32 Tb/s InfiniBand network monitoring tool for the Run 3 LHCb real-time DAQ

CERN Summer Student Programme 2021

Marta Gomis Domènech, martagomisdomenech@gmail.com

Supervised by: Rafal Krawczyk and Niko Neufeld

September 3rd, 2021

Table of Contents

1. Introduction

The Data Acquisition System (DAQ) of the Large Hadron Collider beauty (LHCb) experiment at CERN is being completely upgraded for the upcoming Run 3 of the LHC accelerator. The previous hardware trigger is abandoned because it would not benefit from the upcoming higher luminosity of the accelerator. Thanks to the detector geometry, in addition to the relatively small event size compared to the other three main experiments at CERN (ATLAS, CMS and ALICE); a software-based event selection, with a trigger-less readout system is possible for LHCb. The new system allows online data selection through real-time processing at bunch crossing-rate, which results in a massive bandwidth of 32 Tb/s [1]. It has already been commissioned and it is currently being tested for upcoming operation.

Figure 1 shows an overview of the new network architecture for Run 3 LHCb DAQ. The context of the project concerns the High Level Trigger 1 (HLT1), which is responsible for the event reconstruction and which is therefore called Event Builder (EB). It is decoupled from HLT2 by a large buffer storage network. The new fabric for the EB network relies on InfiniBand switches and links, provided by Mellanox Technologies.

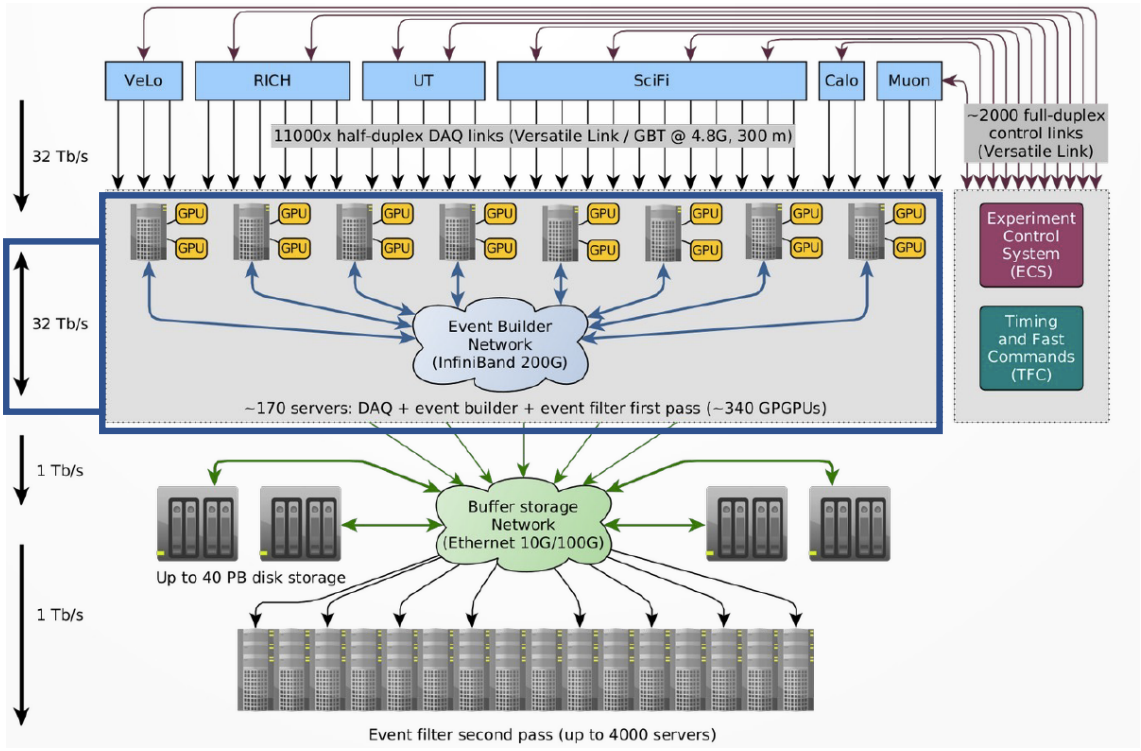

Figure 1 New network topology for Run 3 LHCb DAQ from [1]. Highlighted in blue: Event Builder part of the readout system, as well as its bandwidth. Data flow from the subdetectors front-end electronics up to the data storage.

2. Motivation and project goal

The new EB network topology over InfiniBand introduced in the previous section requires new monitoring tools for network performance and network diagnosis. The main goal of the project is to develop such real-time monitoring tool, which will have a first final usage during the system installation, in order to prevent any installation-related issue. The ultimate usage of the tool will be during run-time: in the LHCb control room, it will allow the end users to control network congestion in order to prevent errors which may lead to undesirable data losses.

3. Event Builder Network point-to-point fabric over InfiniBand

InfiniBand is a network protocol widely used for High Performance Computing (HPC) clusters, low latency and high throughput communication networks [2,3,4], as it supports RDMA (Remote Direct Memory Access). RDMA is based on traditional networks, but as a difference it provides a messaging service such that applications can directly access the virtual memory on remote computers, without any of the two operating systems being involved. Thanks to stack bypass and copy avoidance, it reduces CPU utilization and memory bandwidth bottlenecks [6].

The lossless transmission is possible thanks to a non-blocking fat-tree switched infrastructure, which in our EB network is composed of 200 Gb/s bandwidth InfiniBand links and 28 InfiniBand switches.

As shown in Figure 2, the EB network has a 3-layer fat-tree topology [3], being this 2-stage architecture suitable for efficient all-to-all communication. The nodes are switches (which route packets between ports) and Host Channel Adapters (HCAs, which terminate the links and connect the InfiniBand network to the PCI-based cards into servers):

- First layer: 10 spine switches, which communicate with the second layer through internal connections
- Second layer: 18 leaf switches, which communicate with the spine switches, as well as with the HCAs
- Third layer: about 320 HCAs, which communicate with leaf switches through external connections

The non-blocking feature of the fabric is possible as it has the same number of internal and external connections, which is called constant cross-bisectional bandwidth. Each of the InfiniBand switches has a radix of 40 (number of physical ports) [5].

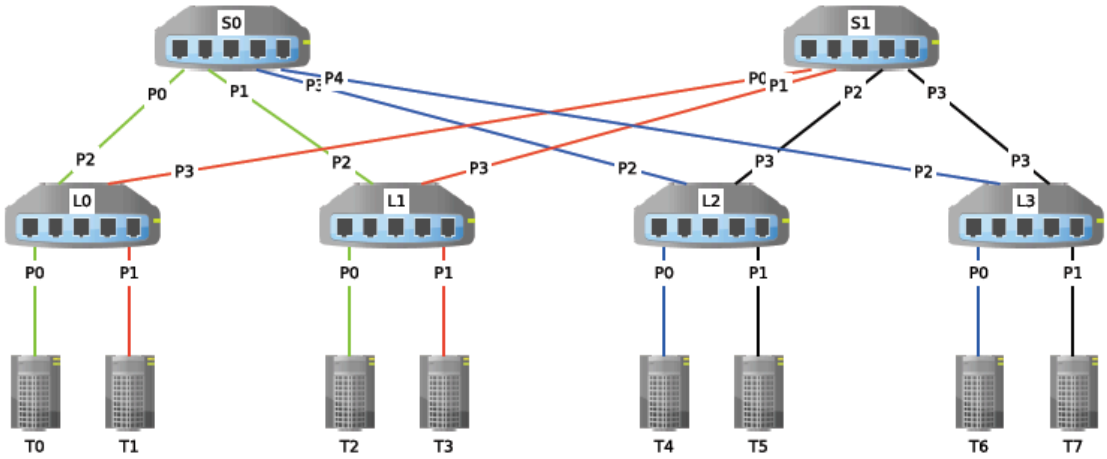

Figure 2 From [19]: example of an InfiniBand non-blocking fat-tree topology with 8 connections and 4-radix switches (2 spines and 4 leafs). The different connections' colours represent the optimal communication paths between the servers.

4. Project constraints

4.1. InfiniBand switches

The InfiniBand switches are not managed, which means that the data they report about network performance has a specific format typical of Mellanox. Instead, LHCb network runtime diagnostic tools are based on an Internet Standard called SNMP (Simple Network Management Protocol) [7]. For this reason, it is within the scope of the project to define a clear mapping between InfiniBand and SNMP in order to develop a protocol-compliant monitoring tool.

4.2. LHCb technologies for data collection and visualization

A **Prometheus monitoring server** is used in LHCb for time series data collection, which is, data scraping and storage. As explained in [8]: "Prometheus collects and stores its metrics as time series data, i.e. metrics information is stored with the timestamp at which it was recorded, alongside optional key-value pairs called labels". The project's goal is to monitor metrics, which are the numeric measurements informing about network performance. For each reading, labels give information about the reported node and port; and they allow to query, filter and group time series. Following the previous paragraph, metrics and labels are to be chosen according to SNMP.

At the end-user side in the LHCb control room, the **Grafana User Interface** [9] is used for time series visualization. The Prometheus server is configured as Grafana's data source, so Grafana queries it for data.

In order to illustrate the functionality of the developed monitoring tool, at the end of the project a dashboard example with user-friendly and custom panels has been

proposed. In this regard, metrics which allow the network congestion control (and therefore likely to be displayed in the ultimate dashboard) are:

- The link throughput for both received and transmitted data in each port
- The time during which data packets have to wait in each port before they are transmitted

Nevertheless, the deeper, in detail analysis of the collected data is beyond the scope of the project.

5. Method

In this section, the step-by-step development of the tool is described, going over the concepts introduced in the previous section.

5.1. InfiniBand diagnosis: starting point and first steps

5.1.1. InfiniBand diagnosis tools description

Given the previously described project constraints, the first task of the tool development pipeline has been obtaining the network performance metrics provided by InfiniBand. With this objective, the starting point of the project has been based on the GitHub repository *linux-rdma/rdma-core* [10]. A set of utilities designed to help configure, debug and maintain InfiniBand fabrics is provided in the packages *linux-rdma/rdma-core/infiniband-diags.* They are written in C and, together with the correponding libraries, they are compiled with CMake cross-platform software. In particular, the utilities: *ibnetdiscover*, *iblinkinfo*, *ibnodes*, *ibswitches*, *ibhosts* provide information about the nodes and the fabrics node-to-node connectivity [11,12] illustrated in section 3.

5.1.2. Black-box approach

Once the context and the network topology have been studied, the tools *ibqueryerrors* and *perfquery* have been explored, as they provide information about the network performance and error counters. With an initial black-box approach of the pre-compiled versions of the utilities (which are not very user-friendly and provide rather verbose outputs), the objective was to decide on a set of function calls with the appropriate arguments that provide all the required information for network monitoring. This black-box approach turned out not to be sufficient, as a few features remained unclear even after the documentation scrutiny; namely, the counters hierarchy and the counters reset mechanisms.

5.1.3. Reverse-engineering approach and results

The final approach has consisted of reverse-engineering the two utilities, using the GNU Project Debugger [13] with a compiled version of the utilities containing the debug information. This final approach has been successful, and the chosen set of InfiniBand diagnosis utilities is as follows:

- Data counters retrieval:
  - $ ibqueryerrors --report-port --details --node-name-map /admin/etc/ib-node-name-map
  - $ ibqueryerrors --report-port --counters --node-name-map /admin/etc/ib-node-name-map

  Where the first call provides the error counters of the faulty ports and the second call, on the other hand, provides the performance counters of all ports. The file "ib-node-name-map.txt" provides a mapping between the switches GUID (Global Unique ID) and their names.

- Data counters reset:
  - $ perfquery --Reset_only --Guid <node_guid> <port_number>
  - $ perfquery --Reset_only --Guid <node_guid> <port_number> --xmtdisc
  - $ perfquery --Reset_only --Guid <node_guid> <port_number> --rcverr

  The three calls are needed in order to reset all the counters of the port, thus avoiding inconsistencies between counters of different hierarchy.

  By default, when any of the counters reaches its maximum value, it saturates rather than overflowing. This is not convenient since the concerned metrics would not be monitored anymore until the counter is reset. However, counters resets may lead to data losses. The counters which grow very quickly have extended versions of 64-bit length, so their maximum values are not reached in several years considering the runtime data traffic.

A second result of this project stage is the modification of the *ibqueryerrors* source code in order to fix a bug concerning the counters hierarchy: the detailed counters for transmitted and received errors are swapped. The task has been carried using Eclipse CDT IDE and stored in the repository: https://gitlab.cern.ch/magomis/rdma-core-lhcb-eb-daq.

Regarding the InfiniBand counters reset mechanism, the source code has not been modified as it is not the project's priority and there is limited time for implementation. Furthermore, for maintenance reasons, only the minimal modifications have been done

in the InfiniBand source code. In this sense, the robust implementation of the application code to deal with the issues has been prioritized.

Once the InfiniBand diagnosis tools providing the performance network information have been selected as described in this section, everything is set to start developing the application that will periodically execute them. The process is described in the following sections.

5.2. Second stage: InfiniBand exporter to Prometheus

The tool architecture was not defined at the beginning of the project. Being the core of the second stage of the project, such choice has been a particularly thorough task for which helpful insights have been obtained during a meeting with the CMS DAQ team. In particular, it was confirmed that during Run 2, managed switches were used in CMS DAQ for SNMP diagnostics; whereas IB diagnostic tools were used otherwise.

Following the project constraints described in 4.2., the overall functionality of the application consists of exporting the data provided by InfiniBand diagnosis tools to the LHCb Prometheus monitoring server.

Due to the limited time for implementation, the use of existing projects has been prioritized. In this regard, the repository *gilbaults/infiniband-exporter* [14] has been chosen as a model for the application. It is written in Python and uses the Prometheus client library [15] for code instrumentation. A custom data collector is created to proxy metrics from InfiniBand diagnosis. The metrics are stored in *Counter* and *Gauge* instances, depending on if they go only up or both up and down, respectively; and they have labels which uniquely identify the nodes and ports. The metrics are exported thanks to a WSGI (Web Server Gateway Interface) application over an HTTP server.

The code for the application can be found on CERN GitLab https://gitlab.cern.ch/magomis/infiniband-exporter-snmp-compliant. Some details about the implementation are given in the next sections.

5.3. Third stage: from InfiniBand to SNMP

Following 4.1., a scrutiny of InfiniBand and SNMP metrics documentation has been done in order to set a clear mapping between the two. The goal of this stage of the project is to define protocol-compliant metrics and labels corresponding to the InfiniBand diagnosis data. The in-depth documentation task still lacked some essential information until the relevant RFC (Request for Comments) Internet Draft was found [16]. Even if the draft is not recent, most of the metrics are still valid for the latest versions of both technologies. The actual mapping table that has been built as a result of this documentation task is included in the project GitLab repository for future reference. The

resulting choice of metrics and labels to expose the data is described in the application repository README file.

5.4. Fourth stage: Prometheus client API – Telegraf

The code snippet corresponding to the data exposition through WSGI over HTTP has been left as in the original project without any modification. The Prometheus client API is called Telegraf, and it has been used in this project stage for debugging, as the Prometheus server was not configured yet. Figure 3 shows three metrics examples (two counters and a gauge) with both metrics names and labels according to the mapping introduced in the previous section.

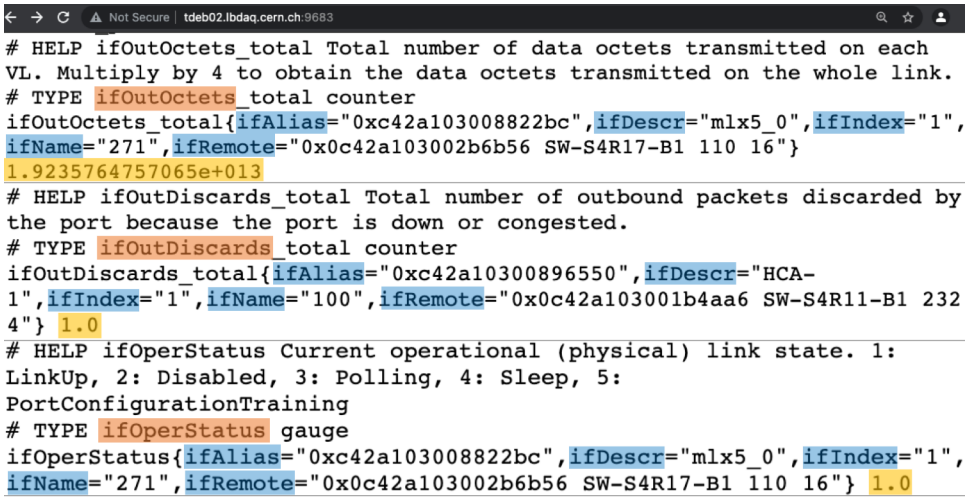

Figure 3 Examples of metrics visualization on the Prometheus HTTP API Telegraf. Metrics names highlighted in orange, labels in blue and values in yellow

5.5. Last stage: data visualization in Prometheus server and Grafana UI

Once the diagnosis data is structured according to the desired metrics and labels on Telegraf, the Prometheus server is configured so that the data can be queried on Prometheus and finally visualized on Grafana. The Prometheus server is handy at finding the queries which provide the suitable time series selection and filtering, prior to their visualization. This is done using PromQL querying language provided by Prometheus [17]. Such queries are finally used to create custom panels on LHCb Grafana dashboards explained in the following section.

6. Test and discussion

From the beginning of its development, the application was tested on the actual, rather quiet network, therefore the observed throughput was far from the nominal bandwidth of the InfiniBand links. On the last stage of the implementation, once the application was functional, it has been tested with a benchmark developed in LHCb which simulates

real event building data traffic. Figure 4 shows a Grafana dashboard as the result of one of the tests. Here the focus has been put on performance counters as they enable dealing with network congestion. Error counters, on the other hand, would mean data losses, so they may be aggregated in a single panel to raise the alarms.

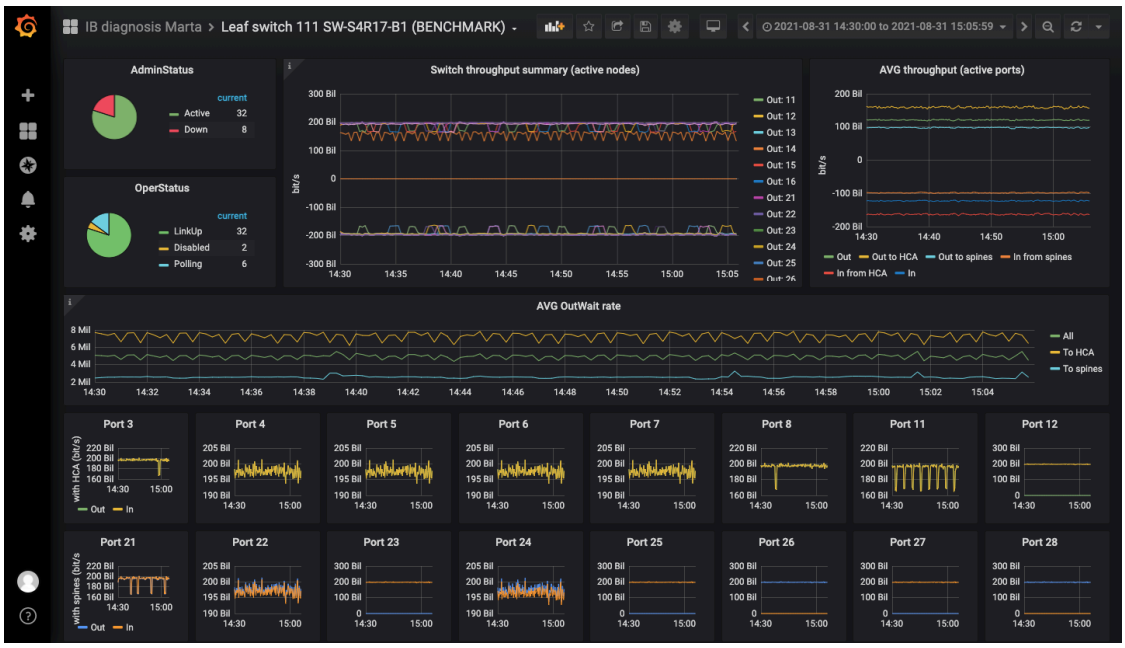

Figure 4 Example of Grafana dashboard [18] during a real event building data traffic simulation

Figure 4 shows a dashboard example with performance counters of a specific leaf switch. The summary panel at centre-top shows the throughput of all its ports in bit/s, for both the outgoing (positive) and incoming (negative) data. At right next to it, there is the averaged throughput for all ports, as well as the distinction of ports connecting to HCA and ports connecting to spine switches. The same has been done for the "OutWait" rate in the middle of the figure. It corresponds to the time during which packets have to wait in the port before being sent. Finally, the bottom-half shows the incoming and outgoing data throughput for each port. The first row corresponds to ports communicating with HCA, while the second one corresponds to ports communicating with spine switches.

As it can be observed, several ports operate at a throughput close to the nominal 200 Gbps. In general, more data is being transferred with HCA than with spine switches. As a consequence, the "OutWait" towards HCA is significantly greater. The individual port panels show that some of them send and receive data at the same rate, while others mainly receive or mainly send data.

7. Deliverables

The deliverables of the project which have been previously detailed in the report are enumerated here as a wrap-up:

- CERN GitLab repository with the InfiniBand source code bug fixes https://gitlab.cern.ch/magomis/rdma-core-lhcb-eb-daq
- CERN GitLab repository with the InfiniBand diagnostics application https://gitlab.cern.ch/magomis/infiniband-exporter-snmp-compliant
- Mapping table IB – SNMP Prometheus labels and metrics, included in the second repository
- Grafana dashboard examples (leaf switches):
    - Final version example: https://lbgrafana.cern.ch/d/EkryqaV7z/leaf-switch-sw-s3r11-b1
    - Benchmark test examples (not final metrics version):
        - Good performance: https://lbgrafana.cern.ch/d/L9055Y4nz/leaf-sw-s4r17-b1-benchmark-good-performance?orgId=1
        - Tracing bad performance: https://lbgrafana.cern.ch/d/1FP97P47z/leaf-sw-s3r10-b1-benchmark-tracing-bad-performance?orgId=1

8. Conclusions

The main objective of the project has been accomplished, as a working diagnosis tool for the new LHCb Event Builder network has been developed. In the way, a better understanding of the InfiniBand diagnosis tools has also been achieved. The main challenge has been the great amount of technologies and platforms to get familiar with. Other than that, the most challenging task has been the definition of the application architecture.

From a personal point of view, I have acquired programming competences as I have learnt a new programming language and debugged existing code which was relatively complex compared to my previous experiences. I have learnt about data acquisition systems, supercomputer networks and communication protocols; which will be very relevant for my future career.

9. Acknowledgements

I would like to thank CERN for giving me the opportunity to participate in the Summer Student Programme and for the great lectures. I am thankful to the LHCb Collaboration for the cutting-edge project in which I have worked, especially to the LHCb Online group and Niko Neufeld for welcoming me and giving me the chance to take part in the very

interesting weekly follow-up meetings. The biggest thank you to Rafal Krawczyk for his patience and help throughout the project, as well as for the knowledge I have acquired from him. Finally, I would also like to thank Tommaso Colombo for his helpful insights about dataflow during the last stage of the project.

References

[1] 32 Tb/s DAQ for the LHCb experiment at CERN
https://indico.cern.ch/event/974424/contributions/4217589/attachments/2186332/3694141/DAQFEET_9_02_21_FINAL.pdf

[2] Introduction to High-Speed InfiniBand Interconnect
https://indico.cern.ch/event/218156/attachments/351724/490088/Intro_to_InfiniBand.pdf

[3] InfiniBand Essentials
http://www.hpcadvisorycouncil.com/events/2014/swiss-workshop/presos/Day_1/1_Mellanox.pdf

[4] Introduction to InfiniBand (White Paper)
https://www.mellanox.com/pdf/whitepapers/IB_Intro_WP_190.pdf

[5] InfiniBand Switch QM8790
https://support.mellanox.com/s/productdetails/a2v50000000wUKeAAM/qm8790

[6] InfiniBand FAQ
https://www.mellanox.com/pdf/whitepapers/InfiniBandFAQ_FQ_100.pdf

[7] Net – SNMP
http://www.net-snmp.org

[8] Prometheus – Monitoring system and time series database
https://prometheus.io

[9] Grafana UI
https://grafana.com/grafana/

[10] rdma-core GitHub project
https://github.com/linux-rdma/rdma-core

[11] Infiniband-diags Linux man page
https://linux.die.net/man/8/infiniband-diags

[12] InfiniBand tools overview
https://downloads.openfabrics.org/ofv/tools/ib-tools.pdf

[13] GDB: The GNU Project Debugger
https://www.gnu.org/software/gdb/

[14] GitHub infiniband-exporter
https://github.com/guilbaults/infiniband-exporter

[15] Prometheus Python Client
https://github.com/prometheus/client_python

[16] Definitions of Managed Objects for InfiniBand Interface Types
https://tools.ietf.org/pdf/draft-ietf-ipoib-ibif-mib-09.pdf

[17] Querying Prometheus
https://prometheus.io/docs/prometheus/latest/querying/basics/

[18] LHCb Grafana dashboard
https://lbgrafana.cern.ch/d/EkryqaV7z/leaf-switch-sw-s3r11-b1

[19] Flavio Pisani, Tommaso Colombo, Niko Neufeld, Umberto Marconi, Rafal Krawzcyk, Domenico Galli, Rainer Schwemmer and Paolo Durante. “Network simulation of a 40 MHz event building system for the LHCb experiment”. *EPJ Web of Conferences* **245**, 01012 (2020), CHEP 2019. DOI: 10.1051/epjconf/202024501012