The ATLAS High Level Trigger Steering

N Berger[1], T Bold[2], T Eifert[3], G Fischer[4], S George[5], J Haller[6], A Hoecker[7], J Masik[8], M zur Nedden[4], V Perez Reale[7], C Risler[4], C Schiavi[9], J Stelzer[7] and X Wu[3]

[1] Institut National de Physique Nucleaire et de Physique des Particules (IN2P3-LAPP) Laboratoire d'Annecy-le-Vieux de Physique des Particules
[2] University of California Irvine, Department of Physics and Astronomy, and AGH University of Science and Technology, Krakow
[3] Geneva University (DPNC) Nuclear and Corpuscular Physics Department
[4] Humboldt-Universität zu Berlin, Department of Physics
[5] Royal Holloway, University of London, Department of Physics
[6] Hamburg University and DESY
[7] CERN
[8] Manchester University, Department of Physics and Astronomy
[9] INFN, Sezione di Genova

E-mail: `S.George@rhul.ac.uk`

Abstract. The High Level Trigger (HLT) of the ATLAS experiment at the Large Hadron Collider receives events which pass the LVL1 trigger at ~75 kHz and has to reduce the rate to ~200 Hz while retaining the most interesting physics. It is a software trigger and performs the reduction in two stages: the LVL2 trigger and the Event Filter (EF). At the heart of the HLT is the Steering software. To minimise processing time and data transfers it implements the novel event selection strategies of seeded, step-wise reconstruction and early rejection. The HLT is seeded by regions of interest identified at LVL1. These and the static configuration determine which algorithms are run to reconstruct event data and test the validity of trigger signatures. The decision to reject the event or continue is based on the valid signatures, taking into account pre-scale and pass-through. After the EF, event classification tags are assigned for streaming purposes. Several new features for commissioning and operation have been added: comprehensive monitoring is now built in to the framework; for validation and debugging, reconstructed data can be written out; the steering is integrated with the new configuration (presented separately), and topological and global triggers have been added. This paper will present details of the final design and its implementation, the principles behind it, and the requirements and constraints it is subject to. The experience gained from technical runs with realistic trigger menus will be described.

1. Introduction

Large Hadron Collider (LHC) at CERN, Geneva, is nearing completion. It will ultimately provide proton-proton collisions at a centre-of-mass energy of 14 TeV, a design luminosity of $10^{34}\,\mathrm{cm}^{-2}\,\mathrm{s}^{-1}$ and a bunch-crossing rate of 40 MHz. ATLAS is a general purpose experiment for the LHC which is described in [1, 2]. The primary physics goals of ATLAS are to understand the mechanism for electroweak symmetry breaking and to search for new physics at the TeV energy scale. The trigger and data acquisition (T/DAQ) system must work in the challenging environment of $\sim 10^9$ p-p interactions per second and the large number ($\sim 10^8$) of electronics channels of the ATLAS detector. This must be reduced to the ~300 MB/s which can be sustained to mass storage, while efficiently retaining rare physics signatures for off-line analysis. To achieve this, ATLAS has designed a three-level trigger system (see Fig. 1) [3].

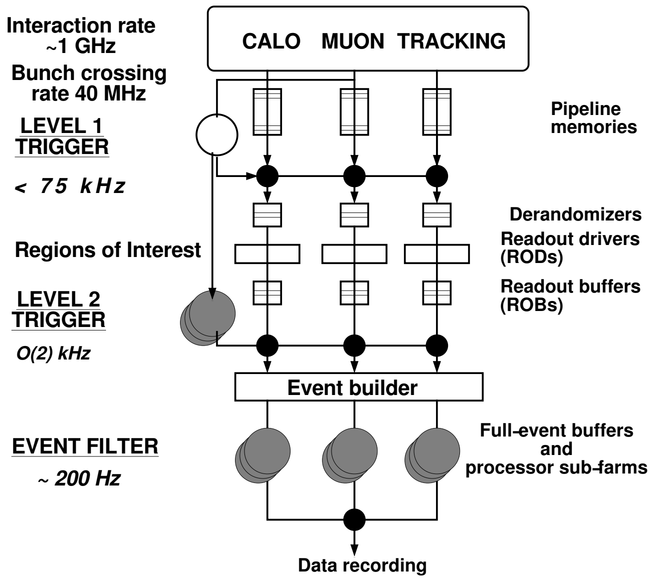

Figure 1. The ATLAS T/DAQ system. The HLT Steering runs in the LVL2 and EF processors which are shown as filled in circles.

The first level trigger (LVL1) is implemented in custom electronics (mainly ASICs and FPGAs). Its decision is based on relatively coarse data from two subsystems, the calorimeters and dedicated muon trigger stations. Event selection is based mainly on inclusive high-p_t objects (muons, electromagnetic/tau/hadronic clusters, jet clusters, missing and scalar transverse energy sums) whose trigger thresholds are programmable. During the LVL1 latency of 2.5 μs the data of all sub-detectors are kept in pipeline memories. For accepted events, the geometrical location of the objects, *Regions of Interest (RoIs)*, and the thresholds they passed, are sent to the second level trigger (LVL2) and the data are then transferred from the pipeline memories to the Read-Out Buffers (ROBs). LVL1 reduces the event rate from the initial 40 MHz to about 75 kHz.

The High Level Trigger (HLT) is a software-based trigger, running on farms built from commodity computing and network technology. It is an asynchronous, distributed system. The HLT is subdivided into LVL2 and the Event Filter (EF). LVL2 reduces the output rate to around 2 kHz. The EF should further reduce the rate to ~200 Hz. Both levels have access to the full granularity of all the detector data and follow the principle of further refining the signatures identified at LVL1. LVL2 must retrieve events fragments from the ROBs via Ethernet. It uses only data in RoIs identified by LVL1, in order to reduce the data transfers within LVL2 to under 5% of the ~120 GB/s total input bandwidth to the ROBs. LVL2 algorithms are highly optimised for speed. If LVL2 accepts an event, all the fragments from the ROBs are combined and sent to one EF processor for further consideration. The EF further refines the classification of LVL2, using the extra time to run more complex algorithms, often based on the same tool set as off-line reconstruction. It also benefits from more detailed calibration and alignment than LVL2.

Each trigger level must reach a decision quickly enough to handle the output rate of the previous level. Given the input rates and the number of processing cores available at each level in the nominal configuration for start up, the average decision times must be less than 40 ms for LVL2 and under 4 s in the EF. This is just an example as the relative allocation of processors between LVL2 and EF is flexible. The majority of this time is available for event processing but at LVL2 it also includes data access via the network. The architecture and present status of the ATLAS T/DAQ system is described further in [4].

The three-level architecture and the use of LVL1 RoIs for guidance keeps the event-building bandwidth under control. To minimise the HLT decision time, and hence maximise the event rate the HLT can handle, the software is designed to reject events as early as possible. The HLT Steering software is at the heart of the HLT and implements these novel features of the ATLAS HLT selection strategy, as will be described in the following sections.

Initial implementations of the HLT Steering were presented before in [6, 7, 8, 9, 10]. In the autumn of 2005 the previous implementation was reviewed. Several new concepts were introduced as a result of new work on the trigger configuration[11]. Use cases, such as *re-running of the trigger* for optimisation studies, were better understood. New functional requirements were identified, for example extended support for error handling and monitoring. The new implementation was completed in the spring of 2007 and is currently being tested. This is the version that will be used for ATLAS data taking during the first LHC collisions, and is presented in this paper.

2. Requirements

The basic requirements on the HLT Steering are:

- support for the RoI mechanism: initial seeding of LVL2 by the LVL1 RoIs, and more generally to start each trigger level or step from the result of the previous one;
- early rejection: minimise the processing time by rejecting events as soon as it becomes clear that the event can no longer pass the trigger;
- the time overhead of the HLT Steering should be small compared to the overall time budget of the trigger, to leave most time for the event selection algorithms;
- operational flexibility to enable and disable triggers, adjust pre-scale and pass-through factors (between runs);
- configuration allows construction of complex menus from simple building blocks;
- work in both on-line and off-line software environments.

Apart from the obvious case of on-line data taking, there are several other scenarios in which the HLT Steering will be run. These use cases, summarised below, put some additional constraints on the implementation of the HLT Steering. Clearly the on-line requirements are the highest priority, but the ability to use the exact same software off-line to emulate, study and tune the on-line performance is highly valued.

2.1. On-line triggering

This is the primary use case. The LVL2 and EF decisions must be reached within very tight time constraints. The data returned by the HLT Steering contains the accept/reject decision, error flags, the status of the different triggers (electrons, muons, taus, jets, etc.), and various other data from intermediate processing. Certain data objects produced by trigger algorithms may be included too. From LVL2, all this information is appended to the raw event and sent to the EF, which uses some of the intermediate information and data objects to set up seeded reconstruction to pick up where LVL2 left off. The EF itself produces similar data which, along with LVL2, are included in the raw event that is ultimately stored off-line if the event is accepted. The amount of detail in these data can be increased for debugging.

The Steering supports pre-scale and pass-through triggers. Pre-scales will be used by the shift crew to control the rates of triggers which will vary with the luminosity and beam conditions.

LVL1 pre-scales can be changed during a run at a luminosity block boundary, while LVL2 and EF pre-scales must be configured before the run begins. Both pre-scale and pass-through are also used to record lower thresholds and rejected events at each level of the trigger. This is essential for commissioning, debugging and efficiency calculations. By using pre-scale and pass-through, debug information is available for rejected events too. Statistical data are also available for both accepted and rejected events through the monitoring system.

The HLT Steering runs in the off-line framework, to which it adds the functionality necessary for triggering. An interface layer based on the off-line framework makes it possible to run the Steering in both off-line and on-line environments. It is called from the on-line software application that runs on HLT farm nodes. The Steering in turn calls the HLT algorithms which have also been written within the off-line framework. The ability to run the same software in both on-line and off-line software environments is especially important in the startup phase of the experiment, at low luminosity, when the HLT will be run in transparent mode to accept all events, then the HLT can be re-run off-line to tune and optimise the algorithms on real data.

2.2. Analysis
The information produced by the steering, either from on-line data taking or an off-line run, can be accessed for subsequent analysis. Basic information about which triggers were passed is easily available in all levels of off-line data, along with the more detailed information described above.

2.3. Off-line studies
The trigger can be run as part of the off-line reconstruction on simulated or real data. In the latter case, the results can be compared to those obtained on-line. It is also possible to take the output data from reconstruction, and re-run the decision part of the trigger with different selection criteria. This functionality is aimed at optimisation studies to tune selection cuts.

3. Configuration
There is a separate paper on the trigger configuration[11] but the essential concepts needed to understand the HLT Steering are introduced here.

Table 1. Sample trigger menu table. PS indicates "pre-scale" and PT means "pass-through".

Generic name	LVL1 item	LVL2 chain	EF chain
e5	L1_EM3 (PS)	L2_e5	EF_e5
e5_PT	L1_EM3 (PS)	L2_e5_PT	EF_e5_PT
e10	L1_EM8	L2_e10	EF_e10
g10	L1_EM8	L2_g10	EF_g10
2e10	L1_2EM8	L2_2e10	EF_2e10
e20_XE12	L1_EM18_XE12	L2_e20_xe12	EF_e20_xe12
XE12	L1_XE12 (PS)	L2_xe12_PT	EF_xe12_PT

Table 1 shows a small selection from a draft trigger menu designed for start-up of the LHC. The full menu contains electron (e), photon (g), muon (mu), tau (tau), jet (j), b-jet (b), missing energy (xe), total energy (te), jet energy (je) and B-physics triggers, in single, multiple and combined triggers, with various thresholds each. It has low threshold, pre-scaled (PS) and pass-through (PT) items to help understand and cross-check the trigger. The numbers in the trigger names represent nominal thresholds in GeV. At LVL1, EM refers to electromagnetic clusters; electrons and photons cannot be separated at this level because there is no inner tracker data available.

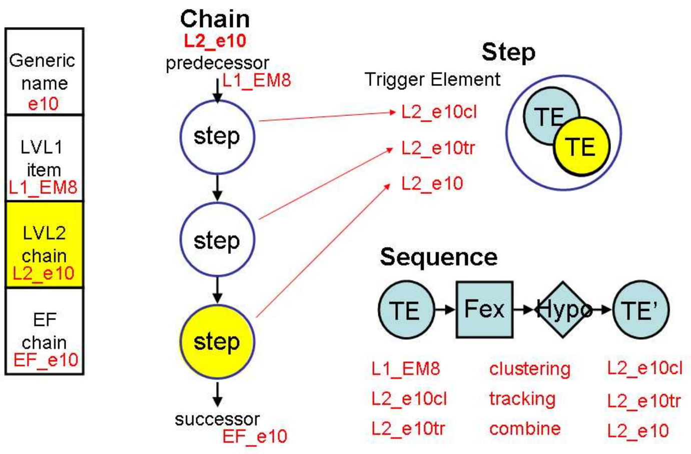

Figure 2. Illustration of the main configuration concepts of the HLT Steering. The labels are an example.

Each trigger has a generic name, with a corresponding LVL1 item, LVL2 chain and EF chain. These HLT chains are central to the design of the HLT Steering. A chain is composed of several steps. These are the steps needed to confirm or reject this particular trigger in an event. Each step represents one or more algorithms, and is specified in terms of Trigger Elements (TEs). TEs are abstract objects which represent the state of the reconstruction. Sequences define how a given TE is transformed to one or more output TEs via one or more algorithms. These concepts are illustrated in figure 2. This diagram includes labels to give an example. The row for the e10 trigger in table 1 is represented as a column on the left side of the diagram. The chain shown is the L2_e10 chain from this trigger. The preceding chain to this is L1_EM8, and the successor is EF_e10. The three steps of the L2_e10 chain are defined by the TEs they require: L2_e10cl, L2_e10tr, L2_e10. Sequences are shown which create these TEs from the previous ones. For example, a calorimeter-clustering algorithm is configured to transform a LVL1 EM cluster (L1_EM8) to a LVL2 EM cluster (L2_e10cl).

A typical sequence consists of a single input TE, a features extraction algorithm (FEX), and hypothesis algorithm (Hypo) and a single output TE. Other types of algorithm and more complex logic (multiple inputs and outputs) are also possible. FEX algorithms are normally seeded by the RoI (either from LVL1 or refined by a previous step). They retrieve detector data from within this RoI, and try to find a feature in these data, such as a track or calorimeter cluster. At the end they update the RoI position if it has been more accurately determined. Hypothesis algorithms follow FEX algorithms. Their job is to compare the features produced by the FEX algorithm against some hypothesis and mark the output TE of the sequence as valid or invalidate according to the success or failure of the hypothesis. Examples of hypotheses are: cut on the shape parameters of a calo cluster; cut on the cluster-track matching variables of an electron candidate; apply an E_T threshold.

Trigger algorithms are described more fully in [12], [13] and [14].

4. Steering Logic

The Steering takes the static configuration described above and applies it to the dynamic event state (which RoIs are active and the status of their reconstruction) in order to determine which algorithms should be run on which RoIs in what order, and ultimately to decide whether the event has fulfilled the criteria for acceptance.

The first task is creation of the initial TEs needed as input to the first sequences. This is known as the *LvlConverter*. At LVL2, this is done from LVL1 RoIs. In the EF, the initial TEs are created from the LVL2 output instead. One TE is created for each threshold of each RoI, since LVL1 RoIs can pass multiple LVL1 thresholds. Following this, all relevant chains are activated. Relevant chains are those whose predecessor was successful from the previous level. Only active chains will be processed by the HLT Steering. Since the configuration may contains hundreds of chains, but only a few (order 10) will typically be relevant for an event, this saves time.

The Steering then proceeds with the execution of all active chains. This loop stops once all chains have become inactive either because they successfully reached their last step or because they failed at any step. Each chain knows its internal step and thus knows what TEs are required when the execute method is called. For each of these required TEs, the chain runs the sequence that is configured to produce the needed type of TE. Sequences link a TE from the previous step in the chain to the TE required by the current step. Hence, executing a sequence will run the list of algorithms belonging to the sequence for each input TE. The result depends on the algorithms: the output TE is created and set either to active or inactive. TEs must be active to be used to satisfy the requirements of the chain step. If insufficient active TEs remain with respect to the requirement of the chain step, the chain is deactivated. This will avoid wasting time processing this chain in any further steps.

At the end of the loop over chains, the event was either successful, or the loop was broken prematurely. Either way, pre-scale and pass-through are applied as this could change the decision. The original outcome, the pre-scale and pass-through results are stored for each chain so the reason the event passed (or failed) can be understood later. This along with other data are compiled to make the HLT Result (one each for LVL2 or EF) which is appended to the event data for use off-line. This procedure is known as *Result Building*.

The algorithms store and exchange data via an inter-algorithm communication mechanism known as the *Steering Navigation*. This is described in [6]. This data structure is included in the LVL2 Result so that the EF can use it to resume the reconstruction from where LVL2 left off.

A caching mechanism is included to avoid unnecessary execution of sequences and algorithms. When configuring chains, it is often the case that the same sequence may be implied in more than one chain, especially in similar chains which differ only by a pre-scale factor, or that share a common starting point but differ in later steps. In this case, the sequence will be run on each RoI only the first time it is needed, and the results taken from the cache after that. It is also common that several sequences will be defined with the same FEX algorithm but different hypothesis algorithms, for example in order to apply different thresholds to the same calorimeter cluster. In this case, the FEX algorithm will only be run once for a given RoI, after which the cached results are used. The different hypothesis algorithms are run in every sequence of course. All this caching is implicit and allows a complex configuration to be built up from a common set of sequences and algorithms.

5. Performance

The performance of the HLT Steering has been measured using a recent release of the ATLAS software (13.0.25) built for the M4 cosmic run. The code was run on a 3 GHz Xeon processor. The configuration used was a LVL2 inclusive electron/photon menu which includes some low thresholds intended for commissioning the trigger at very low luminosity. Four hundred simulated top events were used for the event data. It should be noted that these events are atypically busy with an average of 4 EM RoIs per event rather than the usual 2. This means that absolute times are not very meaningful, although it should be noted that they have been shown to be consistent with the available processing power planned for LVL2 and EF.

The overhead of the Steering was found to be about 6% of the total time to process an event: around 4% for the LvlConverters, 1% for Result Building and 1% for Monitoring framework (described in the next section). This meets the aim that the overhead should be small compared to the algorithm time. Further time improvements are expected as the code is relatively unoptimised at the moment.

To see the effect of caching, the trigger was run with the mechanism disabled. The resulting time distributions are shown in figure 3. It can be seen that caching provides roughly a factor five reduction in time in this situation. The benefit will vary depending on the configuration and the event data. The absolute times in these plots are subject to the caveats expressed above.

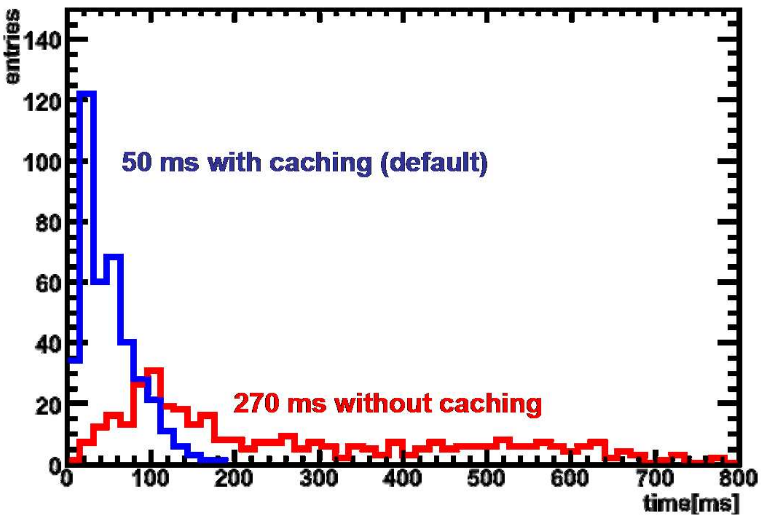

Figure 3. The time to process an event, with caching enable (default, sharp peak to the left) or disabled (long tail to the right). The times shown are subject to the caveats in the main text.

The Steering has been tested in cosmic runs, technical runs and off-line production. It has been run on-line for several hours without crashing, which indicates a reasonable level of robustness has been achieved. These tests will continue in preparation for the start of the LHC in 2008.

6. Monitoring

For successful data taking a continuous monitoring of the trigger and its performance is essential. The shift crew must be able to react immediately to malfunctions of the system in order to minimise the loss of data. Periods with bad trigger conditions or detector performance have to be identified and excluded from the off-line data analysis. Different aspects of the monitoring of the HLT can be considered: the monitoring of the HLT Steering decision and trigger rates, a persistent data quality check of the events processed by the HLT and the operational monitoring of the HLT Steering. While the data quality checks are more important for the off-line quality assessment, the rate measurement is sensitive to a stable on-line operation of the HLT, the accelerator and beam conditions of the HLT and the performance of the sub-detectors used for the trigger decision.

The HLT Steering provides a monitoring framework which is solely based on ROOT histograms. The individual trigger algorithms running in the HLT use this framework in order to fill variables sensitive to the trigger behaviour and the algorithms' performance into histograms. The Steering code itself stays independent of the monitoring code, and the monitoring histograms are configurable via the configuration files. All histograms can be filled continuously. They can also be individually reset for convenient data taking intervals like runs and luminosity blocks. In the on-line environment the monitoring histograms from the individual farm nodes are added by the On-line Histogramming Service (OH) [15] and made available for further processing. The histograms serve as basis for the on-line and off-line assessment of data quality and the trigger performance monitoring, as well as for software validation of the HLT Steering and algorithm code.

In the following some examples of variables monitored in the algorithm selecting electron candidates (*e10 calo hypo*) are given: the transverse energy of electromagnetic clusters, the ratio of the core cells energy to the total energy, the number of electron candidates as function of the pseudorapidity η can be monitored, while in the cluster algorithm e.g. the transverse energy, η and ϕ are filled into monitoring histograms. Furthermore, many histograms of algorithm execution times are produced.

In addition to the monitoring of variables inside the algorithms, the monitoring of the Steering decision itself is performed after each trigger level, where access to the full information of the event decision is available for accepted and rejected events. A brief overview over the monitored information is given below. For each trigger chain at LVL2 and the EF the number

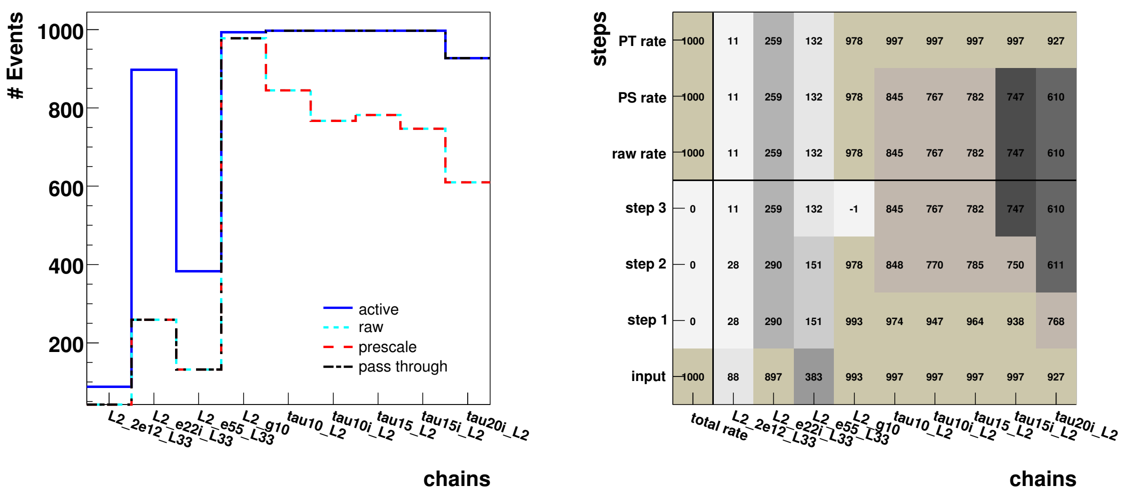

Figure 4. a) Number of events accepted by chains of LVL2: activated chain, accepted before (raw) and after pre-scale and after pass through factors have been applied b) number of events accepted by chains (x-axis) of LVL2 after each step (y-axis), including the number of input events, corresponding to step 0, as well as the accepted events at the end of chain (raw) and after pre-scale and pass-through. The total rate of the whole trigger level is also show, before (input) and after the execution of all chains.

of events for which the chain was active is monitored as well as the number of accepted events before and after pre-scale and pass-through factors have been applied. An example of the corresponding histogram based on a reduced number of chains triggering on electron, photon and tau candidates, is shown in Fig.4a). In order to control the step wise event selection on the trigger, the raw number of accepted events after each step is counted using the same code for LVL2 and EF. The final number of accepted events after the last step of the trigger level is recorded before and after pre-scale and pass-through factors have been applied. The information is stored in a 2-dimensional ROOT histogram where the y-axis corresponds to the step in the trigger decision. The bins of the x-axis correspond to the total level, groups of chains and individual chains. An example of this histogram is shown in Fig. 4b),displaying for each chain the raw number of kept events after each step as well as input and output before and after pre-scale and pass-through. The left most column shows the total number of input and output events. Since chains can have different number of steps, a value of -1 is filled in bins for steps larger than the chain length. The stepwise event information can also be monitored for groups of chains which are to be defined in the configuration. This 2-dimensional histogram is the basis for the on-line HLT rate calculation. The HLT is an asynchronous, distributed system, therefore the trigger rates are not calculated at each node separately, but rather using the summed up information from all nodes assigned to the considered trigger level. The LVL2 trigger rates for the total accepted events as well as for all individual chains at each step are derived using the LVL1 output rate, which is known to the HLT via the central trigger processor [5]. This is done by multiplying the ratio of accepted events to the total number of input events from LVL1 with the LVL1 rate. For the calculation of the EF rates the LVL2 output rate is used correspondingly.

In addition to the trigger chains also the single trigger elements are being monitored by counting all trigger elements of a given type regardless of the step at which they are used in the event decision, including the LVL1 items passed to the LVL2 trigger. All trigger elements that initiate a sequence and all active output trigger elements which were produced by a sequence are counted separately. This information allows for an additional monitoring of the trigger sequences' performance and selectivity.

Event processing at the HLT is seeded by Regions of Interest (RoIs), which also have to be monitored in order to spot malfunctioning sub-detectors or triggers. Therefore the pseudorapidity and angular distribution of the initial RoIs at LVL2 or EF are stored. The difference in the RoIs' η and ϕ between two steps provides control of the step-wise event processing and refined reconstruction at the HLT.

During the event processing different errors may occur, but the processing should continue as long as the rate of failures is acceptable. In order to control the rate and sources of errors error codes produced by the steering are also being monitored. The error codes are used to report problems from the algorithms indicating the reason of the failure e.g. missing feature or event or setup information, time outs or various exceptions as well as the taken action e.g. aborting the chain, event or job. In addition there are internal error codes from the steering itself. The error codes are filled in monitoring histograms storing information how often individual errors occurred in total and also in the execution of individual chains.

7. Conclusions

The HLT Steering implements the key features of the ATLAS HLT event selection strategy: seeded data access and reconstruction through the RoI mechanism, and step-wise reconstruction for early rejection. It supports and facilitates the building of complex menus from the simple building blocks of chains, sequences and algorithms. The built-in caching mechanism saves valuable processing time and simplifies the configuration. The time overhead of the Steering is modest. It is well instrumented for monitoring which is vital for running on-line. It has already been used successfully in technical runs and cosmic runs and these tests will continue.

Acknowledgments

The authors would like to acknowledge the collaboration and support of ATLAS experiment in general, and its T/DAQ and software projects in particular.

References

[1] Jenni, P *et al* 1999 ATLAS detector and physics performance Technical Design Report, part 1 *ser. Technical Design Report ATLAS* (Geneva: CERN)
[2] Jenni, P *et al* 1999 ATLAS detector and physics performance Technical Design Report, part 2 *ser. Technical Design Report ATLAS* (Geneva: CERN)
[3] Jenni, P *et al* 2003 ATLAS high-level trigger, data-acquisition and controls Technical Design Report *ser. Technical Design Report ATLAS* (Geneva: CERN)
[4] Gorini, B *et al* 2007 Integration of the Trigger and Data Acquisition Systems in ATLAS *in these proceedings*
[5] Jenni, P *et al* 1998 ATLAS level-1 trigger Technical Design Report *ser. Technical Design Report ATLAS* (Geneva: CERN)
[6] Berger, N *et al* 2007 The High-Level-Trigger Steering of the ATLAS experiment *Proc. Real Time 2007 (Batavia, IL)*
[7] Comune, G *et al* 2006 Steering the ATLAS High Level Trigger *Proc. CHEP06 (Mumbai)* 199 http://indico.cern.ch/contributionDisplay.py? contribId=199; sessionId=2; confId=048
[8] Grothe, M *et al* 2003 Architecture of the ATLAS high level trigger event selection software *ECONF* vol. C0303241
[9] Armstrong, S *et al* 2005 Design, deployment and functional tests of the online event filter for the ATLAS experiment at LHC *IEEE Trans. Nucl. Sci.* vol. 52 pp. 2846–2852
[10] Santamarina Rios, C *et al* 2006 Implementation and performance of the seeded reconstruction for the ATLAS event filter *IEEE Trans. Nucl. Sci.*, vol. 53 pp 864–869
[11] Stelzer, J *et al* 2007 The configuration system of the ATLAS Trigger *in these proceedings*
[12] Fonseca Martin, T *et al* 2007 Event reconstruction algorithms for the ATLAS trigger *in these proceedings*
[13] Conde Muino, P 2007 Implementation and Performance of the ATLAS Second Level Jet Trigger *in these proceedings*
[14] Emeliyanov, D *et al* 2007 Trigger Selection Software for Beauty physics in ATLAS *in these proceedings*
[15] Barczyk, M *et al* 2003 Online monitoring software framework in the ATLAS experiment *Preprint* hep-ex/0305096