EPJ Web of Conferences **127**, 00011 (2016) DOI: 10.1051/epjconf/201612700011
Connecting The Dots 2016

Graphics Processors in HEP Low-Level Trigger Systems

Roberto **Ammendola**[1], Andrea **Biagioni**[2], Stefano **Chiozzi**[5], Angelo **Cotta Ramusino**[5], Paolo **Cretaro**[2], Stefano **Di Lorenzo**[3,4], Riccardo **Fantechi**[3], Massimiliano **Fiorini**[5,6], Ottorino **Frezza**[2], Gianluca **Lamanna**[8], Francesca **Lo Cicero**[2], Alessandro **Lonardo**[2], Michele **Martinelli**[2], Ilaria **Neri**[5], Pier Stanislao **Paolucci**[2], Elena **Pastorelli**[2], Roberto **Piandani**[3], Luca **Pontisso**[3,a], Davide **Rossetti**[7], Francesco **Simula**[2], Marco **Sozzi**[3,4], and Piero **Vicini**[2]

[1] *INFN Sezione di Roma Tor Vergata, Italy*
[2] *INFN Sezione di Roma, Italy*
[3] *INFN Sezione di Pisa, Italy*
[4] *Università di Pisa, Italy*
[5] *INFN Sezione di Ferrara, Italy*
[6] *Università di Ferrara, Italy*
[7] *NVIDIA Corp., Santa Clara, CA, USA*
[8] *INFN Laboratori Nazionali di Frascati, Italy*

Abstract. Usage of Graphics Processing Units (GPUs) in the so called general-purpose computing is emerging as an effective approach in several fields of science, although so far applications have been employing GPUs typically for offline computations. Taking into account the steady performance increase of GPU architectures in terms of computing power and I/O capacity, the real-time applications of these devices can thrive in high-energy physics data acquisition and trigger systems. We will examine the use of online parallel computing on GPUs for the synchronous low-level trigger, focusing on tests performed on the trigger system of the CERN NA62 experiment. To successfully integrate GPUs in such an online environment, latencies of all components need analysing, networking being the most critical. To keep it under control, we envisioned NaNet, an FPGA-based PCIe Network Interface Card (NIC) enabling GPUDirect connection. Furthermore, it is assessed how specific trigger algorithms can be parallelized and thus benefit from a GPU implementation, in terms of increased execution speed. Such improvements are particularly relevant for the foreseen Large Hadron Collider (LHC) luminosity upgrade where highly selective algorithms will be essential to maintain sustainable trigger rates with very high pileup.

1 Introduction

In High Energy Physics experiments the realtime selection of the most interesting events is of paramount importance because of the collision rates which do not give the possibility to save all the data for offline analysis. For this purpose, different trigger levels are usually used to carefully choose the most meaningful events. Low levels require low and (almost) deterministic latency, and their

[a] e-mail: luca.pontisso@pi.infn.it

standard implementation is on dedicated hardware (ASICs or FPGAs). Our approach aims at exploiting the Graphics Processing Units (GPUs) computing power, in order to build refined physics-related trigger primitives, such as energy or direction of the final state particles in the detectors, and therefore leading to a net improvement of trigger conditions and data handling. GPUs architectures are massively parallel, being designed to optimize computing throughput but with no particular attention to their usage in real-time contexts, such as the online low-level triggers. While execution times are rather stable on these devices, also I/O tasks have to guarantee real-time features along the data stream path, from detectors to GPU memories. The NaNet project arises with the goal of designing a low-latency and high-throughput data transport mechanism for systems based on CPU/GPUs. The GPU-based low-level trigger using the NaNet board is currently integrated in the experimental setup of the Ring-imaging Čerenkov detector (RICH) of the NA62 experiment in order to reconstruct the ring-shaped hit patterns. We report and discuss results obtained with this system along with the algorithms that will be implemented.

2 NaNet

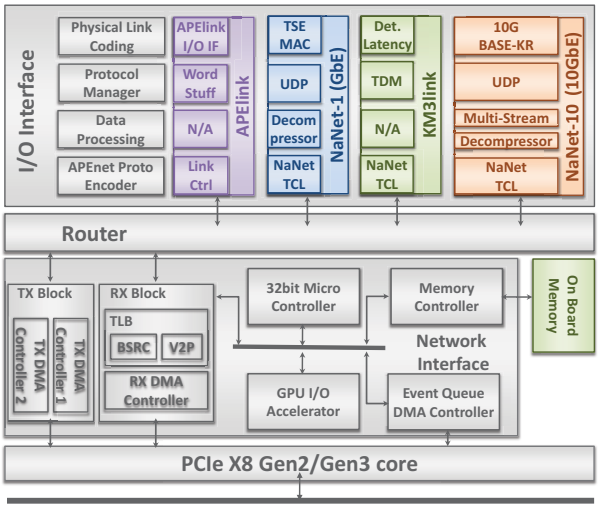

Figure 1. NaNet PCIe Network Interface Card family architecture.

The NaNet project goal is the design and implementation of a family of FPGA-based PCIe Network Interface Cards, equipped with different network links according to their employment, to bridge the detectors' front-end electronics and the computing nodes [1]. To this pourpose NaNet features (i) a low-latency and high-throughput data transport mechanism for real-time systems, (ii) support for several link technologies to be able to fit in different experimental setups, and (iii) a design with multi-port interface to increase the scalability of the entire system to reduce the number of nodes in the farm clusters. The management of custom and standard network protocols is performed in hardware to avoid jitter effects (variance in tasks' execution time) due to operating system and to guarantee a deterministic behaviour of communication latency while achieving the maximum capability out of the adopted channel. Furthermore, NaNet integrates a processing stage which is able to process data coming from detectors on the fly, to improve the efficiency of applications running on computing nodes.

On a per-experiment basis, different solutions can be implemented: data decompression, data reformatting, merging of event fragments. Finally, data transfers to or from application memory in CPU or GPU are performed with zero-copy. The block diagram of the NaNet architecture is in Figure 1. Exploiting the modularity of the hardware design, the Network Interface and Router modules are shared among all boards of the family, ensuring fast and reliable implementation of different NaNet design configurations. The PCIe interface can be customized — *i.e.* Gen2 or Gen3 — and the number or switch ports chosen, to best match experimental requirements. The I/O interface, one of the key features of the NaNet design, consists of 4 elements: Physical Link Coding, Protocol Manager, Data Processing, APEnet Protocol Encoder. The Physical Link Coding covers the Physical and DataLink Layers of the Open Systems Interconnection (OSI) model, managing transmission of data frames over a common local media and performing error detection. The Protocol Manager covers the Network and Transport Layers, managing the reliability of communication data flow control. A processing stage, Data Processing, that applies some function to the data stream in order to offload some work for the applications running on the computing node. Finally, the APEnet Protocol Encoder performs a protocol translation to a format more suited for PCIe Direct Memory Access (DMA) transactions.

NaNet accomplishes zero-copy networking by means of a hardware implemented RDMA (Remote DMA) engine working with both CPUs and nVIDIA GPUs (GPUDirect V2/RDMA [2]), achieving very low I/O latency.

On the host side, a dedicated Linux kernel driver offers its services to an application-level library, which provides the user with a series of functions to: open/close the NaNet device; register and de-register circular lists of persistent data receiving buffers (CLOPs) in GPU and/or host memory; manage software events generated when a receiving CLOP buffer is full (or when a configurable timeout is reached) and received data are ready to be processed.

NaNet-1 was developed in order to verify the feasibility of the project; it is a PCIe Gen2 x8 network interface card featuring GPUDirect RDMA over GbE.

NaNet-10 is a PCIe Gen2 x8 network adapter implemented on the Terasic DE5-net board equipped with an Altera Stratix V FPGA featuring four enhanced small form-factor pluggable (SFP+) cages [3].

Both implementations use UDP as their transport protocol.

In Figure 2, NaNet-10 and NaNet-1 latencies are compared within UDP datagram size range; NaNet-10 guarantees sub-µs hardware latency for buffers up to ≈1 kB in GPU/CPU and it reaches its 10 Gbit/s bandwidth peak capability already at ≈1 kB size (Figure 3).

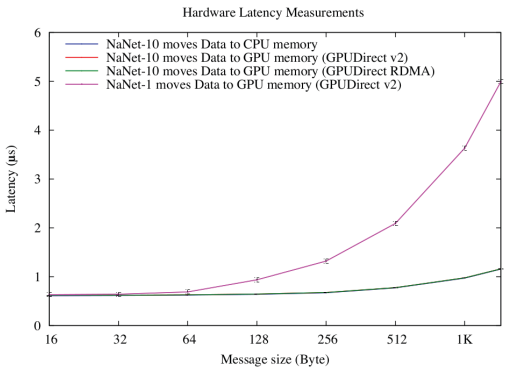

Figure 2. NaNet-10 vs. NaNet-1 hardware latency. NaNet-10 curves are completely overlapped at this scale.

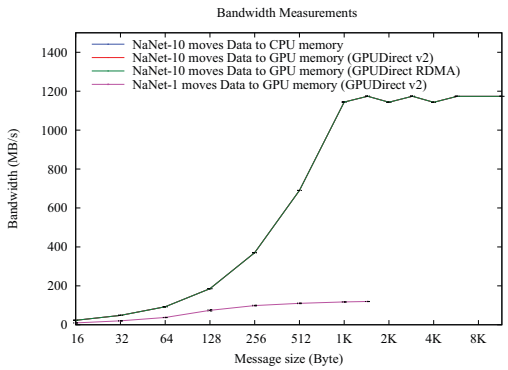

Figure 3. NaNet-10 vs. NaNet-1 bandwidth. NaNet-10 curves are completely overlapped at this scale.

3 GPU-based L0 trigger for the NA62 RICH detector

In order to research the feasibility of a GPU-based L0 trigger system for NA62 experiment, we directed our attention on reconstructing the ring-shaped hit patterns in the RICH of the NA62 experiment. This detector identifies pions and muons with momentum in the range between 15 GeV/c and 35 GeV/c. Čerenkov light is reflected by a composite mirror with a focal length of 17 m focused onto two separated spots equipped with ≈ 1000 photomultipliers (PM) each. Data communication between the readout boards (TEL62) and the L0 trigger processor happens over multiple GbE links using UDP streams. The final system consists of 4 GbE links to move primitives data from the readout boards to the GPU_L0TP (see Figure 4). The overall time budget for the low-level trigger comprising both communication and computation tasks is of 1 ms, so a deterministic response latency from GPU_L0TP is a strict requirement. Refined primitives coming from the GPU-based calculation will be then sent to the central L0 processor, where the trigger decision is made taking in account informations from other detectors.

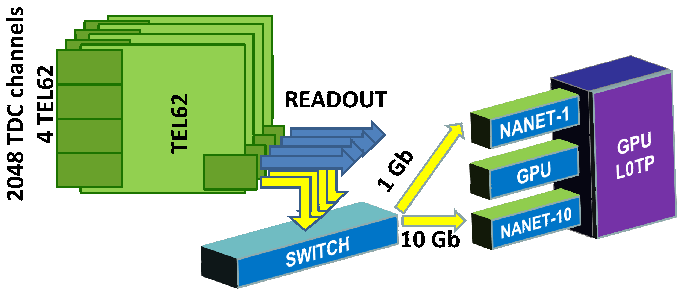

Figure 4. Pictorial view of GPU-based trigger.

3.1 Reconstructing multiple ring events on GPUs

The knowledge of Čerenkov rings parameters allows to build stringent conditions for data selection at trigger level. This implies that circles have to be reconstructed using the coordinates of activated PMs. We take in consideration two multi-rings pattern recognition algorithms based only on geometrical considerations (no other information is available at this level) and particularly suitable for exploiting the intrinsic parallel architecture of GPUs.

3.1.1 Histogram algorithm

The procedure involves dividing the xy-plane into a grid and creating a histogram whose bins contain distances from the gridpoints and the hits of the physics event. Distance bins whose contents exceed a threshold value let identify the rings. In order to limit the use of resources, it is possible to proceed in two steps, starting the histogram procedure with a 8x8 grid and calculating now distances from such squares. Afterwards, to refine their positions, the calculation is repeated with a grid 2x2 only for the candidate centers selected according to the threshold in the previous step.

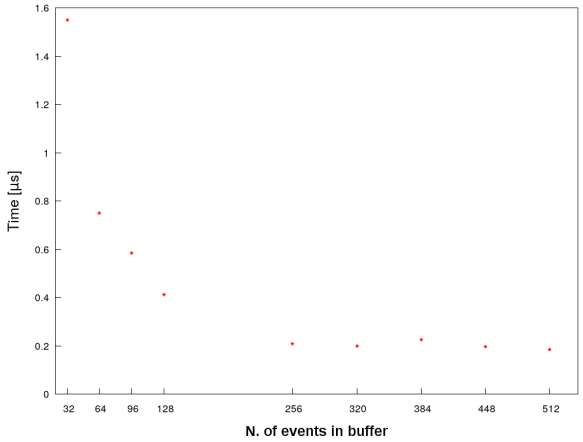

Figure 5. Almagest algorithm performances, time for single event. Test performed on K20c nVIDIA GPU.

3.1.2 Almagest algorithm

One of the Ptolemy's theorems described in the *Almagest*, his treatise on astronomy, states that when four vertices of a quadrilateral (ABCD) lie on a common circle, it is possible to relate four sides and two diagonals: $|AC| \times |BD| = |AB| \times |CD| + |BC| \times |AD|$. By using this formula it is possible to implement a pattern recognition algorithm for multi-rings which exposes different level of parallelism, resulting well-suited for GPUs achitecture and fast in its execution. This is crucial either to directly reconstruct the rings or to choose different algorithms according to the number of circles. The large number of possible combinations of four vertices, given a maximum of 64 points for physics event, can be a limitation to this approach. To greatly reduce the number of tests, one possibility is to choose few triplets — *i.e.* a set of three hits assumed to belong to the same ring — trying to maximize the probability that all their points belong to the same ring and iterating through all the remaining hits to search for the ones satisfying the aforementioned formula [4]. The parallel implementation of this algorithm yields many triplets and events being processed at the same time. Some results are shown in Figure 5, where the computing time is further sped up by greatly reducing accesses to GPU shared memory, mainly using threads private registers through CUDA (a parallel computing platform and programming model invented by nVIDIA) intra-warp veto and shuffle instructions, so that multi-rings events are processed with a latency of 0.5 µs per event. Once the number of rings and points belonging to them have been found, it is possible to apply for example Crawford's method [5] to obtain centre coordinates and radii with better spatial resolution.

4 Results for the GPUbased L0 trigger with NaNet-1

In 2015 the GPU-based trigger at CERN includes 2 TEL62 boards connected to a HP2920 switch and a NaNet-1 [6] board with a TTC HSMC daughtercard plugged into a server made of a X9DRG-QF dual socket motherboard populated with Intel Xeon E5-2620 @2.00 GHz CPUs (*i.e.* Ivy Bridge architecture), 32 GB of DDR3 RAM and a Kepler-class nVIDIA K20c GPU.

Such a system allows testing of the whole chain: the data events move towards the GPU-based trigger through NaNet-1 by means of the GPUDirect RDMA interface. All data landing within a

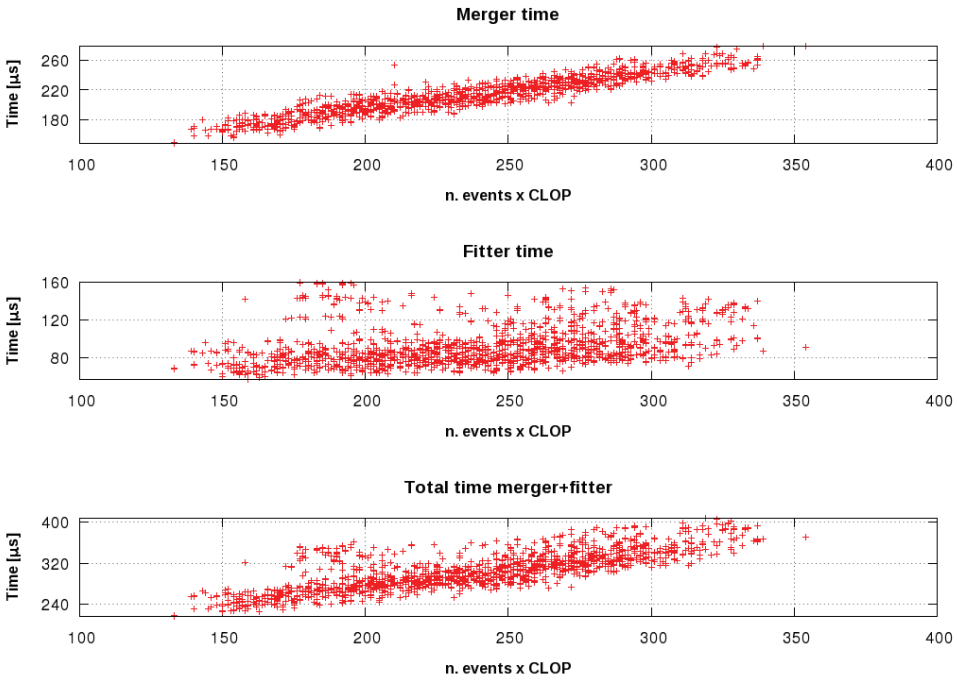

Figure 6. Multi-ring reconstruction of events performed on K20c nVIDIA GPU.

configurable time frame are gathered and organized in a parametrizable Circular List Of Persistent buffers (CLOP) in GPU memory. This time frame must be always shorter or equal to how long multi-ring reconstruction takes on the GPU, to be sure that buffers are not overwritten before they are consumed.

Results are reported in Figure 6. On the x-axis: the CLOP size measured as number of received events; on the y-axis: the latencies of different stages. The computing kernel implemented the histogram fitter with a single step (*i.e.* using an 8x8 grid only). Events coming from two readout boards, for a gathering time of 400 µs, and parameters like events rate (collected with a beam intensity of 4×10^{11} protons per spill), a CLOP's size of 8 kB, time frame were chosen so that we could test the online behaviour of the trigger chain. The merge operation doesn't expose much parallelism, requiring instead synchronization and serialization. As a result it is an ill-suited problem to the GPU architecture. In operative conditions, the merging time only would exceed the time frame. The high latency of the this task when performed on a GPU suggests to offload such duty to a hardware implementation.

Acknowledgements

S. Chiozzi, A. Cotta Ramusino, S. Di Lorenzo, R. Fantechi, M. Fiorini, I. Neri, R. Piandani, L. Pontisso, M. Sozzi thank the GAP project, partially supported by MIUR under grant RBFR12JF2Z "Futuro in ricerca 2012".

References

[1] A. Lonardo et al., JINST **10**, C04011 (2015)
[2] R. Ammendola et al., *GPU Peer-to-Peer Techniques Applied to a Cluster Interconnect*, in *Proceedings of the IEEE 27th International Parallel and Distributed Processing Symposium Workshops and PhD Forum (IPDPSW 2013)* (2013), pp. 806–815, http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6650959
[3] R. Ammendola et al., JINST **11**, C03030 (2106)
[4] G. Lamanna, Nucl. Inst. & Meth. A **766**, 241 (2014)
[5] J. Crawford, Nucl. Inst. & Meth. **211**, 223 (1983)
[6] R. Ammendola et al., JINST **9**, C02023 (2104)