# Measuring Performance Under Failures in the LHCb Data Acquisition Network

Eloïse Stein, Flavio Pisani, Tommaso Colombo and Cristel Pelsser

***Abstract*—For the Large Hadron Collider beauty (LHCb) experiment, achieving high throughput in the data acquisition (DAQ) network is crucial for supporting scientific applications. However, failures within DAQ networks can lead to significant performance degradation.**

**In this study, we investigate the frequency, duration, and causes of failures in the LHCb DAQ network over a two-month period to illustrate how common these events are. This insight is essential for developing strategies to optimize performance during data taking periods.**

**We further study the performance degradation upon failure. We explore the performance for two potential approaches to high-performance event building on the DAQ network: synchronized and non-synchronized designs. We use live experiments to demonstrate that a synchronized design, which carefully schedules network communications to avoid congestion, can achieve significantly better performance when the network is used at full capacity. However, this approach comes at the expense of reduced fault tolerance compared to the non-synchronized approach. This study highlights that it is essential for the network to handle failures more efficiently to sustainably maintain high data rates.**

***Index Terms*—data acquisition, event building, failure analysis, network fault tolerance, networks**

## I. Introduction

Data acquisition (DAQ) systems play a crucial role in the collection of scientific data [1]–[3]. Such systems, for instance, are deployed at experiments along the Large Hadron Collider (LHC) at the European Council for Nuclear Research (CERN) to collect fragmented data from various sensors and assemble them to reconstruct each particle collision event. This process is known as Event Building.

Event building for large collider experiments is enabled by fast computer networks. Each sensor is connected to a computer which receives its data. To synthesize these data fragments into a unified representation of each collision, every computer engages in communication with all others through the network. The resulting network traffic pattern is a continuous succession of all-to-all exchanges.

As particle collision events are continuously produced in the LHC, reconstructing these events is challenging. It requires a considerable amount of bandwidth, as the data exchanged between computers is voluminous and needs to be transmitted rapidly for event reconstruction.

Eloïse Stein, Flavio Pisani (e-mail: flavio.pisani@cern.ch) and Tommaso Colombo (e-mail: tommaso.colombo@cern.ch) are with CERN, Meyrin, 1211 Geneva, Switzerland. Eloise Stein (e-mail: eloise.stein@etu.unistra.fr) is also with the University of Strasbourg, Strasbourg, France.

Cristel Pelsser (e-mail: cristel.pelsser@uclouvain.be) is with the Université catholique de Louvain (UCLouvain), Ottignies-Louvain-la-Neuve, Belgium, and the University of Strasbourg, Strasbourg, France.

The all-to-all exchange is a collective communication that is very demanding in bandwidth as all the computers in the DAQ system must exchange data with all the others. If all computers send data to the same destination simultaneously, the links to this destination become congested. A typical strategy to address this problem is to spread the communications to each computer over time, meaning that the exchange is segmented into distinct phases.

In this approach, the all-to-all collective exchange is synchronized, meaning that the DAQ application ensures that all computers complete their data exchange before moving to the next phase. This synchronized approach achieves high throughput, particularly in systems with data rates close to 100% of the link capacities [4]. An alternative approach involves a simpler non-synchronized all-to-all exchange, where the network is left to deal with the congestion. When the network operates at full capacity without any failed links, the performance of this approach significantly deteriorates in large systems due to the accumulation of congestion [4]. However, these two approaches have never been compared in the context of network link failures, which is the subject of this paper.

As highlighted in the literature [5], link failures in high-throughput networks, such as those used for collective communications between computers in DAQ systems, are common. To further support this point, we propose to study link failures in the DAQ network of the Large Hadron Collider beauty (LHCb) experiment. We analyze network failures observed over a two-month period during which the LHC was fully active and data were exchanged in the DAQ network. Specifically, we present statistics on the duration, frequency, and underlying causes of network link failures to prove that failures are frequent and can last a long time. These statistics motivate our problem. Furthermore, we find that link failures can significantly degrade performance, as bandwidth utilization in the DAQ network is close to the maximum capacity. A single failure leads to congestion and degrades the throughput from approximately 46 Tbps to 30 Tbps in total.

Finally, we derive some design recommendations from our comparison of the throughput achieved by the synchronized and non-synchronized all-to-all applications in the presence of network failures.

## II. Background

In high-energy physics experiments, large particle accelerators are used to accelerate and collide particles at high energies. One of the largest is the Large Hadron Collider (LHC) at the European Council for Nuclear Research (CERN).

Such large physics experiments typically involve numerous sensors that measure the properties of particles generated by

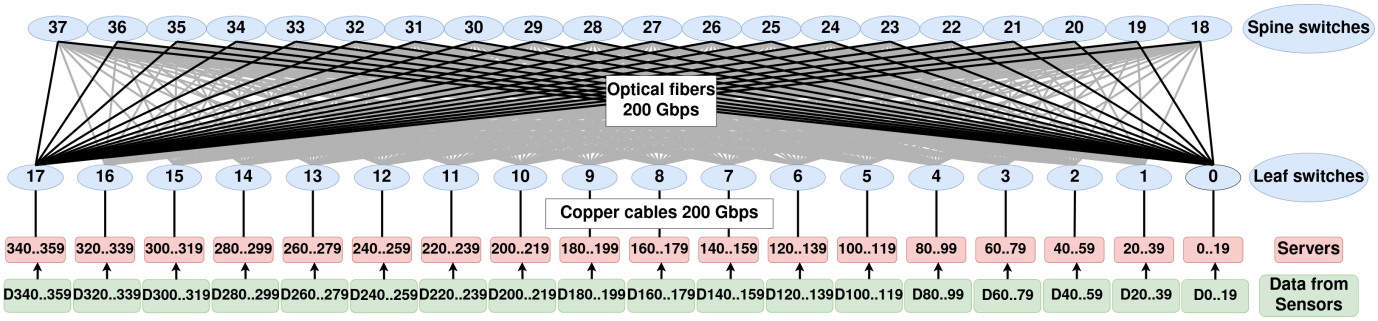


Fig. 1: The Event Builder Network of the LHCb experiment. The data fragments from the sensors are transmitted to the servers, which perform an all-to-all exchange to reconstruct the events. The network topology is a two-layer fat-tree with 28 40-port InfiniBand High Data Rate (HDR) switches. Each leaf switch is directly connected to approximately 20 servers and 20 spine switches with 200 Gbps links. Each source server traverses the fat-tree topology from the leaf switch it is connected to, up to the spine switches, and then back down to the leaf switches to reach the destination server.

the LHC. Data acquisition (DAQ) systems play a crucial role in collecting data from these sensors and processing it to extract relevant information for physics research. To achieve this, one of the main steps is the Event Building process, which merges the fragmented data from each sensor into a unified dataset. The Event Building process, our primary focus in this paper, aims to consolidate all the data fragments into collisions, also known as "events". This operation typically occurs across a high-throughput network of interconnected servers. In the DAQ system of the Large Hadron Collider beauty (LHCb) experiment, such network is known as the Event Builder network and is illustrated in Figure 1. After the Event Building process, interesting events are selected through a two-step High Level Trigger (HLT) process [6].

Unlike other experiments such as ATLAS [7], the LHCb experiment has no hardware trigger, meaning that events are not filtered prior to the event building stage. Instead, LHCb adopts a triggerless DAQ framework, requiring the system, including the communication network, to handle the full collision rate. This results in higher bandwidth requirements. For this reason, the performance of the DAQ network at LHCb is crucial, as it needs to sustain a high data rate of approximately 46 Tbps.

The LHCb Event Builder network makes use of a fat-tree topology, more specifically $k$-ary-$L$-tree [8]. It relies on InfiniBand and consists of 28 40-ports InfiniBand High Data Rate (HDR) switches, depicted by the blue circles in Figure 1. A key feature of a fat-tree topology is its ability to support simultaneous full-bandwidth communication between all node pairs without mutual interference, making it rearrangeably non-blocking—a critical property in our case requiring consistent, high-throughput communication [9].

The fat-tree topology of the Event Builder network comprises two layers: spine switches at the top layer and leaf switches at the bottom layer. A spine switch is connected to every leaf switch via a 200 Gbps optical fiber. Each leaf switch is directly connected to approximately 20 servers using copper cables with a capacity of 200 Gbps, allowing the Event Builder to interconnect 360 servers. In practice, only 326 servers are present in the Event Builder network today because there is currently no need for 360 servers as the generated data can be handled by 326 servers. Furthermore, due to various networking and system overheads, all links are used at approximately 70%, resulting in a network throughput of approximately 46 Tbps in the production network.

The routing of communication flows between servers through the network is managed by OpenSM. OpenSM [10] is the Subnet Manager (SM) on which InfiniBand relies to compute routes and propagate them into the Linear-Forwarding Tables (LFT) of the switches. The routing algorithm used for the Event Builder network is Ftree, as it is optimized for fat-tree topologies [10]. In the event of link failures, the default behavior of OpenSM is to switch to the Min-Hop routing algorithm, as Ftree can only be used on pure fat-trees, i.e. fat-trees with no bandwidth reduction. Min-hop uses shortest paths and balances the routes on the ports of the switches, discarding the failed ports.

In the Event Builder network, the servers receive a partial spacial view of the activity in the beam from a subset of the sensors. Then, the servers send the data from different time intervals to different servers. Each server reconstructs from the received data fragments a complete view of the collision events for its allocated times and is tasked with handling an equal share of the total number of events. This data distribution and reconstruction process results in a continuous succession of all-to-all exchanges, a popular collective operation. More precisely, a data fragment denoted as $D(s_i, t)$ from an event occurring at time $t \in T$, is collected by a sensor, and then transmitted to a designated server, denoted as $s_i$, $i \in [0, S]$, with $S$ being the set of servers IDs in the DAQ application. $T$ defines the set of all possible timestamps at which events occur. To assemble the data fragments, they need to be assigned to a specific server, $s_j$, that receives all the pieces of data for time $t \in T$ through the DAQ application $A$ as shown in Equation 1.

$$\begin{aligned} A : (\boldsymbol{S} \times \boldsymbol{T})^{|\boldsymbol{S}|} \rightarrow \boldsymbol{S} : \\ A(\boldsymbol{D}(\boldsymbol{s}_0, \boldsymbol{t}), \boldsymbol{D}(\boldsymbol{s}_1, \boldsymbol{t}), \ldots, \boldsymbol{D}(\boldsymbol{s}_S, \boldsymbol{t})) = \boldsymbol{s}_j \end{aligned} \tag{1}$$

Collective all-to-all communications are widely used in scientific applications, including other CERN experiments [2], [3], as well as in the implementation of algorithms such as the Fast Fourier Transform (FFT) [11]–[13] and large-scale high-performance computing (HPC) applications in general [14]. More recently, collective exchanges have also been integrated

into machine learning applications [15]–[17]. There are other variants of collective exchanges, such as AllGather and AllReduce, most of which are standardized in Message Passing Interface (MPI) [18].

The nature of the all-to-all exchange poses significant challenges from a network perspective. It demands substantial bandwidth to prevent congestion. A straight forward topology is a full-mesh network where each server is directly connected with all the other servers, necessitating a bandwidth proportional to $n^2$, where $n$ represents the number of servers involved in the collective exchange. This is however very costly and a waste of resources as not all links are needed all the time. This is why, a typical strategy is to route the exchange in a fat-tree topology and divide the exchange into phases.

The collective exchange is typically divided into multiple phases to distribute the necessary bandwidth over time and avoid congestion in the network. The minimum number of phases necessary to realize the all-to-all exchange is $n$. During each phase, every server communicates exclusively with one other server that is not involved in any other communication. The objective is for all servers to have communicated with each other by the end of all phases.

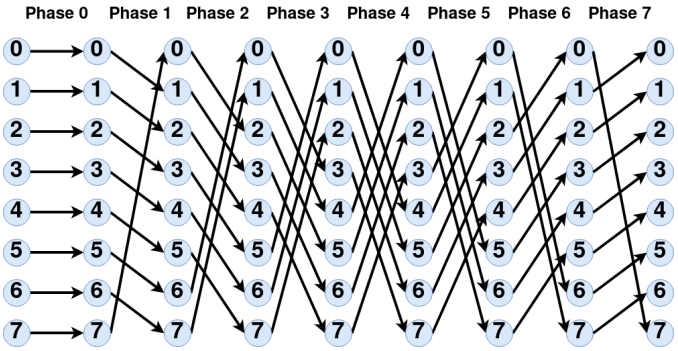


Fig. 2: The Linear-shift scheduling with 8 servers.

Various scheduling algorithms exist to map communications onto phases, including Linear-shift patterns [19], Bandwidth-optimal [20], and XOR. The purpose of these algorithms is to compute the destination ID based on the source ID and the phase ID. The considered network makes use of the linear-shift pattern. In this algorithm, each destination server ID is computed using the formula: $d = (s + p) \mod n$, where $s$ represents the source server ID, $p$ is the phase ID, and $n$ denotes the total number of servers involved in the collective exchange. The main advantage of the linear shift algorithm is that, on a fat-tree topology, when combined with the Ftree routing algorithm proposed by Infiniband, it ensures no congestion on the network in the absence of network failures [10], [19]. To illustrate the operation of the linear-shift pattern, consider an example involving 8 servers that engage in an all-to-all data exchange (Figure 2). This figure depicts the communications at each phase. For instance, at phase 0, every server talks with itself. At phase one, each server $i$ sends its data to the server of id $i + 1$ modulo $n$. At the end of all phases, every server has exchanged data with every other server. The minimum number of phases necessary to realize a full all-to-all exchange is the same as the number of servers.

To execute the linear-shift scheduling correctly, phases need to be synchronized. This implies that when a server completes its exchange with another server in phase $p_0$, it must wait until all other servers finish their exchange for phase $p_0$ before communicating with the server scheduled for phase $p_1$. To synchronize the exchange, Infiniband provides the Tournament Barrier algorithm [21]. The LHCb DAQ makes use of this algorithm as it demonstrates the best performance [4].

In DAQ networks, throughput is the critical performance metric. The real-time nature of event reconstruction demands rapid data exchange between servers, requiring significant bandwidth. Delays in this process can lead to buffer saturation in the servers and potential loss of valuable data. Given these constraints, researchers and practitioners in this domain consistently use throughput as the primary measure to evaluate DAQ network performance.

## III. A Study of Link Failure Occurrences

Link failures in large-scale networks are frequent [22], as evidenced by the study conducted by Gill et al. [5]. Gill et al. present statistical insights into failures from a year-long monitoring of a real large-scale network. The study illustrates the long duration of certain link failures, often attributed to repair difficulties such as the lack of spare network equipment or limited physical accessibility to the affected link. Furthermore, failures occurred on a daily basis throughout the measurement period.

In this section, our contribution is to gather and analyze failure statistics in the Event Builder network of the LHCb experiment to understand the duration, frequency, and nature of network failures in this network. Our statistics cover a two-month period during which the LHC was operational, and physics data was transmitted within the Event Builder network.

### A. Monitoring of Failures

The failures are retrieved using OpenSearch, an open-source tool derived from Elasticsearch and maintained by a community of contributors [23]. OpenSearch comprises various components, including Dashboards for data visualization and analysis, along with other tools for managing and querying data. We use Logstash, an open-source data processing pipeline [24], to collect and process data from the system log messages of OpenSM. The collected data are fed into OpenSearch and include all possible logs from OpenSM, such as changes in topology, routing engine updates, and the status of the subnet. Specifically, we are interested in logs that indicate changes in switch port state, which are provided by messages containing ”osm_spst_rcv_process”.

For instance, when a link goes down, the switch port connecting that link goes from the ACTIVE to DOWN state. Conversely, when the link is repaired, it goes from DOWN to INIT (and then ACTIVE). To obtain this data, we create a query in OpenSearch targeting these specific logs, executing it automatically every day for two months. The output is a Comma-Separated Values (CSV) file containing all the necessary information for failure analysis: timestamp, involved switch and port, and the change in port state. Subsequently,

we parse these files using a Python program to generate failure data, which we can then analyze.

### B. Failure Statistics

In this section, we provide an analysis of the failures that occurred during two months of full network activity. We investigate the duration, frequency, and causes of these failures. The results show that failures can be frequent and prolonged due to flapping links. It is thus of critical importance that the network gracefully adjusts to failures to sustain high data rates.

*1) Duration:* To properly evaluate the duration of failures, we considered the presence of flapping links. Flapping links are common in networks in general, and can cause considerable perturbations to networks due to multiple route re-configurations [25]. A flapping link is defined as a link that oscillates repetitively between an up state and a down state within a short period of time. While the exact duration of a "short period of time" lacks a precise definition in the literature, i.e. there is no overall defined threshold, it generally ranges from a few seconds to a few minutes. To define a threshold for our network, we classified the observed flapping links and their flapping periods over the two-month measurement period to determine the most probable duration between flaps. In the Event Builder network, we observe that when a link starts flapping, 90% of the failures observed during the flapping period of the links have a time interval of less than 10 minutes. This means that when the first failure occurs on the link during its flapping period, the next one probably occurs less than 10 minutes later. Based on this insight, we consider a flapping link period as a single failure by ignoring the time interval between two failures that are inferior to 10 minutes. This means that when two or more failures occur on a link and their time interval is less than 10 minutes, these failures are considered as one, and the failure period starts at the time the first failure occurred until the last one is recovered. By applying the flapping links correction, we discovered that nearly 70% of the failures result from flapping links. Over the 2605 observed failures, 1778 are attributed to flaps. After applying the correction, these flaps were reduced to 287 failures, representing 287 periods during which the links were flapping.

Figure 3 illustrates the cumulative distribution of link failure duration observed throughout March and April 2024 on the Event Builder network. The duration of a link failure refers to the period during which a link transitions from an operational state to a non-operational state until it returns to the operational state. The blue line shows the results for all observed failures, excluding periods during which performance tests were conducted on the Event Builder network. These tests involved manually disabling links to evaluate how the network reacts to failures and would thus bias our experiments. Given that the duration of these failures is predetermined and programmed, they are irrelevant to the study. The dotted orange line represents the cumulative distribution of link failure duration when applying the flapping links correction.

In the results without the flapping links correction (the blue line), we observe that instances of short failures are not

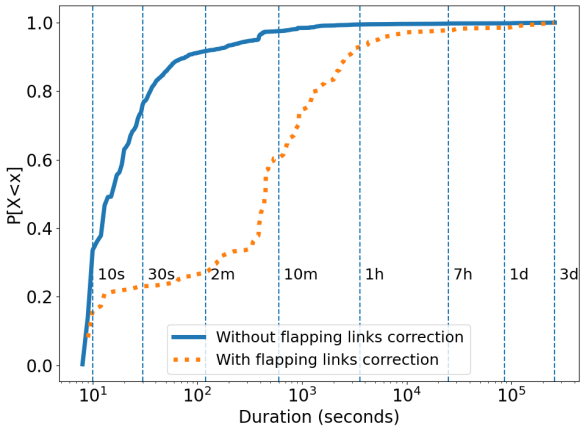


Fig. 3: Cumulative distribution of failure duration. The blue line represents the cumulative distribution of failure duration without the flapping links correction. The orange dotted line represents the cumulative distribution of failure duration with the flapping links correction. Flapping links correction is applied by counting failures of a flapping link as one long failure rather than several small failures spaced over a short period of time.

frequent, 33% of the observed failures have a duration of less than 10 seconds. However, the data reveals that 88% of observed failures are short-lived, lasting less than a minute, while 97% conclude within 10 minutes. Consequently, the majority of failures are brief, with the longest failures lasting around 3 days.

Compared to the duration of failures without the flapping links correction, failures duration are prolonged when the correction is applied (the orange dotted line), with 24% of the failures lasting less than 1 minute and 60% lasting less than 10 minutes. This highlights the severity of flapping links as a network issue, as they often lead to prolonged failures. Furthermore, the impact on the network is even worse because each flap requires computing new routes and updating routing tables. These actions consume time [26], approximately 1.5 seconds in our network. Consequently, these failures can degrade network performance and cause long-lived performance degradation.

*2) Frequency:* Figure 4 illustrates the distribution of failures over the two-month period of March and April 2024. Only the days and links where failures occurred are represented in Figure 4. Each link was assigned an ID based on the order of the switch ID and the port ID, which were sorted in ascending order. In total, 150 links in the Event Builder network failed at least once during the month of March and April 2024. The test days for the Event Builder network occurred on 03-08, 03-10, 03-11 and 03-12, which explains the occurrence of failures for various links on these days. During these tests, various links were manually disabled, which explain why a lot of links experienced a few failures on these days. These are the events removed from Figure 3. Furthermore, two links experienced repeated failures throughout the entire period. These links represent a single flapping link in two directions. The link frequently experienced flapping, with failures occurring nearly every day and recording more than 200 failures on a single

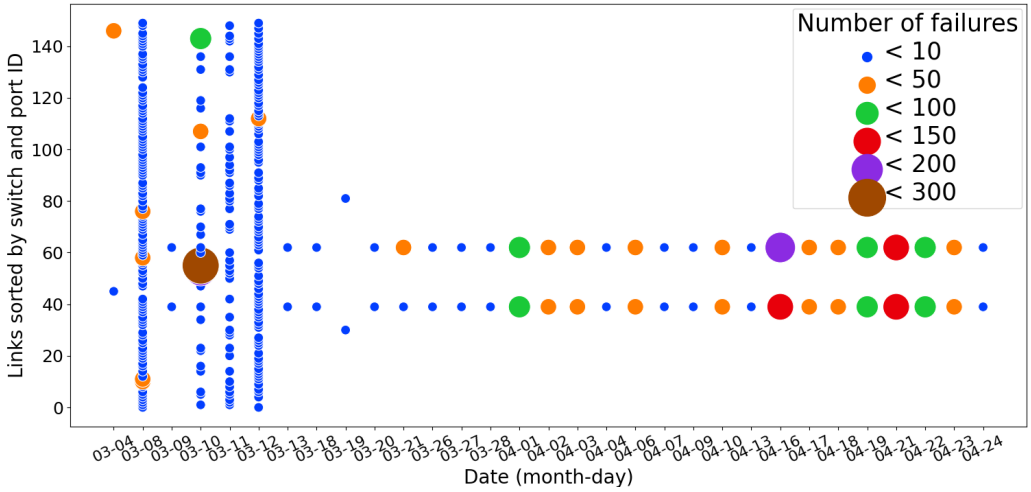


Fig. 4: Distribution of network link failures over a two-month period. The x-axis represents the days in March and April 2024 when failures occurred. The y-axis represents the IDs of the links that failed during this period. The link IDs are assigned based on the switch IDs and the port IDs which are sorted in ascending order.

day (04-16). As detailed in Section V, the consequences of such failures can significantly impact network performance, resulting in substantial loss of throughput. Unfortunately, this link could not be repaired during the measurement period due to the unavailability of spare equipment and the location of the link.

*3) Nature:* During March and April 2024, we observed a total of 2605 failures, out of which 549 were attributed to tests conducted on the Event Builder network, while 2056 were real network failures. Among the 2056 real failures, 1778 are due to flapping links. Flapping links are caused by cable deterioration over time. Excluding the flaps, there remain 278 isolated failures, which are generally due to hardware issues such as dirty optical fibers or software problems. Additionally, maintenance activities were conducted on the Event Builder network servers, resulting in a total of 332 failures of servers. However, failures of servers are not relevant to our study, as they involve complete disconnections of servers from the network. There is no means of recovering from a server failure with the current infrastructure. In such cases, the available bandwidth remains sufficient for the all-to-all exchange as the number of sources is reduced.

## IV. Materials and Methods

Now that we have seen that link failures are common, we investigate the network performance in nominal state and upon failures. Our objective is to devise efficient communication strategies in the two situations. The all-to-all synchronized application currently in production and used by the Data Acquisition (DAQ) system has been developed internally for the LHCb experiment [4]. We evaluate the performance of synchronized all-to-all compared to the non-synchronized one to verify that synchronization always remains the optimal solution. For this purpose, we develop a new all-to-all application without synchronization that supports the same throughput as the synchronized one currently in production.

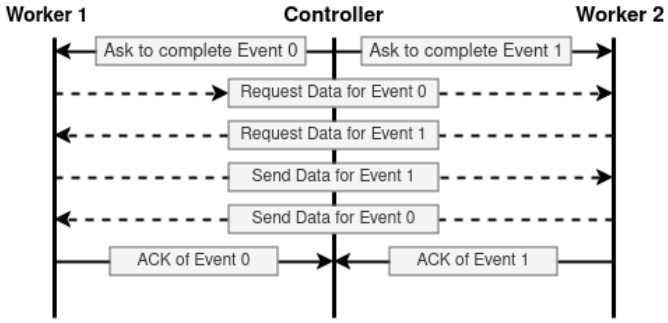


Fig. 5: Description of the MPI all-to-all application without synchronization.

Both the non-synchronized and synchronized all-to-all applications are developed using the Message Passing Interface (MPI). MPI is a tool for high-performance scientific computing that provides libraries to enable multiple processes to communicate with each other and to synchronize. MPI relies on Single Program Multiple Data (SPMD) principle [18], which allows the user to write a single program to be executed by a set of processes, each with a distinct role. The user's challenge lies in defining how these processes communicate with each other based on their roles. As the synchronized all-to-all application used by the LHCb DAQ system relies on MPI, we chose to develop the non-synchronized application using MPI to ensure the two applications are comparable and to take advantage of the simplicity of using MPI.

In the non-synchronized all-to-all application, we simulate the Event Building process similarly to the synchronized one, which involves reconstructing collision events generated by the Large Hadron Collider. Each server in the Event Builder network has one process of the MPI application allocated. The principle of the non-synchronized all-to-all application is that one process serves as the controller while all others act as workers. The controller assigns each worker a collision

event to reconstruct. Each worker has data on every event and sends it to the worker responsible for reconstructing that event. Figure 5 illustrates our non-synchronized application with one controller and two workers. The controller assigns worker 0 to reconstruct the event with ID 0 and worker 1 to reconstruct the event with ID 1. Subsequently, worker 0 and worker 1 exchange requests for data on the corresponding event. Upon sending the data and reconstructing the event, the workers send an acknowledgment to the controller, prompting the assignment of subsequent events to be reconstructed.

In this approach, there is no scheduling; when workers are assigned an event by the controller, they send a request to receive data for this event to all workers in the order of their ID. However, the data request is non-blocking, which means that all sources may send their data concurrently. The only blocking condition for a worker is that it is required to receive all data from other workers on the event it needs to reconstruct before making a request to the controller to assign another event.

To compare the synchronized and non-synchronized all-to-all in the event of failures, we manually deactivated the links between the leaf and spine switches, illustrated in Figure 1. We did not test disabling the links between leaf switches and servers, as these do not affect the bandwidth of the all-to-all exchange for the remaining nodes.

The links that were deactivated were chosen randomly and included 1, 3, and 5 simultaneous failures. We chose this number of failures because the literature has shown that groups of failures containing more than 5 failures are unlikely, with only 10% of groups containing more than 4 failures [5]. Our network shows the same behavior, with a median of only 1 simultaneous failure. The maximum number of coexisting failures is 5.

For each failure scenario, the tests were repeated randomly 10 times. To illustrate the performance degradation during failures, we count the number of events reconstructed by each server over 10 seconds in the all-to-all application. Multiplying this rate by 16 MB, the amount of bytes sent for each event, we obtain the global throughput of the Event Builder network.

## V. Performance Evaluation

In this section, we present the measurements of the throughput obtained on the Data Acquisition (DAQ) system of the LHCb experiment using both the synchronized and non-synchronized all-to-all applications. We evaluate these two approaches at full capacity with increasing load and under failures.

### *A. Scalability of Synchronized and Non-Synchronized All-to-All*

We first re-evaluate the scalability of the synchronized and non-synchronized all-to-all exchange of [4] due to significant changes in the DAQ application since this publication, potentially leading to new results.

We evaluate the scalability of the current network design by measuring the global event building throughput when enabling an increasing number of servers in Figure 6. To increase the number of servers, racks of servers are consecutively added to the system. Each rack contains one leaf switch and 16 to 20 servers.

In our evaluation, synchronization proves to be more advantageous, as the system grows in size, the difference increases. When the DAQ system is used at full capacity, we observe a throughput gain of about 15 Tbps over the non-synchronized all-to-all. By contrast, in the prior test [4] , the throughput gain was approximately 5 Tbps at full capacity of the Event Builder network with 326 servers.

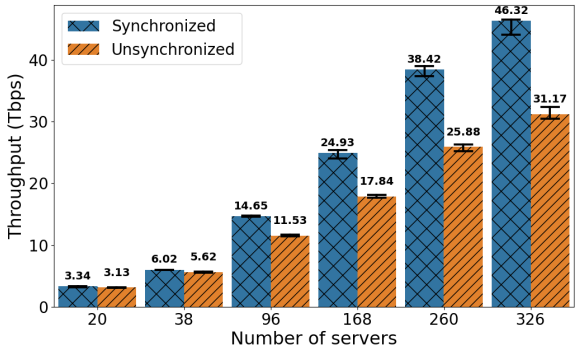


Fig. 6: Scalability of the synchronized and non-synchronized all-to-all applications. The error bars represent the minimum, mean and maximum values.

The difference in performance between synchronized and non-synchronized all-to-all arises from the assurance provided by synchronization that the scheduling of communications during each phase is always respected. With synchronization, every server waits until all others have transmitted and received their data before proceeding to the next communication. This approach effectively prevents network congestion, as proper scheduling and sufficient bandwidth ensures that there will be no congestion on the network links, thus improving throughput. Conversely, in the non-synchronized all-to-all, there is no scheduling as the exchange is not synchronized, which creates network congestion and reduces the achieved throughput. Therefore, the synchronized all-to-all provides better performance, particularly in systems used at full capacity. Although there is cost in synchronization time, this cost can be offset by the performance gained by avoiding congestion on the network.

### *B. Throughput Achieved by Synchronized and Non-Synchronized All-to-All in Case of Failures*

The synchronized approach performs better than the non-synchronized approach without failures and when the system is used at full capacity. In this section, we evaluate the performance of these two approaches in the event of failures, considering that link failure events are common, as demonstrated in Section III.

The throughput of the synchronized exchange can significantly decrease in the event of failures. Figure 7 shows the results of the evaluation of the synchronized and non-synchronized approaches in the event of failures. The failure scenarios have been replicated 10 times for each number

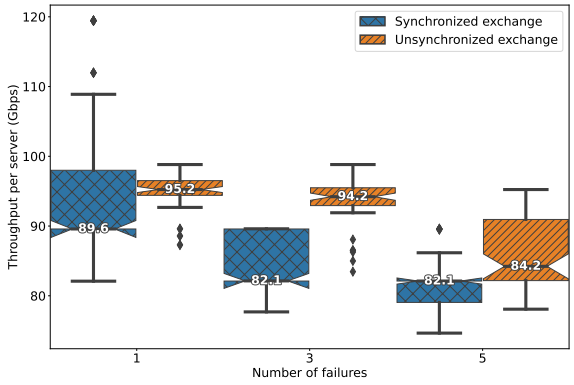


Fig. 7: Synchronized and non-synchronized all-to-all exchange throughput per server as a function of the number of failures. Failure scenarios are randomly generated 10 times for each number of failures. The number of simultaneous failures in the Event Builder network is 1, 3 or 5. The boxplots represent the minimum, 25th percentile, median, 75th percentile, and maximum values. Outliers are also depicted.

of simultaneous failures and randomly generated. In total, there are 30 experiments. In particular, we demonstrate that throughput can significantly decrease in the event of failures, from a median of 142.71 without failures to 89.6 for the synchronized exchange, even with just one failure across the entire network of 360 optical links. The variations in results for synchronized and non-synchronized exchanges are due to some links carrying more flows than others. When they are down, more traffic is impacted by their failure. For instance, the minimum achieved throughput per server for a single failure with synchronized exchange is 82 Gbps, while the maximum value is 120 Gbps. For the 82 Gbps result, the disabled port of one switch served as the output port for 26 destinations, whereas for the 120 Gbps result, the disabled port on another switch was used as the output port for 9 destinations. Consequently, disabling the port used to reach 26 destinations, which support more communication flows, had a much greater impact on traffic compared to the port used for 9 destinations, resulting in a decreased throughput in case of failures.

Synchronized exchange shows lower performance than non-synchronized exchange in the event of failure. In the synchronized all-to-all, the cost of synchronization can be offset by the performance gained from avoiding congestion on the network. However, this method lacks adaptability in the event of network link failures because its performance is dependent on avoiding congestion. When congestion becomes unavoidable due to link failures, performance suffers accordingly. Contrarily, the non-synchronized approach shows better adaptability in case of failures. The lack of synchronization points allows it to make efficient use of the remaining network capacity.

### C. Design Recommendation

In our measurements, we demonstrated that the non-synchronized version of the all-to-all exchange performs better than the synchronized version in the event of failures. As discussed in Section III, some failures can be long-lived failures due to flapping links. For instance, during two months of measurements, a single optical link failed 861 times, resulting in a total downtime of approximately $6.1 \times 10^5$ seconds, which represents approximately seven days and one hour. In Section V-B, we show that the median throughput of the synchronized exchange for a single failure is 89.6 Gbps per server, whereas the non-synchronized application achieves a median throughput of 95.2 Gbps per server. Consequently, the total application throughput during a failure is 29.2 Tbps for the synchronized application compared to 31 Tbps for the non-synchronized application. We obtain these numbers by multiplying the median throughput for a server by the total number of servers, which is 326. This means that every time one link fails, the synchronized exchange loses 1.8 Tbps compared to the non-synchronized exchange, which represents a total loss of approximately $1.1 \times 10^6$ Tb over the months of March and April, given the link's total downtime of $6.1 \times 10^5$ seconds. Therefore, removing synchronization in the event of even a single link failure could prevent the loss of a significant amount of valuable data for particle physics research.

## VI. Related Work

Our evaluation was conducted on a real DAQ network that is actively in use, giving our findings practical relevance. In a related paper [5], the authors investigated failures in a real large-scale network, presenting the impact of various types of link failures. However, while they acknowledged long-lived failures, they did not specifically explore the impact of flapping links, connections that repeatedly go unstable.

Pisani et al. [4] highlighted that the all-to-all exchange without synchronization shows decreased performance in the absence of link failures and face scalability challenges. Our study reinforces this observation by introducing novel measurements. Additionally, we also compare the synchronized and non-synchronized all-to-all, in the event of link failures, to show that the non-synchronized approach reacts better in the event of failures.

Finally, [27] delved into how Infiniband routing algorithms perform in the event of failures. This study presents the throughput obtained for various Infiniband algorithms in a simulated fat-tree network and shows that Ftree handles failures more effectively, resulting in better throughput. In this paper, we introduce a new factor affecting throughput during failures: synchronization. We demonstrate that without synchronization, we can achieve improved performance in terms of throughput during failures.

## VII. Conclusion

In this paper, we present statistics about failures collected over two months. Our results revealed that failures occur frequently, with 2605 failure events recorded during the data taking period. Most of these failures were due to flapping links, which are characterized by fluctuations between operational and non operational states, leading to extended periods of downtime. This highlights the existence of long-lived failures

within the network, which cause a significant loss of throughput. Then, we evaluate two alternative approaches to event building on the LHCb DAQ system, showing a synchronized approach that can reach high throughput in normal conditions. However, upon link failures, the synchronized approach faces significant performance reduction. In these scenarios, eliminating synchronization is a quick solution to reduce the performance degradation.

## REFERENCES


[1] N. Belyaev, D. Krasnopevtsev, R. Konoplich, V. Velikhov, and A. Klimentov, "High performance computing system in the framework of the higgs boson studies," tech. rep., ATL-COM-SOFT-2017-089, 2017.

[2] G. Jereczek, G. Lehmann Miotto, and D. Malone, "Analogues between tuning tcp for data acquisition and datacenter networks," in 2015 IEEE International Conference on Communications (ICC), pp. 6062–6067, 2015.

[3] T. Bawej, U. Behrens, J. Branson, O. Chaze, S. Cittolin, G.-L. Darlea, C. Deldicque, M. Dobson, A. Dupont, S. Erhan, A. Forrest, D. Gigi, F. Glege, G. Gomez-Ceballos, R. Gomez-Reino, J. Hegeman, A. Holzner, L. Masetti, F. Meijers, E. Meschi, R. K. Mommsen, S. Morovic, C. Nunez-Barranco-Fernandez, V. O'Dell, L. Orsini, C. Paus, A. Petrucci, M. Pieri, A. Racz, H. Sakulin, C. Schwick, B. Stieger, K. Sumorok, J. Veverka, and P. Zejdl, "The new cms daq system for run-2 of the lhc," IEEE Transactions on Nuclear Science, vol. 62, no. 3, pp. 1099–1103, 2015.

[4] F. Pisani, T. Colombo, P. Durante, M. Frank, C. Gaspar, L. G. Cardoso, N. Neufeld, and A. Perro, "Design and commissioning of the first 32 tbit/s event-builder," IEEE Transactions on Nuclear Science, pp. 1–1, 2023.

[5] P. Gill, N. Jain, and N. Nagappan, "Understanding network failures in data centers: Measurement, analysis, and implications," SIGCOMM Comput. Commun. Rev., vol. 41, p. 350–361, aug 2011.

[6] R. Aaij et al., "Allen: A high level trigger on GPUs for LHCb," Comput. Softw. Big Sci., vol. 4, no. 1, p. 7, 2020.

[7] G. Aad, B. Abbott, K. Abeling, N. Abicht, S. Abidi, A. Aboulhorma, H. Abramowicz, H. Abreu, and Y. A. et al., "Fast b-tagging at the high-level trigger of the atlas experiment in lhc run 3," Journal of Instrumentation, vol. 18, p. P11006, nov 2023.

[8] F. Petrini and M. Vanneschi, "k-ary n-trees: high performance networks for massively parallel architectures," in Proceedings 11th International Parallel Processing Symposium, pp. 87–93, 1997.

[9] N. Pippenger, "On rearrangeable and non-blocking switching networks," Journal of Computer and System Sciences, vol. 17, no. 2, pp. 145–162, 1978.

[10] Nvidia, "Opensm," 2023. [Online]. Available: https://docs.nvidia.com/networking/display/MLNXOFEDv461000/OpenSM. Accessed on: June 24, 2024.

[11] K. Czechowski, C. Battaglino, C. McClanahan, K. Iyer, P.-K. Yeung, and R. Vuduc, "On the communication complexity of 3d ffts and its implications for exascale," in Proceedings of the 26th ACM International Conference on Supercomputing, ICS '12, (New York, NY, USA), p. 205–214, Association for Computing Machinery, 2012.

[12] J. Doi and Y. Negishi, "Overlapping methods of all-to-all communication and fft algorithms for torus-connected massively parallel supercomputers," in SC '10: Proceedings of the 2010 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis, (New Orleans, LA, USA), pp. 1–9, IEEE, 2010.

[13] E. Namugwanya, A. Bienz, D. Schafer, and A. Skjellum, "Collective-optimized ffts," 06 2023.

[14] E. Chan, M. Heimlich, A. Purkayastha, and R. van de Geijn, "Collective communication: theory, practice, and experience: Research articles," vol. 19, p. 1749–1783, sep 2007.

[15] L. Zhao, S. Maleki, Z. Yang, H. Pourreza, A. Shah, C. Hwang, and A. Krishnamurthy, "Forestcoll: Efficient collective communications on heterogeneous network fabrics," ArXiv, vol. abs/2402.06787, 2024.

[16] A. Shah, V. Chidambaram, M. Cowan, S. Maleki, M. Musuvathi, T. Mytkowicz, J. Nelson, O. Saarikivi, and R. Singh, "TACCL: Guiding collective algorithm synthesis using communication sketches," in 20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23), (Boston, MA), pp. 593–612, USENIX Association, Apr. 2023.

[17] Z. Cai, Z. Liu, S. Maleki, M. Musuvathi, T. Mytkowicz, J. Nelson, and O. Saarikivi, "Synthesizing optimal collective algorithms," PPoPP '21, (New York, NY, USA), p. 62–75, Association for Computing Machinery, 2021.

[18] MPI, "Mpi(3) man page," 2021. [Online]. Available: https://www.open-mpi.org/doc/v4.0/man3. Accessed on: June 24, 2024.

[19] E. Zahavi, G. Johnson, D. Kerbyson, and M. Lang, "Optimized infinibandtm fat-tree routing for shift all-to-all communication patterns," Concurrency and Computation: Practice and Experience, vol. 22, pp. 217 – 231, 11 2009.

[20] B. Prisacari, G. Rodriguez, C. Minkenberg, and T. Hoefler, "Bandwidth-optimal all-to-all exchanges in fat tree networks," in Proceedings of the 27th International ACM Conference on International Conference on Supercomputing, ICS '13, (New York, NY, USA), p. 139–148, Association for Computing Machinery, 2013.

[21] D. Hensgen, R. Finkel, and U. Manber, "Two algorithms for barrier synchronization," International Journal of Parallel Programming, vol. 17, pp. 1–17, 02 1988.

[22] R. Singh, M. Mukhtar, A. Krishna, A. Parkhi, J. Padhye, and D. Maltz, "Surviving switch failures in cloud datacenters," SIGCOMM Comput. Commun. Rev., vol. 51, p. 2–9, may 2021.

[23] OpenSearch, "Find the truth within your data," 2024. [Online]. Available: https://opensearch.org/. Accessed on: June 24, 2024.

[24] Elasticsearch, "Centralize, transform and stash your data," 2024. [Online]. Available: https://www.elastic.co/logstash. Accessed on: June 24, 2024.

[25] P. Merindol, P. David, J.-J. Pansiot, F. Clad, and S. Vissicchio, "A fine-grained multi-source measurement platform correlating routing transitions with packet losses," Computer Communications, vol. 129, 08 2018.

[26] N. Dandapanthula, "Infiniband network analysis and monitoring using opensm," Master's thesis, The Ohio State University, 2011.

[27] B. Bogdanski, B. D. Johnsen, S.-A. Reinemo, and F. O. Sem-Jacobsen, "Discovery and routing of degraded fat-trees," in 2012 13th International Conference on Parallel and Distributed Computing, Applications and Technologies, pp. 697–702, 2012.