ATLAS High Level Trigger within the multi-threaded software framework AthenaMT

Rafał Bielski, on behalf of the ATLAS Collaboration
CERN, 1211 Geneva 23, Switzerland
E-mail: `rafal.bielski@cern.ch`

Abstract. Athena is the software framework used in the ATLAS experiment throughout the data processing path, from the software trigger system through offline event reconstruction to physics analysis. The shift from high-power single-core CPUs to multi-core systems in the computing market means that the throughput capabilities of the framework have become limited by the available memory per process. For Run 2 of the Large Hadron Collider (LHC), ATLAS has exploited a multi-process forking approach with the copy-on-write mechanism to reduce the memory use. To better match the increasing CPU core count and, therefore, the decreasing available memory per core, a multi-threaded framework, AthenaMT, has been designed and is now being implemented. The ATLAS High Level Trigger (HLT) system has been remodelled to fit the new framework and to rely on common solutions between online and offline software to a greater extent than in Run 2. We present the implementation of the new HLT system within the AthenaMT framework, which is going to be used in ATLAS data-taking during Run 3 (2021 onwards) of the LHC. We also report on interfacing the new framework to the current ATLAS Trigger and Data Acquisition (TDAQ) system, which aims to bring increased flexibility whilst needing minimal modifications to the current system. In addition, we show some details of architectural choices which were made to run the HLT selection inside the ATLAS online data-flow, such as the handling of the event loop, returning of the trigger decision and handling of errors.

ATL-DAQ-PROC-2019-004
13 May 2019

1. AthenaMT motivation and design

Athena [1] is a software framework used in the ATLAS experiment [2] at all stages of event data processing path, from simulation and trigger, through event reconstruction to physics analysis. It is based on an inter-experiment framework Gaudi [3] designed in the early 2000s. At the time of the initial design, there was no strong motivation for concurrent event processing in experiments like ATLAS and the architectural choices made for Gaudi and Athena assumed only a single-process and single-thread mode of operation. However, over the last decade, the computing market has transitioned towards many-core processors and the unit price of memory has plateaued. The combination of these two trends presented in Figures 1 and 2 implies that high throughput of data processing in software requires concurrent execution and memory sharing.

In Run 2 of the Large Hadron Collider (LHC) [6], the ATLAS experiment software has already suffered from suboptimal use of resources leading to the throughput of simulation and

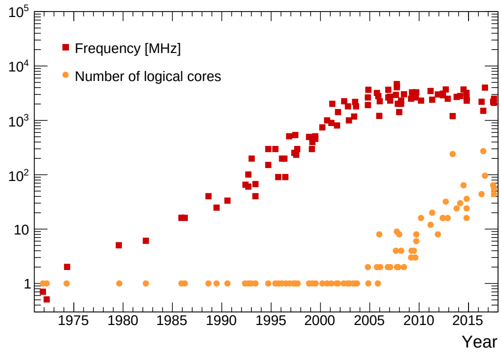


Figure 1. CPU frequency and logical core count trends between 1972 and 2018, based on data collected by K. Rupp [4]. Since around 2005, higher processing throughput is achieved by increasing the number of cores rather than their clock frequency.

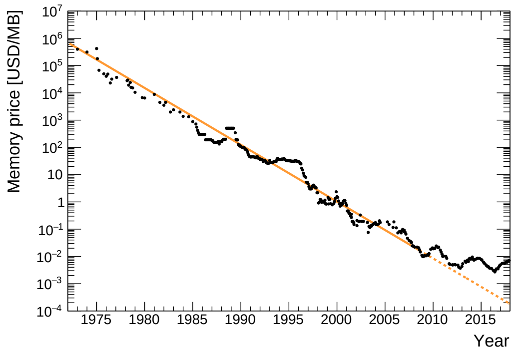


Figure 2. Memory price trends between 1972 and 2018, based on data collected by J.C. McCallum [5]. The orange line represents a fit to data from the years 1972–2009. A deviation from the logarithmic price decrease is observed since around 2012.

reconstruction workflows being limited by the available memory. A multi-process approach based on forking after initialisation (AthenaMP) was adopted as an intermediate solution to this problem. A reduction of the overall memory requirements was possible thanks to the copy-on-write mechanism. A similar approach has been also used in the ATLAS Trigger system. At the same time, work has been started towards redesigning the core framework of both Gaudi and Athena to support multi-threaded execution in a native, efficient and user-friendly manner. The effort is aimed at production-ready software available for Run 3 of the LHC, starting in spring 2021.

The multi-threaded Athena framework, AthenaMT, is based on Gaudi Hive [7] which provides a concurrent task scheduler based on Intel Thread Building Blocks (TBB) [8]. The framework supports both inter-event and intra-event parallelism by design and defines the algorithm execution order based on the data dependencies declared explicitly as ReadHandles and WriteHandles. When all input dependencies of an algorithm are satisfied, it is pushed into a TBB queue which takes care of its execution. The combination of inter-event and intra-event parallelism is depicted in Figure 3, where four threads are shown to execute concurrently algorithms which process data from either the same or different events. More efficient memory usage can be achieved thanks to a large share of event-independent data.

2. High Level Trigger in AthenaMT

Taking the advantage of major changes in the Athena core software, a decision has been made to considerably redesign the ATLAS High Level Trigger (HLT) framework which is part of Athena. In Run 2, the HLT software used a dedicated top-level algorithm implementing a dedicated sub-algorithm scheduling procedure. All trigger algorithms implemented an HLT-specific interface and any offline reconstruction algorithm required a wrapper to be used in the HLT. The Run-3 HLT framework will take full advantage of the Gaudi scheduler and no HLT-specific interfaces will be required.

The main functional requirements of the ATLAS HLT have been incorporated into the design of the Gaudi Hive framework. These include the processing of partial event data (regional reconstruction) and early termination of an execution path if the event is failing the trigger selection. The regional reconstruction is achieved with Event Views which allow an algorithm to be executed in the same way on either partial or full event data. The Event Views are

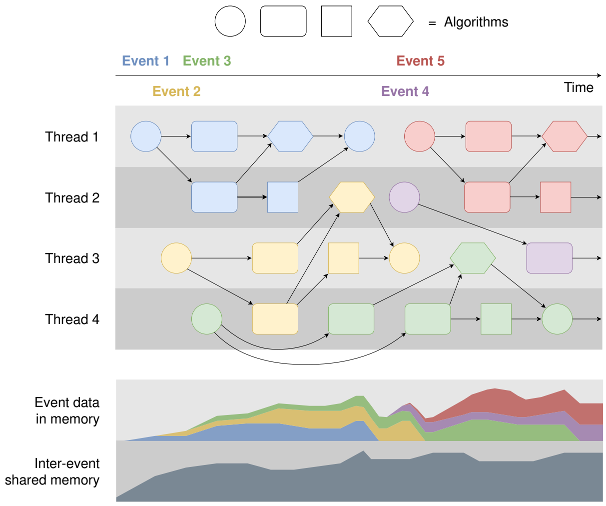


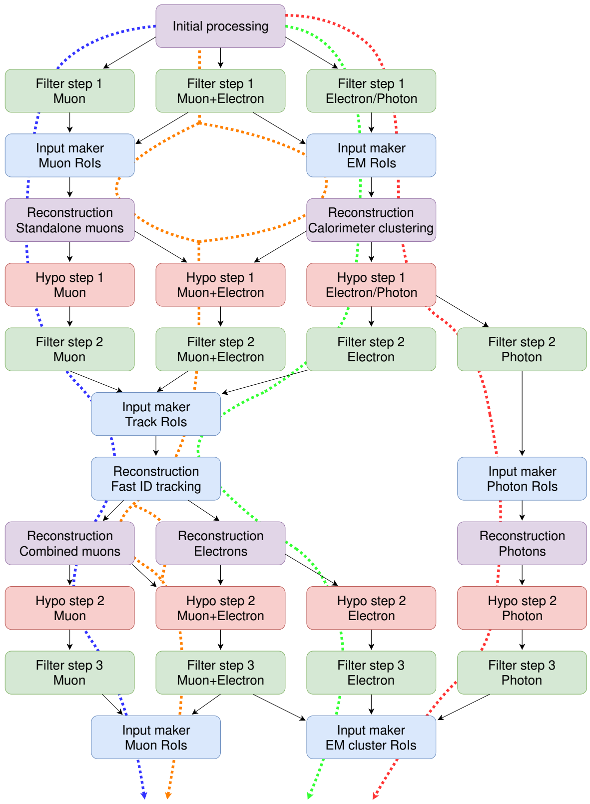
Figure 3. Schematic flowchart describing concurrent event processing in AthenaMT. Each shape corresponds to a type of algorithm and each colour corresponds to data associated with one event. Data flow is represented by arrows.

prepared by Input Maker algorithms scheduled before the reconstruction algorithms. Early termination is achieved with Filter algorithms. The scheduling of all algorithms in the HLT is assisted by configuration-time Control Flow which defines sequences of Filter → Input Maker → Reconstruction → Hypothesis algorithms. If the hypothesis testing fails in one sequence, the filter step of the next sequence ensures the early termination of the given path. The Control Flow creates a diagram including all possible execution paths at configuration time. The diagram is built from a list of all physics selections configured to be executed and does not change during runtime. However, each trigger "chain" corresponding to one path through the diagram can be individually disabled during runtime or executed only on a fraction of events. An example fragment of the Control Flow diagram is presented in Figure 4.

3. HLT within the TDAQ infrastructure

The ATLAS Trigger and Data Acquisition (TDAQ) infrastructure, presented in Figure 5, consists of detector readout electronics, a hardware-based Level-1 (L1) trigger, an HLT computing farm and data flow systems. Data from muon detectors and calorimeters are analysed by the L1 system at the LHC bunch crossing rate of 40 MHz in order to select potentially interesting events and limit the downstream processing rate to a maximum of 100 kHz. In addition to the accept decision, L1 produces also Region-of-Interest (RoI) information which seeds the regional reconstruction of events in the HLT and will serve as the input to Event View creation in the new software. The HLT computing farm consists of around 40 000 physical cores executing Athena to enhance the trigger decision and to reduce the output rate to around 1.5 kHz which can be written to permanent storage.

The HLT software can be used both in online data processing and in offline reprocessing or simulation. Although the event processing sequence is the same in both cases, the online processing requires a dedicated layer of communication to the TDAQ system. This layer implements the TDAQ interfaces for input/output handling, online-specific error handling procedures and an additional time-out watcher thread which is not needed offline. Each node in the HLT farm runs a set of applications presented in Figure 6. The main application responsible for event processing is the HLT Multi-Process Processing Unit (HLTMPPU) which loads an instance of Athena. After initialisation, the process is forked to achieve memory savings in a multi-process execution. The mother process only monitors the children and does not participate in event processing. Each child processes events by executing Athena methods and transferring the inputs and outputs to/from the Data Collection Manager (DCM) which communicates with the data flow infrastructure.



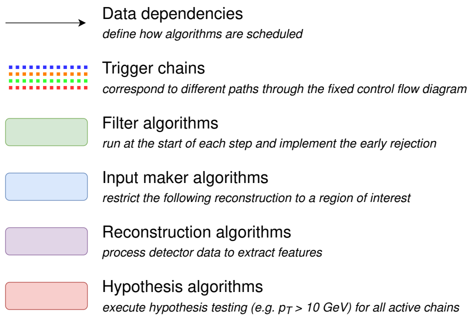
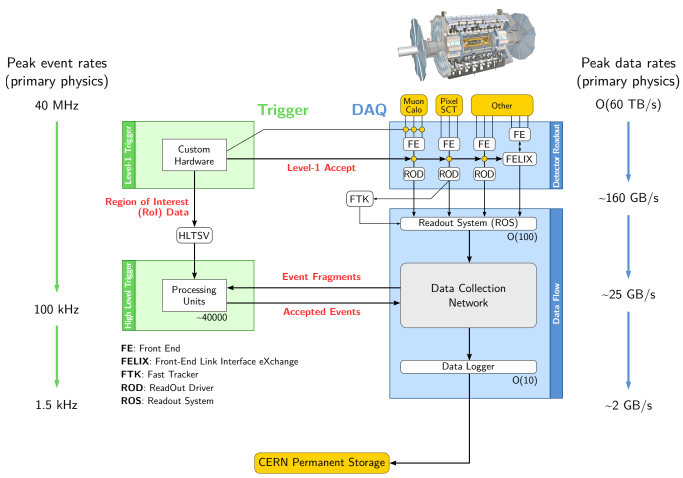


Figure 4. Example fragment of HLT Control Flow diagram presenting several execution paths through sequences of Filter → Input Maker → Reconstruction → Hypothesis algorithms.



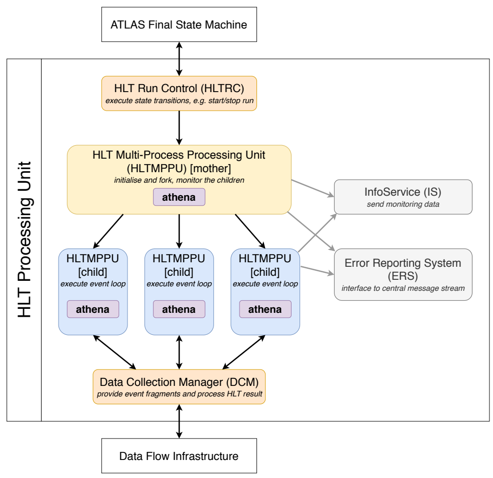


Figure 5. Functional diagram of the ATLAS TDAQ system showing typical peak rates and bandwidths through each component in Run 2 [9].

Figure 6. Structure of the HLT processing unit applications and the communication between them.

The design of the TDAQ infrastructure will not change in Run 3 and the HLT processing unit will consist of the same applications as in Run 2. However, the data flow within HLTMPPU will change considerably to make use of the new AthenaMT HLT software framework. The multi-process approach will remain in use, however the Athena instance within each HLTMPPU child will be able to process multiple events concurrently using multiple threads. This design provides flexibility for optimising the performance of the system.

In Run 2, the event processing loop was steered by HLTMPPU and events were pushed into Athena sequentially. In Run 3, Athena will manage the event processing loop and will actively pull events from DCM through HLTMPPU when processing slots are available. The new interface between Athena, HLTMPPU and DCM is presented in Figure 7.

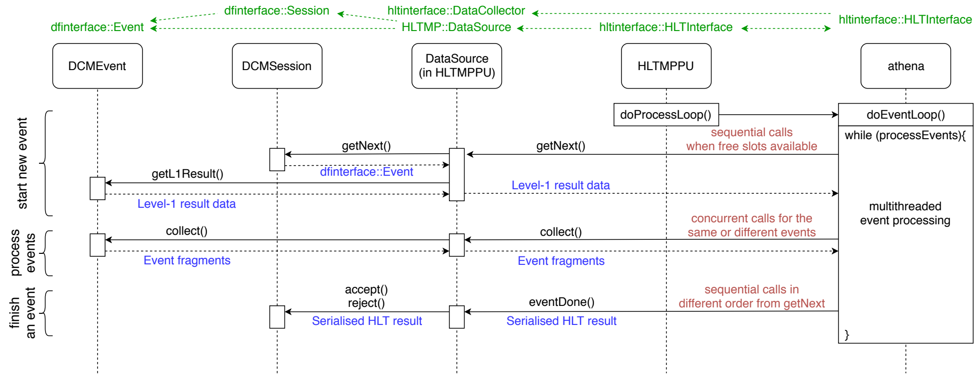


Figure 7. Simplified call diagram presenting the Run-3 interfaces steering the event execution in an HLT processing unit.

4. Summary and outlook

Recent trends in the computing market and software design motivated a large redesign of the Athena framework to fundamentally support multi-threaded event execution. Along this adaptation, the ATLAS High Level Trigger is also being re-implemented to achieve closer integration with the multi-threaded core framework and with the offline processing software. The upgrade of both offline and online Athena software is aimed at Run 3 of the LHC starting in 2021. The fundamental steps of the design and implementation of the core framework as well as the interfaces to the TDAQ infrastructure have already been achieved and the adaptation of the event processing algorithms and the supporting and monitoring infrastructure is ongoing.

In addition to the full implementation, the evaluation and optimisation of the performance of the new system is also required before the start of Run 3. This includes the determination of the optimal number of forks, threads, and concurrently processed events in the online system and large-scale tests with the full TDAQ infrastructure. New classes of software problems are anticipated with the multi-threaded execution and measures to minimise their impact and analyse them have to be defined.

References

[1] ATLAS Collaboration 2019 Athena [software] Release 22.0.1 `https://doi.org/10.5281/zenodo.2641997`

[2] ATLAS Collaboration 2008 The ATLAS Experiment at the CERN Large Hadron Collider *JINST* **3** S08003

[3] Barrand G and others 2001 GAUDI – A software architecture and framework for building HEP data processing applications *Comput. Phys. Commun.* **140** 45–55. See also Gaudi [software] Release v31r0 `https://gitlab.cern.ch/gaudi/Gaudi/tags/v31r0`

[4] Rupp K 2018 Microprocessor Trend Data `https://github.com/karlrupp/microprocessor-trend-data` commit 7bbd582ba1376015f6cf24498f46db62811a2919

[5] McCallum J C 2018 Memory Prices (1957-2018) `http://jcmit.net/memoryprice.htm` [accessed 2019-02-12]

[6] Evans L and Bryant P 2008 LHC Machine *JINST* **3** S08001

[7] Clemencic M, Funke D, Hegner B, Mato P, Piparo D and Shapoval I 2015 Gaudi components for concurrency: Concurrency for existing and future experiments *J. Phys. Conf. Ser.* **608** 012021

[8] Reinders J 2007 *Intel Threading Building Blocks: Outfitting C++ for Multi-core Processor Parallelism* (O'Reilly Media). See also TBB [software] Release 2019_U1 `https://github.com/intel/tbb/tree/2019_U1`

[9] ATLAS Collaboration 2019 Public results page `https://twiki.cern.ch/twiki/bin/view/AtlasPublic/ApprovedPlotsDAQ` [accessed 2019-02-12]