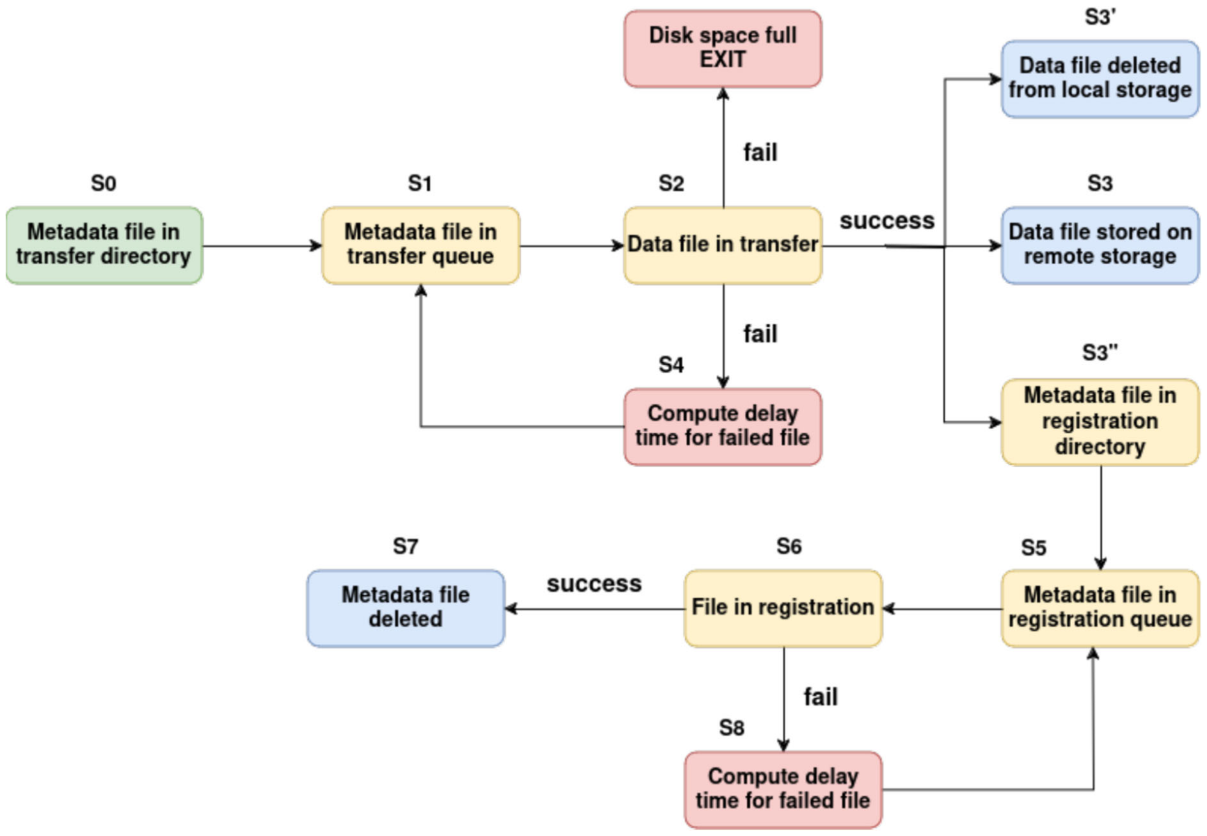

Disk space full EXIT
S3'
Data file deleted from local storage
fail
S0
Metadata file in transfer directory
S1
Metadata file in transfer queue
S2
Data file in transfer
success
S3
Data file stored on remote storage
fail
S4
Compute delay time for failed file
S3"
Metadata file in registration directory
S7
Metadata file deleted
success
S6
File in registration
S5
Metadata file in registration queue
fail
S8
Compute delay time for failed file

Fig. 14 EPN2EOS state machine - represents the sequence of steps involved in transferring data files and registering their metadata within the system

- S2 - S4 - S1 or S5 - S6 - S8 - S5). Additionally, files are placed in unbounded transfer queues, with each node's queue being independent from the others. All file transfers are processed in the same way, regardless of the node handling them. Thus, the design behavior of EPN2EOS is compatible with the Jackson network model.

The following analogy is considered:

- The server presented in Fig. 11 is considered to be an EPN node on which an instance of EPN2EOS is running.
- A job entering the system is associated with the transfer of a data file from the local storage of the EPN node to the remote storage.
- The time a job entered the system is associated with the time a metadata file was placed in the transfer directory.
- The parameter p represents the probability that the transfer of a data file will fail.
- The probability that the transfer of a data file will succeed is represented by $1 - p$.
- The server receives jobs from outside when new metadata files are added to the transfer directory. Thus, the parameter λ denotes the arrival rate of the jobs that come from outside.
- The server receives jobs from inside when the transfer of a data file fails. Thus, the expression $\lambda * p$ denotes the arrival rate of the jobs that come from inside.
- The parameter μ represents the service rate of the server.

Taking into account the general characteristics of the Jackson network model presented in Section 4.1, the following equations are defined:

The total arrival rate for the system, denoted by ($\lambda^{'}$), is defined as [7]

$$\lambda^{'} = \text{outside arrival rate} + \text{inside arrival rate}\,. \tag{2}$$

The system utilization, denoted by ρ, is defined as [7]

$$\rho = \frac{\lambda^{'}}{\mu}\,. \tag{3}$$

The mean number of jobs in the system, denoted by $E[N]$, where N represents the total number of jobs, is defined as [7]

$$E[N] = \frac{\rho}{1-\rho}\,. \tag{4}$$

The mean response time, denoted by $E[T]$, is defined using the Little's Law for open systems [7]

$$E[N] = \lambda^{'} \cdot E[T]\,. \tag{5}$$

$$E[T] = \frac{E[N]}{\lambda^{'}}\,. \tag{6}$$

The total arrival rate of the files into the EPN2EOS is obtained using the (2)

$$\lambda^{'} = \lambda + p \cdot \lambda\,. \tag{7}$$

$$\implies \boxed{\lambda^{'} = \lambda \cdot (1+p)}\,. \tag{8}$$

The EPN2EOS system utilization is obtained using the (3) and (8)

$$\boxed{\rho = \frac{\lambda \cdot (1+p)}{\mu}}\,. \tag{9}$$

The average number of transfers present in the system is obtained using the (4) and (9)

$$E[N] = \frac{\lambda \cdot (1+p)}{\mu} \cdot \frac{1}{1 - \frac{\lambda \cdot (1+p)}{\mu}}\,. \tag{10}$$

$$\implies \boxed{E[N] = \frac{\lambda \cdot (1+p)}{\mu - \lambda \cdot (1+p)}}\,. \tag{11}$$

The average response time to process a file transfer is obtained using the (6), (8), and (11)

$$E[T] = \frac{\lambda \cdot (1+p)}{\mu - \lambda \cdot (1+p)} \cdot \frac{1}{\lambda \cdot (1+p)}\,. \tag{12}$$

$$\implies \boxed{E[T] = \frac{1}{\mu - \lambda \cdot (1+p)}}\,. \tag{13}$$

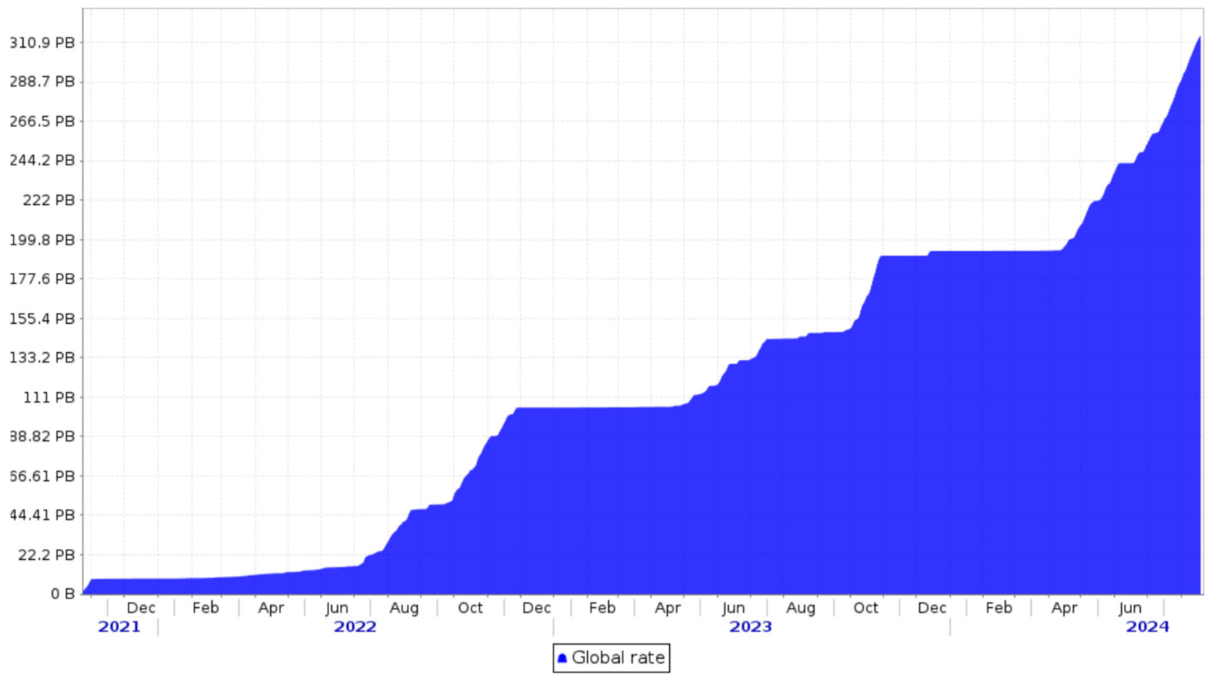


Fig. 15 Data accumulation starting with the end of 2021 - the volume of data transferred by EPN2EOS from the end of 2021 until August 2024

The result obtained in (13) shows that the response time depends on the rate at which new files appear in the system, the processing rate of the server, and the probability that the transfer of a file fails.

6 EPN2EOS Evaluation

This section presents the results obtained after EPN2EOS was deployed into production. EPN2EOS has been running as a daemon service on EPNs since November 2021, following the ALICE experiment upgrade. From then until August 2024, EPN2EOS has transferred approximately 315 PB of data across about 100 million files, with an average file size of 3 GB (Fig. 15).

A rate performance test was carried out on August 24, 2023, involving 330 EPN nodes. During this test, we obtained a maximum aggregated transfer speed of 240 GB/s (Fig. 16), and the maximum transfer speed per EPN node was 2.5 GB/s (Fig. 17). Currently, EPN2EOS runs on 275 EPN nodes.

Figure 18 shows the number of files queued for registration in the ALICE catalog during the rate performance test, demonstrating that the registration process kept pace with the data transfer.

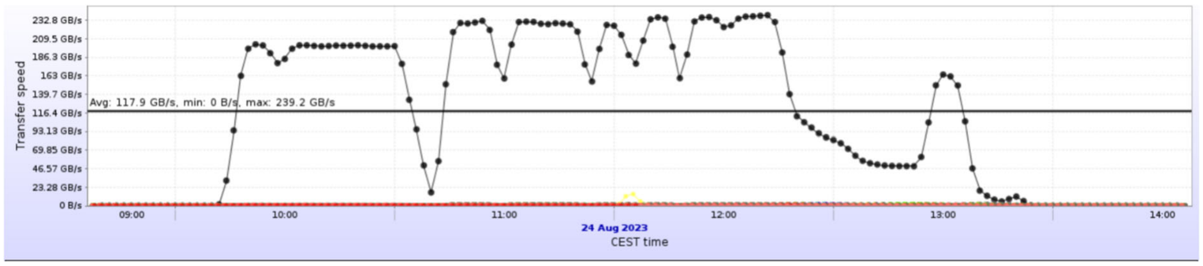


Fig. 16 Aggregated transfer speed - shows the results obtained during the rate performance test conducted on August 24, 2023, involving 330 EPN nodes

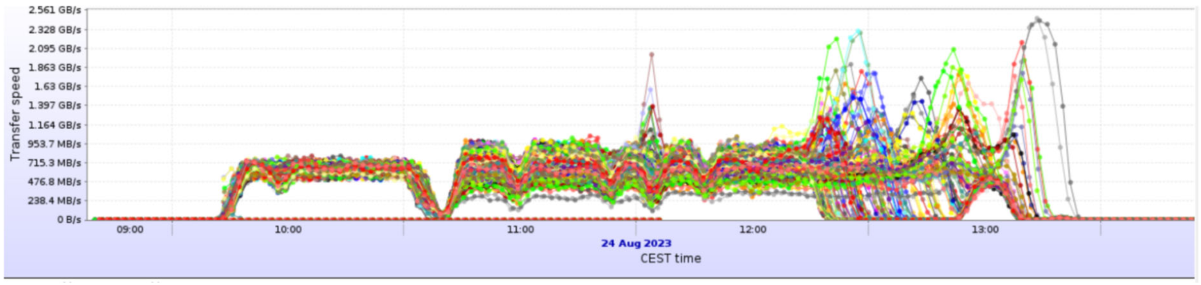


Fig. 17 Transfer speed, where each color represents a single EPN - shows the results obtained during the rate performance test conducted on August 24, 2023, highlighting individual node transfer speeds

The observed limit of 200 GB/s represents the maximum achievable network transfer rate, meaning that any additional transfer requests beyond this threshold result in an accumulation of files in the transfer queues of EPNs. This accumulation does not indicate a freeze in the transfer process but rather a natural consequence of the network operating at full capacity. Once the backlog decreases, all files are successfully transferred (Fig. 19).

The theoretical equations outlined in Section 5 can be applied to the results obtained during the rate performance test. Taking into account the conditions mentioned in Table 3, we can compute for a single EPN node the input rate at which files entered into the system, the system service rate, and the outside arrival rate.

$$\text{input rate} = \frac{\text{total number of transferred files} \cdot \text{average file size}}{\text{evaluation period}} . \tag{14}$$

$$\implies \text{input rate} = 120 \text{ GB/s} . \tag{15}$$

$$\mu = \frac{\text{maximum transfer speed}}{\text{number of EPN nodes} \cdot \text{average file size}} . \tag{16}$$

$$\implies \boxed{\mu = 0.18 \text{ files/s}} . \tag{17}$$

$$\lambda = \frac{\text{input rate}}{\text{number of EPN nodes} \cdot \text{average file size}} . \tag{18}$$

$$\implies \boxed{\lambda = 0.09 \text{ files/s}} . \tag{19}$$

From Fig. 20 we can notice that the transfer error rate during the performance test was approximately 0.08

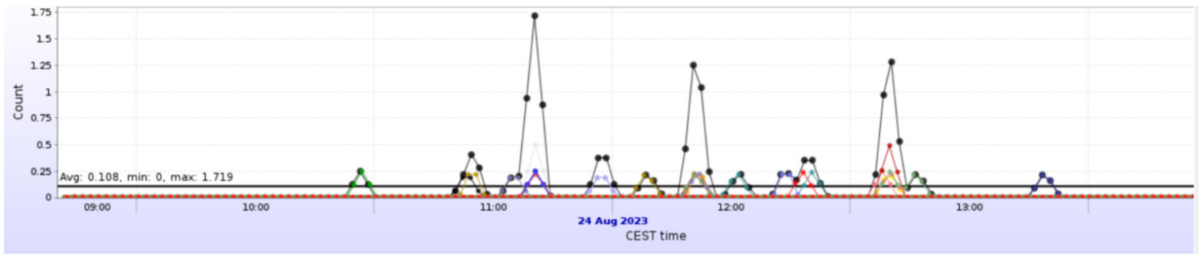


Fig. 18 Files queued for catalogue registration - illustrates the pending files at the last stage of the process during the rate performance test

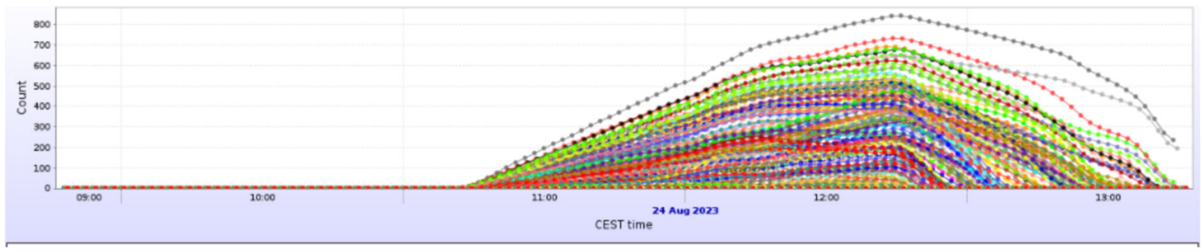


Fig. 19 Queued transfers during the rate performance test, where each color represents a single EPN - illustrates the accumulation of transfer requests when the network reaches its 200 GB/s limit, showing how files queue up until bandwidth becomes available for successful transfer

files/s. Thus, we can determine the probability that a file transfer failed (p)

$$p = \frac{\text{number files whose transfer failed}}{\text{number files successfully transferred} + \text{number files whose transfer failed}}. \tag{20}$$

$$p = \frac{18000\ \text{s} \cdot 0.08\ \text{files/s}}{542642 + 18000\ \text{s} \cdot 0.08\ \text{files/s}}. \tag{21}$$

$$\implies \boxed{p = 0.002}. \tag{22}$$

The probability of a file transfer failing within this system is shown to be very low.

From (8), (19), and (22) we can compute the total arrival rate ($\lambda^{'}$) obtained during the rate performance test for a single EPN node

$$\lambda^{'} = 0.09\ \text{files/s} \cdot (1 + 0.002). \tag{23}$$

$$\implies \boxed{\lambda^{'} = 0.09\ \text{files/s}}. \tag{24}$$

From (9), (17), and (24) we can calculate the utilization of one EPN node (ρ) during the rate performance test

$$\rho = \frac{0.09\ \text{files/s}}{0.18\ \text{files/s}}. \tag{25}$$

$$\implies \boxed{\rho = 0.50}. \tag{26}$$

The mean response time ($E[T]$) can be obtained using the (13), (17), and (24)

$$E[T] = \frac{1}{0.18\ \text{files/s} - 0.09\ \text{files/s}}. \tag{27}$$

$$\implies \boxed{E[T] = 11\ \text{s}}. \tag{28}$$

Therefore, during the rate performance test, each EPN2EOS instance used only 50% of the capacity of

Table 3 Metrics extracted from the rate performance test

Metric name	Metric value
Maximum transfer speed	240 GB/s
Total number of transferred files	542642
Average file size	4 GB
Number of EPN nodes involved in the performance test	330
Evaluation period	August 24 09:00 until August 24 14:00 => 5 hours => 18000 seconds

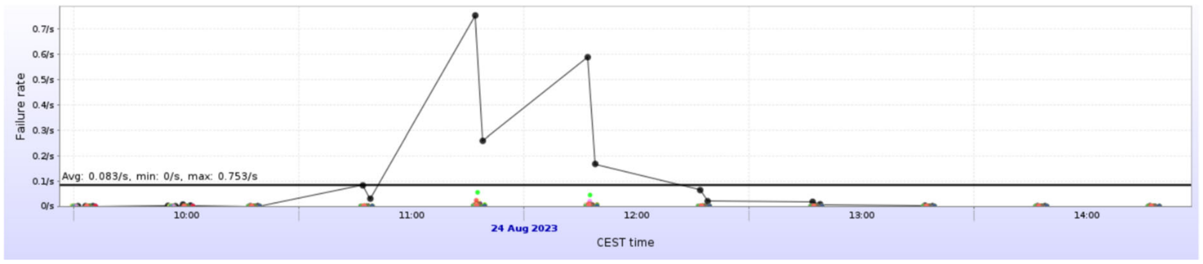


Fig. 20 Transfer error rate - shows that the transfer error rate during the rate performance test remained low, at approximately 0.08 files/s, indicating a stable and reliable transfer process

the EPN node it operated on to facilitate the data transfer. This indicates that the remaining resources of the EPN node are used to collect, compress, and store the data received from the ALICE experiment so that the EPN2EOS can further handle the data transfer. Additionally, the average time a file spent in the system was 11 seconds.

7 Conclusion

EPN2EOS represents the exclusive mechanism for data transfer between the ALICE computing cluster (EPNs) and the 150 PB disk buffer located in CERN IT. Written in Java, it has two main tasks: transferring files containing data collected from the ALICE experiment and registering the metadata associated with each data file in the ALICE catalog. It operates reliably in the demanding environment of real-time data acquisition, having capabilities such as transfer and registration priority scheduling, error handling, monitoring, and a mail alert system. This article also describes the state machine modeling the tool's operation and demonstrates that it respects the Jackson network model of distributed systems.

EPN2EOS has been in production since November 2021, and from then until August 2024, it has transferred approximately 315 PB of data in about 100 million files, with an average file size of 3 GB.

During the rate performance test, we obtained a maximum aggregated transfer speed of 240 GB/s, and a maximum transfer speed per EPN node of 2.5 GB/s. We applied the Jackson network model to analyze the rate performance test results. Our findings show that EPN2EOS instances utilize only half of the EPN node's CPU capacity for data transfer. This implies that the remaining resources are dedicated to data collection, compression, and storage. Consequently, EPN2EOS does not dominate the EPN node, enabling it to prioritize the collection and processing of experimental data.

Regarding future work, the following tasks should be considered for implementation:

- Adding support for automatic cleanup of EPN local disks. For example old data files for which the metadata was not created or is corrupted could be automatically deleted if the disk space runs low, prioritizing data-taking over a potential data recovery.
- Implementing a workflow to delete data from main storage when transferred files are flagged unsuitable for further analysis.
- Implementing an automatic workflow to move data from the main storage to lower storage elements. This could be triggered at a watermark of total storage occupancy, ensuring sufficient space on the main storage to support continuous data-taking.

Acknowledgements The authors would like to thank the ALICE Grid Computing team at CERN for their valuable feedback and discussions, as well as the members of the ALICE Editorial Board for their thorough review and constructive suggestions that helped improve this manuscript.

Author contributions A.F.S was responsible for research, implementation, and manuscript writing. L.B. oversaw project management and provided continuous feedback throughout the development and writing process. N.T coordinated the work and ensured alignment with the research goals. C.G contributed to the implementation and provided incremental reviews and feedback on the manuscript. All authors reviewed and approved the final version of the paper.

Funding Information Open access funding provided by CERN (European Organization for Nuclear Research) The authors have no relevant financial or non-financial interests to disclose.

Data Availability No datasets were generated or analysed during the current study.

Material Availability Not applicable

Code Availability The code developed for this study is available in GitHub at https://github.com/alicesuiu/FileSpooler.

Declarations

Competing Interests The authors declare no competing interests.

Ethics Approval and Consent to Participate Not applicable

Consent for Publication All authors have read and approved the final manuscript and consent to its publication.

References

1. CERN Organization.: CERN Accelerating science. Last accessed: 20 August 2024. https://home.cern/about
2. CERN Organization.: ALICE detects quark-gluon plasma, a state of matter thought to have formed just after the big bang. Last accessed: 20 August 2024. https://home.cern/science/experiments/alice
3. Shreyasi, A., et al.: ALICE upgrades during the LHC Long Shutdown 2. JINST **19**(05), 05062 (2024). https://doi.org/10.1088/1748-0221/19/05/P05062 arXiv:2302.01238 [physics.ins-det]
4. Şuiu, A F., Grigoraş, C., Weisz, S., Betev, L.: EPN2EOS Data Transfer System. EPJ Web Conf. **295**, 01023 (2024). https://doi.org/10.1051/epjconf/202429501023
5. Şuiu, A F., Grigoraş, C., Ţăpuş, N., Betev, L.: Automatic Data Workflow and Disk Management Tool for the ALICE Experiment. (2023). https://doi.org/10.1109/ICCP60212.2023.10398714
6. Şuiu, A F., Carabaş, M., Weisz, S., Grigoraş, C., Ţăpuş, N.: File Spooler and Copy System for Fast Data Transfer, **18**, pp. 1–5 (2021). ECBS 2021: 7th Conference on the Engineering of Computer Based Systems. https://doi.org/10.1145/3459960.3461559
7. Prof. Mor Harchol-Balter.: Performance Modeling and Design of Computer Systems: Queueing Theory in Action, 1st edn. Cambridge University Press, New York (2013)
8. Maurizio Di, P E.: Data Acquisition Systems. Springer, New York – Heidelberg – Dordrecht – London (2013)
9. Richter, M.: for the ALICE Collaboration: A design study for the upgraded ALICE O^2 computing facility. J. Phys. Conf. Series **664**, 082046 (2015). https://doi.org/10.1088/1742-6596/664/8/082046
10. Rohr, D.: Usage of GPUs in ALICE Online and Offline processing during LHC Run 3. EPJ Web Conf. **251**, 04026 (2021). https://doi.org/10.1051/epjconf/202125104026 arXiv:2106.03636. 11 pages, 8 figures, proceedings for vCHEP 2021
11. CERN Organization.: EOS - Open Storage. https://eos-web.web.cern.ch/eos-web/. Last accessed: 20 August 2024
12. Konopka, P., von Haller, B.: The ALICE O2 data quality control system. EPJ Web Conf. **245**, 01027 (2020). https://doi.org/10.1051/epjconf/202024501027
13. Saiz, P., Aphecetche, L., Bunčić, P., Piskač, R., Revsbech, J.-E., Šego, V.: AliEn—ALICE environment on the GRID. Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment **502**(2), 437–440 (2003). https://doi.org/10.1016/S0168-9002(03)00462-5 . Proceedings of the VIII International Workshop on Advanced Computing and Analysis Techniques in Physics Research
14. Cirstoiu, Catalin, Grigoraş, Costin, Betev, Latchezar, Costan, Alexandru, Legrand, Iosif: Monitoring, Accounting and Automated Decision Support for the ALICE Experiment Based on the MonALISA Framework. Int. Symp. High Perform. Distrib. Comput. (2007). https://doi.org/10.1145/1272680.1272688
15. Shortle, John F., Thompson, James M., Gross, Donald, Harris, Carl M.: Fundamentals of Queueing Theory, 5th edn. Wiley, New Jersey (2018)
16. Medhi, J.: Stochastic Models in Queueing Theory. Academic Press, New York (2002)
17. Baskett and Chandy and Muntz and Palacios-Gomez: Open, Closed, and Mixed Networks of Queues with Different Classes of Customers. J. Ass. Comput. Mach. (1975). https://doi.org/10.1145/321879.321887