Balancing the Resources of the High Level Trigger Farm of the ATLAS Experiment

N Garelli1, M T Morar1,2 and W Vandelli1
1 CERN, European Organization for Nuclear Research, CH-1211, Geneva, Switzerland
2 School of Computer Science, The University of Manchester, Oxford Road, Manchester M13 9PL, UK
E-mail: nicoletta.garelli@cern.ch, tudor.morar@cern.ch, wainer.vandelli@cern.ch

Abstract. The ATLAS High Level Trigger (HLT) is organized in two trigger levels running different selection algorithms on heterogeneous farms composed of off-the-shelf processing units. The processing units have varying computing power and network connectivities. The ATLAS working conditions are changing due to the continuous optimization of the LHC operations and the consequent trigger adjustment. In addition, the rolling expansion and replacement of the HLT hardware changes the HLT farm composition. Therefore, balancing the available resources is essential for optimizing the HLT farm exploitation. In this paper, a tool for managing the HLT resources will be presented. The tool allows for showing, modifying and generating the HLT farm configuration, keeping the resource balance, in terms of computing power and bandwidth, across the farms under control.

1. Introduction

ATLAS (A Toroidal LHC ApparatuS) [1] is one of two large general purpose particle detector operating at the Large Hadron Collider (LHC) [2]. The LHC is designed to deliver proton-proton collisions at the center-of-mass energy of $\sqrt{s} = 14$ TeV, which will be reached in the next years. In 2012 the LHC operates at 8 TeV.

The instantaneous design luminosity is $10^{34}$ cm$^{-2}$s$^{-1}$. While this value has not been reached yet, the instantaneous luminosity steadily increased in the course of the years. As of 4 June 2012, a peak of more than $5.5 \cdot 10^{33}$ cm$^{-2}$s$^{-1}$ has been reached.

By the end of 2011 LHC delivered a total integrated luminosity of 5.6 fb$^{-1}$ and ATLAS successfully recorded about 94% of it.

A direct consequence of the instantaneous luminosity rise is the increase of the amount of data the ATLAS trigger computing farm must process. The data acquisition farm undergoes a rolling replacement and expansion to keep up with the processing demands.

This paper describes the HLT racks tool, referred to as “Hilltrot”, which helps with the configuration management of the ATLAS trigger computing farm, balancing resources to match the processing needs at any given moment.

2. The ATLAS HLT

The ATLAS trigger and data acquisition system, referred to as TDAQ [3], conveys data from the detector front-end to the mass-storage system, reducing the event rate from 40 MHz down
to an average rate of several hundred Hz. A multi-level trigger system is exploited to select interesting events.

The first trigger level (Level 1) is implemented in hardware and uses only the information coming from the muon detectors and the calorimeters to reduce the event rate to 75 kHz. The other two levels, namely Level 2 and Event Filter (EF), are software based and collectively called High Level Trigger (HLT).

The Level 2 takes decisions based only on a subset of the event data. This subset comes from the so-called Regions of Interest, which are limited areas of the detector defined by the Level 1. The Event Filter (EF), instead, analyzes complete events, assembled by the Event Builder (EB).

The TDAQ system is based on two Ethernet network domains, coupled by the EB. On the first network, referred to as “Data Collection” (DC), the data are transferred from the detector read-out system to the Level 2 and the EB. The second network, called “Back-End” (BE), moves data from the EB to the EF, then to the ATLAS local mass-storage. Both network domains are internally split into two distinct subnets, implemented on separate hardware to provide additional redundancy.

The HLT computing farm accounts for about 1600 machines, organized in 35 so-called XPU racks and 14 dedicated EF racks. The XPU racks are connected to both TDAQ data networks, and can execute either the Level 2 or the EF workload. This introduces extra flexibility to the computing resource assignment, which can be completely managed at the software level.

Currently, three generations of XPU racks are in use. All the XPU racks are connected to the data collection network by a 10 Gbps link. The two newer generations are equipped with 10 Gbps links also for the back-end, and host 40 or 32 nodes respectively. The XPU racks of the oldest generation host 31 nodes, and are connected to the back-end network by two trunked 1 Gbps links each. The 14 dedicated EF racks are connected only to the back-end network with 10 Gbps link and host 31 nodes each. The CPU models vary across the HLT farm, contributing to the heterogeneity of the system. Currently, three types of processor are in use: Xeon E5420, Xeon E5540 and Xeon X5650. In order to classify the performance, the HEP-SPEC benchmark [4] has been used. The actual composition of the HLT farm is illustrated in Table 1.

| # racks | Type | DC connectivity | BE connectivity | CPU model | # cores per node | # nodes per rack |
|---------|------|-----------------|-----------------|-----------|-----------------|-----------------|
| 11      | XPU  | 10 Gbps         | 2-1 Gbps        | E5420     | 8               | 31              |
| 10      | EF   | -               | 10 Gbps         | E5540     | 8               | 31              |
| 12      | XPU  | 10 Gbps         | 10 Gbps         | X5650     | 12              | 40              |
| 4       | EF   | -               | 10 Gbps         | X5650     | 12              | 31              |
| 12      | XPU  | 10 Gbps         | 10 Gbps         | X5650     | 12              | 32              |

Table 1. The HLT racks currently in use

3. The HLT resource balancing

The TDAQ point of operation depends on evolving parameters, such as the trigger requirements as defined by the physics program. Therefore, a mathematical model of the TDAQ operational capabilities has been developed to evaluate the operational envelope and the best HLT resource usage.

In the model we assume: a fixed Level 1 trigger rate, a constant average event size, and all the HLT racks, both XPU and dedicated EF, available and in use.

Figure 1 shows the outcome of the model with the current hardware configuration, a Level 1 trigger rate of 70 kHz, and an average event size of 1.6 MB. The two axes report the Level 2
and the EF maximum average processing time per event allowed by a given farm configuration. The oblique lines represent a fixed Level 2 accept rate. Each step in the lines corresponds to moving an XPU rack between EF and Level 2. The performance of the XPU rack generations is reflected in the different step sizes. The dashed vertical lines delimit three regions according to the rack generation. The racks in the leftmost region belong to the oldest generation. The central region corresponds to the generation with 40 nodes, and the right one to the generation with 32 nodes.

A point on an oblique line splits the steps in two groups: the upper-left one represents XPU racks used as Level 2, while the lower-right one XPU racks used as EF. Depending on the desired Level 2 and EF latency and the Level 2 accept rate, one can determine the minimum number of XPU racks needed to run as Level 2 and as EF.

The narrow, rightmost region of Figure 1 represents the area where the average Level 2 processing time is so high, that even using all the XPU racks as Level 2 would not be enough to sustain the workload. The EB is bandwidth limited to a rate of 6.6 kHz, as shown by the bottom area below the 6.6 kHz line.

The model provides a good initial HLT rack sharing with respect to the predicted trigger requirements. The sharing is then refined, closely following up and predicting the changes of the operation point.

![Figure 1](image.png)

**Figure 1.** Outcome of the model of the TDAQ operational capabilities. The x-axis reports the Level 2 average processing time per event and the y-axis the EF one. The oblique lines represent a fixed Level 2 accept rate. A detailed description of the plot interpretation is provided in Section 3.

4. **Hilltrot**

Hilltrot is a tool to balance the HLT resources, defining the rack sharing across the two farms, and more in general, used and idle racks. It handles the TDAQ configuration files and accounts for a user-friendly GUI.
4.1. Use cases
A recalibration phase for the HLT software is required after any optimization of the LHC performance, in particular when the instantaneous luminosity increases, and after changes of the trigger algorithms. The decision time and the rejection factors of the trigger algorithms have to be carefully tuned with respect to the physics program.

Therefore, balancing the computing power between the Level 2 and the EF farms is crucial during the calibration of the trigger to set the optimal configuration for every use case.

The HLT computing resources are progressively installed following the needs of ATLAS and older components are replaced to keep the system up to date with the progress of technology.

The LHC duty cycle foresees a week without beam every few months. During these periods the TDAQ system is tested and frequent changes of the HLT rack sharing are required.

The TDAQ experts therefore need a tool for changing the rack sharing to invoke before the start of the data taking session, even between two consecutive sessions, if necessary. The tool has to provide information about the available bandwidth and the CPU power in the HLT farms for each configuration.

4.2. The TDAQ configuration
The ATLAS TDAQ system uses a modular approach for configuring all its components. It keeps the settings split in a hierarchical structure of files, which sum up to a size of more than 160 MB [5]. The information in the ATLAS TDAQ configuration files is stored in an object-based format. From a practical point of view, changing the rack sharing requires to interact with the TDAQ configuration system. At this level the state and the usage of each rack is defined.

4.3. Hilltrot implementation
The TDAQ libraries for handling configuration files are at the core of the project, while the Qt4 widget toolkit [6] was used for the graphical user interface. In particular, we choose to use Python [7], taking advantage of the existing bindings to Qt4 and the TDAQ libraries.

The implementation is in the form of loosely coupled classes. The module structure is shown in Figure 2. The design of Hilltrot is steered by a clear separation between the core part, which

![Figure 2. Internal structure of Hilltrot, showing the main modules and their hierarchy. The white modules implement the main functionality, while the gray ones implement the user interface. In the dashed rectangle the two external libraries, PyQt4 and the TDAQ libraries, are depicted.](image-url)
is an interface to handle the HLT configuration, and a user interface part. Because of this structure, user interfaces, like a command line or text interface can be implemented without code duplication.

The white modules in Figure 2 are the core of Hilltrot. The most basic class is the Segment class, following a specific ATLAS naming convention. Instances of this class are the Level 2 and the EF farms, as well as all the HLT racks. A segment object representing a rack has different states: Enabled, Disabled (it means that it is disabled in the current farm, but enabled in the other), Conflict (it means it is enabled in both farms), and Unused (it means it is not in use in any farm).

The Hilltrot module defines the Database class, which deals with loading and saving the TDAQ configuration files and creates the instances of the Segment class. The XPU racks are all set to the Conflict when a new TDAQ configuration file is loaded, since they are ready to be used in both farms. The Segment class handles conflicts automatically by disabling the rack in Level 2 if one enables it in EF, and vice versa. In this way, it is very easy for the user to avoid overbooking.

The other modules, hWidgets, guiCore and hilltrotQt, implement the GUI. The hWidgets module contains classes which wrap the PyQt widgets, changing the default settings and providing convenience methods. The guiCore module, instead, provides the main window content. It is used both by the hilltrotQt, which provides the window itself and the menus, and by the Hilltrot plugin for dbe [8], the general purpose TDAQ configuration editor.

![Figure 3](image.png)
Figure 3. A typical Hilltrot session, displaying the main parts of the GUI

4.4. User interface
An easy to use GUI has been developed as part of the Hilltrot project. As can be seen in Figure 3, the main design item is a two-panel view, one for the Level 2 farm and one for EF farm. Each panel shows the racks, colored differently based on their states. The HLT farm configuration can be easily modified by selecting one or more racks, then enabling or disabling the selected racks from a right click pop-up menu. Below each of the two panels there is the number of racks for each state, together with the respective color for explanatory purposes. Along with the number of racks, cumulative farm information is displayed:
• the total available bandwidth for the data collection (on the Level 2) and the back-end network (for the EF);
• the total number of cores in use;
• the CPU weighted number of cores in use - this takes into account the measured performance difference for the installed CPU models.

Of course, the number of cores has a direct impact on the available latency at each trigger level. This cumulative farm information greatly improves the overview of the available resources the user has, and allows for better and easier tuning of the HLT farm. Within the information, the number of racks connected to each individual subnet is also shown. A good balance of the resources connected to the two data collection and back-end networks is important to maximize the benefit of the network redundant structure and avoid bottlenecks.

At the bottom a log window is used to display errors, warnings, or the current status of the segments in text format - an useful feature for logging purposes. Warnings are displayed on automatic conflict handling, to inform the user of the exact action done.

A search bar can be found at the top, and it is used to quickly locate a rack, filtering out the other ones.

5. Conclusions
Hilltrot is a tool to balance the resources of the ATLAS HLT farm. It has been successfully deployed at the beginning of 2012 and it has been extensively used by the TDAQ experts during the trigger recommissioning phases and between data taking sessions. Hilltrot contributes to the outstanding performance of the TDAQ system which has to cope with the frequent changes of working point due to the LHC operational evolution.

References
[1] ATLAS Collaboration 2008 “The ATLAS experiment at the CERN Large Hadron Collider” J. Instrum. 3 S08003
[2] Evans L and Bryant P (editors) 2008 “LHC machine” J. Instrum. 3 S08001
[3] ATLAS Collaboration 2003 “ATLAS, High-Level Trigger, Data Acquisition and Controls: Technical Design Report” CERN/LHCC/2003-022 (Geneva: CERN)
[4] Michelotto M et al. 2010 “A comparison of HEP code with SPEC benchmarks on multi-core worker nodes” J. Phys.: Conf. Ser. 219 052009
[5] Soloviev I “The Version Control Service for ATLAS Data Acquisition System Configuration Files” in the proceedings of this conference
[6] Nokia Corporation 2012 “The Qt framework” http://qt.nokia.com/
[7] Python Software Foundation 2012 “The Python programming language” http://www.python.org/
[8] Bianchi R M et al. 2011 “Control and configuration of the ATLAS trigger and data acquisition system during data taking activities” PoS(EPS-HEP2011)401