Studies for a common selection software environment in ATLAS: from the Level-2 Trigger to the offline reconstruction

S. Armstrong, J.T. Baines, C.P. Bee, M. Biglietti, A. Bogaerts, V. Boisvert, M. Bosman, S. Brandt, B. Caron, P. Casado, G. Cataldi, D. Cavalli, M. Cervetto, G. Comune, A. Corso-Radu, A. Di Mattia, M. Diaz Gomez, A. dos Anjos, J. Drohan, N. Ellis, M. Elsing, B. Epp, F. Etienne, S. Falciano, A. Farilla, S. George, V. Ghete, S. González, M. Grothe, A. Kaczmarska, K. Karr, A. Khomich, N. Konstantinidis, W. Krasny, W. Li, A. Lowe, L. Luminari, C. Meessen, A.G. Mello, G. Merino, P. Morettini, E. Moyse, A. Nairz, A. Negri, N. Nikitin, A. Nisati, C. Padilla, F. Parodi, V. Perez-Reale, J.L. Pinfold, P. Pinto, G. Polesello, Z. Qian, S. Resconi, S. Rosati, D.A. Scannicchio, C. Schiavi, T. Schoerner-Sadenius, E. Segura, J.M. de Seixas, T. Shears, S. Sivoklokov, M. Smizanska, R. Soluk, C. Stanescu, S. Tapprogge, F. Touchard, V. Vercesi, A. Watson, T. Wengler, P. Werner, S. Wheeler, F.J. Wickens, W. Wiedenmann, M. Wielers, H. Zobernig

THE ATLAS HIGH LEVEL TRIGGER GROUP [†]

presented by W. Wiedenmann
University of Wisconsin, Madison, Wisconsin

Abstract— **The ATLAS High Level Trigger's primary function of event selection will be accomplished with a Level-2 trigger farm and an Event Filter farm, both running software components developed in the ATLAS offline reconstruction framework. While this approach provides a unified software framework for event selection, it poses strict requirements on offline components critical for the Level-2 trigger. A Level-2 decision in ATLAS must typically be accomplished within 10 ms and with multiple event processing in concurrent threads. In order to address these constraints, prototypes have been developed that incorporate elements of the ATLAS Data Flow -, High Level Trigger -, and offline framework software. To realize a homogeneous software environment for offline components in the High Level Trigger, the Level-2 Steering Controller was developed. With electron/gamma- and muon-selection slices it has been shown that the required performance can be reached, if the offline components used are carefully designed and optimized for the application in the High Level Trigger.**

I. Introduction

THE Large Hadron Collider (LHC) currently under construction at CERN will produce pp-collisions with a center of mass energy of $\sqrt{s} = 14$ TeV at a design luminosity of 10^{34} cm^{-2}s^{-1}. With a bunch crossing rate of 40 MHz and about 23 interactions per bunch crossing it requires however highly selective trigger systems to reduce the expected 10^9 interactions per second to an acceptable rate of a few hundred Hz. ATLAS [1] is one of the two large general purpose experiments at the LHC and covers a widely diversified physics program [2], ranging from discovery physics to precision measurements of Standard Model parameters. ATLAS has an inner detector for precision tracking mounted inside a superconducting magnet with 2 T field strength. Outside the solenoid follow electromagnetic and hadronic calorimeters. Muon identification is achieved with a high precision muon spectrometer. The total number of readout channels is about 10^8.

Given the required large selectivity of the ATLAS trigger ($\simeq 10^{-7}$) and the rare nature of the most interesting physics signatures at the LHC collider, it is essential to understand the efficiencies at each step of the event selection process. Sharing a large number of software components across all platforms from the trigger event selection software to the offline physics analysis and reconstruction environment helps in achieving this goal and allows for a common development and run environment.

II. The ATLAS Trigger

The ATLAS trigger is based on three levels of online selection: Level-1, Level-2, and Event Filter (EF). The second and third level triggers, together known as the High Level Trigger (HLT) [3], are implemented on PCs running the Linux operating system.

The Level-1 trigger [4] is implemented in custom hardware and will reduce the initial event rate to about 75 kHz. The Level-1 decision is based on data from the calorimeters and the muon detectors. For accepted events small localized regions

[†] http://atlas.web.cern.ch/Atlas/GROUPS/DAQTRIG/HLT/AUTHORLISTS/nss2003.pdf

in rapidity η and azimuthal angle ϕ centered on the high p_T objects identified by the Level-1 trigger are determined. Each Region of Interest (RoI) contains the type and the thresholds passed of the associated high p_T candidate objects.

The Level-2 trigger's selection process is guided by the RoI information supplied by the Level-1 trigger and uses full granularity event data within a RoI from all detectors. In this way, only 2% of the full event data are needed for the decision process at Level-2, thus reducing dramatically the size of the network needed to serve the Level-2 trigger. The selection algorithms request data from the Read Out Buffers (ROB) for specific detectors in a Level-1 defined RoI for each processing step. The data are held in the ROBs until the Level-2 trigger accepts or rejects the event. The Level-2 event selection algorithms are controlled by the HLT selection framework and run inside the Level-2 Processing Units [5] [6] (L2PU) in concurrent worker-threads, each processing one event. The multi-threaded approach minimizes overheads from context-switching and avoids stalling the CPU when waiting for requested RoI data to arrive from the ROBs. Asynchronous services, like input of data and application monitoring, are executed in separate threads. This allows an efficient use of multi-CPU processors but requires also all software running in the L2PU to be thread-safe. The technical aspects of multi-threading are handled by the data flow software itself, including creation and deletion of threads and any locking mechanism that may be required. The Level-2 output rate is about 2 kHz with typical event decision times of 10 ms.

If an event is accepted by Level-2, the Event Builder collects all the event data fragments from the ROBs. The complete event is then made available to the EF for the final stage of trigger processing. Here, more complex algorithms provide a further rate reduction to about 200 Hz with typical event decision times of 1-2 s. While the Level-2 reconstructs localized regions, the baseline for the EF is a full offline-like event reconstruction guided by the *Level-2 Result*. It will also use more complete calibration, alignment and magnetic field data.

To achieve a fast rejection, the event processing in the HLT selection proceeds in steps for feature extraction and hypothesis decisions. At the end of each step the step results are checked against abstract physics signatures defined in trigger menus.

III. High Level Trigger Selection Software

The HLT selection framework [7] constitutes the run environment for the trigger algorithms. It is common to Level-2 and EF and is composed of four main components. The *Steering* schedules the *HLT Algorithms* corresponding to the input seed, so that all necessary data for a trigger decision are produced. Information about event specific quantities is exchanged via components of the *Event Data Model* (EDM). During event processing data are stored and accessed through a *Data Manager*. This allows to hide platform- and storage technology-specific details of event data access from the algorithms. The *HLT Algorithms* either reconstruct new event quantities or check trigger hypotheses with previously computed event features.

As the main purpose of the HLT selection software is event selection, it has to run efficiently and reliably in the online environment. In addition, critical selection components must be transferable to the offline environment for development and testing purposes. By providing a common code base for the online and the offline software, the HLT guarantees the consistency of trigger performance evaluations. It also provides a "physicist-friendly" environment for trigger algorithm development. In addition, studies have already shown [8] that great cost savings can be obtained with the proper global optimization of the trigger. Having a single common framework, where the different trigger levels can be cross-optimized, greatly facilitates these studies.

Since the EF provides an offline-like environment, the HLT selection software is naturally based on the ATLAS offline reconstruction and analysis environment ATHENA [9], which itself is based on the GAUDI [10] framework. This allows for the reuse of the storage manager, the EDM, the detector description and many algorithms, which are already developed by the offline community. Only the *Steering* framework and certain algorithms remain as HLT specific developments. In the case of the Level-2 trigger, a similar ansatz is more difficult due to the multi-threaded selection process and the more severe performance requirements. Even though Level-2 algorithms are specially developed to meet the tight timing requirements, they use the same EDM- and detector description objects present in the EF and offline. A transparent use of such components is possible and a common implementation of the HLT framework for both Level-2 and EF can be realized if the same interfaces are available at Level-2. This is provided by the *Level-2 Steering Controller* (SC).

IV. The Level-2 Steering Controller

The SC [11] is the software component that interfaces the L2PU, the data flow application which provides access to the event data stored in ROBs and the HLT event selection software. The purpose of the SC is threefold: to allow the L2PU to host and control the selection software; to allow the reuse of the same trigger algorithm steering software as in the EF; and to provide a mechanism for transmitting the results of Level-1 and Level-2 processing between the data acquisition system and the event selection software. All data flow applications follow for control a state model implemented in form of *Finite State Machines* (FSM). The key to the SC design is to place this interface where the functionality of the data flow and event selection frameworks can be cleanly separated. The location chosen is the FSM of the L2PU.

The SC provides the means for forwarding state changes from the data flow software (Fig. 1) to the event selection software. An important aspect of this approach is that the Level-2 event data access is managed entirely by the data flow. The SC then does not need to interact directly with the data input threads or other data flow components.

Fig. 2 illustrates the sequence of interactions of the SC with the L2PU and the event selection software. The figure shows

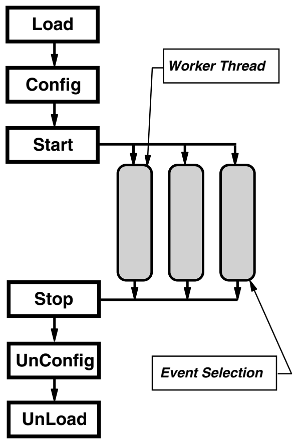


Fig. 1. The L2PU Finite State Machine. All algorithms and services are created at initialization time in the *Configure* step. The Level-2 selection code is executed in parallel in multiple worker-threads.

three states: *Configure*, *Start*, and *Stop*. During the *Configure* phase, configuration and conditions data are obtained from external databases via an HLT-online interface. These data are then used to configure the selection software and all associated components.

After a *Start* the SC receives an 'execute event' directive with a *Level-1 Result* as argument. The result of event processing is directly returned as *Level-2 Result* to the L2PU. A *Stop* command terminates algorithm execution and produces run summary information.

Since the trigger event selection software is being developed in the ATLAS offline framework, which is itself based on the GAUDI framework, the SC also has been designed to re-use the framework interfaces defined in GAUDI. In this way, there is a unified environment for event reconstruction and selection from the second level trigger to the offline analysis.

Since the event selection software executes in multiple worker threads, the SC must provide a thread-safe environment. At the same time, and in order to provide an easy-to-use framework for offline developers, the SC must hide all technical details of thread handling and locks. Thread safety has been implemented in the SC by using GAUDI's name-based object and service bookkeeping system. Copies of components that need to be thread-safe are created for each worker thread with different labels. The labels incorporate the thread-ID of the worker thread, as obtained from the data flow software. The number of threads created by the data flow software is transferred to the SC, which transparently creates the number of required copies. In this scheme, the same configuration can be used in the offline and in the Level-2 environments; the thread-ID collapses to null in the offline environment, as it is not needed there.

In contrast to the EF and offline environment, all Level-2 algorithm- and service instances need to be created and initialized in the configuration state of the L2PU FSM, since only in this phase a L2PU has access to external databases. Later in the event loop, only access to information stored locally on the processor is possible. The *Configure* step is still executed in a single thread and only later after the transition to the *Start* state of the FSM the thread specific copies of the algorithms and services are attached to the worker-threads.

The implementation of the SC consists of three components:

- An interface class, which forwards the L2PU state changes to the algorithm execution framework. Different interface implementations, e.g. for data flow testing without algorithms, can be specified in the L2PU configuration and are loadable as shared libraries. This implementation also helps to minimize cross-dependencies in the respective data flow and offline repositories.
- The multi-threaded algorithm execution environment. The necessary changes for multi-threading support have been incorporated in the GAUDI base libraries and are available with the recent official releases of the GAUDI framework.
- A modified ATHENA event loop handler. Contrary to offline, the event loop is controlled by the data flow software. The modified event loop handler makes the *Level-1 Result* available to the HLT selection software, executes the algorithms for a given event and forwards the Level-2 decision to the L2PU interface class.

The implementation is complemented by special utility services, which connect e.g. the GAUDI framework messaging to the corresponding data flow implementations.

V. Software Development Model

HLT software developers follow a typical edit, compile and run cycle in the ATHENA offline environment, when creating new software components. For running an application, the ATHENA main program together with a job configuration file would be used: `athena <job-configuration>`.

Since the same interfaces are available in the EF and the L2PU environment the code developed in the offline environment can be directly downloaded in binary form to the processors. For Level-2 the developer has to follow however a set of simple coding rules [12] to produce thread safe code

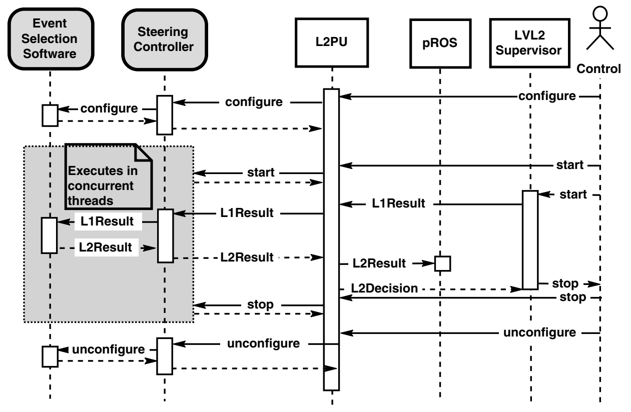


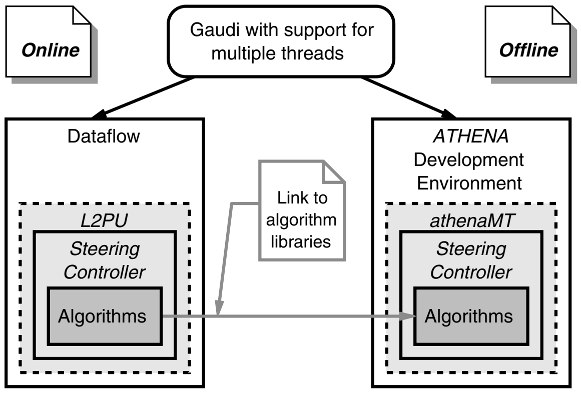
Fig. 2. The sequence of interactions of the SC with the L2PU and the Event Selection Software. Three states are shown: *Configure*, *Start*, and *Stop*. The gray area shows the interactions which happen in multiple Worker Threads. The *pROS* is the component that forwards the *Level-2 Result* to the EF.

and to make its algorithms or services compatible with the automatic creation of multiple copies in the L2PU. Furthermore it should be not necessary to use *Locks* or *Mutexes* to adapt the code to the multi-threaded environment. To meet the timing requirements for the second level trigger, the number and type of available services is restricted to the necessary minimum, which is a subset of the services available in the EF and in offline. In this way it should be always possible to move a Level-2 component to the EF and offline environment. The other direction, however, is only possible if the software component finds all its necessary resources in the restricted L2PU environment.

The data flow software can be configured to run either as single or multi node system. The single node system starts a ROS emulator, a Level-2 supervisor, and a L2PU in the same processing node, while the multi-node system distributes these applications over various nodes. The setup of a complex data flow system for application testing is in both cases a nontrivial task and most HLT developers lack also the necessary hardware resources. A multi-threaded test application called *athenaMT* was therefore created, which presents internally the same run environment as a L2PU, but can be started as simple as the normal ATHENA main program: `athenaMT <number of worker-threads> <job-configuration>`. *athenaMT* uses the SC and most of the data flow components that are also used in a standard L2PU. It differs, however, in the supervision aspect of the L2PU and in the way detector raw data are made available to the processing unit. The application can be used on single- or multi-CPU machines. HLT developers need not to be familiar with detailed technical aspects of the data flow software and are also shielded from changes in the data flow part of the software. They can concentrate exclusively on the HLT software and are able to perform a large variety of useful tests, from thread safety to performance measurements, in a realistic L2PU environment. Fig. 3 shows the relation between an online data flow setup and an offline development environment for Level-2 software. In this way the development effort for HLT and data flow can be widely parallelized. It is clear, however, that the final certification of the HLT software has to be done on a large distributed system.

VI. Performance Measurements

After integrating the SC with the data flow software, both performance and robustness tests were carried out on a dual-processor 1.533 GHz AMD Athlon machine. The SC ran for over 50 hours with three threads with an early version of the selection software prototype. The prototype ran successfully on both single-CPU and double-CPU machines, showing it to be thread safe. A direct measurement of the SC overhead yielded 13 μs per event. The overhead was estimated by comparing the number of events per second a L2PU can handle when running without or with the SC and executing a simple algorithm (first and second row in Tab. I). Tab. I shows also the obtained rate, when in addition the *Level-1 Result* is transfered to the *Data Manager*. The quoted overhead includes all GAUDI framework steps to schedule and execute algorithms and to execute the used base services. The measurements were based on runs of at least 100,000 events and almost perfect scaling of the overheads was observed with an algorithm containing a CPU burning loop from 0 to 8 ms.



Online
Gaudi with support for multiple threads
Offline
Dataflow
L2PU
Steering Controller
Algorithms
Link to algorithm libraries
ATHENA Development Environment
athenaMT
Steering Controller
Algorithms

Fig. 3. A typical development and prototype setup. GAUDI with support for multiple threads is the basic algorithm execution environment. It is shared by the online and the offline environments. Algorithms are developed in the offline environment and tested with the L2PU emulator *athenaMT*. The same binary libraries containing the event selection algorithms are then used by the real L2PU running in a data flow setup.

More tests with increasing complexity were performed on a three node data flow system build with a dual processor Intel XEON 2.2 GHz machine hosting the L2PU, a dual-processor 1.533 GHz AMD Athlon machine hosting the ROS emulator and a single processor Intel Pentium 4 machine running the Level-2 Supervisor. The dual processor machines were connected via a Gigabit Ethernet network. All machines were running the Redhat 7.3 Linux operating system.

In sequence, the offline *Data Manager* StoreGate [13] the HLT *Steering* and a fast inner tracker feature extraction algorithm [14] were included in the setup. The algorithm used an early version of the *Event Data Model* and the raw data conversion process from byte stream format to EDM classes. More complex tests were recently done with complete e/γ and μ selection slices. They contained the complete HLT steering framework with decoding of the *Level-1 Result*, scheduling of the feature extraction algorithms, and sending the results of the Level-2 processing to the EF. The *Data Manager*, the *Event Data Model*, the detector description for the calorimeters and the muon system and services for conversion from raw data byte stream format to high level data containers were used from offline. The feature extraction algorithms were specially developed for Level-2 but used the above mentioned offline services. In the case of the e/γ slice, over 95 % of the events were processed within 5 ms for a sample of di-jet events at low luminosity and a RoI size of $\Delta\eta \times \Delta\phi = 0.3 \times 0.3$. Fig. 4 shows similar results for the μ slice.

Prototype configuration	Measured rate	Overhead/event
L2PU	21.7 kHz	46 μs
L2PU+SC	17.0 kHz	59 μs
L2PU+SC+Data Manager	15.3 kHz	65 μs

TABLE I

RATES ON A DUAL-PROCESSOR 1.533 GHZ AMD ATHLON MACHINE.

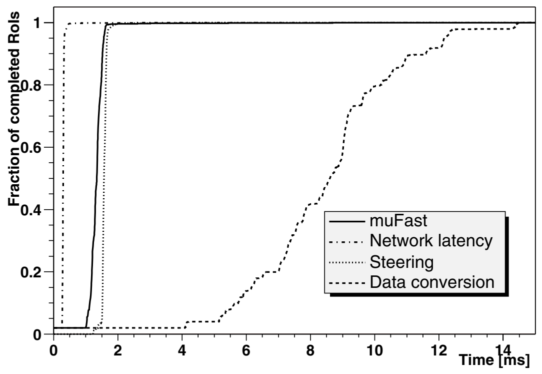


Fig. 4. The main contributions to the latency shown as curves of integrals for the μ selection slice. The slice used offline software components for data conversion, detector description, data manager and steering. The contributions in order of decreasing importance are data preparation and conversion, framework overheads, algorithmic processing and network access time. The curves show e.g. that in this setup algorithmic processing is terminated for 95% of the events in less than 2 ms. The data sample consisted of 200 GeV muons at high luminosity. The muon background from the ATLAS cavern was boosted by a factor of two, so that the results give a conservative estimate of the processing times.

VII. EXPERIENCES

During the tests it was observed that the event throughput in a L2PU didn't scale in the expected way with the number of worker-threads. This was due to the use of a common memory pool for container objects in the default memory allocation scheme of the *Standard Template Library* (STL). The event processing model of Level-2 favors a scheme where every thread allocates its own memory pool. Such an allocation scheme is available in the STL. After carefully optimizing the code with it, the expected scaling behavior was observed. Such an optimization poses limitations on the offline components used and their external dependencies. For instance, external utility libraries may not be available with this allocation scheme. To avoid frequent memory allocation during event processing, all large containers that hold detector data were designed to allocate memory only during initialization time and to reset their data for each new event.

The reconstruction of full events in offline favors a processing model where utility services and external data are retrieved on demand. In the case of Level-2 this model cannot be applied since during event processing only locally stored meta-data are accessible. All required data need to be known at configuration time and need to be prefetched by the processor. Furthermore, the creation of large configuration objects on demand may lead to time-outs during event processing. These restrictions required a redesign and alternative initialization methods for some offline components, especially for detector description and raw data conversion.

The use of offline components in multiple worker threads and the requirement to avoid *Locks* in the event selection code limits certain design and implementation choices, e.g. the use of *Singletons*. These restrictions were communicated to the HLT developers at an early design stage.

VIII. CONCLUSIONS

The presented implementation of the SC for the ATLAS Level-2 trigger enables the reuse of offline software components throughout the ATLAS High Level Triggers. It realizes a homogeneous software and development environment from the Level-2 trigger to offline. Realistic prototypes have shown that the required performances can be reached if the offline components are carefully optimized and designed for reuse in the triggers. This may limit architectural, design and implementation choices that are otherwise available in a pure offline environment. An understanding of these restrictions is necessary for all contributing developers.

ACKNOWLEDGMENT

The authors would like to acknowledge the help and support of the ATLAS Data Acquisition Group and the ATLAS Offline and Detector software groups.

REFERENCES

[1] ATLAS Collaboration, ATLAS: Technical proposal for a general-purpose pp experiment at the Large Hadron Collider at CERN, CERN/LHCC/94-43 (1994)
[2] ATLAS Collaboration, ATLAS Detector and Physics Performance Technical Design Report, CERN/LHCC/99-014 and 99-015 (1999)
[3] ATLAS Collaboration, ATLAS High-Level Triggers, DAQ and DCS Technical Proposal, CERN/LHCC/2000-017 (2000)
[4] ATLAS Collaboration, ATLAS Level-1 Trigger Technical Design Report, CERN/LHCC/98-014 (1998)
[5] A. Bogaerts and F. Wickens, LVL2 Processing Unit Application Design, EDMS Note, ATL-DH-ES-0009 (2001)
[6] A. dos Anjos et al., The Second Level Trigger of the ATLAS Experiment at CERN's LHC, Nuclear Science Symposium, Portland, Oregon, USA, October 19-25,2003
[7] M. Elsing et al., Analysis and Conceptual Design of the HLT Selection Software, ATLAS Internal Note, ATL-DAQ-2002-013 (2002)
[8] J.T.M. Baines et al., First study of the LVL2-EF boundary in the high-pT e/gamma High-Level Trigger, ATLAS Internal Note, ATL-DAQ-2000-045 (2000)
[9] Athena: User Guide and Tutorial. Available: http://atlas.web.cern.ch/Atlas/GROUPS/SOFTWARE/OO/architecture/General/Tech.Doc/Manual/2.0.0-DRAFT/AthenaUserGuide.pdf
[10] Gaudi Project. Available: http://cern.ch/proj-gaudi/
[11] S. González et al., Use of Gaudi in the LVL2 Trigger: The Steering Controller, EDMS Note, ATL-DH-EN-0001 (2002)
[12] A. dos Anjos et al., Guidelines for offline preparation and testing of LVL2 code. Available: http://www.cern.ch/steve.armstrong/algorithms/guidelines
[13] P. Calafiura et al., StoreGate: A Data Model for the ATLAS Software Architecture, Computing in High Energy and Nuclear Physics Conference, Beijing, 2001. Available: http://atlas.web.cern.ch/Atlas/GROUPS/SOFTWARE/OO/architecture/EventDataModel/Tech.Doc/StoreGate/Chep01.pdf
[14] A. Bogaerts et al., Initial LVL2 Tests with the Si Tree Algorithm, EDMS Note, ATL-DH-TN-0001 (2003)