Multi-Threaded Algorithms for GPGPU in the ATLAS High Level Trigger

P. Conde Muíño on behalf of the ATLAS Collaboration

LIP (Laboratório de Instrumentação e Física Experimental de Partículas),
Elias Garcia 14, 1000-149 Lisbon, PT
Departamento de Física, Faculdade de Ciências, Universidade de Lisboa, PT

Abstract. General purpose Graphics Processor Units (GPGPU) are being evaluated for possible future inclusion in an upgraded ATLAS High Level Trigger farm. We have developed a demonstrator including GPGPU implementations of Inner Detector and Muon tracking and Calorimeter clustering within the ATLAS software framework. ATLAS is a general purpose particle physics experiment located on the LHC collider at CERN. The ATLAS Trigger system consists of two levels, with Level-1 implemented in hardware and the High Level Trigger implemented in software running on a farm of commodity CPU.

The High Level Trigger reduces the trigger rate from the 100 kHz Level-1 acceptance rate to 1.5 kHz for recording, requiring an average per-event processing time of ~ 250 ms for this task. The selection in the high level trigger is based on reconstructing tracks in the Inner Detector and Muon Spectrometer and clusters of energy deposited in the Calorimeter. Performing this reconstruction within the available farm resources presents a significant challenge that will increase significantly with future LHC upgrades. During the LHC data taking period starting in 2021, luminosity will reach up to three times the original design value. Luminosity will increase further to 7.5 times the design value in 2026 following LHC and ATLAS upgrades. Corresponding improvements in the speed of the reconstruction code will be needed to provide the required trigger selection power within affordable computing resources.

Key factors determining the potential benefit of including GPGPU as part of the HLT processor farm are: the relative speed of the CPU and GPGPU algorithm implementations; the relative execution times of the GPGPU algorithms and serial code remaining on the CPU; the number of GPGPU required, and the relative financial cost of the selected GPGPU. We give a brief overview of the algorithms implemented and present new measurements that compare the performance of various configurations exploiting GPGPU cards.

ATL-DAQ-PROC-2016-045
15/05/2017

1. Introduction

ATLAS [1] is a multipurpose detector installed at the Large Hadron Collider (LHC) [2] at CERN, designed to study a wide range of physics processes. At the nominal conditions, the LHC provides protons with an instantaneous luminosity of 10^{34} cm^{-2}s^{-1}, a bunch crossing rate of 40 MHz and an average number of proton-proton collisions per bunch crossing (pile-up) of around 22. These numbers already were exceeded by 20% during 2016 data taking. The ATLAS Trigger and Data Acquisition (TDAQ) system processes the large volume of data produced by the ATLAS detector, $\mathcal{O}$(PB/s), to reduce the incoming event rate from the 40 MHz nominal value to around 1.5 kHz that can be stored to disc for further offline analysis. This is achieved with a trigger system divided in two main levels. The first one, Level-1, is hardware based and uses coarse granularity from the calorimeter and muon detectors to identify high transverse

momentum, p_T, objects like electrons, photons, taus, jets, muons or missing transverse energy produced by particles escaping detection. It has to provide a decision within a latency of 2.5 μs, reducing the incoming rate to 100 kHz. The event is further processed by the High Level Trigger (HLT), a software based system that runs sequences of algorithms to confirm/reject the selections made by the Level-1 within an average processing time of $\sim$ 250 ms and reducing the rate to the desirable 1.5 kHz.

To increase the physics reach of the experiments, an LHC Upgrade program has been planned, divided in two phases: in the first one, Phase I (2019-20), the accelerator will be upgraded to provide up to three times higher instantaneous luminosity than currently, while after the Phase II (2024-26) upgrade it will be able to deliver an instantaneous luminosity up to 7.5×10^{34} cm^{-2}s^{-1}. The higher instantaneous luminosity means a much larger number of collisions per bunch crossing and therefore larger volumes of data that will have to be processed by the ATLAS TDAQ system. Selecting physics processes of interest, however, requires low transverse momentum thresholds. Control of trigger rates will be achieved by using advanced algorithms, including pile-up correction, that are capable of providing larger rejection than current algorithms for the same physics signal efficiency. This will result in longer execution times.

The HLT processing farm, that relies on commodity computing, is limited in size due to power and cooling restrictions in the ATLAS facilities. The expected increase in CPU processing time at higher luminosity is dominated by the tracking algorithms, due to their combinatorial nature. This could be addressed by the use of hardware accelerators such as General Purpose Graphical Processing Units (GPGPU) that have the capability for massive parallelization.

In order to evaluate the potential use of GPGPU in the ATLAS HLT, a demonstrator HLT trigger prototype is under development. Tracking, calorimeter clustering, muon and jet reconstruction algorithms are being redesigned and implemented to obtain maximum benefit from the GPGPU architecture. The prototype will be evaluated in terms of the number of events processed per second (throughput) per unit cost. NVIDIA Kepler GPGPU and the CUDA programming language were chosen for the implementation of the demonstrator. The performance of the system was tested on a host containing two Intel(R) Xeon(R) E5-2695 v3 with 14 cores each and a clock speed of 2.3 GHz and an NVIDIA Tesla K80 with two Tesla chips of 2496 CUDA cores per chip, with 12 GB of RAM and 2505 MHz memory clock.

The design and implementation of the demonstrator prototype will be explained in detail, followed by a presentation of initial performance measurements.

2. The ATLAS Trigger GPGPU Demonstrator

In the current system, one HLT processing unit (HLTPU) is run per physical core of the HLT farm. The HLTPU runs the ATLAS event reconstruction and trigger framework Athena. The latter is responsible for the execution of the algorithms and the trigger steering that runs sequences of selection algorithms and takes the decision of which events to accept or reject, according to a predefined trigger menu. Athena also provides the data and the monitoring services for the trigger algorithms.

To abstract the use of hardware accelerators from the main trigger processing, a client-server architecture was chosen, as illustrated in Figure 1. An offloading service was implemented in Athena to export and receive the data to/from the GPGPU. This service is also responsible for the transformation of the Athena-specific C++ classes of the ATLAS Event Data Model into simple structures of arrays (SoA), more suitable for GPGPU processing. The communication with the GPGPU is made through the Accelerator Process Extension (APE) server, who is independent of Athena and acts as an efficient resource manager, serving many clients at the same time. APE can exploit different accelerator technologies using plug-in modules that implement the specific algorithms. It is also responsible for the allocation of resources, for sending the modules and the associated input event data and receiving the algorithm output

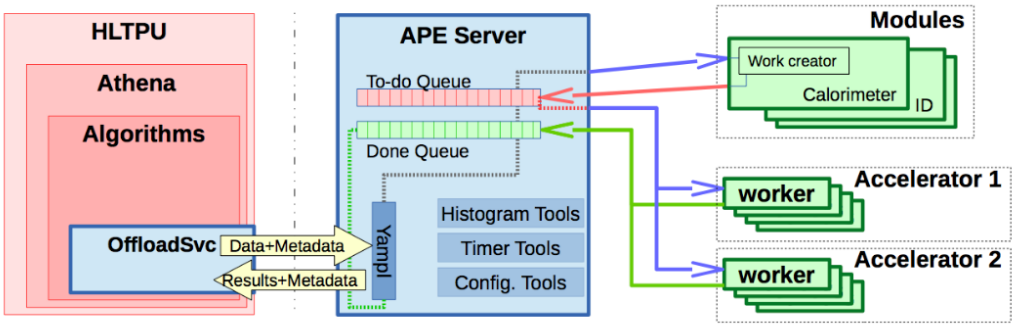

Figure 1. Schematic diagram of the ATLAS Trigger GPGPU client-server architecture.

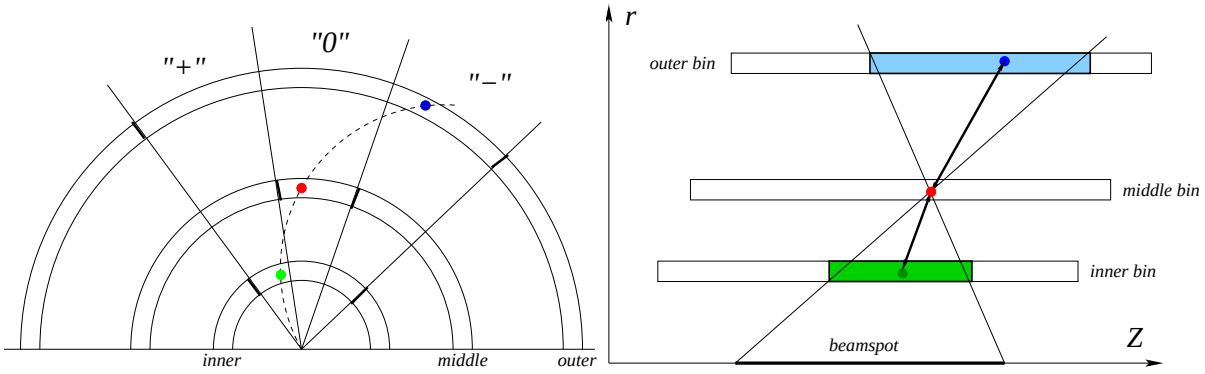

Figure 2. Schematic representation of the Inner Detector space point formation algorithm.

from the different accelerators available. It provides in addition monitoring services.

2.1. Inner Detector Tracking Algorithms

The Inner Detector (ID) track reconstruction is run as a sequence of steps, starting from the conversion of the raw byte-stream data, provided by the detector, into C++ classes representing detector hits; the formation of three-dimensional space points (SP) from the hits; and the reconstruction of track seeds from SP in three layers, that are then propagated to the following ID planes to reconstruct full tracks. The process finishes with the removal of duplicate tracks, i.e. those with shared hits.

For the demonstrator, the track seeding step was outsourced to the GPGPU while the other steps were executed on the CPU. The track seeding comprises two parts: the formation of triplets of space points in the silicon tracking detectors and the application of kinematic and quality criteria to select the SP that will be further propagated. To give an idea of the complexity of the different steps, starting from around 10^5 SP, the seeding step forms around 10^9 triplets that are then reduced to around 10^4 seeds and will form around 10^3 tracks. Therefore, the seeding is not only the most time consuming step but also very appropriate, due to its combinatorial nature, to the parallel processing architecture of the GPGPU.

The algorithm to form seeds arranges the space points (SP) in wedge-shaped slices in the plane transverse to the beam direction as shown schematically in Figure 2 (left), such that a 1 GeV track will have SP in at most three consecutive slices. Then, each SP (except for the ones in the inner layer) is treated as a potential middle SP of a triplet. Additional SP in the inner and outer layers are searched for, in the same slice or the immediate neighouring ones. The triplet

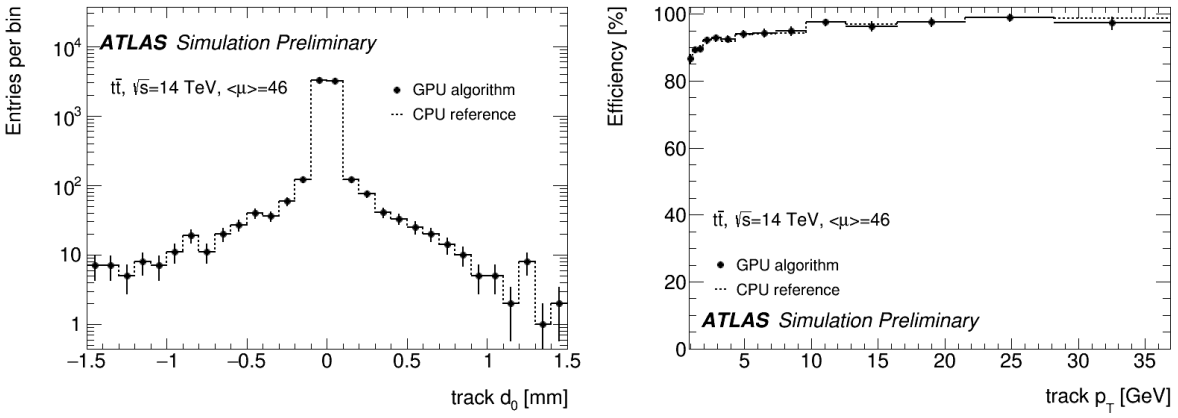

Figure 3. The transverse impact parameter (left) and track reconstruction efficiency as a function of transverse momentum (right) distributions for the simulated tracks correctly reconstructed by the GPGPU-accelerated tracking algorithm and the standard CPU-only algorithm[3].

formation is restricted requiring that the z coordinate projection is compatible with the beam-spot position, as indicated schematically in Figure 2 (right). The algorithm was implemented in three GPGPU kernels: the first calculates the number of doublets with middle and inner or outer SP and allocates the global storage space for doublets accordingly; the second module forms the doublets and stores them as 32-bit indices of the corresponding outer/inner SP; and the last module forms the triplets and applies kinematic and quality cuts. The typical size of the storage is about 40 MB and therefore the global memory of the GPGPU has to be used.

The performance of the GPGPU tracking algorithm was studied using a simulated data sample of top quark pair production with an average pile-up of 46 interactions per bunch crossing. The distribution of the reconstructed impact parameter of the tracks with respect to primary vertex is shown in Figure 3 (left) for the GPGPU algorithm with full circles and for the reference CPU tracks with a dashed line. There is very good agreement between the two distributions. The efficiency was studied with respect to a set of simulated reference tracks produced within the geometrical acceptance of the tracking detector and with $p_T \geq 1$ GeV. Each reconstructed track was required to be within a $\Delta R = \sqrt{\Delta\phi^2 + \Delta\eta^2}$ distance of 0.05 with respect to the reference. The results for the GPGPU and CPU algorithms are compared in Figure 3 (right). The same efficiency is obtained for both algorithms, demonstrating the good performance of the GPGPU tracking algorithm.

The results of timing measurements are shown in the pie charts in Figure 4 for the total event processing time in the CPU and the GPGPU, as well as the breakdown of the different kernels and the data conversion time for the GPGPU. The tracking seeding processing time is reduced by a factor 5 when executed in the GPGPU, corresponding to a 25% reduction in the overall event processing time. The performance is limited due to memory access. The inter-process communication (IPC), data transfer and data conversion overhead accounts for around 13% of the algorithm processing time.

2.2. Calorimeter Reconstruction Algorithms

The calorimeter cluster reconstruction algorithm in ATLAS, called TopoCluster [4], groups calorimeter cells into three-dimensional clusters to reconstruct particle energy depositions. In a first step cells are classified as seed, growing or terminal, according to three programmable thresholds on their energy to noise ratio, S/N. Typical thresholds are 4, 2, 0 for seed, growing

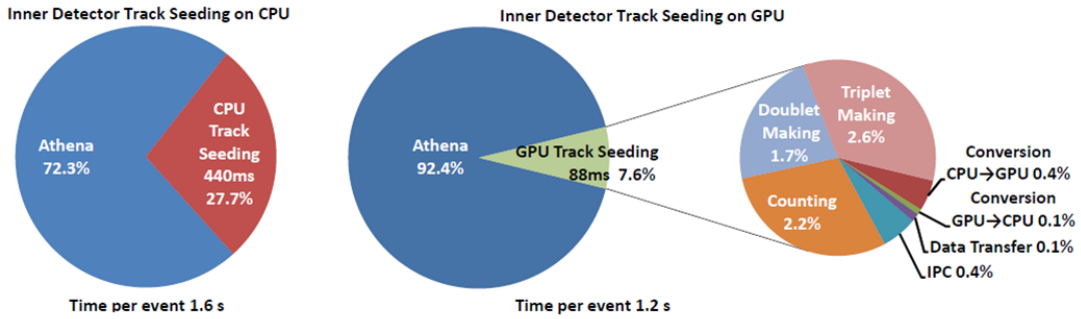

Figure 4. Breakdown of the time per event for the ATLAS trigger process (Athena) running ID track seeding on the CPU or offloaded to a GPGPU showing the time fraction for the counting, doublet making and triplet making kernels running on the GPGPU and the overhead associated with offloading the work (data transfer, data conversion and inter-process communication) [6].

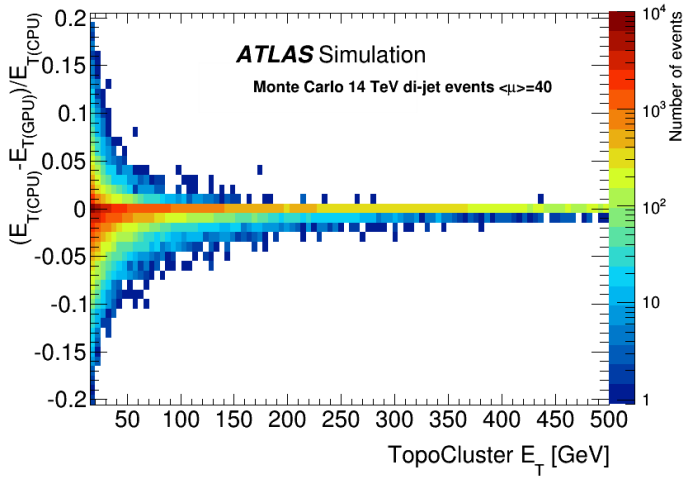

Figure 5. The relative transverse energy difference of the matched calorimeter clusters reconstructed using the TopoCluster algorithm and the logically similar algorithm ported to GPGPU, as a function of the transverse energy returned by TopoCluster [5].

or terminal cells respectively. The algorithm starts with seeds ordered in S/N and joins neighbouring cells if their S/N is above the terminal threshold. The procedure is iterated for all the growing cells in a cluster. Clusters with growing cells that touch each other are merged. Terminal cells are assigned to the first cluster that reaches them. The last part of the algorithm splits the cluster to separate energy depositions from different particles.

For the results presented in this paper only the cluster growing step, the most time consuming part in the TopoCluster reconstruction, was ported to the GPGPU using a cellular automaton algorithm, the Topo-Automaton Clustering (TAC). The geometrical information concerning the 170 k calorimeter cells and their neighbours is stored at the beginning of the run in the global memory of the GPGPU. Event by event, data corresponding to the calorimeter cell energy and noise is sent to the GPGPU in the form of SoA. The algorithm starts with the classification of cells according to the three levels of S/N. Then, the seed cells are sorted in S/N and attributed a tag, corresponding their position in the sorted array. All tags are stored in a new array, organized

Table 1. Time reduction factor of the TAC algorithm with respect to TopoCluster algorithm for different simulated data samples, with different pile-up conditions [5][6].

Data sample	Pile-up	Time reduction factor
$t\bar{t}$	138	2
$t\bar{t}$	46	2
di-jet	40	1.3

according to the cell index. To maximize parallelism, the work space is simplified into pairs of a cell and a neighbour, for all the seed/growing cells. In each iteration, the cellular automaton evaluates all the neighbour pairs and assigns the largest tag to both cells. Terminal cells are not allowed to propagate their tag and they are always attributed to the cluster with largest tag, for reproducibility. If a new tag is proposed to a seed or growing cell that has already been assigned to another cluster, a tag-pair is formed for a later merging step and the cell tag remains unchanged. The algorithm iterates over the array of cell pairs until no tag is changed. Finally, the list of tag-pairs is processed, attributing the largest tag to the clusters to be merged. The TAC algorithm was implemented in three different kernels, the first classifies cells and builds cell pairs, the second generates a unique tag for each cell-pair and the third, the cluster-growing kernel, propagates the tags to form clusters.

The performance of the TAC algorithm was studied using three different simulated data samples, with different occupancies: di-jet events with an average pile-up of 40 interactions per bunch crossing and two samples of top quark pair production with average pile-up of 46 and 138 collisions/bunch crossing, respectively. Detailed comparisons were made between the clusters reconstructed with TAC and TopoCluster. They were found to differ only in the terminal cells, as it is expected due to different assignment conditions in the two algorithms. The relative difference between the reconstructed transverse energy of the clusters reconstructed by the TAC and TopoCluster algorithms is shown in Figure 5 as a function of the TopoCluster transverse energy (E_T). The energy difference is larger for low E_T clusters due to the larger impact of terminal cells. For $E_T > 50$ GeV the differences are always smaller than 5%.

The algorithm execution times are presented in Figure 6 that shows the total event processing time, the fraction of time spent by the TopoCluster and TAC algorithms and the break down of the different kernels and processing steps in TAC. The TopoCluster growing algorithm takes only a small fraction (8%) of the total event processing time. This is reduced by around a factor of two when run on the GPGPU. The data conversion and inter-process communication overhead takes a large fraction of the processing time, $\sim 40\%$. The whole Athena event processing takes slightly longer when the offloading to the GPGPU is performed because of a small increase in the processing time of the cluster splitting phase, which was not ported to the GPGPU. The time reduction factor of the TAC algorithm with respect to TopoCluster ranges between 1.3 and 2 depending on the occupancy of the events, as summarized in Table 1. This is due to the larger impact of the overheads for shorter processing times. The performance of the algorithm is memory bound, due to access to the global memory. Larger speed-up factors are expected once the cluster-splitting step has been parallelized and implemented on the GPGPU especially since this will not require additional data conversion and inter-process communication overheads.

2.3. Muon Track Reconstruction Algorithms

The reconstruction of muons in ATLAS is achieved by combining information from the Muon Spectrometer (MS) and the ID system. A series of software trigger algorithms reconstructs the

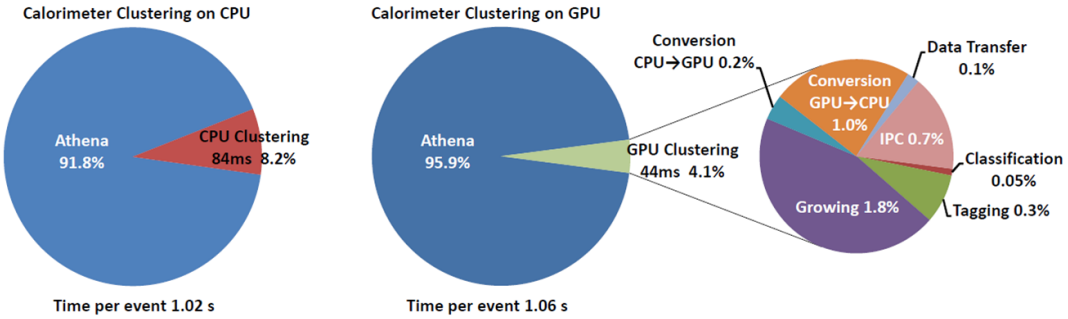

Figure 6. Breakdown of the time per event for the ATLAS trigger process (Athena) running calorimeter clustering on CPU and offloaded to a GPGPU showing the time for the classification, tagging and clustering kernels running on the GPGPU (GPGPU execution) and the overhead associated with offloading the work [6].

trajectory of a candidate muon starting from the position of all the MS signals above threshold (hits). The Hough Transform method is used to associate hits to straight track segments in different detector planes that are then geometrically matched and all the hits associated to them used to perform a fit of the full particle trajectory. The Hough Transform algorithm is intrinsically parallel; hence very appropriate for implementation on a GPGPU. The GPGPU implementation of the algorithm is split between different kernels that execute the various reconstruction steps. The first one arranges hits into subsets that are used both to fill the Hough matrix and for association to a reconstructed segment. Next the Hough matrix is filled and local maxima identified. The list of maxima is sorted according to the height of the maximum. Finally, for each maximum the corresponding geometric and kinematic parameters are evaluated and a hit pattern, that will be transmitted to the next reconstruction algorithm, is created. The performance of the muon tracking GPGPU algorithms is currently under evaluation.

3. Event processing rates of the GPGPU Demonstrator Prototype

The performance of the demonstrator prototype was evaluated in terms of the event rate processed (throughput) when running the calorimeter-only reconstruction, ID tracking only, or both; with one or two GPGPU as a function of the number of Athena processes running. The ratio of throughput with GPGPU acceleration to CPU-only is shown in Figure 7 (left). The maximum gain corresponds to the case when only ID-tracking is run, and ranges between 20-40% depending on the number of Athena processes. One GPGPU can efficiently serve up to 14 processes without a reduction in throughput. This is demonstrated by the average number of Athena processes waiting for the completion of the offloaded work (Figure 7, right), that increases rapidly after 14 Athena processes when only one GPGPU is used. With two GPGPU, the number of processes waiting is stable up to around forty. For the calorimeter-only reconstruction, the small increase in the processing time of the non-accelerated code leads to a small decrease in the throughput for the case of offloading to GPGPU. When both ID and calorimeter reconstruction are offloaded, the throughput increases by around 20% relative to CPU-only processing. The gain obtained in the event rates is expected to improve once more code is offloaded to the GPGPU, such as track following, cluster splitting or the jet reconstruction. In addition, cheaper gaming GPGPU's, capable of providing similar acceleration at cheaper cost, are being tested.

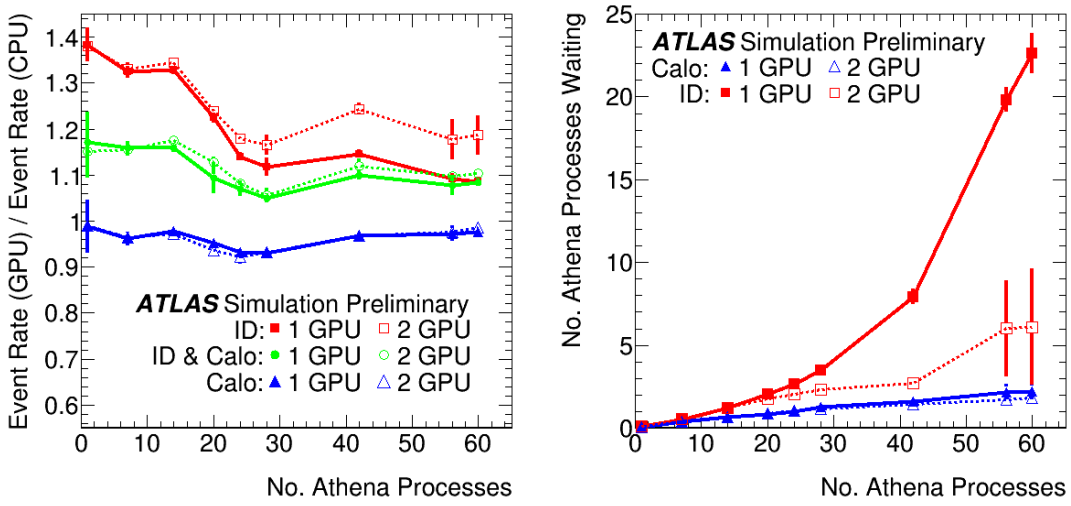

Figure 7. The ratio of event throughput rates with GPGPU acceleration to the CPU-only rates as a function of the number of ATLAS trigger (Athena) processes running on the CPU. Separate tests were performed with Athena configured to execute only Inner Detector tracking (ID), only Calorimeter topological clustering (Calo) or both (ID & Calo) [6].

4. Summary and Conclusions

The LHC Upgrade program will impose stringent requirements on the ATLAS trigger system that will have to handle larger volumes of data and run advanced algorithms to provide higher background rejection with the same efficiency. To face this challenge, ATLAS is evaluating the use of GPGPU for triggering purposes. Calorimeter cluster and inner detector tracking reconstruction algorithms have been implemented on GPGPU, reducing the execution time by a factor of five for the tracking and a factor two for the calorimeter clustering, compared to the CPU algorithm. The overheads of data transformation and inter-process communication range from 13% to 40%, depending on the algorithm. The data transformation time, accounting for almost half of this overhead, could be reduced using data structures optimized for both CPU and GPGPU. The number of events processed per second can be increased up to 20-40% when GPGPU are used. Potential improvements are expected once more code is offloaded to the GPGPU. As an alternative, it may be possible to improve the CPU code performance using vectorization or exploiting some of the techniques used for the GPGPU demonstrator.

Acknowledgments

The author acknowledges the support of the grant CERN/FIS-NUC/0005/2015, the program *Investigador FCT*, POPH, EU and FCT, Portugal.

References

[1] ATLAS Collaboration, *JINST* **3** S08003, 2008.
[2] L. Evans and P. Bryant ,*JINST* **3** S08001, 2008.
[3] ATLAS Collaboration, ATL-COM-DAQ-2016-119,
https://twiki.cern.ch/twiki/bin/view/AtlasPublic/TriggerSoftwareUpgradePublicResults.
[4] ATLAS Collaboration, ATL-LARG-PUB-2008-002, https://cds.cern.ch/record/1099735.
[5] ATLAS Collaboration, ATL-COM-DAQ-2016-133,
https://twiki.cern.ch/twiki/bin/view/AtlasPublic/TriggerSoftwareUpgradePublicResults.
[6] ATLAS Collaboration, ATL-COM-DAQ-2016-116,
https://twiki.cern.ch/twiki/bin/view/AtlasPublic/TriggerSoftwareUpgradePublicResults.