PROCEEDINGS OF SCIENCE

GPGPU for triggering in High Energy Physics Experiments

S.Amerio[1], **M.Belgiovine**[2], **R.Di Sipio**[2], **A.Gabrielli**[2], **A.Gianelle**[1], **D.Lucchesi** [1], **M.Negrini**[3], **L.Rinaldi**[1]*, **F. Semeria**[3], **A.Sidoti**[3], **M.Villa**[2]
1) Padova University and INFN, 2) Bologna University and INFN, 3) INFN-Bologna
E-mail: rinaldi@bo.infn.it, belgiovi@bo.infn.it

Graphical Processing Units (GPUs) provide exceptional massive parallel computing power with small power consumption. General Purpose Computing on GPU (GPGPU) brings high performance computing with off-the-shelf products. However the full exploitation of this new computing paradigm will not be possible if software applications only partially employ massive parallelism. High Energy Physics (HEP) experiments have much to gain adopting this new computing paradigm. The expected gain in performance in reducing the application latency and dealing with the data high throughput increase, will allow to employ systems based on GPGPU for data acquisition, increasing the available computing power with smaller electric power consumption. All these features suggest the application of GPGPU at the trigger level to provide fast decision and high rejection power. In view of possible applications in a trigger system we will show, using realistic examples based on HEP experiments, the improvement in performance on HEP computing applications after tuning and optimization for running on GPUs. The methodology to improve the performance will also be shown together with results using different GPU architectures. Tracking in the bustling LHC environment is very challenging with multiple minimum bias interactions superimposed to the high transferred momentum one. In particular the porting to GPU architecture of two track reconstruction algorithms will be shown: track fitting with the SVT algorithm and track recognition with the Hough Transform. Both pattern recognition and track fitting would benefit from massive parallelism with high throughput processing that can be fully exploited at trigger level.

Technology and Instrumentation in Particle Physics 2014,
2-6 June, 2014
Amsterdam, the Netherlands

*Speaker.

PoS(TIPP2014)408

1. Introduction

Most of the High Energy Physics (HEP) experiments, designed to detect large amount of physics events produced with a very high rate, can profit from the development of new computing paradigms for fast data-processing mainly based on massive parallelisation. Such intense computing power can be provided by General Purpose Computing on Graphical Processing Units (GPGPU) [1]. In this paper we discuss two typical use-cases where a parallel approach is expected to reduce the execution time: a trigger model based on track fitting and a track pattern recognition algorithm based on the Hough transform.

2. Track fitting at trigger level

This track fitting model is based on the CDF Silicon Vertex Trigger (SVT) [2]. SVT performed online trigger decisions in about 20 μs using a 2-steps algorithm: a pattern recognition to identify low resolution tracks (called roads) performed by Associative Memory chips and a linear track fitting with a scalar product on all hits combinations inside each road. The second step is suitable for parallelisation: after unpacking the input data (24-bit words) containing the hits and the associated roads, all the possible combinations of hits are computed for each event and for each road. Then, for each combination, the fit constants are retrieved from the device memory and a scalar product with the hits coordinates is performed and the output tracks are returned. The second step has been ported into the CUDA programming language [3]; first the data are unpacked in parallel, then the data structures used to process the input event data are setup on the GPU memory, depending on the chosen payload size, and for each iteration two functions are executed: the first one determines the number of *hits combination* of possible tracks in each event and the second one performs the fit of *track combinations*. Each *CUDA-threadblock*, at local problem scope, is assigned to a single event workload (e.g. for finding the number of combinations in each road of the event and for fitting each combination for every roads in the event), while at global scope, the total execution is divided in subtasks in terms of number of events to be processed (e.g. for a 100 events payload, 100 *CUDA-threadblocks* are executed).

A first study on the SVT algorithm porting and the comparison to CPU performance are described in [4]. The algorithm timing has been improved using shared memory (on-chip) for shared data within the same *CUDA-threadblocks*, allowing faster memory accesses inside the *CUDA-kernel*, and allocating data with *CUDA-pinned* memory for faster transfers between CPU and GPU. Also, four asynchronous *CUDA-streams* have been used to queue every iteration relative to a single payload, in order to gain a higher parallelism level (e.g. mem. transfers/execution overlapping; addressing iterations on multiple GPUs).

The test has been executed on different devices, like the new NVIDIA k40m [5] (devices details are reported in Table 1), using a dataset of 9000 simulated events. The test results are shown in Figures 1 and 2. The total execution time increases for large event payload (more than 30 events processed together). The event complexity affects the performance for low numbers of tracks. In general, the total execution time is dependent on the device, with a time reduction for the latest GPU boards and faster bus connections, as observed when the pure computation and host-to-device data transfer timing are measured separately (Fig 2).

Device specification	NVIDIA Quadro 600FX	NVIDIA GeForce GTX770	NVIDIA Tesla K20m	NVIDIA Tesla K40m
Performance (Gflops)	354	3213	3542	4291
Mem. Bandwidth (GB/s)	25.6	224.2	208	288
Bus Connection	PCIe2	PCIe3	PCIe2	InfiniBand
Mem. Size (MB)	1024	2048	5120	12228
Number of Cores	96	1536	2496	2880
Clock Speed (MHz)	350	1046	706	1502

Table 1: Computing resources setup

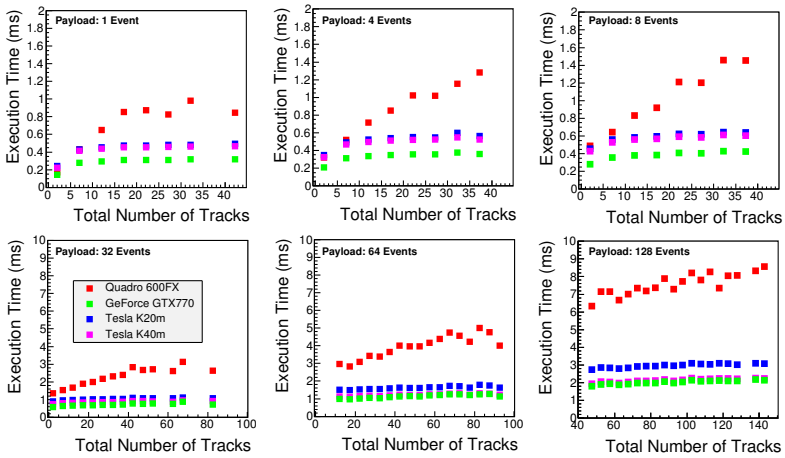

Figure 1: Total execution time as a function of the number of tracks found by the algorithm, for different event payloads and for several NVIDIA GPU boards

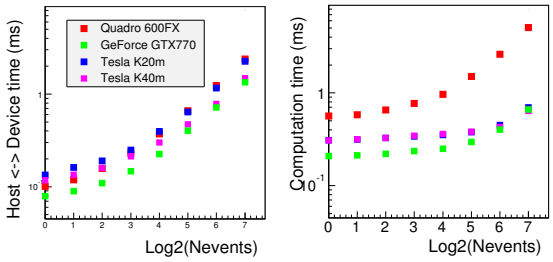

Figure 2: Host-Device data transfer time (left) and pure computation time (right) as a function of the number of processed events.

3. Track recognition using the Hough Transform

The Hough Transform [6, 7, 8] is a pattern recognition technique for feature extraction in image processing. In our model, a typical cylindrical detector for inner tracking composed by 12 layers has been considered: the Hough Transform has been used for the extraction of the tracks curvature from simulated tracking detector signals. The test has been carried out using a dataset of 1000 simulated events, each containing up to 5000 hits. The timing measurements have been performed as a function of the number of hits. A circle detection Hough algorithm has been developed, the Hough parameters being the curvature ρ and the track angle ϕ (constraining the tracks to the nominal vertex). In addition, the algorithm performed the track searching in smaller sectors along the longitudinal (θ) and transverse (φ) regions. For the CUDA implementation, the Hough

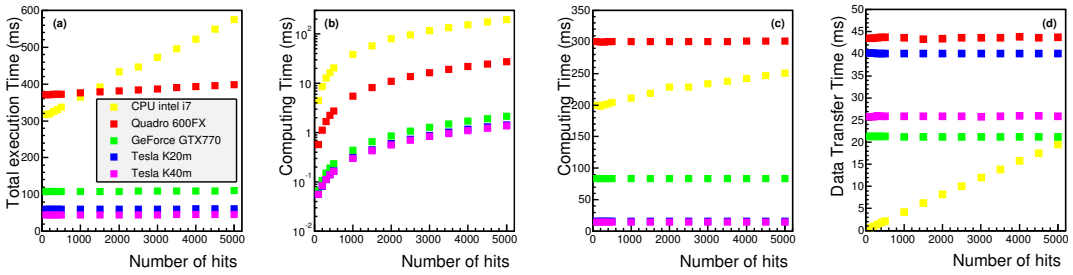

Figure 3: Timings as a function of the number of hits: (a) total execution time, (b) voting procedure, (c) relative maxima search, (d) host-to-device data transfer time (CPU time is the data input reading time).

matrices have been initialised at first iteration only; the parameter vote has been performed assigning each hit to a block and the Hough parameters to the threads, using shared memory allocation within each thread. Instead, for the local maxima search, longitudinal (16 bins) and transverse (4 bins) sectors form the *CUDA-grid* and the *CUDA-threadblocks* run over ρ (1024 bins) and ϕ (1024 bins), using the *CUDA-pinned* memory to speedup the the output transfer. The results of the test are shown in Fig. 3: compared to a traditional CPU, the total execution time on GPUs is faster up to a factor 15. For the voting procedure, the speed-up increases of two orders of magnitude. As expected, the slower parts of the execution are the data transfer between host and device and the local maxima search, because of an irreducible serial component. Also in this test, the execution times show a remarkable dependence on the GPU devices and their bus connections to hosts.

4. Conclusions

Two algorithms for tracking reconstruction and track pattern recognition with the Hough transform have been implemented on GPU devices. In both cases, a dramatic reduction of execution time has been observed. The obtained timing performance are encouraging in view of a future use at trigger level. Good performance have been obtained on pure computational algorithms, although the execution time is driven mainly by data transfer between CPU and GPU and by the GPU board specifications and bus connection, and therefore the best timing performance has been achieved using the latest GPUs (NVIDIA k40m) and PCIe3 [9] bus.

References

[1] NVidia Corporation URL http://www.nvidia.com/object/gpu.html

[2] W A Ashmanskas et al, 2004 *Nucl. Instrum. Methods Phys. Res., Sect. A* **518** p 532

[3] NVidia Corporation URL http://www.nvidia.com/object/cuda_home_new.html

[4] S Amerio et al; 2014 *J. Phys. Conf. Ser.* **513** 012002

[5] NVidia Corporation URL http://www.nvidia.com/

[6] P. Hough, Machine analysis of Bubble Chamber Pictures; 1959 *Proc. Int. Conf. High Energy Accelerators and Instrumentation* **C590914** 554âĂŞ558
P. Hough, Method and mean for recognizing complex patterns, 1962, United States Patent 3069654.

[7] V Halyo et al; 2014 *JINST* **9** P04005

[8] M R Buckley, V Halyo, P Lujan; 2014 http://arxiv.org/abs/1405.2082

[9] PCI-SIG URL http://www.pcisig.com/specifications/pciexpress/base3/