EPJ Web of Conferences **214**, 01046 (2019) https://doi.org/10.1051/epjconf/201921401046
CHEP 2018

Implementation of the ATLAS trigger within the multi-threaded AthenaMT framework

Stewart Martin–Haugh[1,*], on behalf of the ATLAS collaboration[**]

[1]Particle Physics Department
STFC Rutherford Appleton Laboratory
Harwell Campus
Didcot
OX11 0QX

Abstract.
We present an implementation of the ATLAS High Level Trigger (HLT) that provides parallel execution of trigger algorithms within the ATLAS multi-threaded software framework, AthenaMT. This development will enable the HLT to meet future challenges from the evolution of computing hardware and upgrades of the Large Hadron Collider (LHC) and ATLAS Detector. During the LHC data-taking period starting in 2021, luminosity will reach up to three times the original design value. In the following data-taking period (2026) upgrades to the ATLAS trigger architecture will increase the HLT input rate by a factor of 4-10, while the luminosity will increase by a further factor of 2-3.
AthenaMT provides a uniform interface for offline and trigger algorithms, facilitating the use of offline code in the HLT. Trigger-specific optimizations provided by the framework include early event rejection and reconstruction within restricted geometrical regions. We report on the current status, including experience of migrating trigger selections to this new framework,and present the next steps towards a full implementation of the redesigned ATLAS trigger.

1 Introduction

Individual collision events in a particle collider can be treated as independent of the previous. In practice, there are time-dependent variations in detector response coming from e.g. out-of-time pile-up and bunch train position, but events can still be simulated or reconstructed independently from one another (taking appropriate account of detector conditions). LHC computing therefore parallelises naturally by event, and indeed historically LHC event processing frameworks have processed single events serially on a single CPU core, with events distributed between independent processing node. This allowed for the success of grid computing, with event processing streamed to different grid sites around the world. Unfortunately, processing one event on each CPU core does not match current trends in computer architecture: the number of CPU cores available in a standard compute node has increased, but the amount of memory per core is increasing at a lower rate. Reducing the memory requirement is only possible by sharing memory between CPU cores. Sharing event processing between

*e-mail: stewart.martin-haugh@stfc.ac.uk

multiple threads ("multi-threading) allows more memory to be shared between cores, reducing the overall memory footprint per core. Additionally, co-processors such as Graphical Processing Units (GPUs) and Field Programmable Gate Arrays (FPGAs) can be used most effectively for asynchronously offloading compute-intensive tasks, freeing up CPU cores for work better suited to the CPU. Reducing the memory usage per core and effectively using co-processors is important for future LHC running conditions in Run 3 (2021-2023) and Run 4 (2026-2029). The future running conditions and upgrades to the ATLAS [1] detector are detailed in [2] and [3], while the adaptation of existing software to a multi-threaded environment requires changes at the framework level as detailed in the next section.

2 The ATLAS Software Framework

2.1 ATLAS Software in Run 2

The ATLAS software framework Athena [4] is mainly written in C++, with a configuration layer written in Python. The underlying Gaudi framework [5] is shared with the LHCb experiment. Gaudi defines the basic classes used for event processing, and also provides the scheduler for optimal algorithm execution. In Run 2, the Gaudi scheduler was used by offline reconstruction, but the HLT used a custom layer known as "Trigger Steering". This was needed to implement additional HLT functionality: data flow, control flow and regional reconstruction to minimise processing time. Data flow refers to the configuration of each sequence of trigger algorithms with explicit input and output data, to remove duplicate algorithms and minimise processing time. Control flow refers to the ability to flag each event for acceptance/rejection, and to apply the flag for each algorithm. Finally, regional reconstruction refers to the ability to restrict event processing to a smaller geometrical Region of Interest (RoI). The Trigger Steering layer fulfilled these requirements successfully during Run 1 and Run 2 but with a significant development and maintenance overhead.

2.2 ATLAS Software for Run 3

During Run 2 a design effort for a new software framework, suitable for Run 3 and beyond, was undertaken. The key requirements were:

- to reduce maintenance overhead.
- to make effective use of hardware.

As already discussed, effective use of modern multi-core machines requires multi-threading. To reduce the maintenance overhead, a common multi-threaded framework, AthenaMT [6] was proposed. From the beginning, the framework was designed to meet offline and trigger requirements, eliminating the need for a custom trigger-specific layer. Data and control flow, as well as regional reconstruction were designed to be part of the scheduler.

Data flow is expressed via data dependencies, in the form of ReadHandles and WriteHandles. An algorithm takes one or more inputs ReadHandles (either from a previous algorithm or directly from the initial detector data) and performs a transformation to create one or more output objects, published via WriteHandles. This information is known before the execution of the first event, and is used by the Run 3 Gaudi scheduler [7] to generate a directed acyclic graph[1] of dependencies controlling the order in which algorithms run for each event. The graph is fixed during initialisation and is then the same for all processed events. Algorithms

[1] In practice this means that graphs with problems such as circular dependencies cannot be constructed.

with no common dependencies (e.g. tracking and calorimeter reconstruction) can run in parallel. A simple example is shown in figure 1 for an electron algorithm with independent tracking and calorimeter reconstruction.

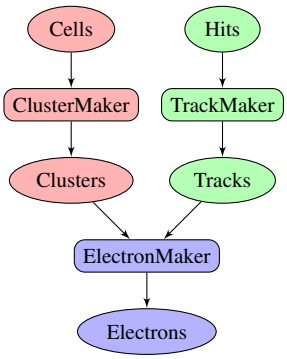

Figure 1. Data dependencies for electron reconstruction.

Control flow is a set of conditions (AND/OR) which allow the scheduler to avoid unnecessary processing. For instance, if an event contains only low momentum jets, subsequent algorithms (e.g. jet substructure calculations) can be skipped. An example for jet reconstruction is shown in figure 2. Note that control flow is also applied if a complete selection is disabled during the run - in this case the selection remains in the graph but is always rejected.

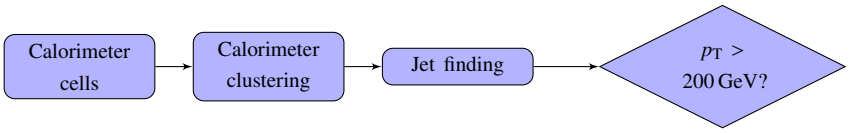

Figure 2. A simple jet reconstruction chain. The mandatory reconstruction steps are shown as rectangles, while the transverse momentum hypothesis is shown as a diamond.

Regional reconstruction is expressed via EventViews, which confine an algorithm to a given geometric region. This is designed to be entirely transparent to the algorithm, which simply requests data via ReadHandles, and receives only the data in the given region.

2.3 Menu and configuration

During Run 2, more than 2000 unique selections (e.g. one electron above 20 GeV, one electron and one muon, one electron and missing energy) were used. Each unique selection is known as a chain, and the set of chains is known as a menu. The configuration for each selection was generated by a large (several hundreds of thousands of lines) Python package known as TriggerMenu. It is important to know exactly which menu is used both for data-taking and production of simulated samples, so the configuration is stored in a database [8]. This configuration system also allows for chains to be enabled or disabled during the run, allowing e.g. for more expensive selections at lower luminosity. As part of the software upgrade for Run 3,

both the actual configuration and its storage in a database is currently being redesigned. The key requirements are

- Reduce the complexity of the configuration and ease maintenance/development overhead
- Improve the performance of the configuration serialisation/deserialisation

The reduced complexity will come from a redesign of the configuration (both in the trigger and across Athena as a whole), while the improved serialisation/deserialisation performance will come from storing the configuration as a blob rather than in relational format.

2.4 Integration with the online infrastructure

The AthenaMT software framework must interact with the online trigger and data-acquisition infrastructure, as well as new hardware triggers installed for Run 3. During Run 2, each node in the HLT forked multiple sub-processes thus sharing memory between them via the copy-on-write mechanism [9]. In Run 3, each forked sub-process will additionally run with multiple threads, and potentially process multiple events across these threads. This has a few implications:

- Tuning of the number of forks, threads and concurrent events will be necessary to ensure maximum performance.
- If a sub-process crashes or takes too long to process, all concurrently processed events must be saved for offline reprocessing.

Due to rolling replacements of hardware, there are several different generations of machine in operation in the ATLAS HLT farm. In principle the different parameters could be tuned according to machine generation.

A simplified example of three concurrently processed events, each taking more or less time to process, is shown in figure 3. The job crashes during the processing of events #4, #5 and #6, so the processing must be stopped for all three event slots, and the events flagged for later reprocessing.

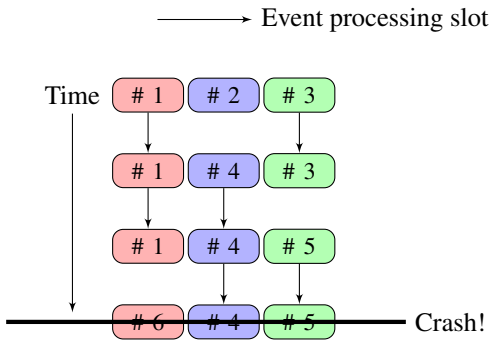

Figure 3. Three events processed in different event slots on the same node, with a crash in an unknown event slot.

2.5 Outlook

The project to upgrade the ATLAS High Level Trigger software is well advanced, with core functionality already in place. Current work is focused on migration and implementation of algorithmic code for event reconstruction, and the design and implementation of the trigger menu. The upgrade effort is on track to make better use of modern hardware, and improve the physics reach of the ATLAS experiment against the additional challenges of Run 3 and Run 4.

References

[1] ATLAS Collaboration, Journal of Instrumentation **3**, S08003 (2008)
[2] ATLAS Collaboration, Tech. Rep. CERN-LHCC-2013-018. ATLAS-TDR-023 (2013), https://cds.cern.ch/record/1602235
[3] ATLAS Collaboration, Tech. Rep. CERN-LHCC-2017-020. ATLAS-TDR-029 (2017), https://cds.cern.ch/record/2285584
[4] ATLAS Collaboration, European Physical Journal C **70**, 823 (2010), arXiv:1005.4568 [hep-ex]
[5] G. Barrand et al, Comput. Phys. Commun. **140**, 45 (2001)
[6] C. Leggett et al, Journal of Physics: Conference Series **898**, 042009 (2017)
[7] I. Shapoval et al, IEEE Nuclear Science Symposium and Medical Imaging Conference (NSS/MIC) (2015)
[8] C. Chavez et al, Journal of Physics: Conference Series **664**, 082030 (2015)
[9] S. Binet et al, Journal of Physics: Conference Series **219** (2010)