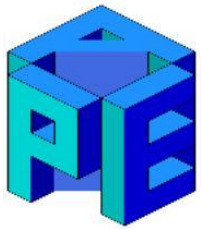



GPU for online reconstruction

Luca Pontisso (INFN Roma – APE Lab)

on behalf the NaNet collaboration

XIV International Workshop on Hadron Structure and Spectroscopy 2017

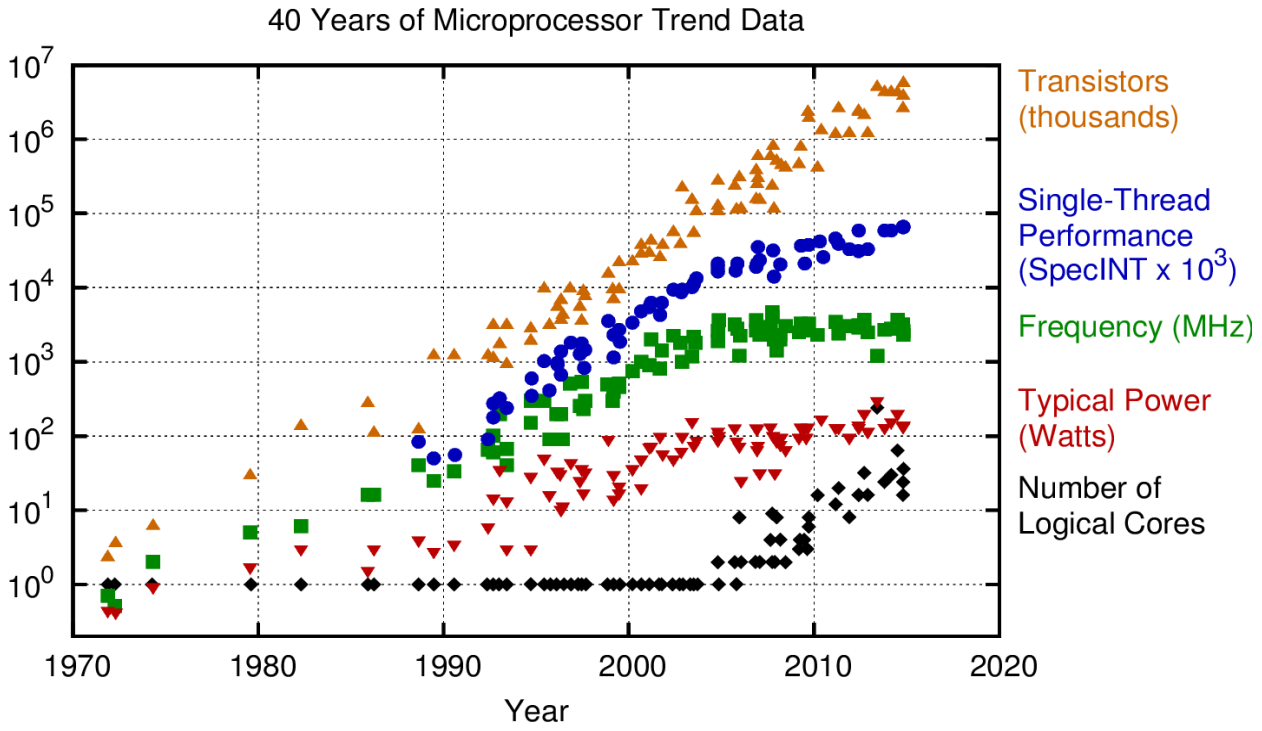
Cortona (Italy), 2 - 5 April 2017

Overview

- Quick glance to Heterogenous computing: CPU+Graphics Processing Unit+ Field-Programmable Gate Array
- The Graphic Processing Units
- Helping (maybe) triggers in the future upgrades of LHC
- Physics case:
  GPU based L0 trigger for NA62 RICH detector
- Algorithms implemented
  - Histogram
  - Almagest
- Work in progress and concluding remarks

Cheating the Moore's Law

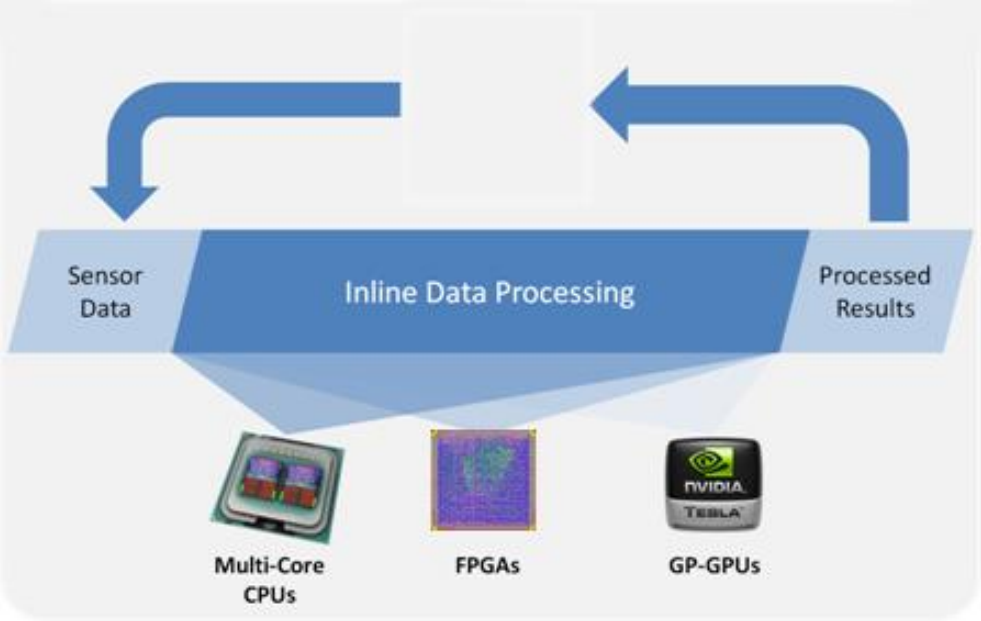
Most machines now, from laptops to tablets to smartphones, have heterogeneous architectures (multi-core processor, a GPU, dedicated processors – possibly integrated in a MPSoC), and more heterogeneity is expected in the near future.



Original data up to the year 2010 collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond, and C. Batten
New plot and data collected for 2010-2015 by K. Rupp

Going heterogeneous

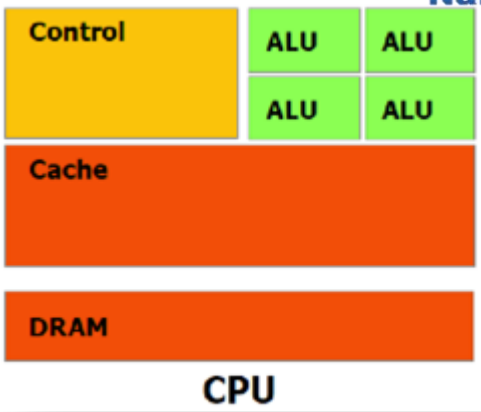
Using specialized computing units (CPU, GPU, DSP, FPGA…) for different kind of tasks…

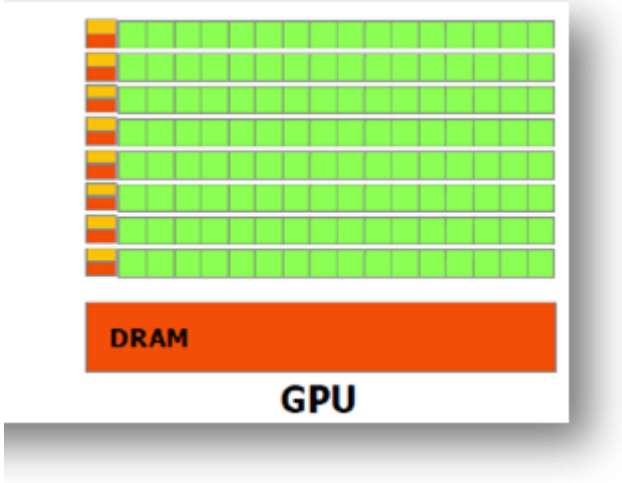


...is a solution required for a growing number of applications including services under Real-Time performance constraints such as vision based systems, mobile devices, AR/VR environments, and servers.

Going heterogeneous

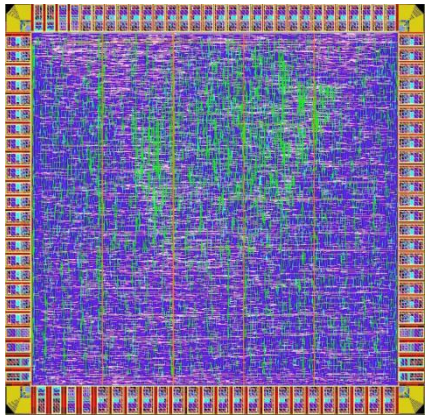
CPU cores have large data cache and they are optimized for efficient execution of general-purpose control code with low memory latency.





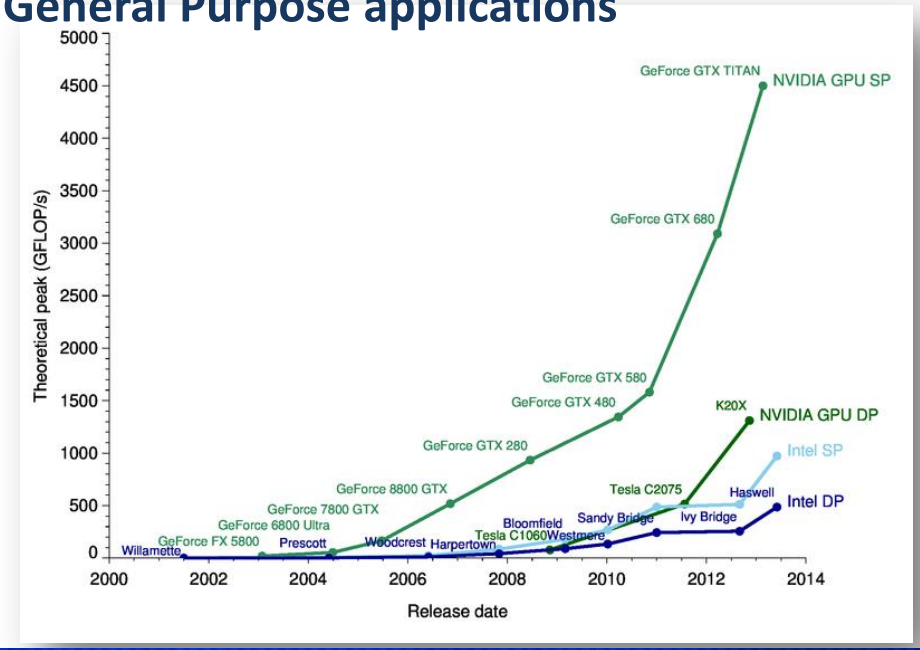
- Based on a massively parallel architecture
- Thousands of cores to process parallel workloads efficiently
- Less control units, many more Arithmetic Logic Units (ALU).

A field-programmable gate array (FPGA) is an integrated circuit that can be programmed or reprogrammed to the required functionality or application after manufacturing.

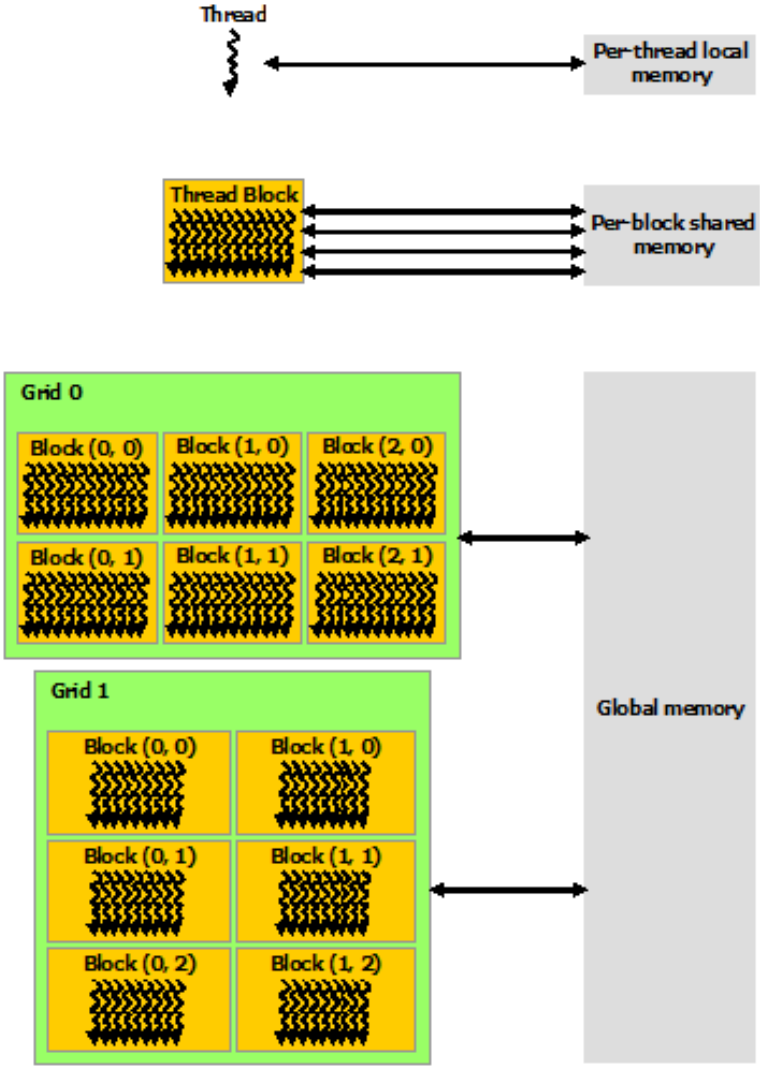


Graphics Processing Unit (GPU)

- GPU's advanced capabilities were originally used primarily for 3D game rendering (OpenGL)
- Still someone implemented non-graphical applications using OpenGL (see Z. Fodor et al. "Lattice QCD as a video game" - 2006)
- Since 2007 high-Level programming languages (CUDA, OpenCL) have been introduced, life became much easier.
- Now this devices are largely deployed in General Purpose applications (GPGPU)
- Thousands of cores to process parallel workloads efficiently
- Faster evolution with respect to traditional CPU
- Easy to have a desktop PC with TERAFLOPS of computing power, with thousands of cores.



GPU memory hierarchy



The memory hierarchy is fundamental in GPU programming

Global Memory

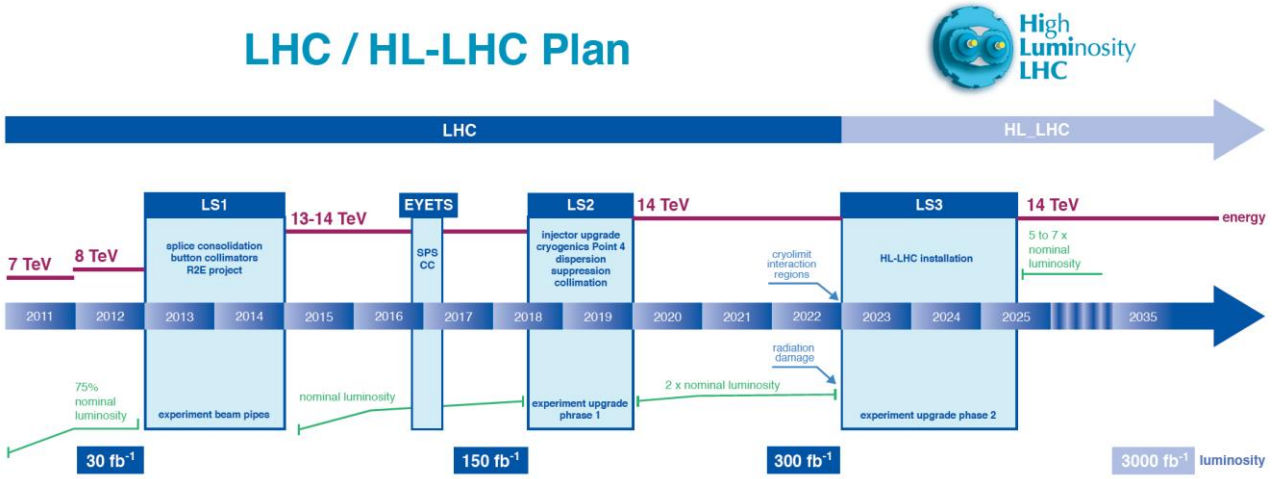
On board, relatively slow, lifetime of the application, accessible from host and device

Shared memory/registers

On Chip, very fast, lifetime of blocks/threads, accessible from kernel only

Data alignment and memory access pattern are crucial to achieve high memory bandwidth

Not too distant future challenges



Standard trigger requirements

- High reduction factor
- High efficiency for interesting events
- Fast decision
- High resolution

The higher background and Pile Up will limit the ability to trigger on interesting events

The primitives will be more complicated with respect today: tracks, clusters, rings

Higher energy

Resolution for high pt leptons → high-precision primitives

High occupancy in forward region → better granularity

Higher luminosity

Track-calo correlation

Bunch crossing ID becomes challenging, pile up

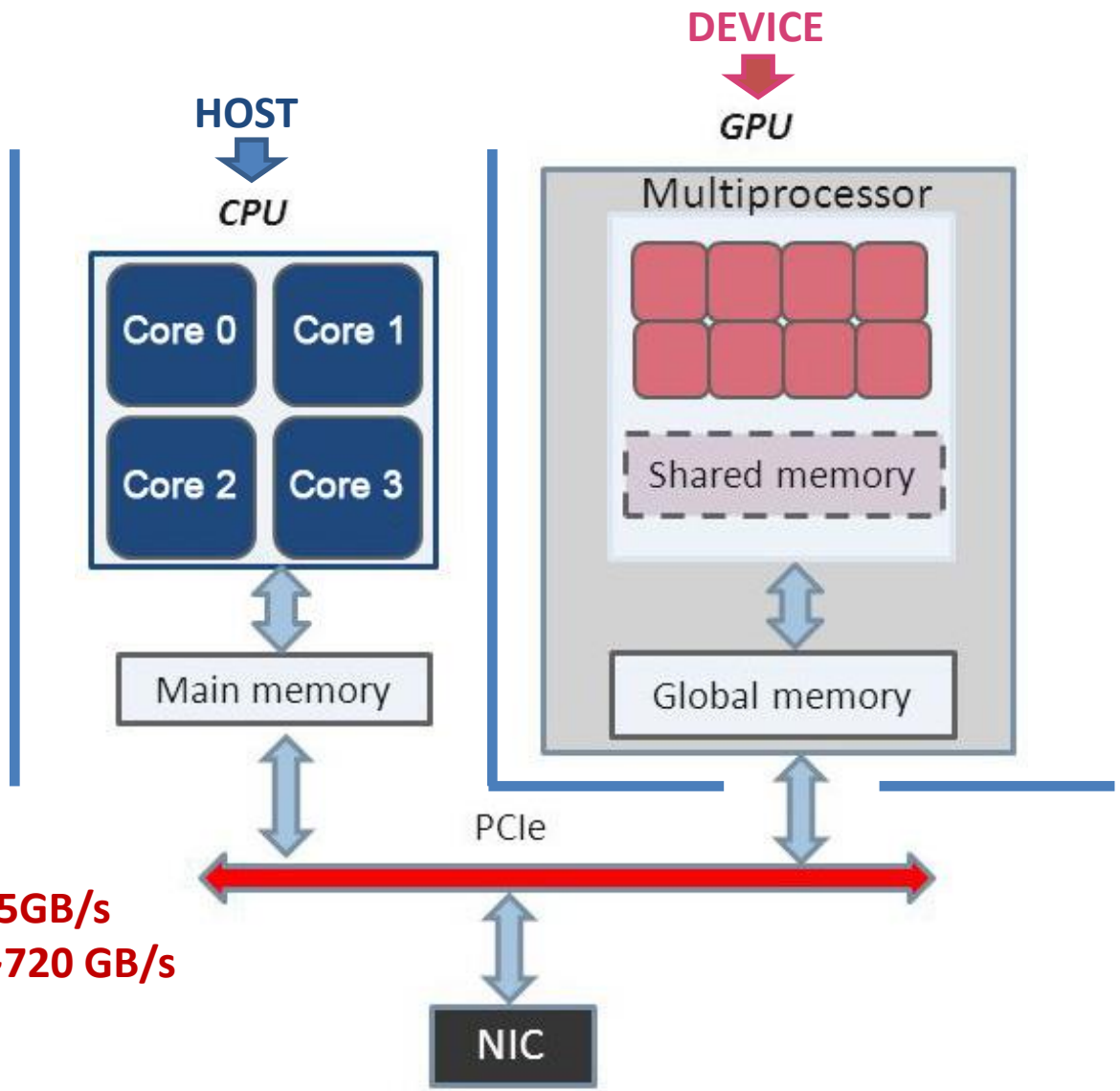
All of these effects go in the same direction

More resolution & more granularity → more data & more processing

Heterogeneous computing could help… with caveats… (1)

Transport problem

- Data must be moved/copied between Host and Device
- Data coming from the network are received from the Network Interface Card connected to the Host (again data to be precessed must reach the Device memory)



Bandwidth
Host memory (DDR4-3200)~25GB/s
GPU memory (nVidia Pascal)~720 GB/s
PCIe v.3 16x ~16GB/s

Heterogeneous computing could help... with caveats (2)

Low level triggers are synchronous environments
hard realtime online computing
(data cannot be lost)

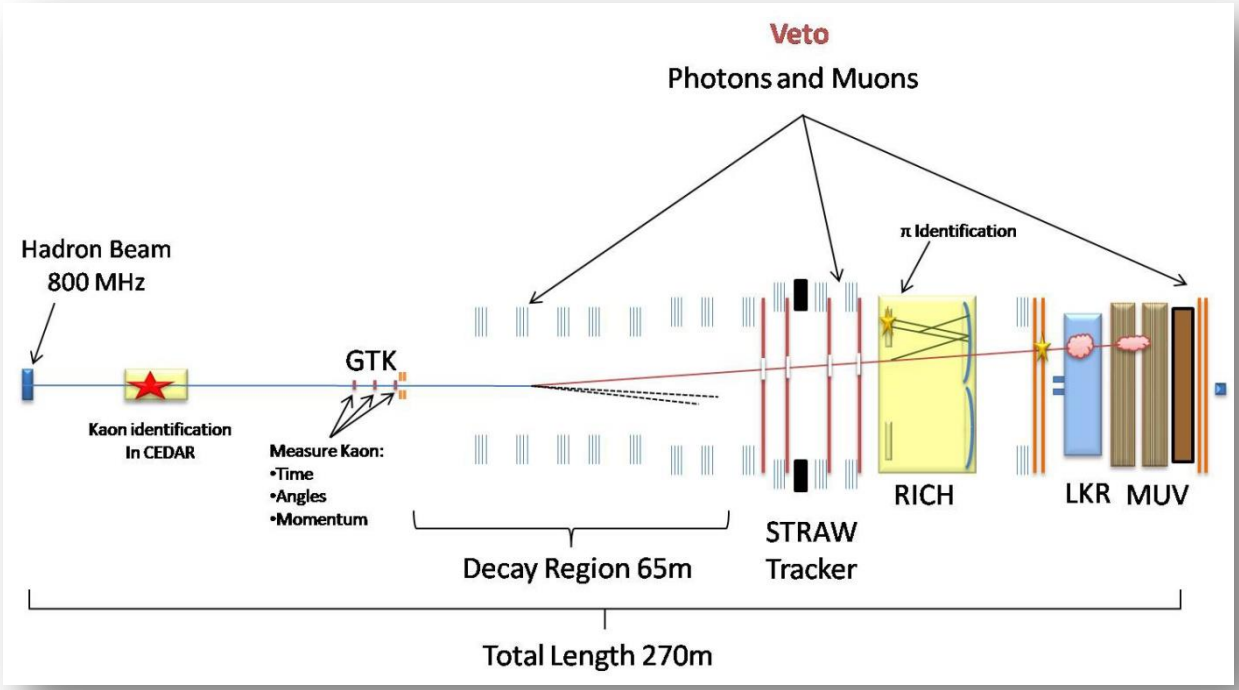
High event rate (tens of MHz)

are GPUs fast enough to take trigger decision at tens of MHz events rate?

Tiny latency

Is the GPU latency per event small enough to cope with the tiny latency of a low level trigger system? Is the latency stable enough for usage in synchronous trigger systems?

Physics case: Low Level Trigger system in NA62



- Measurement of ultra-rare decay

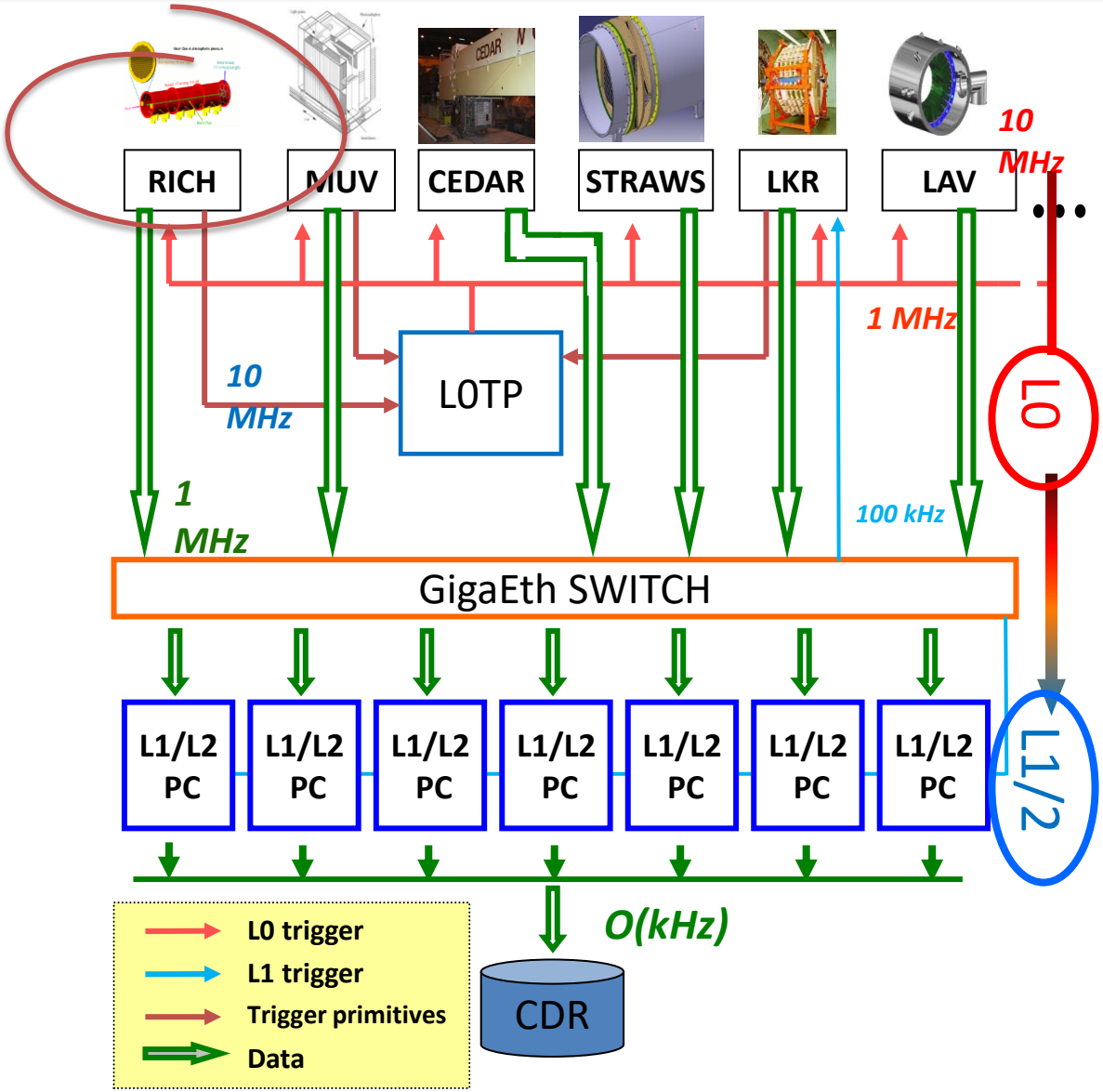
$$K^+ \to \pi^+ \nu \bar{\nu}$$

(BR ~8×10^{-11})

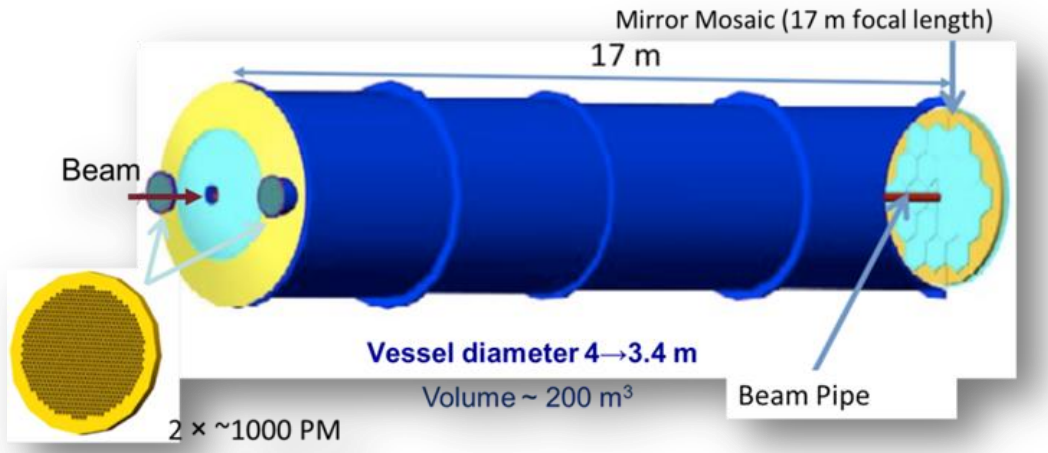
- Kaon decays in flight
- High intensity unseparated hadron beam (6% kaons)

- L0 trigger: synchronous level must reduce rate from 10 MHz to 1 MHz
  - 1 ms max. latency

NA62 DAQ and Trigger

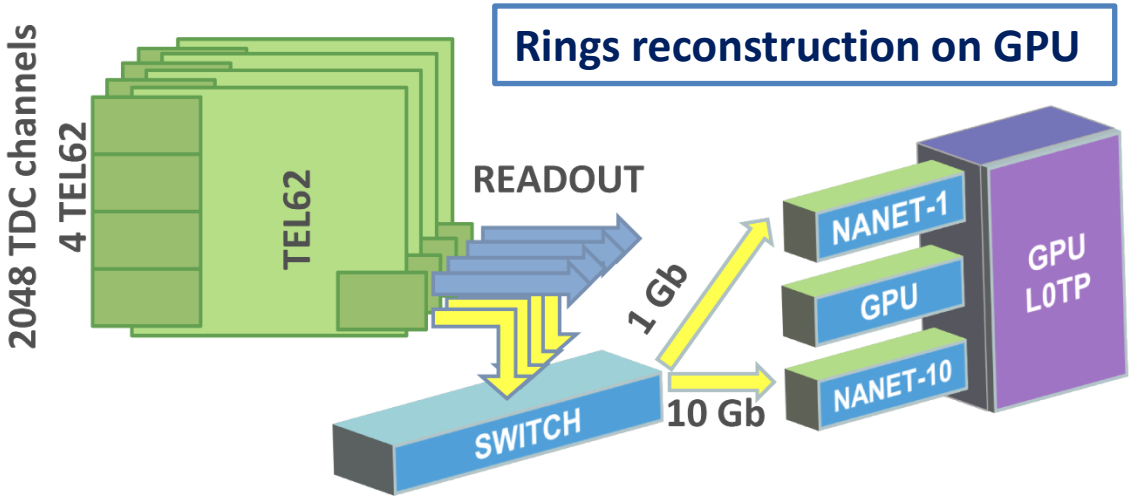


GPU-based L0 trigger general scheme

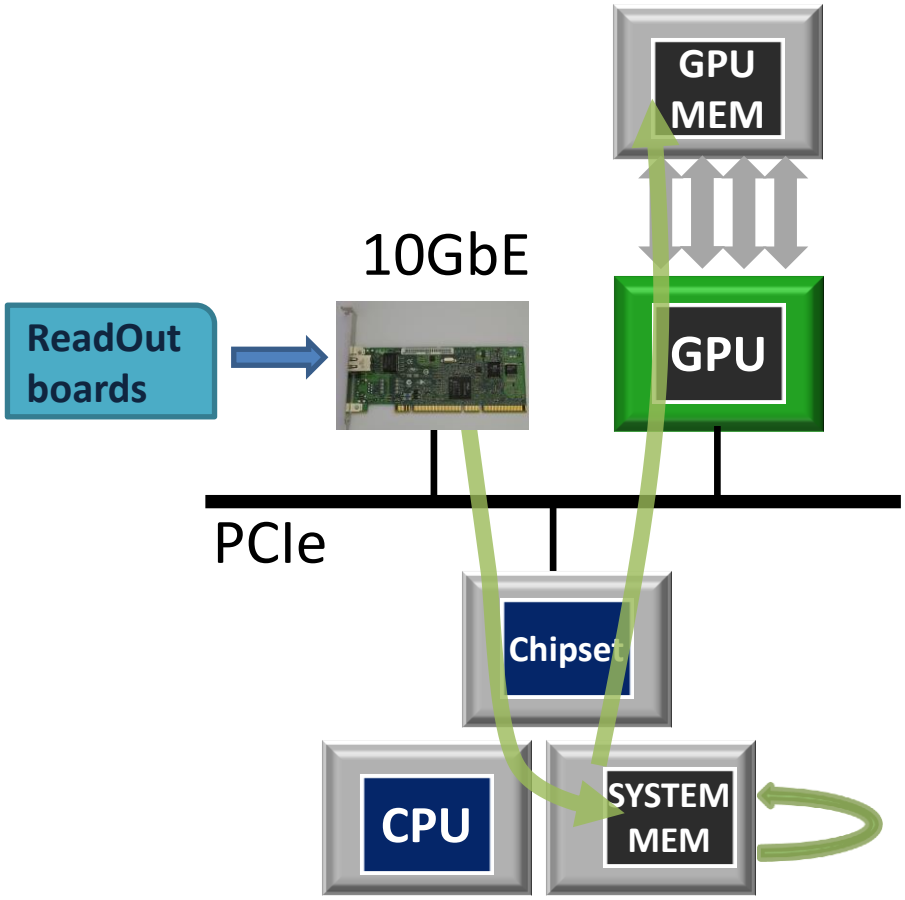


NA62 Rich detector

- Distinguish between pions and muons from 15 to 35 GeV (inefficiency < 1%)
- 2 spots of 1000 PMs each
- 2 read-out boards for each



Minimize Latency: Data Transfer



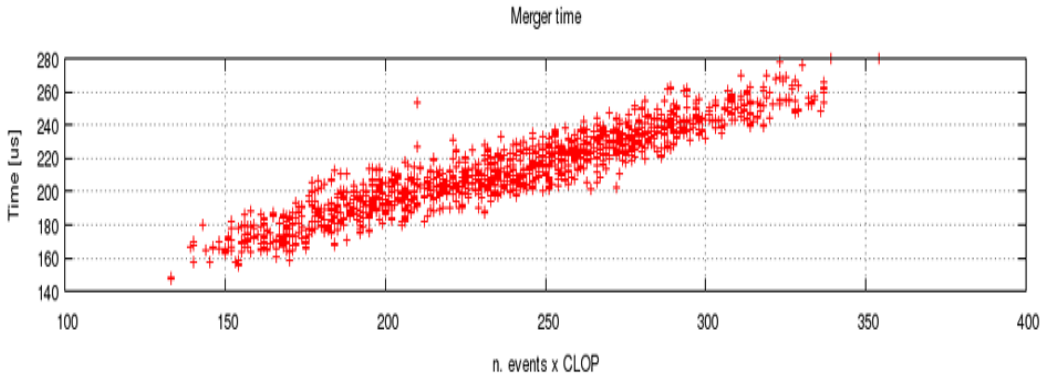
Commercial Network Interface Cards

- Data are copied in a kernel buffer then to destination buffer in user space (on CPU)
- Data are decompressed in a GPU-friendly format (on CPU)
- Data are copied from CPU memory to GPU memory

How to reduce data transfer time:

- Perform decompression on data stream on the NIC!
- DMA (Direct Memory Access) from the network channel directly in GPU memory!

Minimize Latency: Merge Events Split in Four Streams

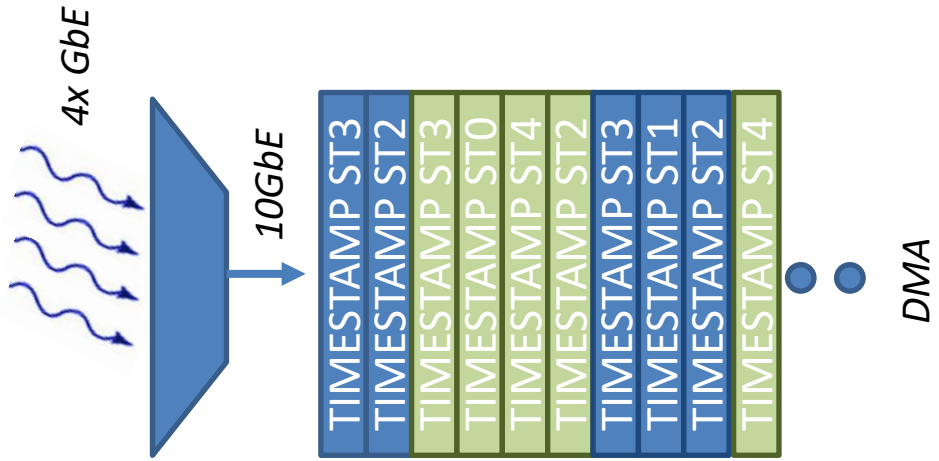


Merging the events coming from the RICH on 4 streams on GPU…

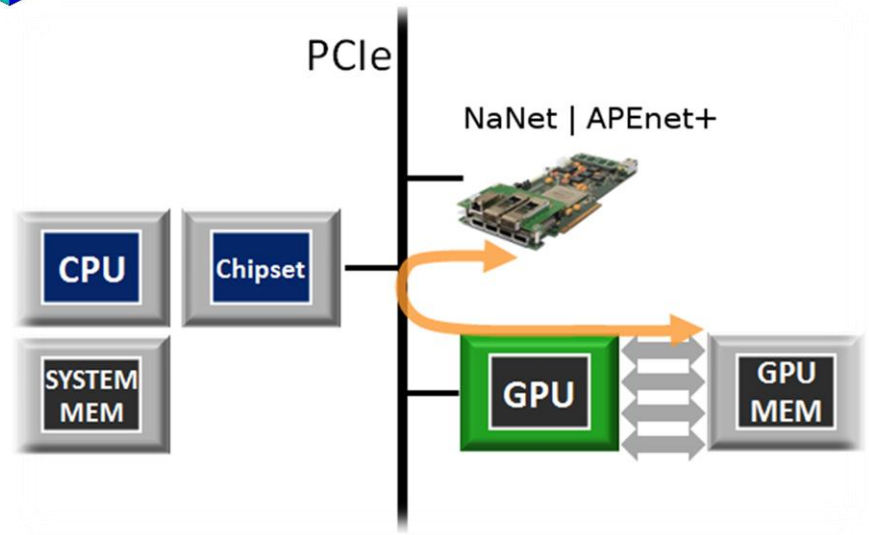
NOT the right way!

It requires frequent synchronization operations and serialization

Implement an Event Merger Stage on the Network Card!



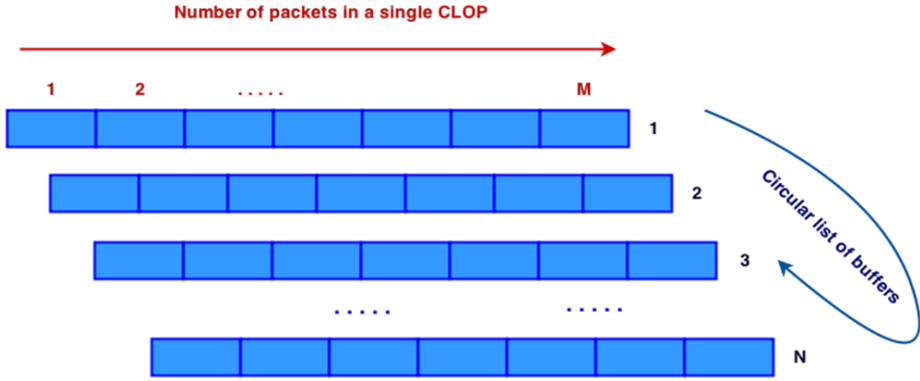
NaNet: a PCIe NIC family for HEP



Developed at INFN Roma APE Lab

- FPGA-based PCIe Network NIC with real-time data transport architecture
- GPUDirect allows direct data exchange on the PCIe bus with no CPU involvement
- No bounce buffers on host memory
- Zero copy I/O
- nVIDIA Fermi/Kepler/Maxwell/Pascal

- Receiving buffers on GPU arranged in a Circular List Of Persistent (CLOP) buffers.

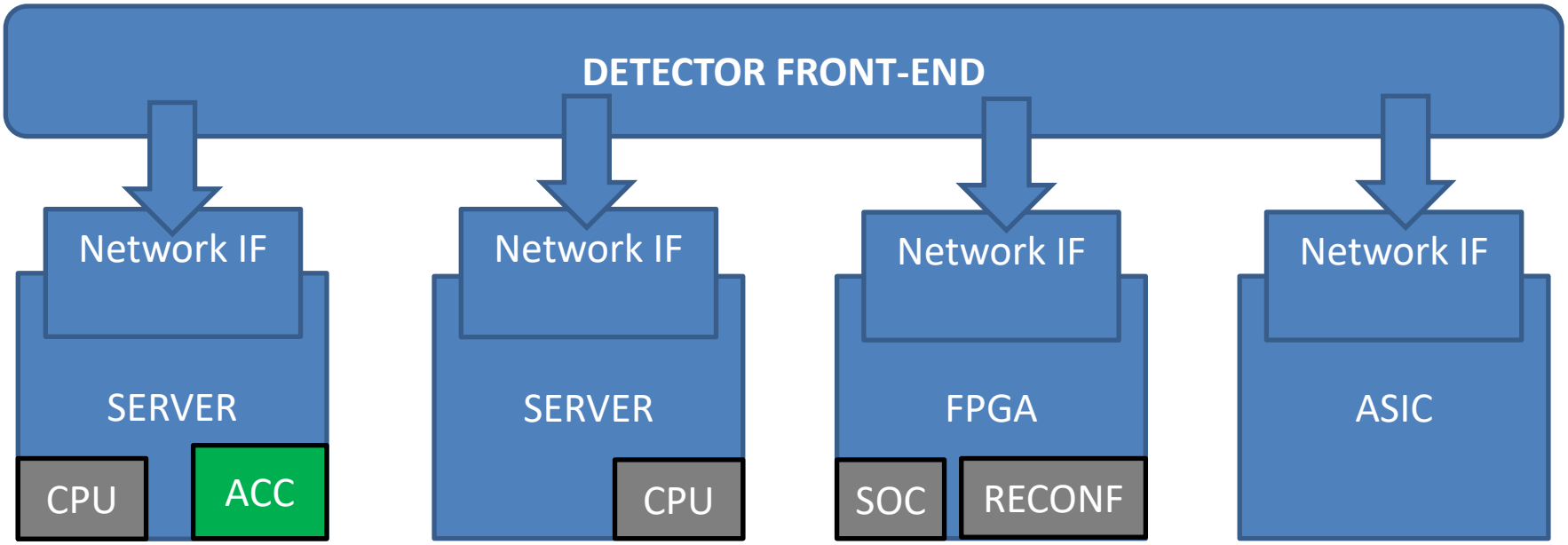


NaNet: a PCIe NIC family for HEP

OBJECTIVES:

- **Bridging the front-end electronics and the software trigger computing nodes.**
- **Supporting multiple link, multiple network protocols.**
- **Low and stable communication latency.**
- **Having a high bandwidth.**
- **Processing data streams from detectors on the fly.**
- **Optimizing data transfers with GPU accelerators.**

Current and Future Platforms



NaNet-10

10GbE

GPU nVIDIA

NaNet³

Network IF: Custom (KM3link)

Deterministic Latency for Event Timing Reconstruction

NaNet-40

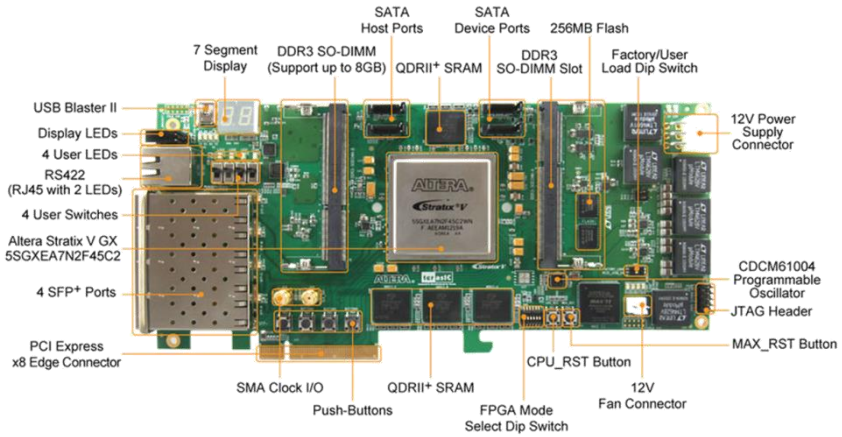
40GbE

OpenCL/ARM v9

GPU

NaNet-10

- ALTERA Stratix V FPGA (Terasic DE5-NET)
- PCIe x8 Gen2
- 4 SFP+ ports (four 10GbE 10GBASE-KR)
- GPUDirect RDMA capability
- UDP protocol offload
- Real-time stream processing on the FPGA
  - Decompression & Merger



NaNet-10 at CERN

Merging in HW - indexing events in GPU

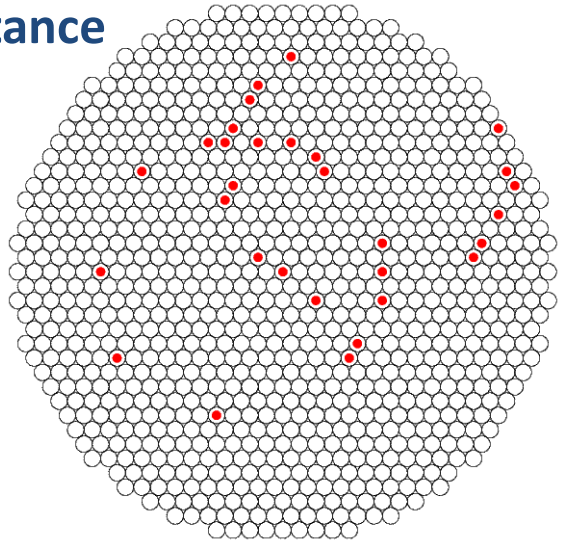
- **Merging stage is performed in HW**
- **Events are arranged in CLOPs with a new format more suitable for GPU's threads memory access Multi Merged Event GPU Packet (M^2EGP).**
  - **Problem: searching for events position inside a CLOP using 1 thread on GPU takes $>$ 100us for hundreds of events**
  - **Solution: it must be parallelized. We can use all the threads looking for a known bytes pattern at the begin of every event: it takes $\sim$ 35us for 1000 events in a buffer**

STR 3 MGP	STR 2 MGP	STR 1 MGP	STR 0 MGP	STR 3 HIT	STR 2 HIT	STR 1 HIT	STR 0 HIT	PATTERN		TOTAL HIT		TIMESTAMP			
STREAM 1; HIT 1		STREAM 1; HIT 0		STREAM 0; HIT 5		STREAM 0; HIT 4		STREAM 0; HIT 3		STREAM 0; HIT 2		STREAM 0; HIT 1		STREAM 0; HIT 0	
STREAM 2; HIT 0		STREAM 1; HIT 8		STREAM 1; HIT 7		STREAM 1; HIT 6		STREAM 1; HIT 5		STREAM 1; HIT 4		STREAM 1; HIT 3		STREAM 1; HIT 2	
STREAM 2; HIT 8		STREAM 2; HIT 7		STREAM 2; HIT 6		STREAM 2; HIT 5		STREAM 2; HIT 4		STREAM 2; HIT 3		STREAM 2; HIT 2		STREAM 2; HIT 1	
STREAM 3; HIT 4		STREAM 3; HIT 3		STREAM 3; HIT 2		STREAM 3; HIT 1		STREAM 3; HIT 0		STREAM 2; HIT 11		STREAM 2; HIT 10		STREAM 2; HIT 9	
PADDING										STREAM 3; HIT 7		STREAM 3; HIT 6		STREAM 3; HIT 5	
127...120	119...112	111...104	103...96	95...88	87...80	79...72	71...64	63...56	55...48	47...40	39...32	31...24	23...16	15...8	7...0

L0 RICH trigger algorithm

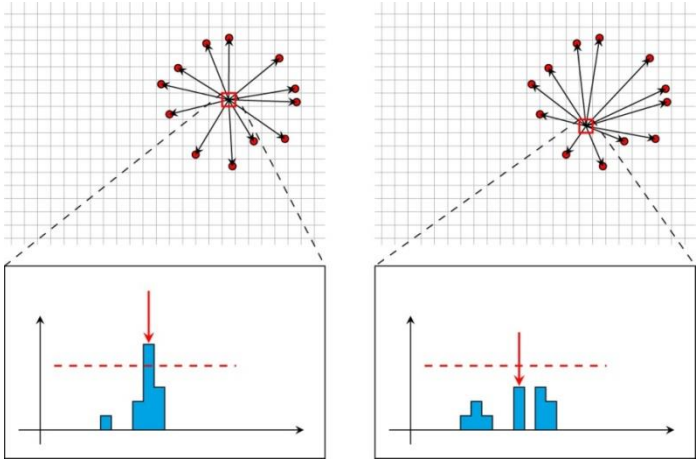
Requirements for an on-line RICH reconstruction algorithm:

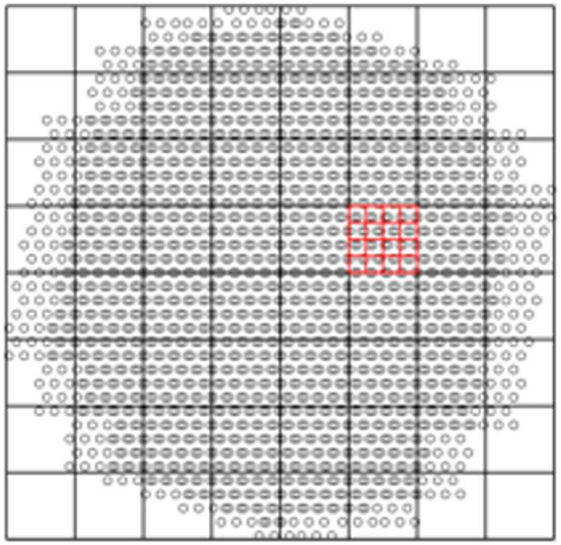
- **Trackless**
  - No information from the tracker
  - Difficult to merge information from many detectors at L0
- **Multi-rings**
  - Many-body decays in the RICH acceptance
- **Fast**
  - Events rate at ~10 MHz
- **Low latency**
  - Online (synchronous) trigger
- **Accurate**
  - Offline resolution required



Histogram: a pattern recognition algorithm

- XY plane divided into a grid
- An histogram is created with distances from these points and hits of the physics event
- Rings are identified looking at distance bins whose contents exceed a threshold value





Pros: naturally mapped on the GPU threads grid

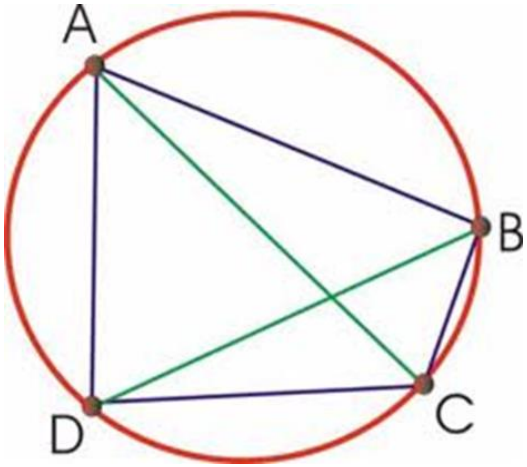
Element of the grid <--> thread Event <--> block

Cons: local memory limited, performances depending on number of events/hits

Almagest: a new multi-ring algorithm

Based on Ptolemy's theorem:

"A quadrilater is cyclic (the vertex lie on a circle) if and only if is valid the relation: AD*BC+AB*DC=AC*BD"



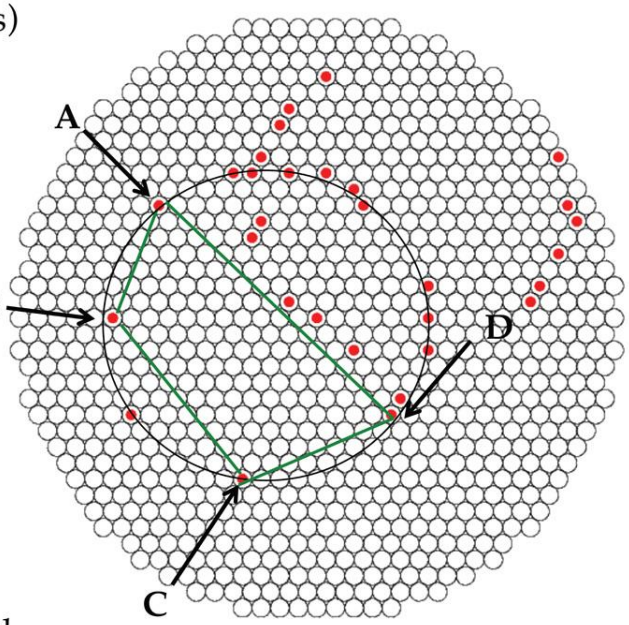
Almagest: a new multi-ring algorithm

i) Select a *triplet* (3 starting points)

ii) Loop on the remaining points: if the next point does not satisfy the Ptolemy's condition then reject it

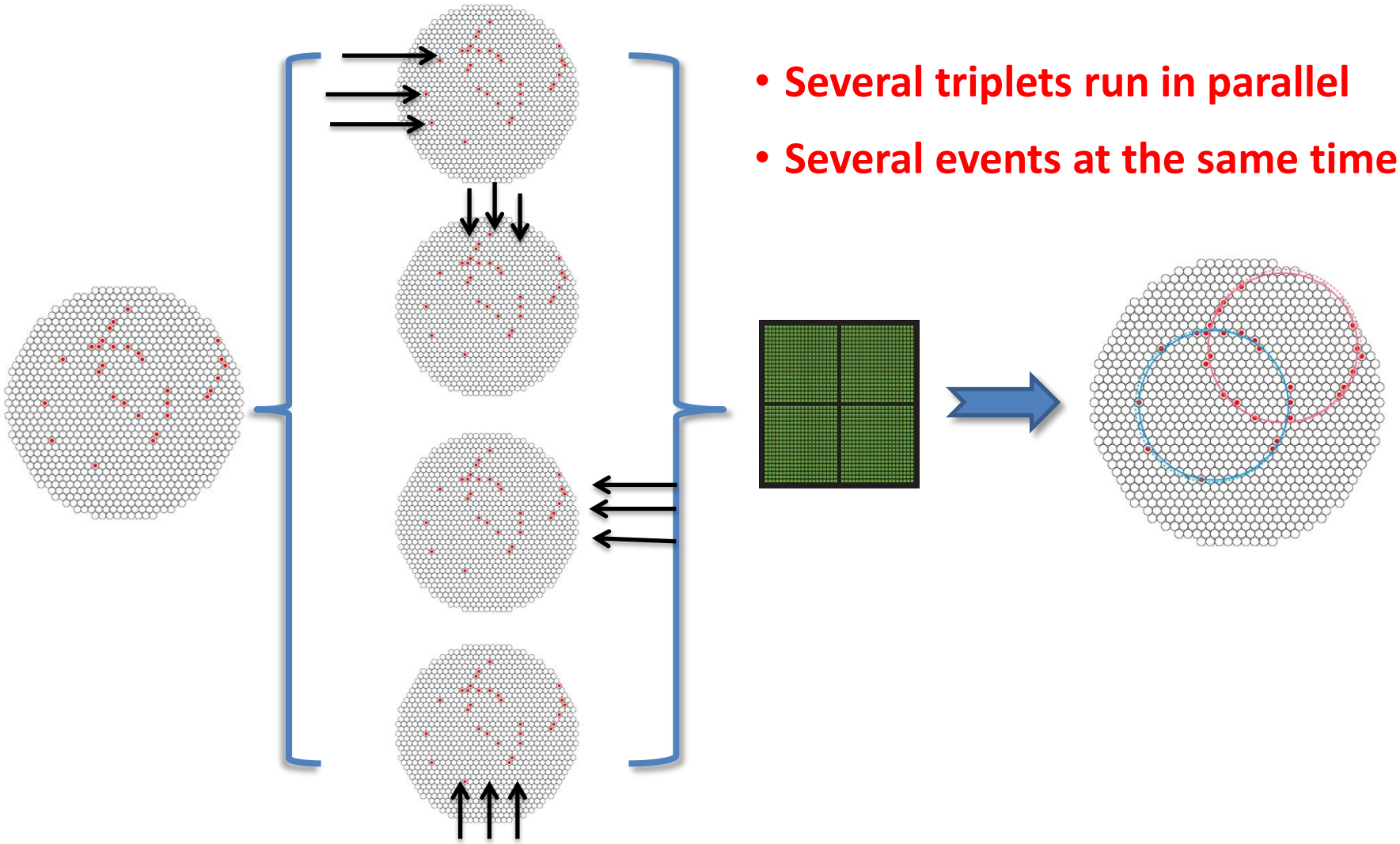
iii) If the point satisfy the Ptolemy's condition then consider it for the fit

iv) …again…



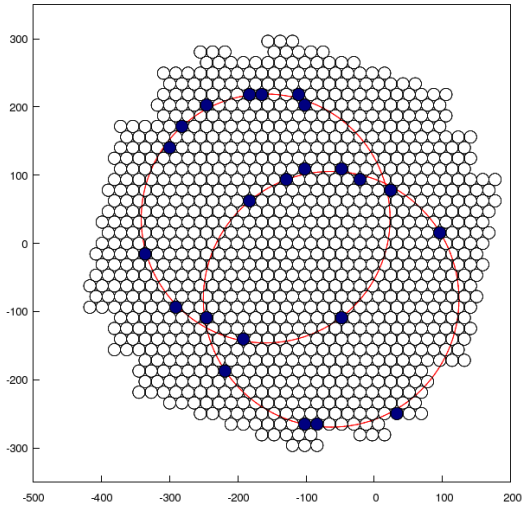
This algorithm exposes two levels of parallelism…

Almagest: a new multi-ring algorithm

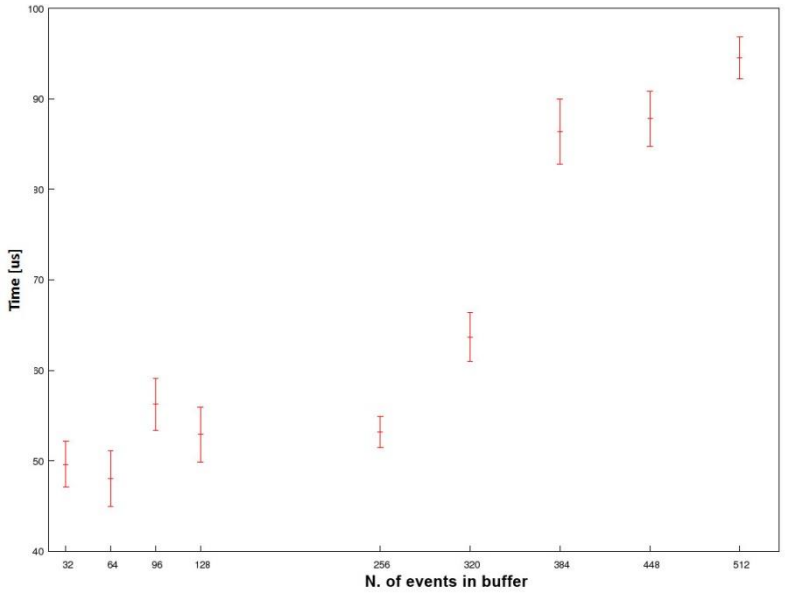


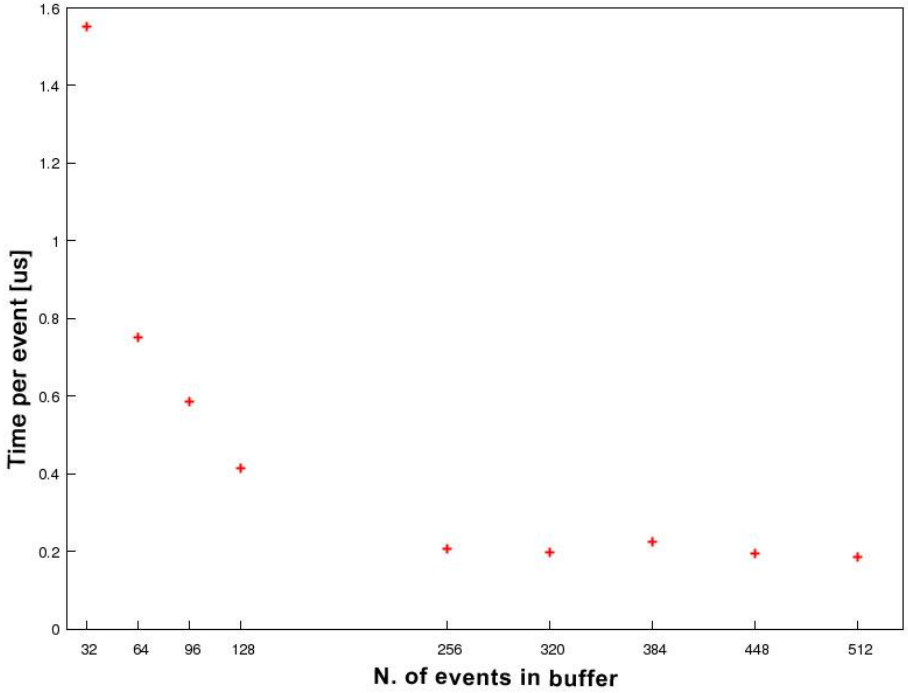
- Several triplets run in parallel
- Several events at the same time

Almagest



- nVIDIA K20c
- Only computing time
- <0.5 us per event (multi-rings) for large buffers



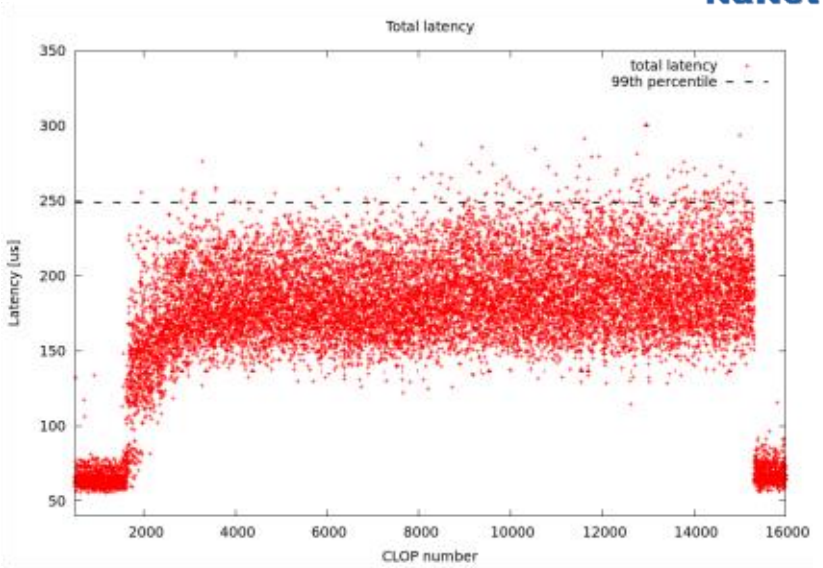


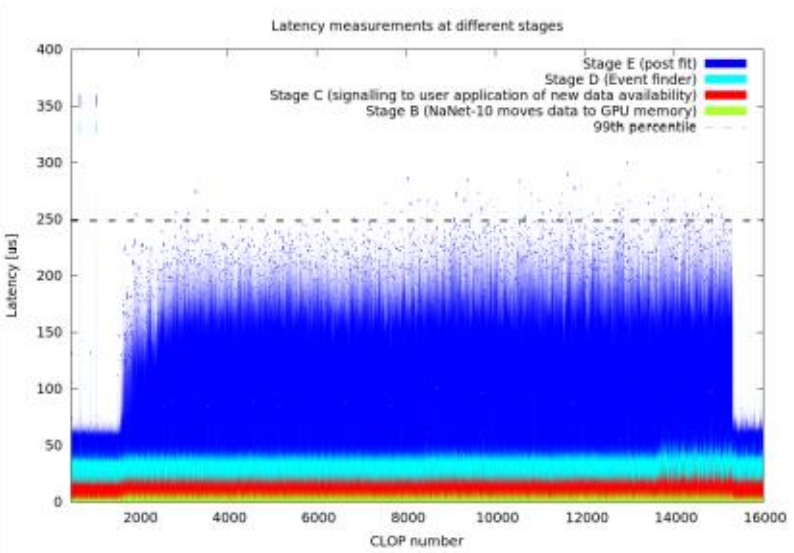
Results in 2016 NA62 Run

Testbed

- Supermicro X9DRG-QF Intel C602 Patsburg
- Intel Xeon E5-2602 2.0 GHz
- 32 GB DDR3
- **nVIDIA K20c**
- ~ 30% target beam intensity ($12*10^{11}$ Pps)
- Gathering time: 350us
- Histogram algorithm

- **Processing time per event: 1 us (to be improved)**
- **Processing latency: below 200 us (compatible with the NA62 requirements)**





Work in progress and conclusions

- Heterogeneous systems can really help in achieving online reconstruction for triggers, offering
  - low latency data transport
  - on the fly data preprocessing
  - great flexibility in managing link/protocols
- Preparing for 2017 RUN
  - adapting the Histogram algorithm to the new nVIDIA Pascal architecture allowed to greatly reduce the time per event, from~1us to 0.2us
  - exploring the feasibility to send data ring parameters to the HLT, in order to reduce its processing time and increase selection
- NaNet-40 is on the way...

R. Ammendola [(b)], A. Biagioni[(a)],
S. Di Lorenzo[(f,g)], R. Fantechi[(f)],
M. Fiorini[(d,e)], O. Frezza[(a)],
G. Lamanna[(g)], F. Lo Cicero[(a)],
A. Lonardo [(a)], M. Martinelli[(a)],
I. Neri[(d)], P.S. Paolucci[(a)],
E. Pastorelli[(a)], R. Piandani[(f)],
L. Pontisso[(f)], F. Simula[(a)],
M. Sozzi[(f,g)], P. Vicini[(a)].

[(a)] INFN Sezione di Roma
[(b)] INFN Sezione di Roma Tor Vergata
[(c)] INFN - Laboratori Nazionali di Frascati
[(d)] INFN Sezione di Ferrara
[(e)] Università di Ferrara
[(f)] INFN Sezione di Pisa
[(g)] Università di Pisa

Thank You