Triggering events with GPUs at ATLAS

S Kama[1], J Augusto Soares[2,6], J Baines[3], M Bauce[4], T Bold[5], P Conde Muino[2], D Emeliyanov[3], R Goncalo[6], A Messina[4], M Negrini[7], L Rinaldi[8], A Sidoti[7], A Tavares Delgado[6], S Tupputi[9], L Vaz Gil Lopes[6] on behalf of the ATLAS Collaboration.

[1] Southern Methodist University, Dallas, TX, USA
[2] INESC-ID and Faculdade de Ciências, Universidade de Lisboa, Portugal
[3] STFC and Rutherford Appleton Lab, GB
[4] Sapienza Università di Roma and INFN, Italy
[5] AGH Univ. of Science and Technology, Krakow, Poland
[6] LIP, Lisboa, Portugal
[7] Istituto Nazionale di Fisica Nucleare (INFN), Italy
[8] Università di Bologna and INFN, Italy
[9] INFN-CNAF, Italy

E-mail: `sami.kama@cern.ch`

Abstract. The growing complexity of events produced in LHC collisions demands increasing computing power both for the online selection and for the offline reconstruction of events. In recent years there have been significant advances in the performance of Graphics Processing Units (GPUs) both in terms of increased compute power and reduced power consumption that make GPUs extremely attractive for use in a complex particle physics experiments such as ATLAS. A small scale prototype of the full ATLAS High Level Trigger has been implemented that exploits reconstruction algorithms optimized for this new massively parallel paradigm. We discuss the integration procedure followed for this prototype and present the performance achieved and the prospects for the future.

1. Introduction

The Large Hadron Collider (LHC) [1] located near Geneva Switzerland has been going through maintenance and upgrade operations since 2013. During Run-2 (2015-2017) it will operate at a higher centre-of-mass energy and instantaneous luminosity than in Run-1. The increase in luminosity and energy will lead to a higher number of pile-up interactions at each bunch crossing resulting in an increase in the number of tracks. Due to the combinatorial nature of tracking, the CPU requirements increase exponentially with the number of tracks. The effect of increased pileup on the average event reconstruction time can be seen in Figure 1. Significant improvements will be required to increase the speed of the algorithms that perform tracking as well as those that process information from the other detector systems. General Purpose GPUs (GPGPUs) provide good compute to power ratios and so are good candidates to maximize compute power within the constraints of available rack-space and cooling power.

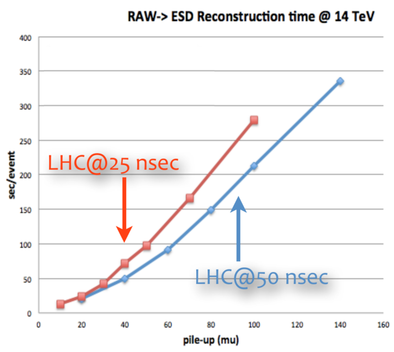

Figure 1. Average event processing times for full ATLAS offline reconstruction on simulated data at a centre of mass energy of 14 TeV versus the mean number of pileup interactions for 25 ns and 50 ns LHC bunch-crossing intervals.

1.1. *ATLAS*

ATLAS [2] is one of the two general purpose detectors located at the LHC. It is composed of concentric layers of detectors with a total of about 100 M readout channels. The Inner Detector (ID) tracker, shown in Figure 2, is located inside a solenoid magnet and contains three different detector technologies. Pixel detectors form the innermost layers surrounded by the Semiconductor Tracker (SCT) and Transition Radiation Tracker (TRT) respectively. Figure 3 shows the Calorimeter systems that surrounds the ID. The Liquid Argon (LAr) Electromagnetic calorimeter is located inside the Hadronic calorimeter that contains both LAr and scintillator tile technologies. Finally toroidal magnets and muon detectors form the Muon Spectrometer at the outermost layers of the ATLAS detector.

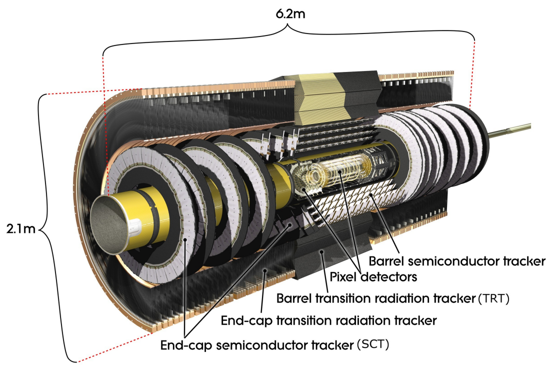
6.2m
2.1m
Barrel semiconductor tracker
Pixel detectors
Barrel transition radiation tracker (TRT)
End-cap transition radiation tracker
End-cap semiconductor tracker (SCT)

Figure 2. Components of ATLAS ID. Pixel detectors form innermost components, surrounded by silicon strip trackers (SCT). Transition radiation trackers (TRT) are situated in outermost layer.

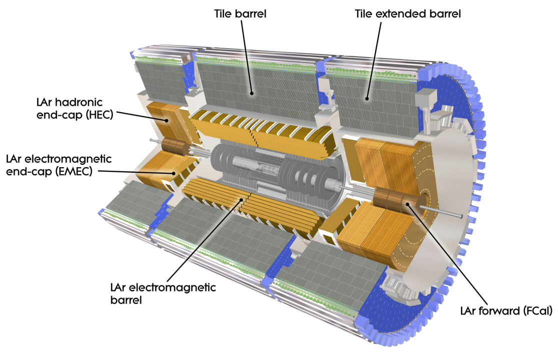
Tile barrel
Tile extended barrel
LAr hadronic end-cap (HEC)
LAr electromagnetic end-cap (EMEC)
LAr electromagnetic barrel
LAr forward (FCal)

Figure 3. Layout of ATLAS Calorimeter systems. LAr calorimeters are surrounded by Tile calorimeters.

1.2. *Trigger*

ATLAS will use a two level trigger system in Run 2. The Level 1 (L1) trigger is based on custom hardware located near the detector that reduces the 40 MHz interaction rate to a maximum of

100 kHz for input to the software based High Level Trigger (HLT). The L1 identifies interesting activity in small geometrical regions of detector called Region of Interests (RoI). Detector data is read from the front-end electronics for events accepted by the L1 trigger and stored in buffers in the Readout System (ROS) pending the HLT decision. The HLT is implemented in about 30000 software processes running on a PC farm with a per-event processing time budget of approximately 300 ms. The HLT analysis of the event is guided by the L1 RoI information. The HLT reduces the event rate to about 1000 Hz corresponding to a bandwidth of approximately 1500 MB/s. Events accepted by the HLT are sent to Data Loggers for permanent storage and offline processing,

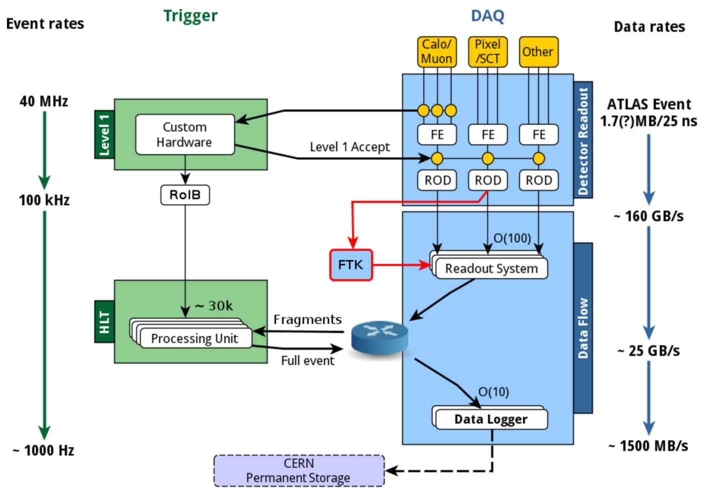

Figure 4. Schematic diagram of the ATLAS Trigger system showing the input and output event rates and the expected data rates at the different trigger levels.

The HLT uses a stepwise processing of sequences called Trigger Chains that are seeded by the RoIs identified by L1. The same RoI can seed multiple chains and the same chain can be scheduled for execution on different RoI. Each chain contains a sequence of algorithms and each algorithm builds on the output of the previous one. Successive algorithms in the chain perform increasingly complex reconstruction and selection steps. If a selection fails then the chain is marked as failed and further algorithms in the chain are not executed. If all chains are marked as failed, the event is rejected and the event data is flushed from the ROS buffers. This provides early rejection of uninteresting events with a minimum of processing. If an algorithm is scheduled for execution in multiple chains on the same data, the results of the first execution of the algorithm is cached. Other chains use the cached result instead of executing the algorithm again on the same data. This approach, using early rejection and caching, provides effective use of CPU resources while maintaining a good selection efficiency. Another feature that saves data-access bandwidth and CPU time (compared to full-event reconstruction) is that in the earlier steps algorithms work on partial event data, requiring 2-6% of data, while the algorithms in later steps have access to the full detector readout. About 95% of the events are rejected before the full event data is read from the detector. If at least one chain accepts, then the event is saved for offline processing. A schematic representation of the HLT Trigger Chains is shown in Figure 5.

1.3. GPU Demonstrator
ATLAS is investigating GPGPUs as a potential power-efficient compute resource to meet the demand for increasing computing power as higher luminosity and pile-up is delivered by the

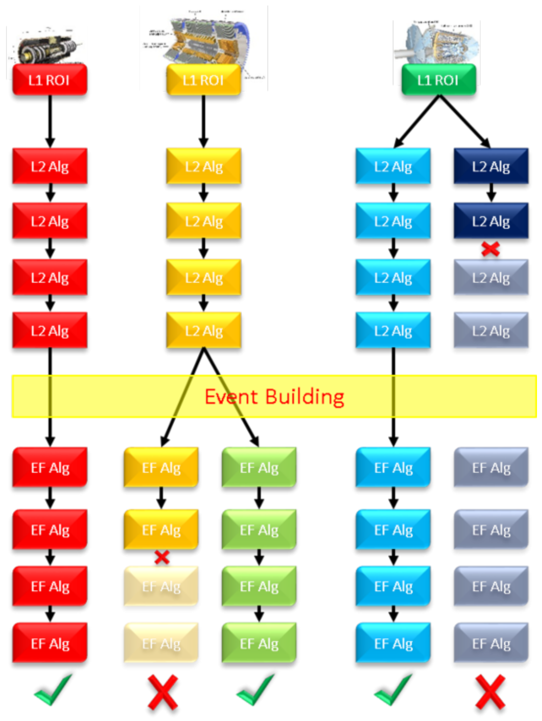

Figure 5. Schematic representation of Trigger Chains within the ATLAS HLT. Chains are composed of algorithms and are seeded by RoI from L1. The L2 algorithms work on partial event data while the EF algorithms have access to full event data.

LHC. Tracking is a natural candidate to exploit the massive parallelism offered by GPGPUs. An earlier prototype demonstrator that re-implemented the ID Trigger in CUDA [3] to run on a Nvidia Fermi architecture GPU achieved up to a factor of 12 speed-up compared to the original serial CPU implementation running on a single CPU core [4]. Figure 6 shows a comparison of the average per event tracking time as a function of the number of input space-points for three different cases: all algorithmic steps performed in a single CPU thread, all but one step (the last step that identifies and removes duplicate tracks) implemented on the GPU and all steps implemented on the GPU. ATLAS has established a working group to expand on the ID prototype and include calorimeter and muon reconstruction in order to study the potential benefit of using GPUs in the trigger farm in terms of throughput per unit cost.

2. Offloading

ATLAS uses multi-process event processing in the HLT. One HLT process (HLTPU) is executed for each virtual CPU core on the host. The HLTPU wraps the ATLAS reconstruction software, Athena, and interfaces it to the DAQ system. In order to manage and share GPUs between multiple HLTPU processes, a server-client approach has been chosen. The client side is implemented in the HLTPU and the server side as a separate process called the Accelerator Process Extension (APE).

The APE server manages offload requests from the clients and executes algorithm on GPUs. It implements a plug-in mechanism and is composed of three major parts. The main part is the Manager. It is responsible for client communication, management and execution of offload requests. The second part is the Module. A Module manages the GPU resources such as constant and time-varying data in the GPU memory, GPU streams in use, event bunching and

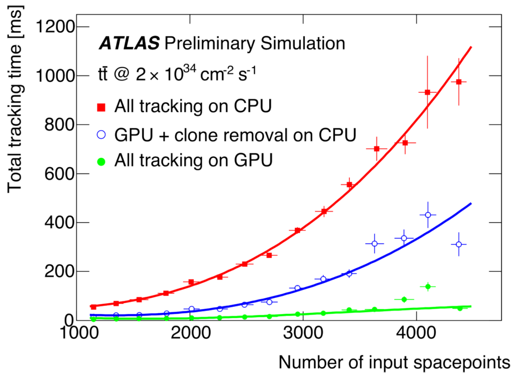

Figure 6. The average tracking time as a function of the number of space-points in the ID for CPU only, GPU only and CPU+GPU based algorithms [4].

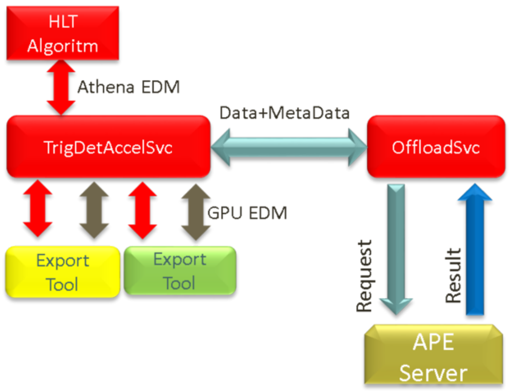

Figure 7. Diagram of data flow in the HLTPU offloading process.

creation of Work items. The third part is the Work item. A Work item is usually composed of series of GPU kernels. When a Work item is created by a Module it is added to a queue for execution. The Manager executes Work items and sends its results to the requester client. Each Module implements multiple Work items to offload different algorithms. The server supports multiple Modules and each detector implements their own Module to handle offload requests. The communication between the server and client is made using the yampl [5] library. Yampl abstracts different communication technologies such as sockets, pipes, shared memory and ZeroMQ [6], allowing in-host as well as across hosts offloading. The modular structure of the server allows utilization of different kinds of compute resources such as CPU, GPU, Xeon-Phi or FPGA accelerator cards without any change on the client side. It also abstracts the hardware from the client allowing deployment in diverse environments such as grid farms or HPCs if Modules for relevant hardware are implemented.

The client side is implemented in the HLTPU and consists of several cooperating components. An HLT algorithm makes a request for offload to the Acceleration Service. The Acceleration Service (TrigDetAccelSvc) converts C++ classes for raw and reconstructed quantities from the Athena Event Data Model (EDM) to an EDM optimized for GPU using Data Export Tools. Then it adds metadata to the exported data and asks the Offloading Service (OffloadSvc) to send the request to the APE server. The Offloading Service ships the request to the APE server and returns the results back to the Acceleration Service. Then the Acceleration Service converts the results back to the Athena EDM and passes them to the HLT algorithm. Each detector implements their own Acceleration Service and Data Export Tools. Due to the serial nature of the process, the data export tools are required to be fast, taking no more than a few percent of the total time in order to maximize scaling from the GPU. A diagram illustrating the client data-flow is shown in Figure 7.

3. Modules

After investigation of the HLT processing time the three most time consuming algorithms were selected for the demonstrator, namely the Inner Detector tracking, topological cluster finding in the Calorimeters and track finding in the Muon Spectrometer. The most time consuming of these is the Inner Detector tracking due to its computational complexity which has highly non-linear dependence on the detector occupancy and indirectly the luminosity. The following strategy has been adopted for implementing the Modules.

(i) It starts with analysis of the Athena execution flow in order to identify components that are performing algorithmic tasks. Replacements for these components are prepared such that offloading the data to APE leaves the overall HLTPU execution flow intact.

(ii) Next the Object Oriented data structures used in Athena are analysed and simpler variants optimized for GPU designed.

(iii) Tools to convert between Athena and GPU EDM formats are written, tested and timed in order to ensure that the conversion time does not offset the gains from the faster execution of the algorithms on the GPU.

(iv) Next geometry, noise levels and similar conditions data are treated in similar manner to the event data.

(v) Algorithms using the GPU data structures are initially implemented on CPU for algorithmic validation. An integration stage is envisaged after these steps are accomplished.

(vi) An implementation of the GPU algorithms follows.

(vii) The optimizations of GPU code and timing of the integrated system is planned.

3.1. ID Module

Tracking is the most time consuming part of the event processing and it will be more time consuming as pileup increases. The ID APE Module implements several CPU intensive tracking steps on GPU. There are 4 major steps. The first step is the raw data (Bytestream) decoding. In this step detector encoded outputs from the Pixel and SCT detectors are decoded and converted to hits on detector elements. The Bytestream is composed of words containing encoded information about hits in a module between a header and trailer word. In the GPU implementation, each word is assigned to a GPU thread and the decoding is done in parallel.

Charged particles typically activate one or more neighbouring sensors such as pixels or strips. Before track reconstruction, neighbouring activated sensors (hits) are merged into clusters. The hit clustering algorithm that merges activated pixel neighbours into clusters uses a Cellular Automata (CA) algorithm. Initially all hits start as an independent cluster and each hit is assigned to a GPU thread. At each iteration of the algorithm adjacent clusters are merged into one. Iteration continues until all adjacent hits belong to same cluster. A schematic representation of the algorithm is shown in Figure 8.

Once the clustering is completed, compatible clusters in inner layers are paired to form track seeds. A 2D thread array loops over the clusters in the inner layers and identifies suitable pairs. These pairs are then fed into a triplet making kernel. A 2D array of threads scans for clusters on outer layers for each pair, forming triplets. Then triplets are merged to form track candidates. An illustration of the algorithm can be seen in Figure 9. Finally in a clone removal stage track candidates which share the same outer hits but have different seeds are merged. Clone removal consists of two steps. In the first step, clone track candidates are identified and in the second step they are merged to form the final tracks.

Bytestream decoding and clustering algorithms have been integrated into the framework and run as part of the full reconstruction chain. Timing measurements for the different steps are compared for the GPU implementation and a serial CPU implementation as a function of the input data volume. The GPU implementation is found to be up to 21 times faster than the CPU implementation. A plot showing the timing comparison is shown in Figure 10. A new and upgraded tracking algorithm is now being implemented.

3.2. Calorimeter Module

The iterative nature of a topological clustering algorithm [8] and the large number of calorimeter cells make it a good candidate for GPU offloading. In the CPU implementation, the algorithm starts by classifying calorimeter cells by their signal-to-noise ratio. The cells where $S/N > 4$

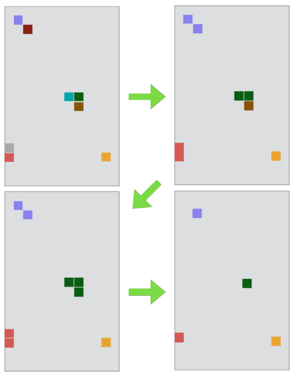

Figure 8. An illustration of the pixel hit-clustering CA algorithm. Each hit starts as an independent cluster (top-left). At each iteration, adjacent clusters are merged into same cluster. Iteration ends when all adjacent clusters are merged.

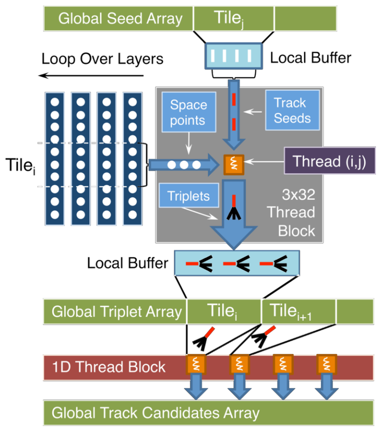

Figure 9. An illustration of the track forming algorithm. Track seeds are extended to outer layers by searching for compatible hits. Each thread scans outer layers for a seed.

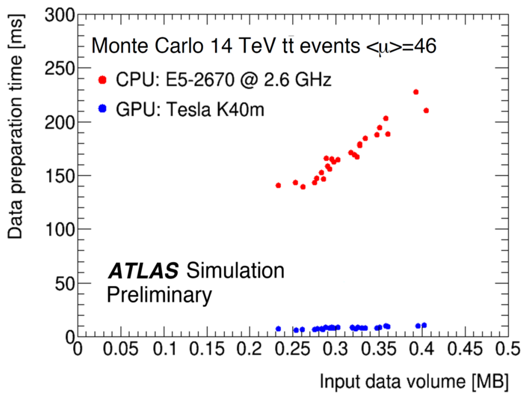

Figure 10. Comparison of the per-event execution time of Bytestream decoding and clustering algorithms for a GPU implementation on a Nvidia Tesla K40m and a serial CPU implementation running on a E5-2670@2.6 GHz for different input data sizes. The GPU implementation shows up to 21x speed-up compared to the original unmodified CPU implementation running on a single CPU core [7].

are labelled as *Seeds*, $4 > S/N > 2$ are *Growing* and $2 > S/N > 0$ are *Terminal* cells. After classification, the algorithm starts from Seed cells and adds all Growing cells around it into the cluster. Then all Growing or Terminal cells around the Growing cells in cluster are included into the cluster until either the cluster reaches a predefined maximum size or there are no cells left to include. In the parallel GPU implementation a CA approach has been chosen. After classification, each cell starts as an independent cluster handled by a GPU thread. At each iteration, cells join the neighbouring cluster with the highest S/N ratio. The procedure continues until the cluster reaches a maximum size or there are no more clusters that it can join. The GPU algorithm being implemented and the existing CPU algorithm are both illustrated in Figure 11.

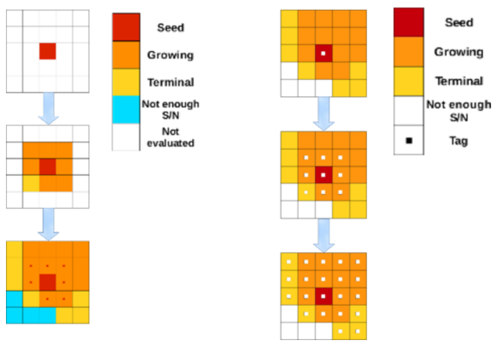

Figure 11. Illustration of the Topo-Clustering algorithm implementations for GPU (left) and CPU (right) .

4. Summary and Future Outlook

The increasing instantaneous luminosity of the LHC necessitates the exploitation of parallel processing. The massive parallelization of trigger algorithms implemented on GPUs is being investigated as a way to increase the compute power of the HLT farm. A framework, APE, is being developed to integrate GPUs into the multi-processing ATLAS software framework, Athena. Work to offload the CPU intensive parts of the ID tracking, Calorimeter Topo-Clustering, and Muon tracking are ongoing. The Data Preparation step of the ID tracking has already been successfully integrated into Athena and a speed-up of about 21x is observed compared to a serial CPU implementation.

Development of the GPU algorithms and Modules is ongoing. In the near term a new implementation of the ID track-finding and clone-removal algorithms will be included in an ID Module and the respective parts will be offloaded to GPU. This will be followed by the implementation of the Topo-Clustering algorithm in a Calorimeter APE Module. Finally, after implementation and inclusion of Muon tracking, detailed measurements including throughput per unit cost will be completed by the end of the year. These measurements will be used to estimate the potential benefits of GPUs in future upgrades of the ATLAS HLT farm.

References

[1] Evans L and Bryant P 2008 *JINST* **3** S08001
[2] Aad G *et al.* (ATLAS Collaboration) 2008 *JINST* **3** S08003
[3] *NVidia/Cuda* URL `https://developer.nvidia.com/cuda-zone`
[4] Baines J, Bristow T, Emeliyanov D, Howard J, Kama S, Washbrook A and Wynne B 2014 URL `http://cds.cern.ch/record/1754968`
[5] *Yampl Library* URL `https://github.com/vitillo/yampl`
[6] *ZeroMQ* URL `http://zeromq.org`
[7] Emeliyanov D 2015 URL `http://twiki.cern.ch/twiki/bin/view/AtlasPublic/HLTTrackingPublicResults`
[8] Lampl W, Laplace S, Lelas D, Loch P, Ma H, Menke S, Rajagopalan S, Rousseau D, Snyder S and Unal G 2008 URL `http://cds.cern.ch/record/1099735`