The GAP project: GPU applications for High Level Trigger and Medical Imaging

Matteo Bauce[1,2], *Andrea Messina*[1,2,3], *Marco Rescigno*[3], *Stefano Giagu*[1,3], *Gianluca Lamanna*[4,6], *Massimiliano Fiorini*[5]

[1]Sapienza Università di Roma, Italy
[2]INFN - Sezione di Roma 1, Italy
[3]CERN, Geneve, Switzerland
[4]INFN - Sezione di Frascati, Italy
[5]Università di Ferrara, Italy
[6]INFN - Sezione di Pisa, Italy

DOI: http://dx.doi.org/10.3204/DESY-PROC-2014-05/1

The aim of the GAP project is the deployment of Graphic Processing Units in real-time applications, ranging from the online event selection (trigger) in High-Energy Physics to medical imaging reconstruction. The final goal of the project is to demonstrate that GPUs can have a positive impact in sectors different for rate, bandwidth, and computational intensity. Most crucial aspects currently under study are the analysis of the total latency of the system, the algorithms optimisations, and the integration with the data acquisition systems. In this paper we focus on the application of GPUs in asynchronous trigger systems, employed for the high level trigger of LHC experiments. The benefit obtained from the GPU deployement is particularly relevant for the foreseen LHC luminosity upgrade where highly selective algorithms will be crucial to maintain a sustainable trigger rates with very high pileup. As a study case, we will consider the ATLAS experimental environment and propose a GPU implementation for a typical muon selection in a high-level trigger system.

1 Introduction

The Graphic Processing Units (GPU) are commercial devices optimized for parallel computation, which, given their rapidly increasing performances, are being deployed in many general purpose application. The GAP project is investigating GPU applications in real-time environments, with particular interest in High Energy Physics (HEP) trigger systems, which will be discussed in this paper, but also Medical Imaging reconstruction (CT, PET, NMR) discussed in [1]. The different areas of interest span several orders of magnitude in terms of data processing, bandwidth and computational intensity of the executed algorithms, but can all benefit from the implementation of the massively parallel architecture of GPUs, optimizing different aspects, in terms of execution speed and complexity of the analyzed events. The trigger system of a typical HEP experiment has a crucial role deciding, based on limited and partial information, whether a particular event observed in a detector is interesting enough to be recorded. Every

experiment is characterised by a limited Data Acquisition (DAQ) bandwidth and disk space for storage hence needs real-time selections to reduce data throughput selectively. The rejection of uninteresting events only, is crucial to make an experiment affordable, preserving at the same time its discovery potential. In this paper we report some results obtained from the inclusion of GPU in the ATLAS High Level Trigger (HLT); we will focus on the main challenges to deploy such parallel computing devices in a typical HLT environment and on possible improvements.

2 ATLAS trigger system

The LHC proton-proton accelerator provides collisions at a rate of 40 MHz, which corresponds, for events of a typical size of 1-2 MByte, to an input data rate of the order of tens of TB/s. The reduction of this input rate to a sustainable rate to be stored on disk, of the order of ∼100 kHz, is achieved through a hybrid multi-level event selection system. Lower selection stages (Lower Level Triggers) are usually implemented on customized electronics, while HLT are nowadays implemented as software algorithms executed on farms of commodity PCs. HLT systems, in particular those of LHC experiments, offer a very challenging environment to test cutting-edge technology for realtime event selection. The LHC upgrade with the consequent increase of instantaneous luminosity and collision pile-up, poses new challenges for the HLT systems in terms of rates, bandwidth and signal selectivity. To exploit more complex algorithms aimed at better performances, higher computing capabilities and new strategies are required. Moreover, given the tendency of the computing industry to move away from the current CPU model towards architectures with high numbers of small cores well suited for vectorial computation, it is becoming urgent to investigate the possibility to implement a higher level of parallelism in the HLT software.

The GAP project is investigating the deployment of GPUs for the HLT in LHC experiments, using as a study case the ATLAS muon HLT. The ATLAS trigger system is organized in 3 levels [2], as shown in Figure 1. The first trigger level is built on custom electronics, while the second level (L2) and the event filter (EF) are implemented in software algorithms executed by a farm of about 1600 PCs with different Xeon processors each with 8 to 12 cores. During the Run II of the LHC (ex-

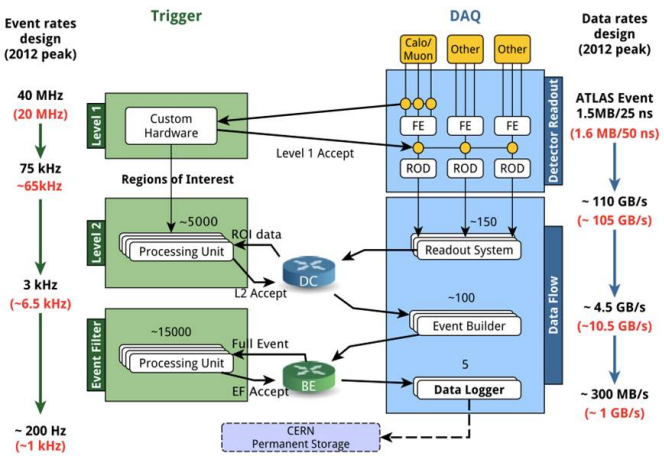

Figure 1: A scheme of the ATLAS trigger system; values in red are indicating the data-taking conditions during the first run of the LHC ('10-'12) while values in black represents the design conditions, expected to be reached during the upcoming run (starting in 2015).

pected to start in 2015) the L2 and EF will be merged in a single software trigger level. Currently, a first upgrade is foreseen in 2018 [3], when real-time tracking capabilities will also be available, followed by a complete renovation of the trigger and detector systems in 2022. We intend to explore the potential improvements attainable in the near future by deploying GPUs in the ATLAS LVL2 muon trigger algorithms. Such algorithms are now implemented as simplified versions and are based on the execution for a large number of times of the same algorithms that reconstruct and match segments of particle trajectories in the detector. The high computing capabilities of GPUs would allow the use of refined algorithms with higher selection efficiency, and thus to maintain the sensitivity to interesting physics signals even at higher luminosity.

In the current ATLAS data acquisition framework it is not possible to include directly a parallel computing device; the integration of the GPU in this environment is done through a server-client structure (Accelerator Process Experiment - APE [4]) that can manage different tasks and their execution on an external coprocessor, such as the GPU. This implementation is flexible, able to deal with different devices having optimized architecture, with a reduced overhead. With the help of this structure it is possible to isolate any trigger algorithm and optimize it for the execution on a GPU (or other parallel architecture device).

This will imply the translation into a parallel computing programming language (CUDA[1] [7]) of the original algorithm and the optimization of the different tasks that can be naturally parallelized. In such a way the dependency of the execution time on the complexity of the processed events will be reduced. A similar approach has been investigated in the past for the deployment of GPUs in different ATLAS algorithms with promising results [5]. The evolution of the foreseen ATLAS trigger system, that will merge the higher level trigger layers in a unique software processing stage, can take even more advantage from the use of a GPU since a more complex algorithm, with offline-like resolution can be implemented on a thousand-core device with significant speedup factors. The timing comparison between the serial and the parallel implementation of the trigger algorithm is done on the data collected in the past year.

3 Muon reconstruction isolation algorithm

The benchmark measurements that has been carried out has focused on one of the algorithms developed for muon reconstruction in ATLAS. In the L2 trigger a candidate muon particle is reconstructed combining three different and sequential algorithms: the first one reconstructs a charged particle track segment in the muon spectrometer [6], a second algorithm matches in space such track segment to a charged particle track reconstructed in the ATLAS Inner Detector (ID) [6], the third evaluates the energy deposits in the electromagnetic and hadronic calorimeters (ECAL, HCAL) [6], as well as the track density in the detector region around the candidate muon trajectory, to check the consistency with a muon crossing the whole ATLAS detector. This third step of the muon reconstruction has been considered for our first test deploying a GPU in the ATLAS trigger system.

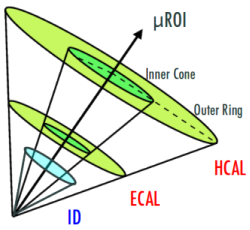

Figure 2: Scheme of the cone used in the muon isolation algorithm.

[1] We perform the tests described in this article on a server set up for this purpose including an NVIDIA graphic accelerator, hence the GPU code has been developed in CUDA.

The muon isolation algorithm starts from the spatial coordinates of the candidate muon trajectory and consider a cone of fixed width $\Delta R = \sqrt{\Delta\phi^2 + \Delta\eta^2}$ in the (η,ϕ) ([6]) space around such trajectory. In order to pass the muon track isolation requirement there should be only a limited number of tracks in the considered cone, and these must be consistent with background in the inner ATLAS tracking system. The calorimeter isolation is applied summing the energy deposits in the electromagnetic and hadronic calorimeter cells lying within the considered cone, and requiring this is only a small fraction of the estimated candidate muon energy[2]. Figure 2 shows the definition of the cone used to evaluate the muon isolation in the calorimeter and in the inner detector.

The integration of the GPU and the execution of such algorithm within the ATLAS trigger framework can be summarized in the following steps:

1. retrieve information from the detector: access to the calorimeter cells information;
2. format information needed by the algorithm, namely the cell content, data-taking conditions, and calorimeter configuration information, into a memory buffer;
3. transfer the prepared buffer to the server (APE) which handles the algorithm execution and transfer the buffer to the GPU;
4. algorithm execution on the GPU (or on a CPU in the serial version of the algorithm);
5. transfer of the algorithm results through the APE server back into the ATLAS trigger framework;

Step 1 is the same also in the current implementation of the muon isolation algorithm; in the standard ATLAS trigger implementation (serial) this step is followed directly by the algorithm execution (step 4) on the CPU. Step 2 is needed to optimize the data-transfer toward the GPU; it is important at this stage to convert the object-oriented structures to a *plain* buffer containing the minimum amount of information needed by the algorithm to minimize the CPU→GPU communication latency. Step 3 is implemented through libraries dedicated to the client-server communication, as a part of the APE server. Such server manages the assignement of the task to the GPU for the execution and waits to retrieve the results. To accomplish step 4, the simple algorithm which evaluates the calorimeter isolation has been translated into the CUDA language optimizing the management of the GPU resources in terms of computing CUDA cores and memory usage. Step

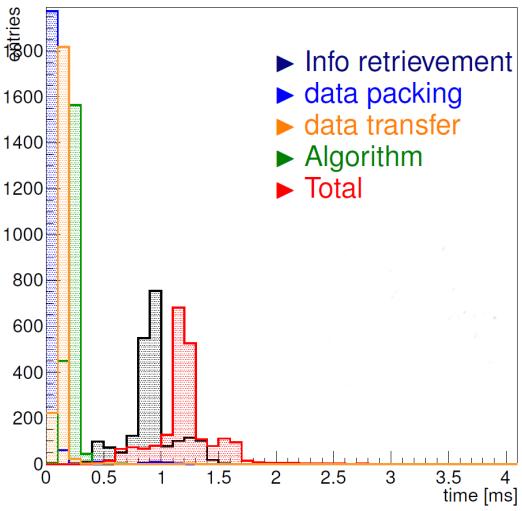

Figure 3: Measurement of the muon isolation algorithm execution time using a Nvidia GTX Titan GPU.

[2]The requirement on this energy fraction varies depending on the region of the detector and the desired quality of the reconstructed muon, but this aspect is not relevant for the purpose of these tests.

5 is implemented within the APE server, similarly to step 3, and completes the CPU-GPU communication, reporting the algorithm results in the original framework.

The measurement of the trigger algorithm execution latency has been performed using a server machine containing an Intel Xeon E5-2620 CPU and a Nvidia GTX Titan GPU, *re*-processing a sample of ATLAS recorded data with no dedicated optimization of the machine. Figure 3 shows the execution latency measured for the several steps and their sum. The overall execution time resulted in being Δt^{tot}_{GPU} ≈1.2±0.2 ms when using the GPU, with respect to Δt^{tot}_{CPU} ≈0.95±0.15 ms, obtained with the standard execution on CPU. As it is shown in Figure 3 the largest fraction of the time ($\sim$ 900 μs) is spent to extract the detector data and convert them from the object-oriented structure to a *flat* format which is suitable for the transfer to the GPU. This contribution to the latency is independent from the the serial or parallel implementation of the algorithm, since it's related to data structure decoding; the current version of the ATLAS framework heavily relies on object-oriented structures, which are not the ideal input for GPUs. The contribution due to the CPU-GPU communication through the client-server structure is found to be $\Delta t^{trans.}_{GPU}$ $\sim$250 μs, which is within the typical time budget of the ATLAS HLT ($\mathcal{O}$(10 ms)). This result confirms the possibility to include GPUs into a HLT system for the execution of parallel algorithms, hence motivates further studies in this direction.

4 Conclusions and Perspectives

From this first study we succesfully deployed a GPU within the ATLAS pre-existing trigger and DAQ framework through a flexible client-server scheme. We observed that the CPU-GPU communication does not introduce a dramatic overhead, which is indeed well within the typical execution latencies of software trigger algorithms, $\mathcal{O}$(10 ms). This result shows that is feasible to consider the GPUs as extremely powerful computing devices for the evolution of the current asynchronous trigger systems. Most of the execution time is devoted to the data extraction from object-oriented structures, which is currently an *external* constraint from the ATLAS trigger framework. This observation, confirmed by similar studies in the ATLAS experiment, focused the attention on this topic; a common effort is ongoing to overcome this problem and benefit at most from the GPU computing power. At the moment two viable approaches are being considered: on one hand it's interesting to consider more complex algorithms that can reduce the time spent for data structure handling to a negligible fraction; on the other hand it is possible to handle *simpler* data structures (raw byte streams from the detector), bypassing most of the currently existing framework. As an example of the first approach the possibility to execute at the trigger level, a refined evaluation of

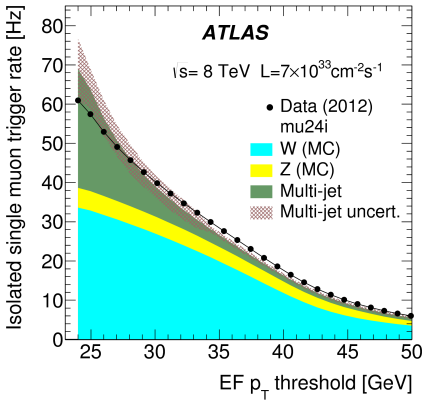

Figure 4: Isolated single muon trigger observed rate as a function of the muon transverse momentum (p_T), compared to simulations for the different expected contributions.

the muon isolation, which would be not sustainable in single-threaded execution on a CPU, is currently under investigation. The muon isolation evaluated in the offline data reprocessing takes into account also environmental corrections due to multiple interactions leading to additional noise in the calorimeter and tracking system. Figure 4 shows the trigger rate as a function of the muon transverse momentum, p_T, for the isolated muon trigger in a data-taking period during 2012. One can see that a relevant fraction of the trigger rate is due to spurious events (multi-jet production). A refined calculation of the muon isolation would reduce such trigger rate, maintaining the efficiency for prompt muon reconstruction high .

As a typical scenario for the second approach, it is possible to decode and process simple data from the muon spectrometer (position and time information) [6] to reconstruct the muon track segment. A similar strategy has been developed in a standalone test considering track reconstruction in the ATLAS Inner Detector, with interesting results [5].

5 Acknowledgements

The GAP project and all the authors of this article are partially supported by MIUR under grant RBFR12JF2Z Futuro in ricerca 2012.

References

[1] Proceeding of *Estimated speed-up from GPU computation for application to realtime medical imaging* from this same workshop (2014).

[2] ATLAS Collaboration, JINST **3** P08003 (2008).

[3] ATLAS Collaboration, -CERN-LHCC-2011-012 (2012).

[4] The client-server structure is obtained using APE, an ATLAS tool developed independently from this project.

[5] D. Emeliyanov, J. Howard, J. Phys.: Conf. Ser. **396** 012018 (2012).

[6] ATLAS Collaboration, JINST **3** S08003 (2008).

[7] http://docs.nvidia.com/cuda/cuda-c-programming-guide