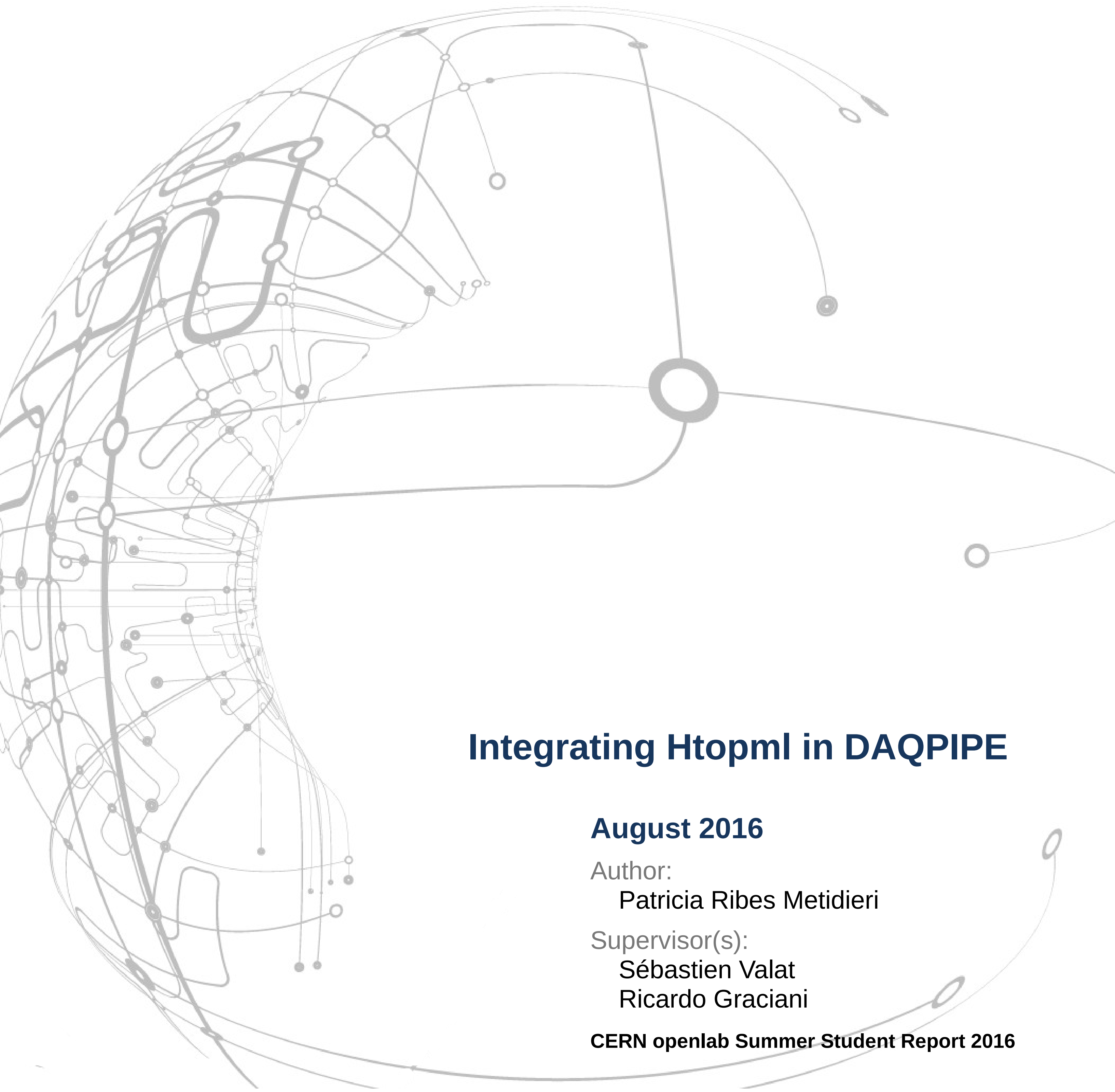


Integrating Htopml in DAQPIPE

August 2016

Author:
Patricia Ribes Metidieri

Supervisor(s):
Sébastien Valat
Ricardo Graciani

CERN openlab Summer Student Report 2016

Abstract

After upgrade of the LHCb experiment in 2020, the Data Acquisition (DAQ) system will require a 500 nodes network with 100 Gb/s each of bandwidth to handle the data coming of the detector. Currently, DAQPIPE (Data Acquisition Protocol Independent Performance Evaluator) is being used to simulate and evaluate the performance of such a DAQ system on existing hardware.

This project, *Integrating Htopml into DAQPIPE*, is divided in two stages: the goal of the first stage is to extend Htopml, a profiling tool for DAQPIPE, enabling to monitor new features of DAQPIPE for deeper analysis; and the goal of the second stage is to enable the dynamic modification of parameters in DAQPIPE through its modification in Htopml.

Table of Contents

1. Introduction

After the upgrade of the LHCb experiment in 2020 the Data Acquisition system of the experiment will have to handle 40 Tb of data per second and, to build a DAQ network able to handle such amount of data, different network fabrics are being tested using DAQPIPE (Data Acquisition Protocol Independent Performance Evaluator).

This project consists in the integration of a light profiling tool named Htopml into DAQPIPE and it is divided in to stages: the goal of the first stage is to extend Htopml, a profiling tool for DAQPIPE, enabling to monitor new features of DAQPIPE for deeper analysis; and the goal of the second stage is to enable the dynamic modification of parameters in DAQPIPE through its modification in Htopml.

This technical report first provides an overview of the topics which constitute the framework for the project, as the current state of the LHCb's DAQ system and the planned changes for 2020 upgrade, a brief description of DAQPIPE and Htopml and their state before the realization of the project. Secondly, it gives a detailed description of the tasks performed in every stage of the project, specifying the interest behind each task and the tools used to perform them. Finally, it shows some results obtained and the extracted conclusions.

2. Context

2.1. The LHCb DAQ system

Particle collisions in the LHCb experiment produce data at a rate of 40 MHz, but currently that amount is reduced to 1.1 MHz in the called L-0 trigger (implemented in hardware) before being sent to the Data Acquisition (DAQ) system. [1]

However, this hardware trigger is where the largest inefficiencies in the data selection occur, so one of the main objectives for the LHCb upgrade during the 2nd Long Shutdown (LS2) of the LHC will be to remove it in order to build a trigger-less readout system. [2]

This implies that the DAQ network of the future LHCb will have to support the complete amount of data coming out from the detector (i.e.100 KB of data per event at a rate of 40 MHz).

Therefore, in order to fulfill the requirements of the High Luminosity LHCb, it will be necessary to build a 500 nodes network and to ensure up to 100 Gb/s of readout bandwidth, which is what the current hardware provides per node.

Every second, the Readout units of all the ~500 nodes of the DAQ network will receive data coming from particle collisions and, since the information corresponding to a single event will be spread out through all the network nodes, it will have to be aggregated in a single CPU node before being sent to the Filter Units (where event selection will be performed). This process, called *Event building*, takes place in one of the Builder Units placed in every node of the network, and the Event Manager is the responsible for selecting the builder unit in which each event is processed. [3]

2.2. About Daqpipe

Providing a 100 Gb/s of readout bandwidth network is a great challenge and currently different network architectures as Infiniband, Omni-Path, 100 Gbs/s Ethernet, are being tested as candidates for the future LHCb. For this purpose, LHCB-DAQPIPE was developed.

LHCB-DAQPIPE (*Data Acquisition Protocol Independent Performance Evaluator*) is a small benchmark application which emulates reading data from input "cards" (in the real DAQ would be connected to the detector.) and the process of event building in order to test HPC fabrics for the High Luminosity LHCb. [4]

As in the real DAQ, DAQPIPE is formed on top of 4 components :

- The **ReadoutUnits (RU)** which have to read the data from the input cards (simulated detectors) and send it to the BuilderUnits.
- The **BuilerUnits (BU**) which have to aggregate the data of an event and to send it to a FilterUnit.
- The **FilterUnits (FU)** which have to select the interesting events for storage and discard those which only contain well known physics.
- The **Filter Manage**r (**FM)** which coordinates the job of the FU.
- The **EventManager (EM)** which is a sort of leader to manage the job between the ReadoutUnit and BuilderUnit.
- The **Profiling Units (PU)** where the information related to the bandwidth of the network is gathered to be exported.

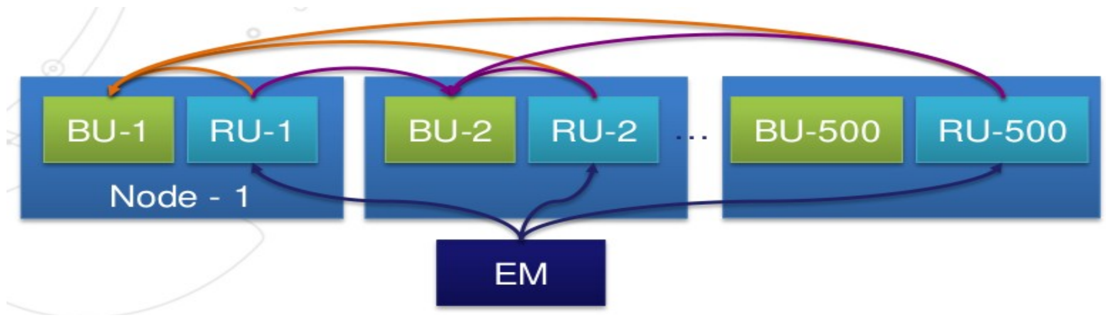


Figure 1. Event building process in the DAQ system as simulated by DAQPIPE. Image extracted from [5].

2.3. About Htopml

To optimize the usage of the network it is important to get a view on what happens in the application over time and, to this end, DAQPIPE uses Htopml. [5]

Htopml is a light profiling tool which provides a framework to export on the fly some data in JSON format and provides a nice visualization in html/javascript of the exported data to understand how DAQPIPE is working.

Htopml is also a powerful application which greatly simplifies the monitoring and debugging of different processes.

Previously to my work Htopml, it was constituted by 7 plugins:

1. **Top:** Is constituted by two graphs. The first one (on the top of the page) show the usage of CPU of the launched processes running in parallel and the second one (on the bottom of the page) the total usage of CPU.

2. **Binding**: This entry uses hwloc (Portable Hardware Locality) to display the location and binding of threads.

3. **Pull Gather**: displays the usage of the buffers.

4. **Comm tracer**: Contains 3 graphs that keep track of different aspects of the sent, received or sent and received Rdma communications between the EM and the BU's, i.e. which provides a view of the communication scheduling.

5. **Histogram**: This entry of the menu shows the distribution of communications time.

6. **Bandwidth**: this plug-in contains 2 graphs. The graphic on the top shows the bandwidth distribution over nodes, while the bottom graph displays the aggregated bandwidth of all the nodes over time.

and could not support the introduction of data by the user.

3. Integration of Htopml into DAQPIPE

3.1. General idea

The first stage of the project was, on the server side, to get familiar with DAQPIPE in order to decide where to place additional code to export relevant data into JSON format and, on the client side, to build new pages in Htopml to read the exported data and visualize the extracted information as new plugins in the application.

As Htopml is a profiling tool, it is important to ensure that DAQPIPE can run independently and, therefore, to ensure that the export of data to JSON format only takes place when DAQPIPE is launched with Htopml.

The work carried out to build the new plugins in Htopml was performed using HTML and javascript libraries like HighCharts, d3js and jQuery.

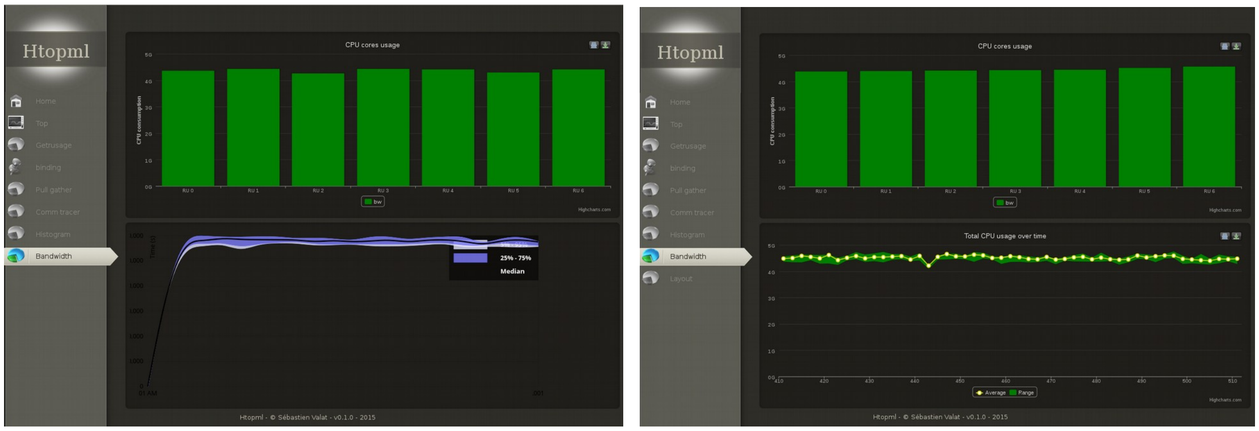
- **HighCharts:** :charting library written in javaScript which allows to easily set interactive charts into web sites.

- **jQuery:** is a widely used cross-platform javaScript library with a notation to simplify javaScript programming.

- **d3.js (Data-Driven Documents):** javaScript library for data manipulation and visualization.

3.2. Implementation

3.2.1. Improvement of one of Bandwidth plugin's graphs



(a) (b)

Figure 2. Screenshot of the Bandwidth menu of the Htopml application previous to the project (a) and after the modifications performed as part of the project (b). When the screenshots were taken, DAQPIPE was running locally with 8 processes.

Bandwidth plugin consists on two graphs, as shown in Figure . The graphic on the top of both images (a) and (b) the bandwidth in each of the cores, while the bottom graphic shows the maximum, minimum and mean of the aggregated bandwidth of the network over time.

The original graph of aggregated bandwidth over time was implemented using javascript library d3js. It was replaced by another graph with the same functionality but implemented using the javascript libraries jQuery and HighCharts.

JQuery's *$.Ajax()* method is used to handle the Http requests to Bandwidth.json node, so the values of Bandwidth can be updated.

3.2.2. Implementation of Layout plugin

As in the real Data Acquisition network, in DAQPIPE each of the nodes of the network (or ranks) can perform several tasks, i.e., is constituted by more than one unit. Therefore, it is interesting to know the layout of the different ranks of the network in order to better monitor the flow of communications between units and the process of event building, that is the motivation behind building Layout plugin as part of the first stage of the project.

Moreover, DAQPIPE currently provides a support to simulate failure and manage to continue running. The failure simulation is currently done on software side by ignoring all messages on the failed node.

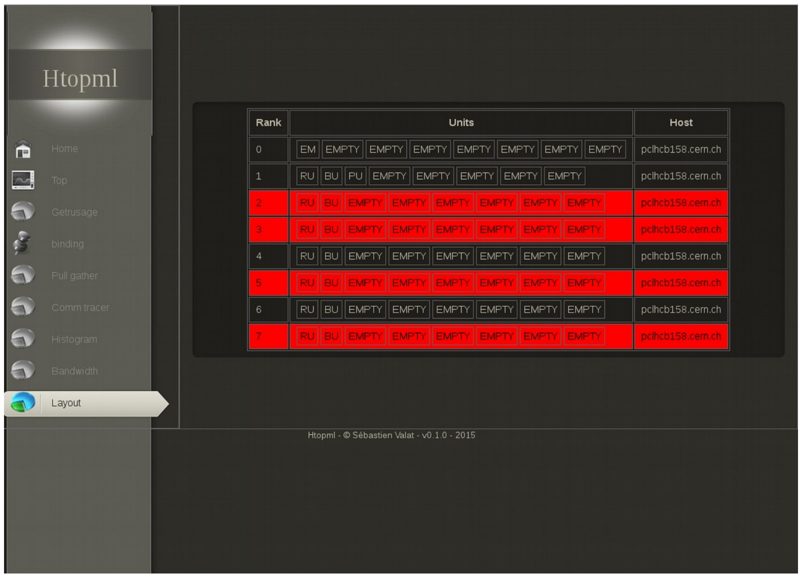
This plugin shows the internal configuration of the different nodes or ranks that conform the DAQ network, i.e. (BU, RU, EM, PU, FU or FM) and the state (functional or failing) of a node when the failure simulation mode of DAQPIPE is active.

In DAQPIPE, *HtopmlLayoutHttpNode* class and *Layout structure* were created in order to collect the internal configuration of the different ranks and the information, collected in the transport layer, was sent to the Units layer to be exported.

The exported data from DAQPIPE for this purpose consists in an array of objects with length equal to the number of nodes in the network. Each object contains three properties

- **Host:** shows the identification label of the node the layout of which is being studied.
- **Units:** Array which contains the type of units hosted in each node up to a maximum number of units per node.
- **Fail:** boolean which gives the state of each node. Its value is set to true when DAQPIPE simulates the fail of a node.

In Htopml, a table with the exported information was created in Layout plugin using jQuery and d3.js.



Rank	Units	Host
0	EM EMPTY EMPTY EMPTY EMPTY EMPTY EMPTY EMPTY	pclhcb158.cern.ch
1	RU BU PU EMPTY EMPTY EMPTY EMPTY EMPTY	pclhcb158.cern.ch
2	RU BU EMPTY EMPTY EMPTY EMPTY EMPTY EMPTY	pclhcb158.cern.ch
3	RU BU EMPTY EMPTY EMPTY EMPTY EMPTY EMPTY	pclhcb158.cern.ch
4	RU BU EMPTY EMPTY EMPTY EMPTY EMPTY EMPTY	pclhcb158.cern.ch
5	RU BU EMPTY EMPTY EMPTY EMPTY EMPTY EMPTY	pclhcb158.cern.ch
6	RU BU EMPTY EMPTY EMPTY EMPTY EMPTY EMPTY	pclhcb158.cern.ch
7	RU BU EMPTY EMPTY EMPTY EMPTY EMPTY EMPTY	pclhcb158.cern.ch

Htopml - © Sébastien Valat - v0.1.0 - 2015

Figure 3. Screenshot of the added menu. The table shows, in the first column, the rank (number) of the node, in the second column, the unit layout of each node and on the last column, the name of the host. The rows in the table are shown red if the node fails.

3.2.3. Implementation of Balance plugin

As previously mentioned, the Event Manager is the element responsible of the coordination of all the nodes in the network in the event building process, as it is the element which selects the BU in which every event is built.

In DAQPIPE there are two different protocols of communication implemented between the EM and the BU/ RU, represented in Figure :

- PULL: the EM first sends a command to the BU of one of the nodes of the network, in which the event will be build. Then, the BU sends commands to all the RU to gather their information.
- PUSH: the EM sends commands directly to all RU indicating the BU to which they have to send their information.

  To improve the performance of the network, the EM should grant the same number of built events in each BU, because otherwise some nodes would not be completely used while others would be saturated. For that reason, the Balance plugin was implemented to obtain feedback on the quantity of tasks that the Event Manager sends to the different he BU for both communication protocols, PULL and PUSH.

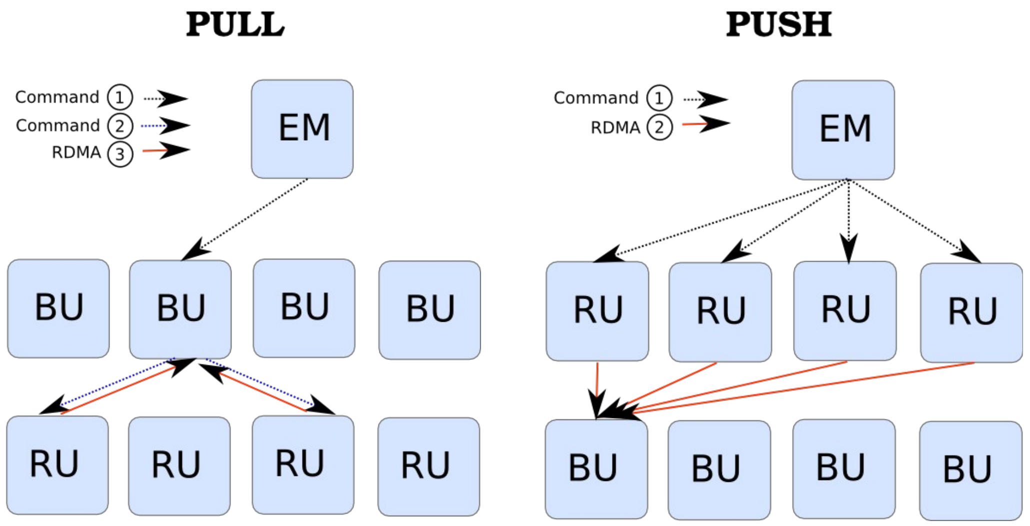


Figure 4. Communication protocols PULL and PUSH implemented in DAQPIPE. Images extracted from [5]

In order to export the relevant data in this case, a new type of command, UNIT_CMD_BALANCE_TRAKING has been introduced in the transport layer, so every time that the EM receives this type of command, it sends the identification of the node in which an event is being built to the PU in which the number of events built in each rank is counted and exported to JSON format.

In Htopml, Balance Plugin presents a single live data bar graph implemented with HighCharts and jQuery, as shown in Figure 5.

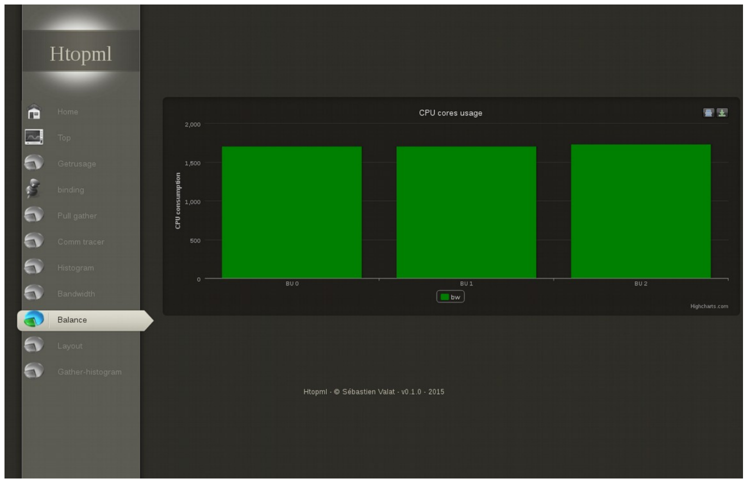


Figure 5. Screenshot of Balance plugin. The graph shows the number of built events in every BU.

3.2.4. Implementation Gather-Histogram plugin

Once a BU receives from the EM the command to build an event, it sends commands to all the RU to start the gather of the spread information, but the time it takes for the information from the different ranks to reach the BU depends on the topology of the network. Gather-Histogram plugin was built as part of the first part of the project in order to study the time it takes for the information to gather in a given BU.

In DAQPIPE, the time for each gather is recorded in the BU and the histogram is built using Histogram class. Finally, the data is exported to JSON format.

In Htopml, Gather-Histogram plugin shows a histogram with the time to gather the spread out information on the x axis and the number of times a gather takes a given time in the y axis, as it is shown in Figure 6:

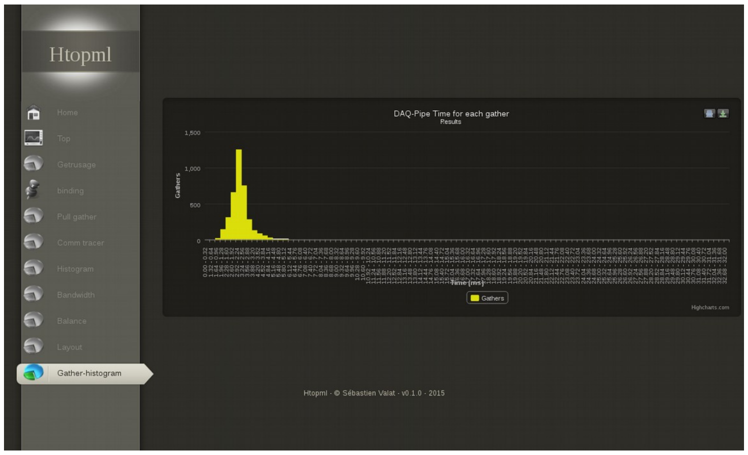


Figure 6. Screenshot of Gather-Histogram plugin.

4. Implementation of the dynamic configuration in Htopml and DAQPIPE

4.1. General idea

The second stage of the project was to enable the introduction of data by the user in Htopml (which was not supported originally) in order to allow changing DAQPIPE's configuration dynamically. The dynamic configuration was applied to different relevant parameters for the bandwidth of the network as Rdma send, Rdma received, the number of threads running in parallel, the maximum size of the built events and buffer size because originally, the previously mentioned parameters were set in DAQPIPE through a config file when the program was launched and their values couldn't be changed in execution time.

In this part of the project, the work was developed mostly on server side using C++ language and Mongoose C++ library to handle Http requests. For the interaction between the user and Htopml, buttons and sliders were added in Bandwidth plugin using jQuery, d3.js and bootstrap-slider javaScript libraries.

4.2. Implementation

From the client side, $.Ajax() was used to send, using the GET method, the information introduced by the user and Mongoose C++ library was used to handle the Http requests. The class DynamicConfigHttpNode and the struct DynamicParam were created in Htopml to decode the information introduced by Mongoose functions as an url, store their values and modify them in DAQPIPE.

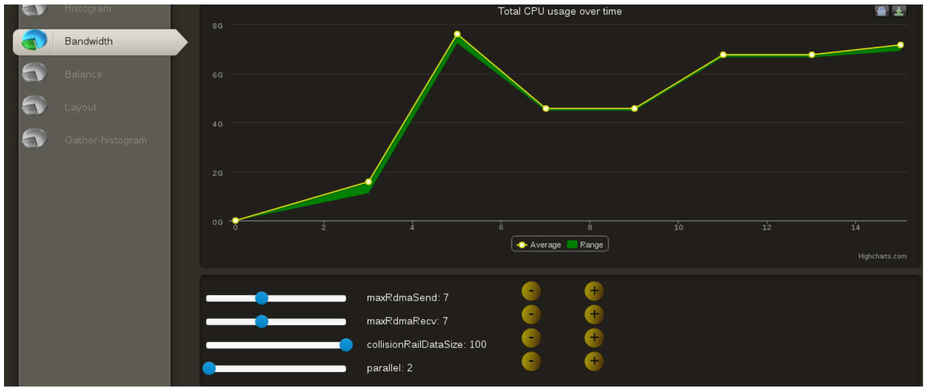
The Dynamic Configuration mode was implemented in general and then used in some relevant parameters for the bandwidth of the network.

4.3. Applications

One of the challenges the LHCb experiment has to face before the hardware trigger retrieval is to provide a 100 Gb/s bandwidth and 500 nodes network. For this purpose, different network fabrics like 100 Gb/s Ethernet, Omni-Path, ...are being tested, and it has been found out that the bandwidth strongly depends on the number of Remote Direct Memory Access (Rdma) communications on the fly, the number of threads running in parallel, the maximum size of the built events and on the size of the buffers. For this reason, the dynamic configuration was applied to those parameters so their values can be dynamically changed in DAQPIPE by modifying their value in Htopml.

Buttons and slider were added in Bandwidth plugin of Htopml's user interface, as shown in Figure 7, to allow their easy modification. The information with the new values is codified in an URL and sent as an input using GET protocol to *http://localhost:8080/daqpipe/dynamicConfig.json*, so it can be read from DAQPIPE.

The sliders have been added in Bandwidth plugin so the user can directly see the impact of changing the values of those parameters in the Bandwidth.



maxRdmaSend: 7
maxRdmaRecv: 7
collisionRailDataSize: 100
parallel: 2

Figure 7. Screenshot of Bandwidth plugin showing the implementation of buttons and sliders for different parameters.

5. Results

On one hand, the goal of this project was to further extend Htopml, a profiling tool for DAQPIPE. As a result of the work developed, 4 more plugins were integrated in Htopml which will help to monitor new import ant features of the DAQ network. Some of these features are the change of the bandwidth, the internal constitution of the nodes, the quantity of tasks the EM sends to the BU and the time it takes to gather the information spread out through all the nodes.

On the other hand, the effect of changing dynamically the parameters mentioned in section 4.3. was tested in our local test cluster. It was observed that the bandwidth ranged from for 1Gb/s to 16 Gb/s.

As the values of the relevant parameters can be modified in execution time, the task to find their optimum combination (i.e. the combination of values that grants the highest bandwidth) becomes faster. As an example of this last part, last month DAQPIPE ran in a supercomputer in Italy named MARCONI and it took 4 hours to perform an automatic analysis to determine the optimized values.

6. Conclusions

The goal of the first stage of the project was to integrate into DAQPIPE a profiling tool, Htopml, in order to offer a better view of the processes taking place into DAQPIPE to allow performing deeper analysis and it was achieved with the introduction of 4 new features in Htopml which will provide feedback of relevant data in the testing of the DAQ network.

In particular, when the failure simulation (currently in software) provided by DAQPIPE is implemented in the real DAQ system, the display of the failing units in Layout plugin will provide useful information to determine both the number and identification of failing nodes of the network.

The goal of the second stage of this project was to enable the dynamic modification of parameters in DAQPIPE through its modification in Htopml, which we were able to implement.

In the tests performed on LHCb local test cluster, this new feature has proved to decrease the time required to find the optimal value of the parameters which affect the bandwidth of the network.

7. References

[1] G. Antichi , M. Bruyere, D. H. Cámpora Pérez , G. Liu , N. Neufeld , S. Giordano , P. Owezarski and A. W. Moore, *Time Structure Analysis of the LHCb DAQ Network.*

[2] *Technical Design Report. LHCb Trigger and Online, 22 May 2014.*

[3] N. Neufeld, *LHC Trigger & DAQ - an Introductory Overview.*

[4] S. Valat, *LHCB-DAQPIPE,* https://gitlab.cern.ch/svalat/lhcb-daqpipe-v2

[5] S. Valat, *HTCC coffee, DAQPIPE status, 1 February 2016*