CUDA in NA62

Luca Pontisso (INFN)

on behalf of GAP and NaNet collaborations

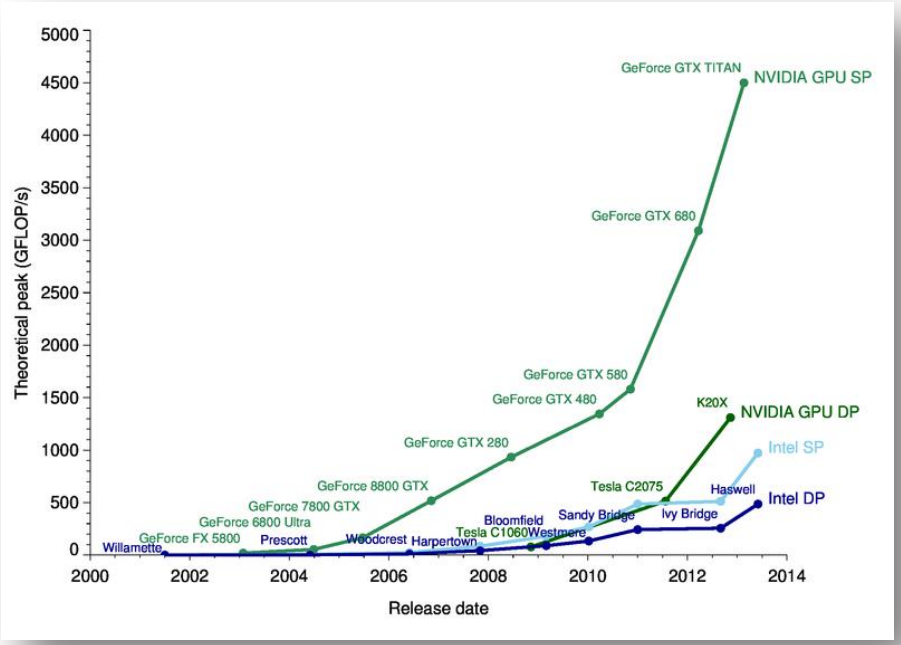
Workshop di CCR: La Biodola 2016

Overview

- Quick glance to GPU
- CUDA: a parallel computing platform/programming model
- Physics case:
  GPU based L0 trigger for NA62 RICH detector
- Algorithms implemented
  - Histogram
  - Almagest
- Concluding remarks

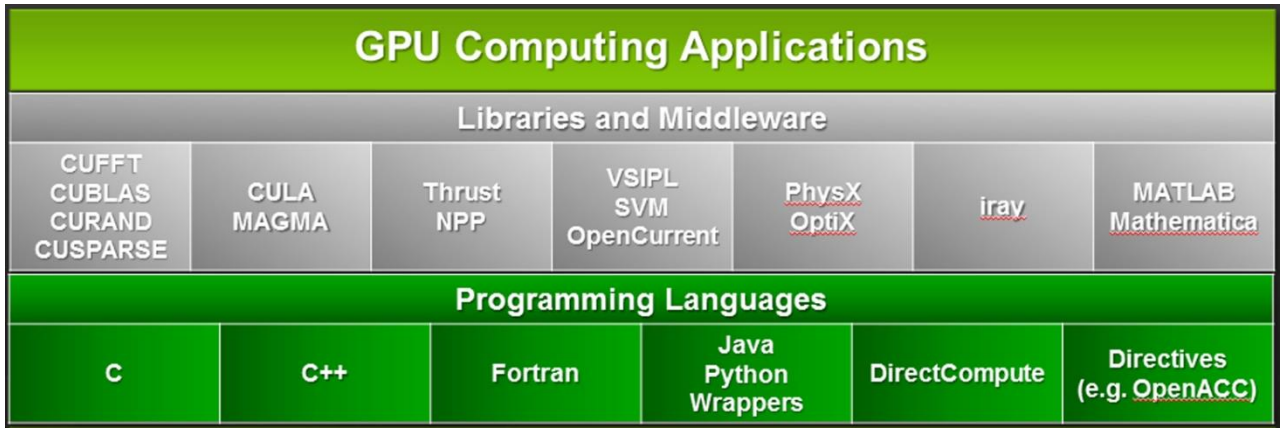
Graphics Processing Unit (GPU)

- GPU's advanced capabilities were originally used primarily for 3D game rendering
- Since 2007 high-Level programming languages (CUDA, OpenCL) have been introduced
- Now this devices are largely deployed in General Purpose applications (GPGPU)
- Based on a massively parallel architecture
- 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.



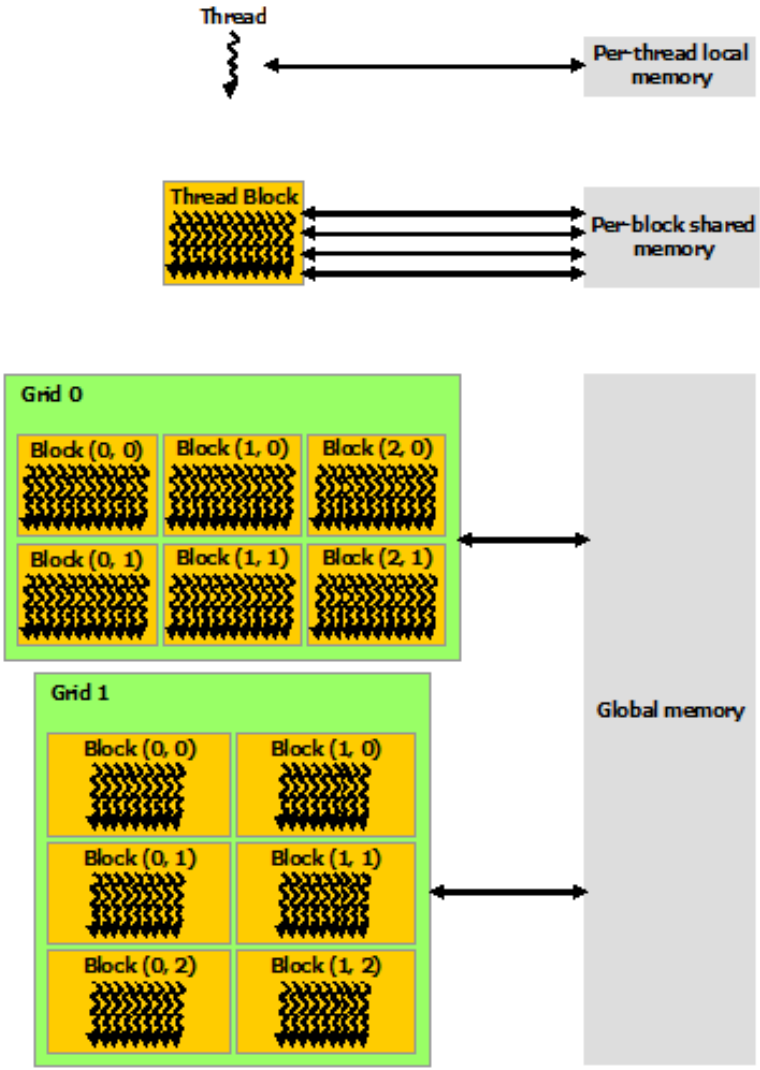
CUDA

- A general purpose parallel computing platform and programming model
- It allows developers to use C as a high-level programming language
- Supporting various languages and application programming interfaces



- CUDA extends C defining functions (kernels) that, when called, are executed N times in parallel by N different CUDA threads,

Memory hierarchy

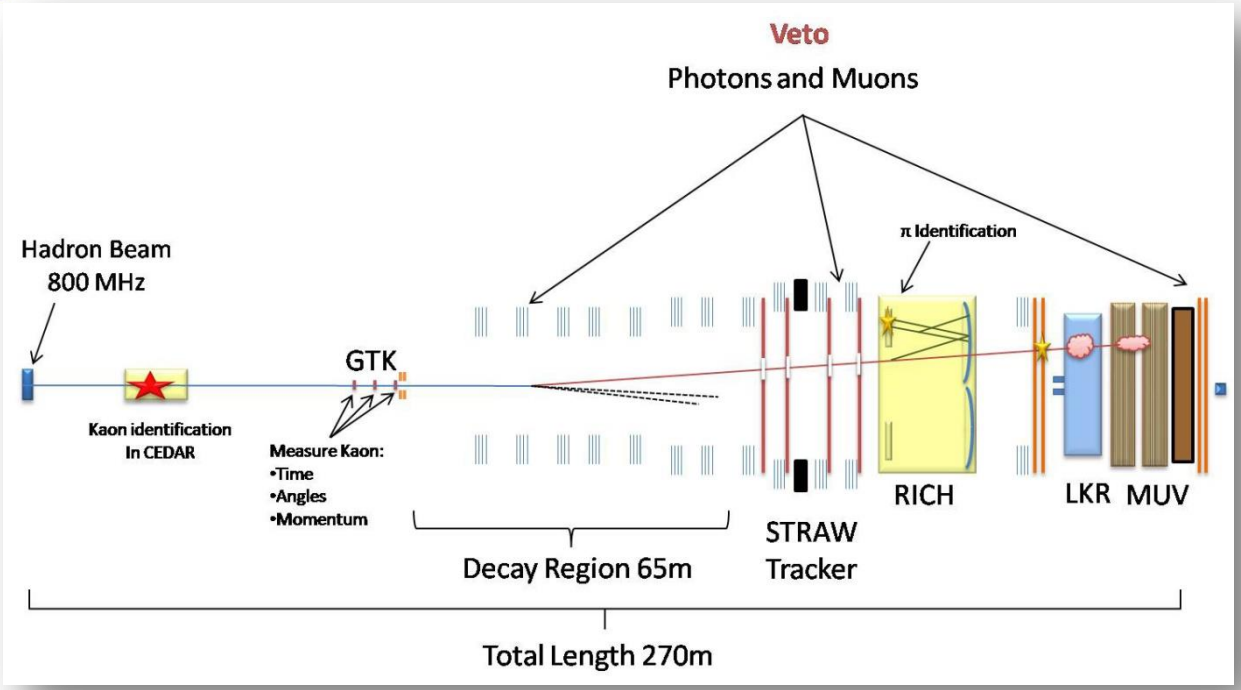


registers

But... inside a warp:

- ***Inter thread communication***
  Shuffle/Vote functions
- ***Implicit synchronization***

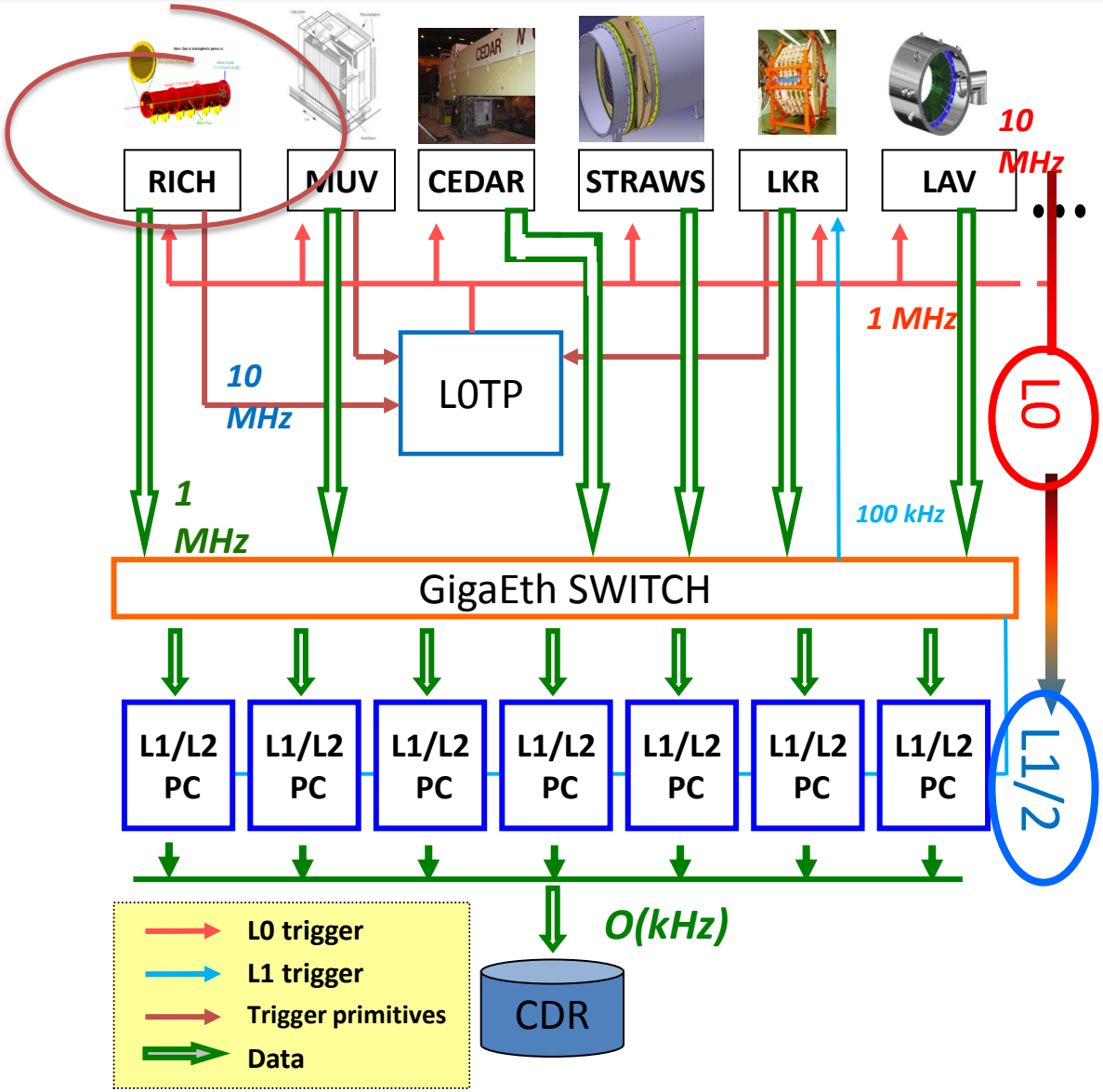
Physics case: Low Level Trigger system in NA62



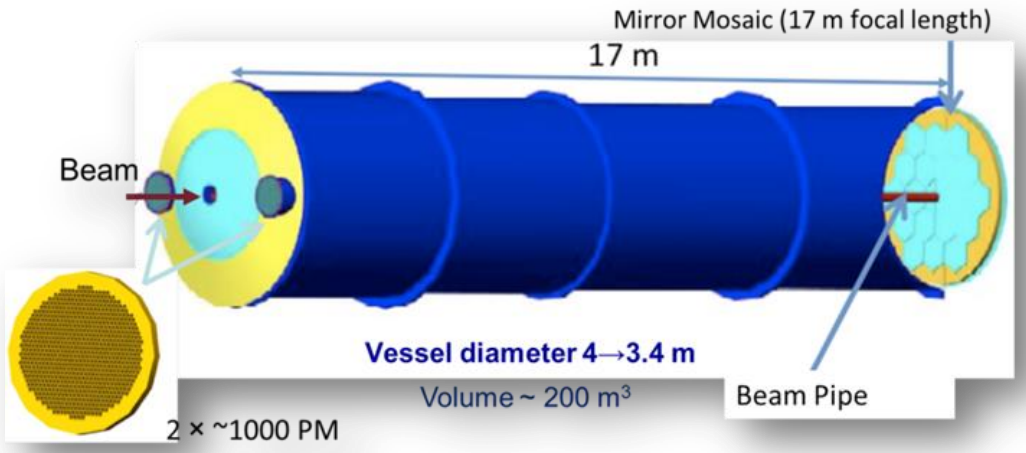
- **Measurement of ultra-rare decay**
  $K^+ \to \pi^+ \nu \bar{\nu}$
  (BR ~8×10^{-11})
- **Kaon decays in flight**
- **High intensity unseparated hadron beam (6% kaons)**

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

NA62 DAQ and Trigger

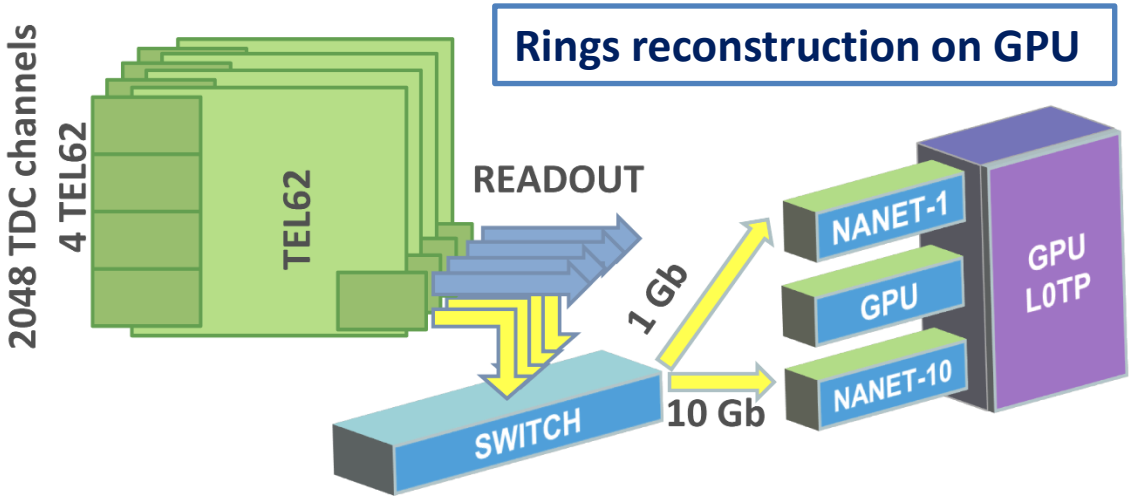


GPU-based L0 trigger general scheme



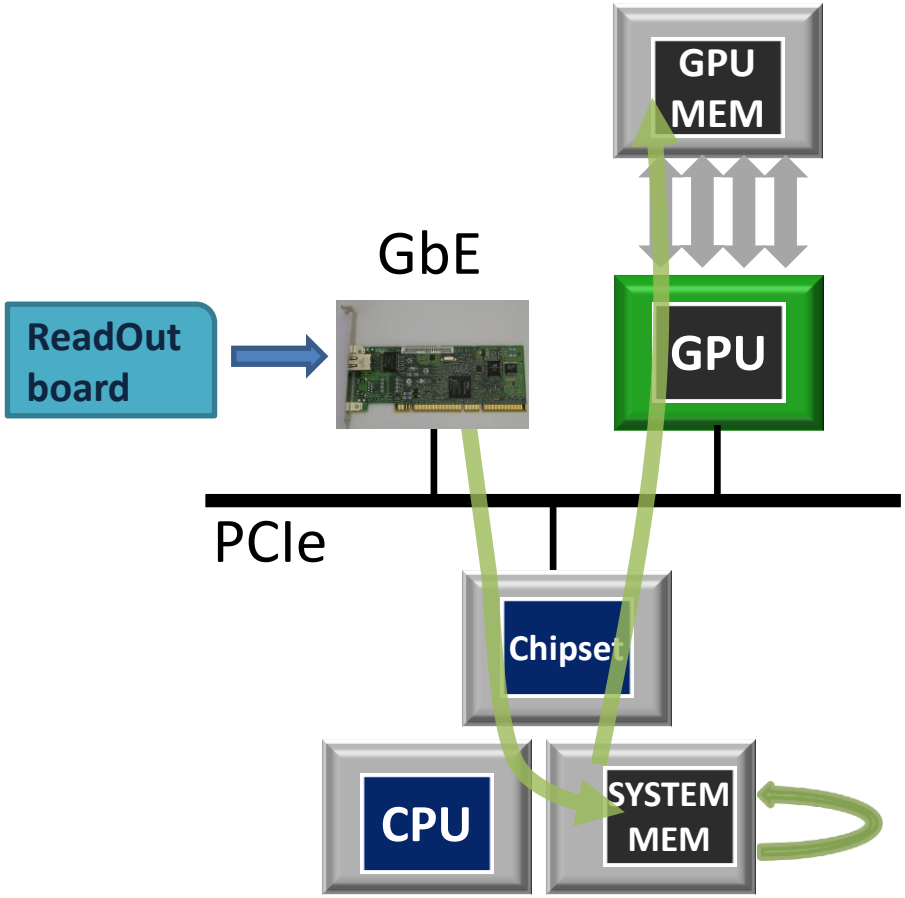
NA62 Rich detector

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



NaNet

Latency communication problem



Total latency dominated by double copy in Host RAM:

- **Data are copied from kernel buffer to destination buffer in user space**
- **Data are copied from CPU memory to GPU memory**

How to reduce data transfer time:

- **DMA (Direct Memory Access) from the network channel directly in GPU memory**
- **Custom management of NIC buffers**

NaNet: a PCIe NIC family for HEP

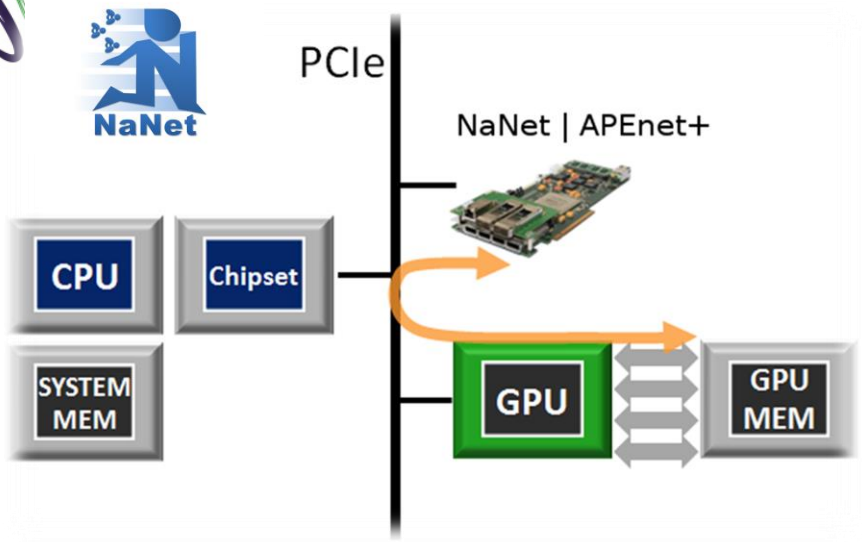
Developed at INFN Roma APE Lab

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.**

NaNet-10 at CERN

NaNet: GPUDirect and Software



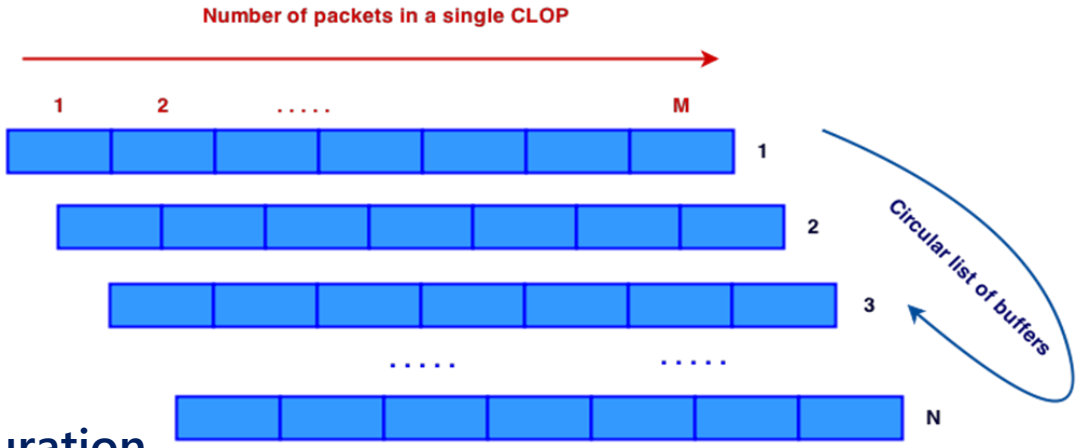
- GPUDirect allows direct data exchange on the PCIe bus with no CPU involvement
- No bounce buffers on host memory
- Zero copy I/O
- Buffers on GPU arranged in a circular list of persistent receiving buffers (CLOP)
- nVIDIA Fermi/Kepler/Maxwell

Host

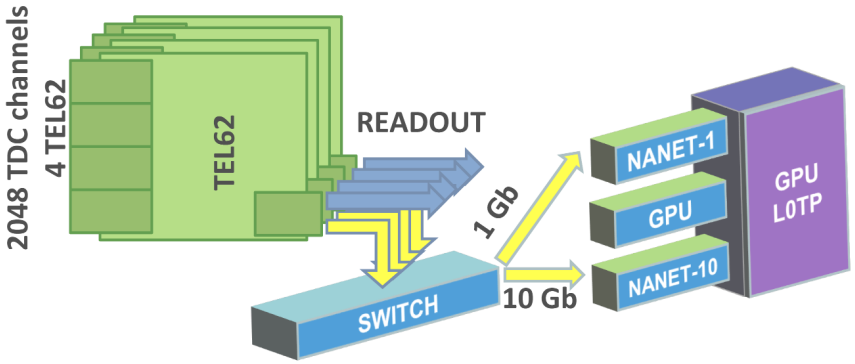
- Linux Kernel Driver
- User space Library (open/close, buf reg, wait recv evts, …)

Nios II Microcontroller

- Single process program performing System Configuration & Initialization tasks

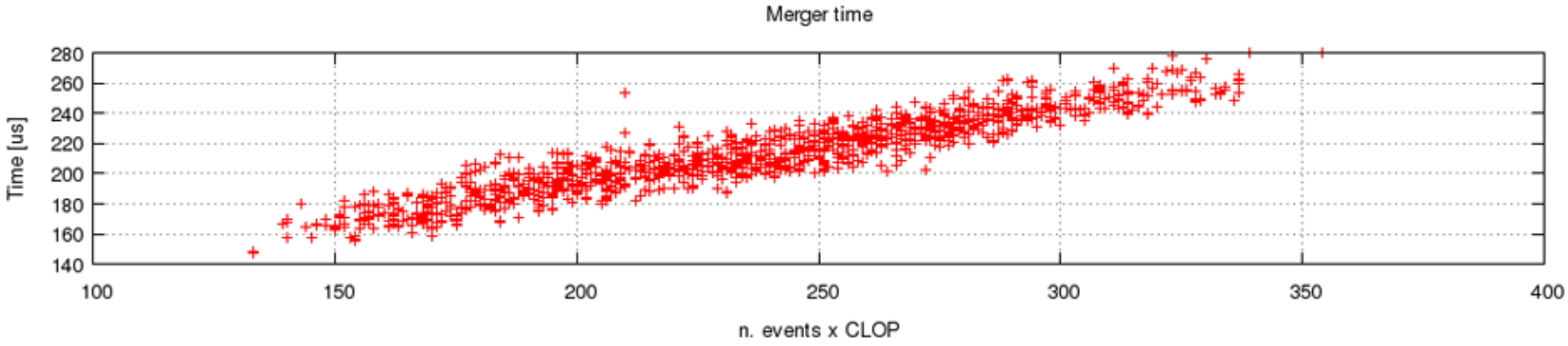


Don’t do it in GPU



Merging the events coming from the RICH on GPU… *NO WAY*

- **it requires synchronization and serialization**
- **computing kernel launched after merging**



Merging in HW - indexing events in GPU

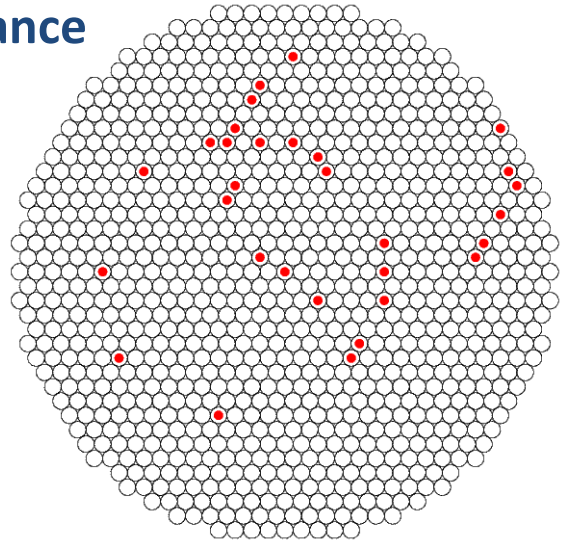
- **Merging stage will be performed in HW**
- **Events are arranged in CLOPs with a new format more suitable for GPU's threads memory access Multi Merged Event GPU Packet (M^2EGP).**
  - **Problem: searching for events position inside a CLOP using 1 thread on GPU takes $>$ 100us for hundreds of events**
  - **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

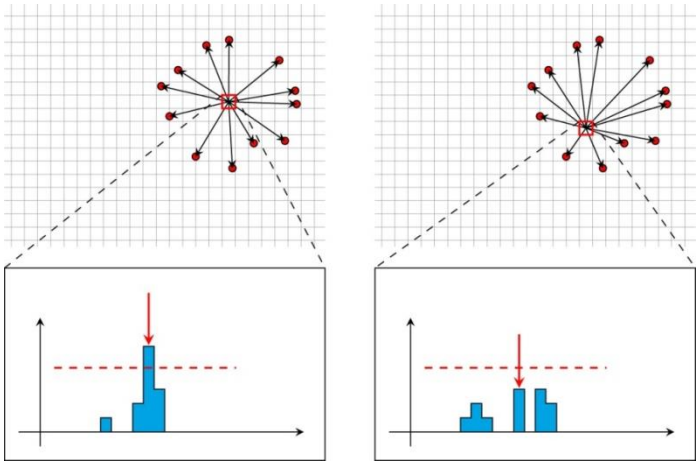
Requirements for an on-line RICH reconstruction algorithm:

- **Trackless**
  - No information from the tracker
  - Difficult to merge information from many detectors at L0
- **Multi-rings**
  - Many-body decay 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



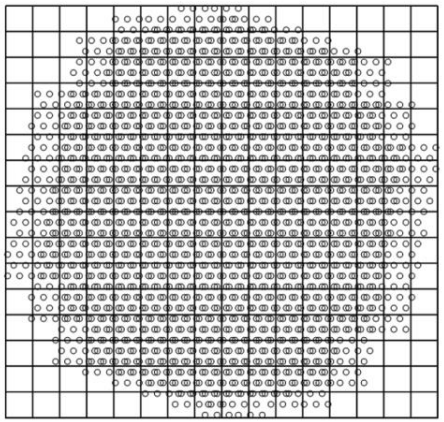
Pros: naturally mapped on the GPU threads grid

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

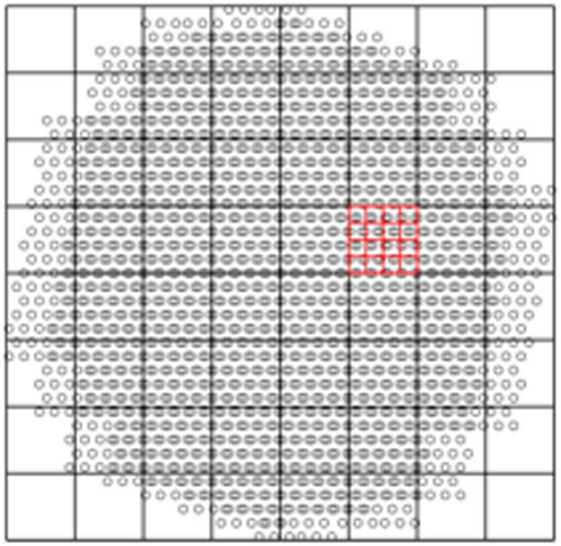
Cons: memory limited, performances depending on number of hits

Histogram: a pattern recognition algorithm

First implementation
16x16 grid -> 256 threads x event

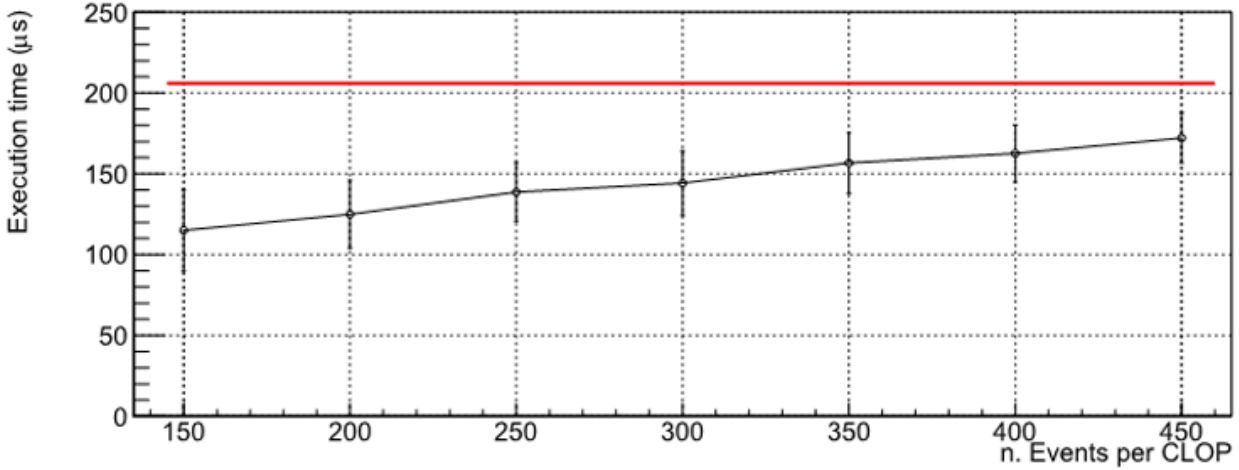


2-step implementation
8x8 grid -> 64 threads x event
4x4 grid only around maximum

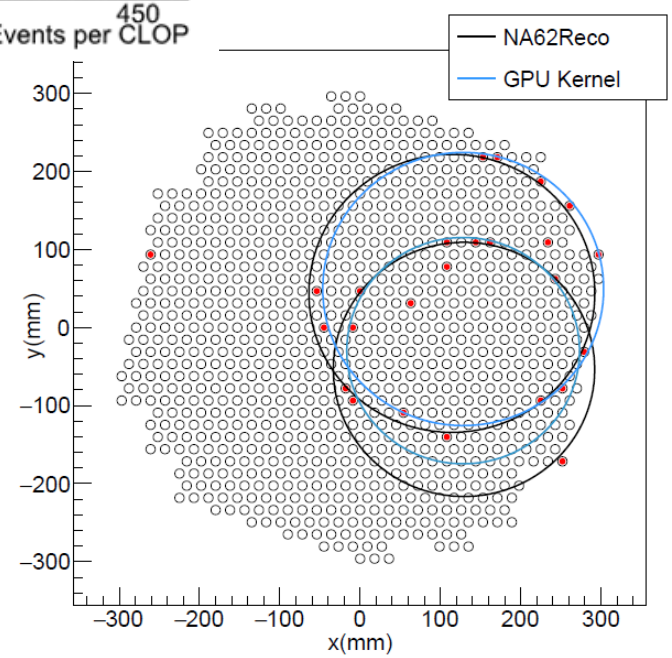


- More blocks run concurrently
- Increased precision

Histogram: a pattern recognition algorithm



Testing histogram kernel with MC data

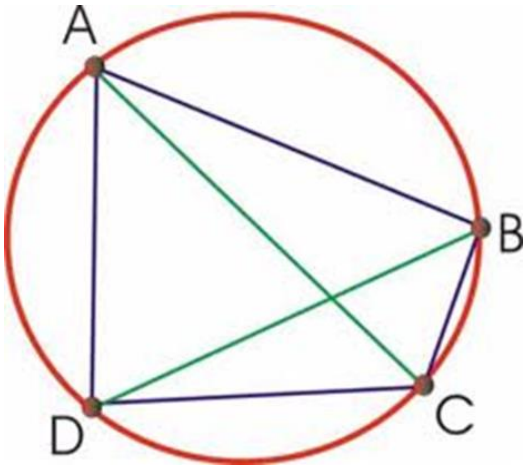


Good quality of online ring fitting

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"

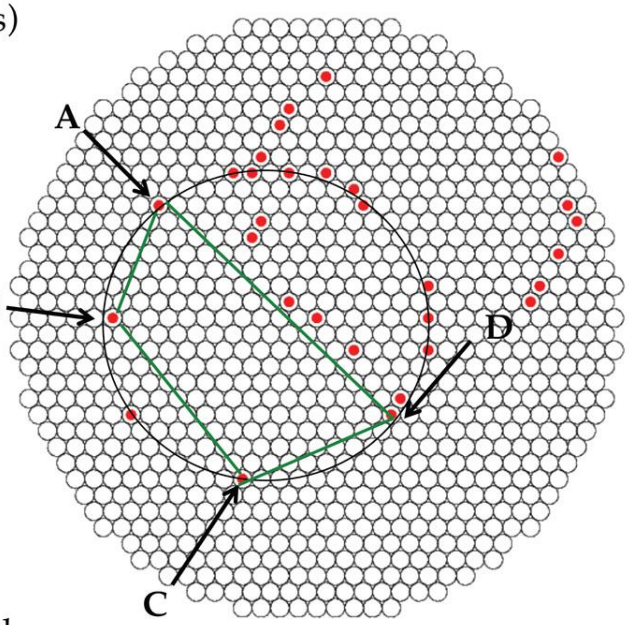


Almagest: a new multi-ring algorithm

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

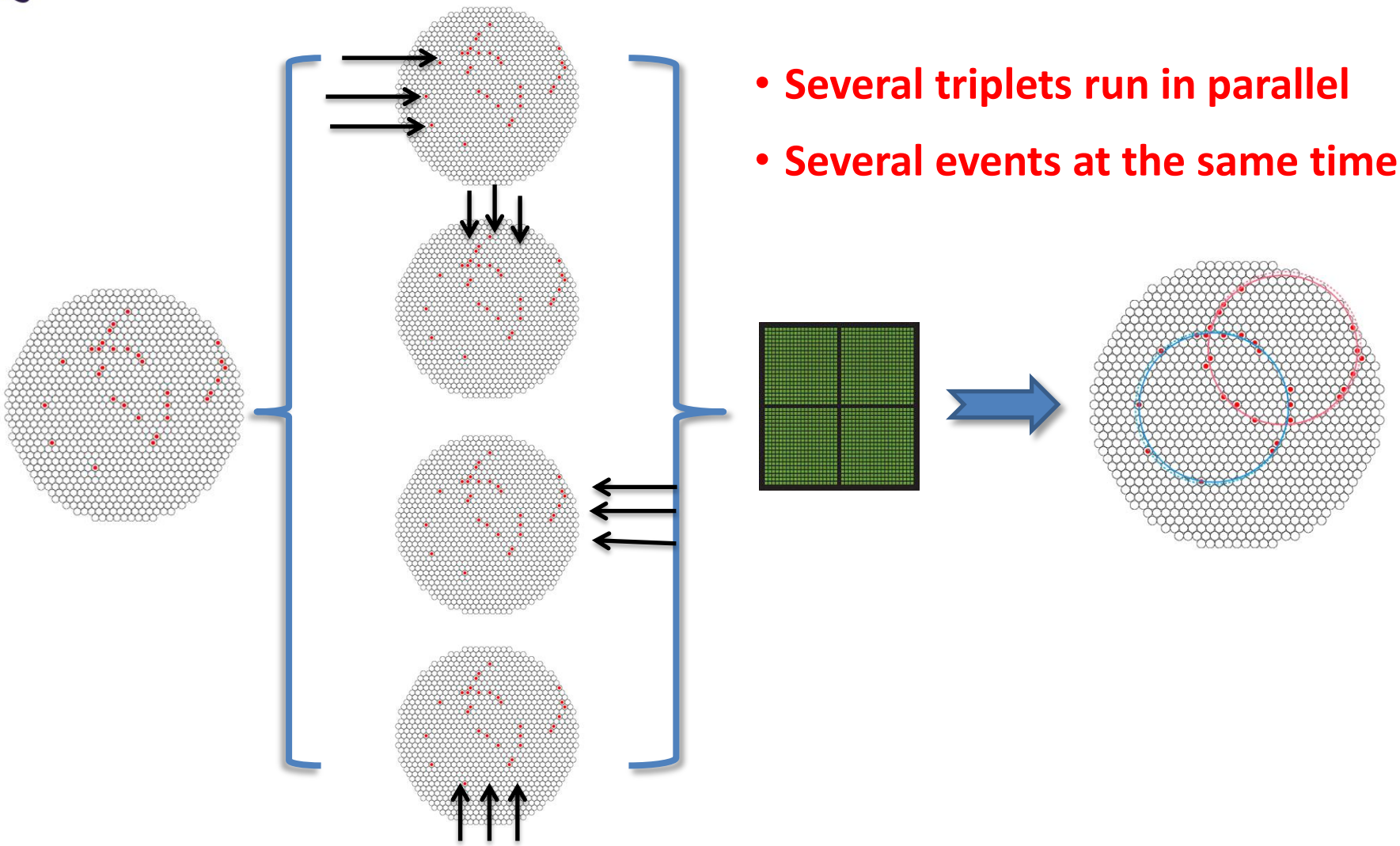
iii) If the point satisfy the Ptolemy's condition then consider it for the fit



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: selecting triplets on GPU

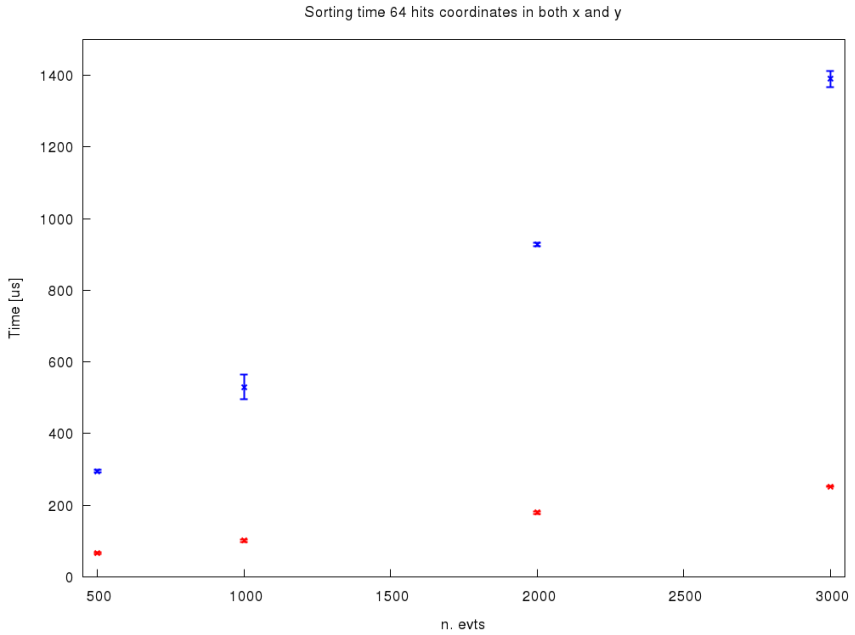
64 hits per events
Sort in both x and y

Coordinates stored in built-in types float2

No multiple kernels

1 thread <-> 1 hit
2 warps
Shared memory is needed

1 thread <-> 2 hit
1 warp
Sorting using «shuffle» btw threads
No shared memory
No __syncthreads()



Almagest: selecting triplets on GPU

```
nvcc -m64 --ptxas-options=-v -O2 -arch=sm_35  ....

ptxas info    : Compiling entry function '_Z5sortfv' for 'sm_35'
ptxas info    : Function properties for _Z5sortfv
   0 bytes stack frame, 0 bytes spill stores, 0 bytes spill loads
ptxas info    : Used 17 registers, 24576 bytes smem, 320 bytes cmem[0]
```

```
nvcc -m64 --ptxas-options=-v -O2 -arch=sm_35  ....

ptxas info    : Compiling entry function '_Z5sortfv' for 'sm_35'
ptxas info    : Function properties for _Z5sortfv
   0 bytes stack frame, 0 bytes spill stores, 0 bytes spill loads
ptxas info    : Used 21 registers, 320 bytes cmem[0]
```

More resources available for the computing stage...

Almagest

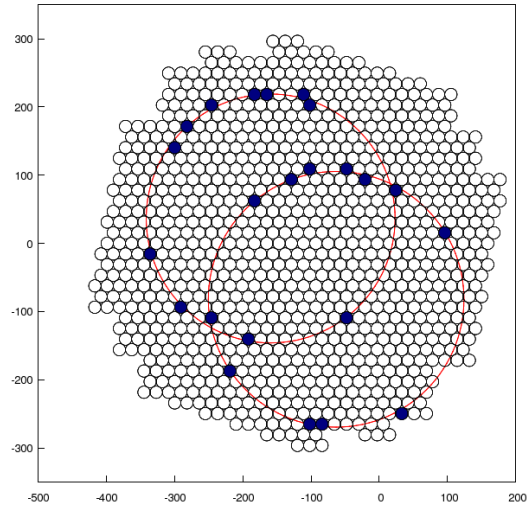
1 thread <-> 1 hit
2 warps

Every thread checks if Tolomeo's condition is satisfied starting with one of the triplets previously found

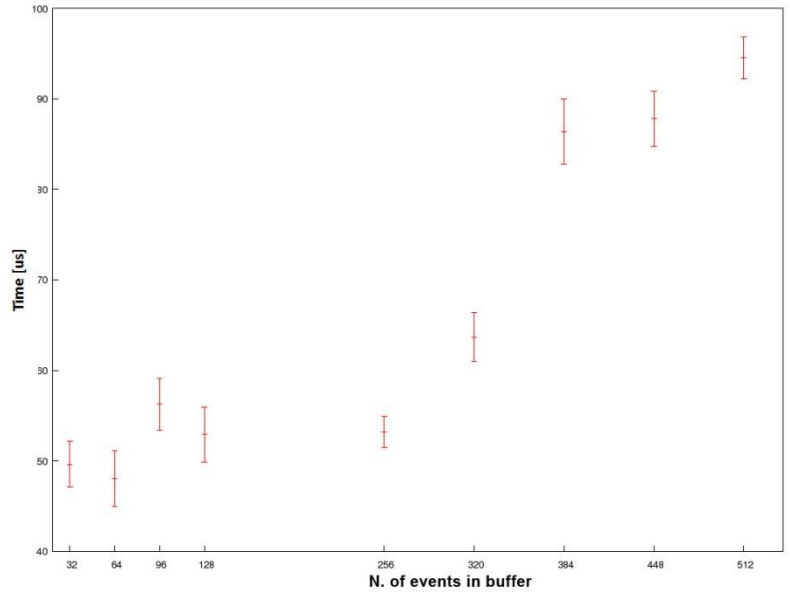
Vote function __ballot()
Results propagated through the warp

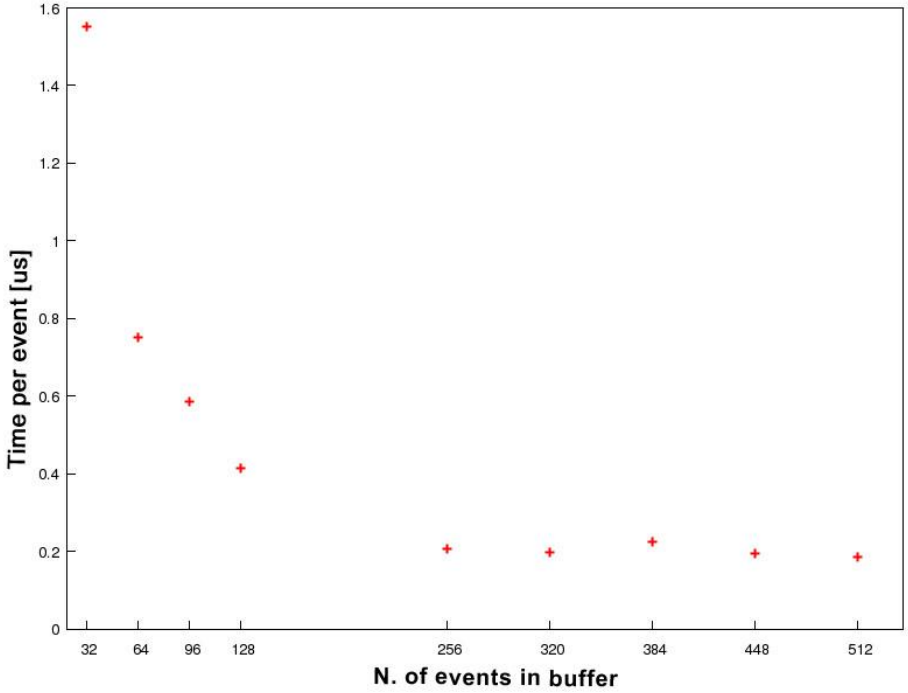
Shared memory and barrier synchronization because of working with 2 warps

Almagest



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





Concluding remarks

- Tested a working solution with the NaNet-1 board during NA62 2015 Run
- Merging events on GPU not feasible
- Multi-ring algorithms such as Histogram and Almagest are implemented on GPU

  There is still room for improvement -> Optimize computing kernels
- The GPU-based L0 trigger with the new board NaNet-10 is installed and being currently tested during the 2016 run

R. Ammendola [(b)], A. Biagioni[(a)],
S. Chiozzi[(d)], A. Cotta Ramusino[(d)],
S. Di Lorenzo[(f,g)] , R. Fantechi[(f)],
M. Fiorini[(d,e)], O. Frezza[(a)],
G. Lamanna[(c)], 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