The GAP Project: GPU for Online Processing in Low-Level Trigger

Massimiliano Fiorini
(Università di Ferrara and INFN Ferrara)

On behalf of the GAP Collaboration

GPU in High Energy Physics
Pisa, 10-12 September 2014

GAP Project

- **GAP** (**G**PU **A**pplication **P**roject) for real-time in HEP and medical imaging is a 3 years project funded by the Italian Ministry of research, started in 2013
- Collaboration between INFN Sezione di Pisa, University of Ferrara and University of Roma "La Sapienza"
- Demonstrate the feasibility of using off-the-shelf computer commodities to accelerate real-time scientific computations
- Application in different fields:
  - High Energy Physics (low and high level triggers)
  - Medical Imaging (NMR, CT and PET)

GAP Project

- **GAP** (**G**PU **A**pplication **P**roject) for real-time in HEP and medical imaging is a 3 years project fund
- Collaboration h … University of … "La Sapienza"
- … the feasibility of … commodities …
- … and high level triggers)
- … imaging (NMR, CT and PET)

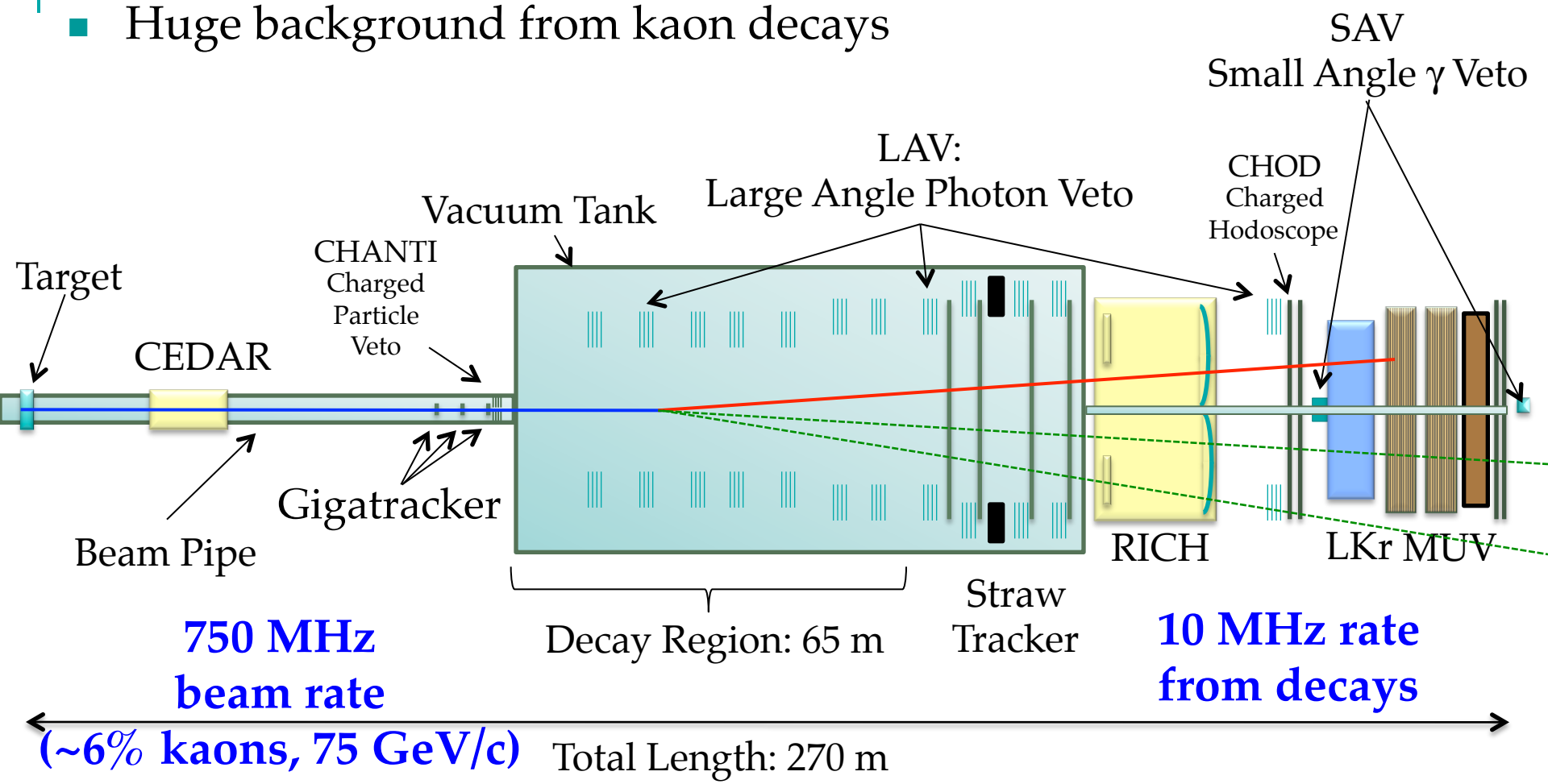
Talk: M. Bauce "The GAP Project: GPU Applications for High Level Trigger and Medical Imaging"

Talk: G. Di Domenico "Fast Cone-beam CT reconstruction using GPU"

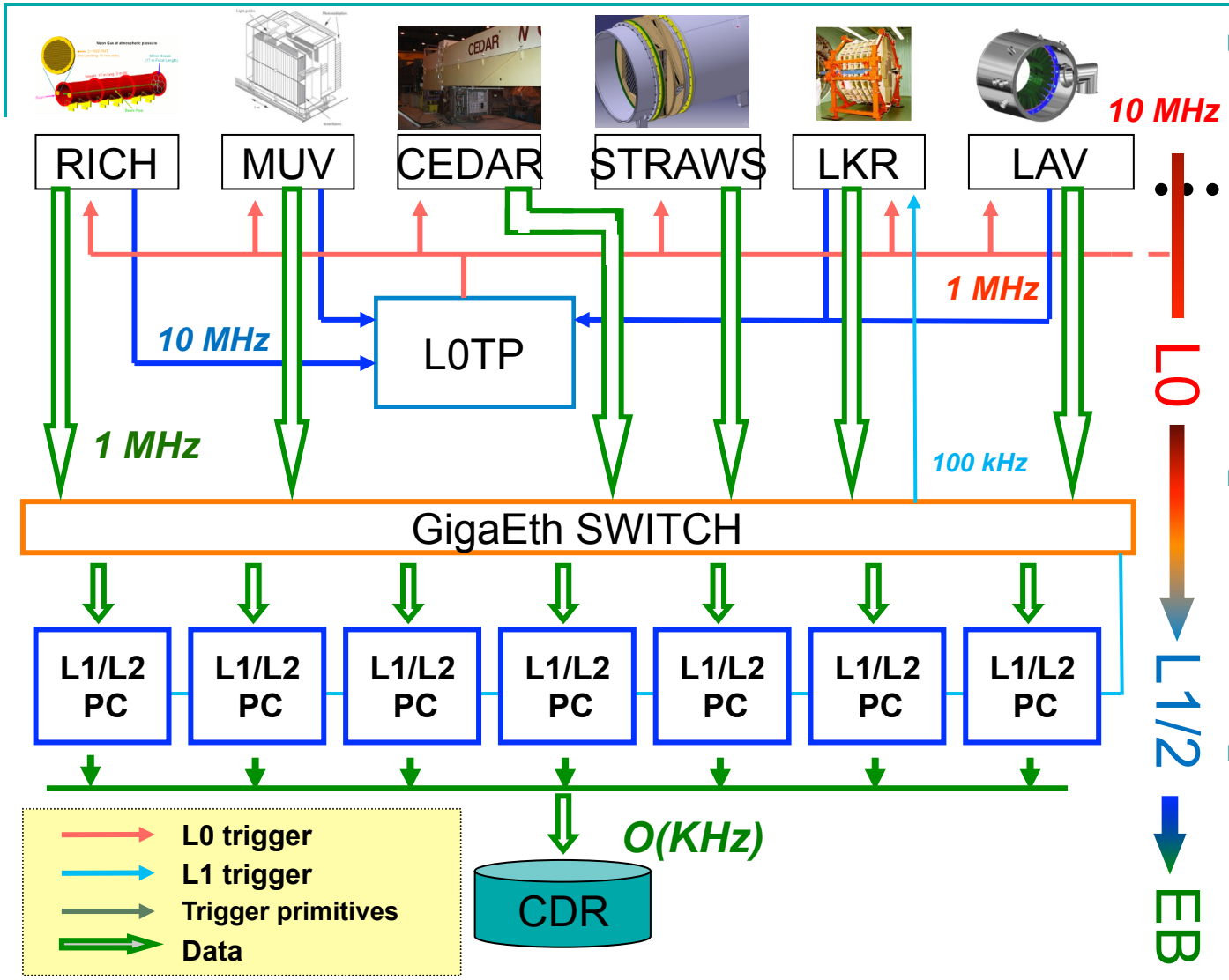
Talk: M. Palombo "GPU-parallelized Levenberg-Marquardt model fitting towards real-time automated parametric diffusion NMR imaging"

Physics case: NA62

- $K^+ \rightarrow \pi^+ \nu\nu$ decay (BR~8×10^{-11})
- Huge background from kaon decays



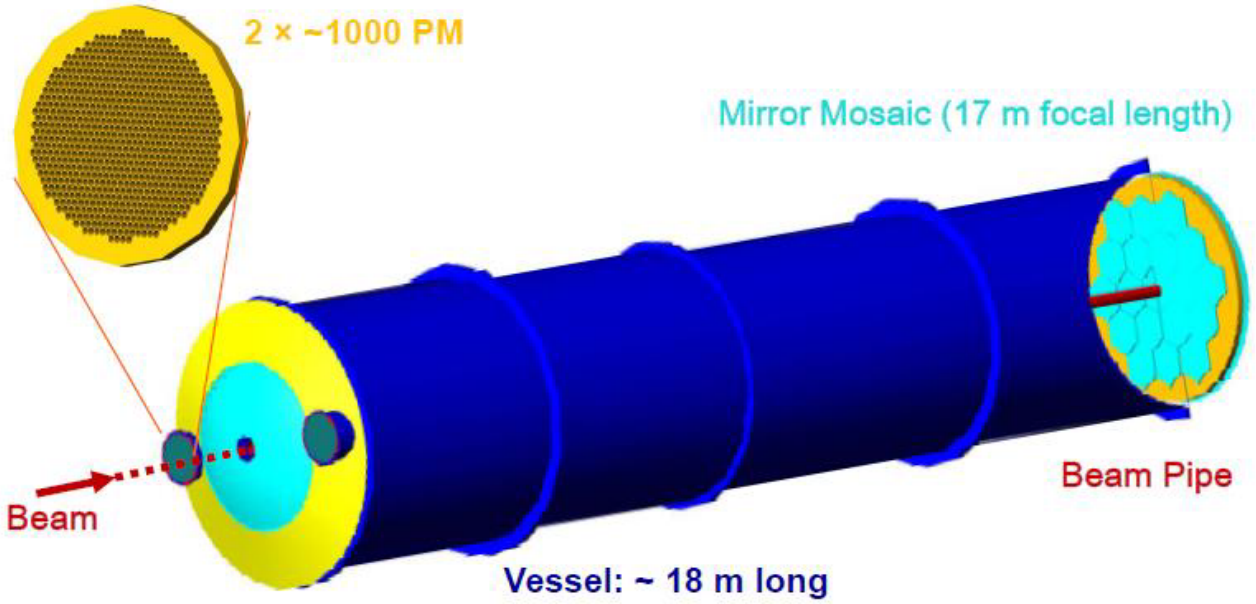
Trigger and DAQ



- **L0**: Hardware synchronous level
  - 10 MHz to 1 MHz, **1 ms max. latency**
  - Primitives (MUV, RICH, LAV, LKR)
- **L1**: Software level
  - "Single detector", 1 MHz to 100 kHz
- **L2**: Software level
  - "Complete information", 100 kHz to 10 kHz

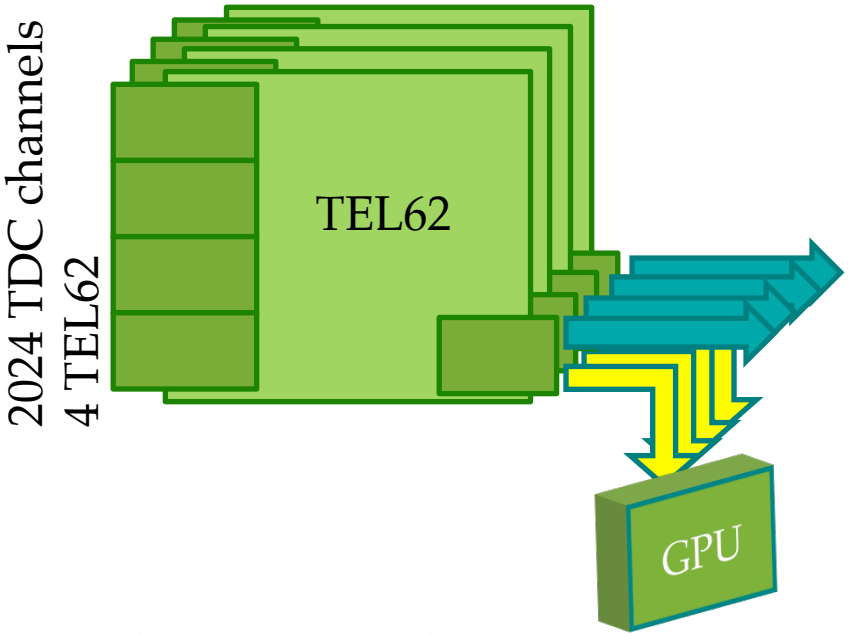
The RICH detector

- 17 m focal length, ~4 m in diameter, filled with Ne at 1 atm
- Pion/muon separation in the range 15-35 GeV/c
- Time resolution ~70 ps
- Mis-identification 5×10^{-3}
- 2 spots of 1000 PMTs each



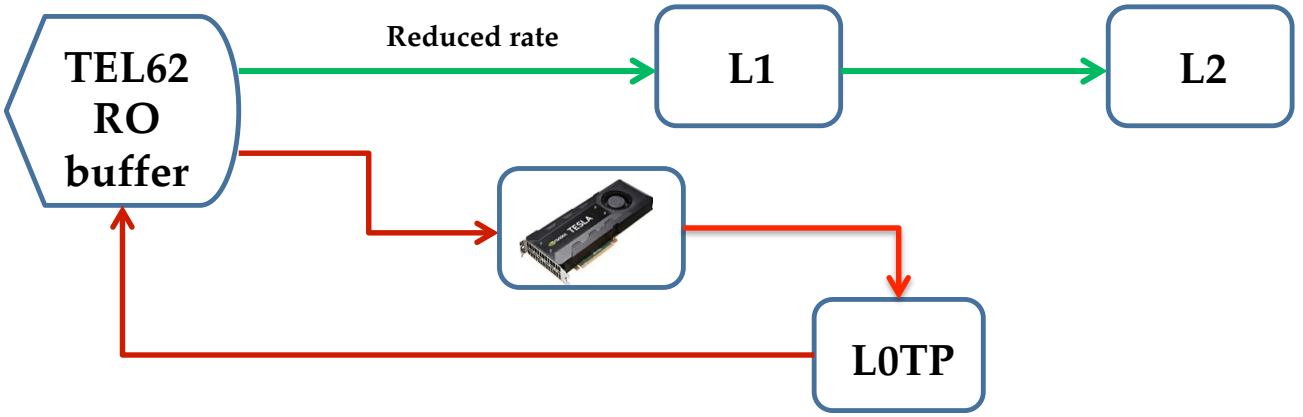
- 10 MHz events rate in the RICH (~20 hits/track)
  - Main contribution from kaon decays (~1 MHz from halo muons and pion decays)

The goal: GPU in L0 RICH



- 4 TEL62 for RICH detector
  - 8×1Gb/s links for data r/o
  - 4×1Gb/s trigger primitives
  - 4×1Gb/s GPU trigger
- Events rate: 10 MHz
- L0 trigger rate: 1 MHz
- Max Latency: 1 ms

- This is not the L0 trigger baseline version for the NA62 RICH detector



GPUs in Low Level Triggers

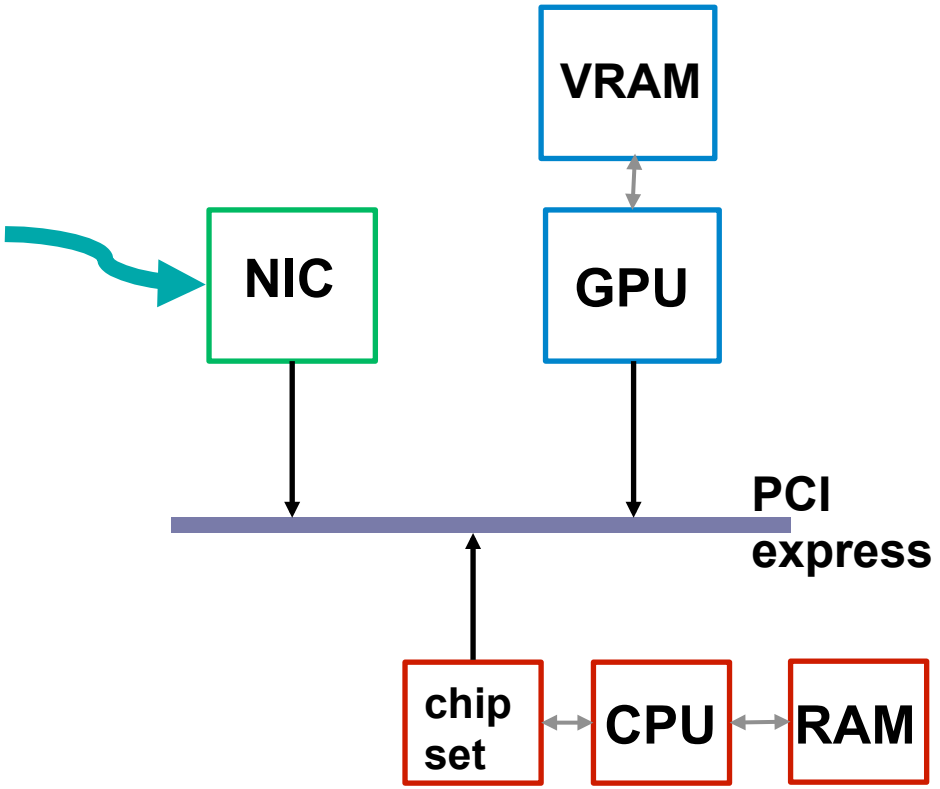
- Two main issues to be solved:

- **Latency**
  - Is the GPU latency per event small enough to cope with the tiny latency of low level triggers?
  - Is the latency stable enough for usage in synchronous trigger systems?

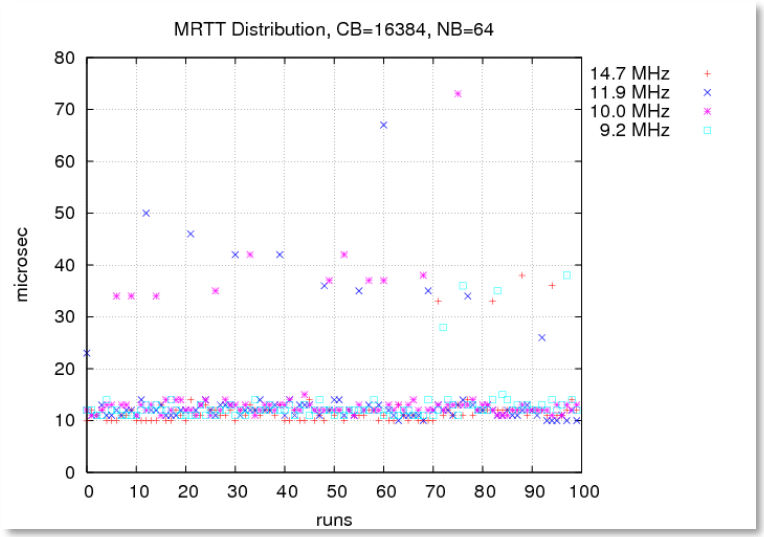
- **Computing power**
  - Is the GPU fast enough to take a trigger decision at tens of MHz events rate?

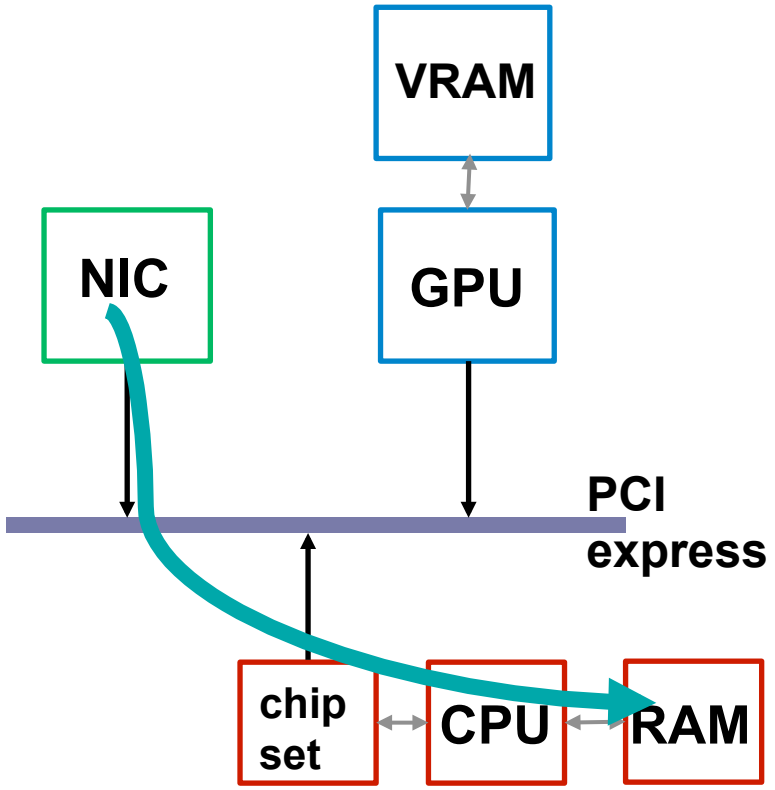
GPU Processing

- Example: packet with 1404 B (few tens of events in NA62 RICH application)
- T=0

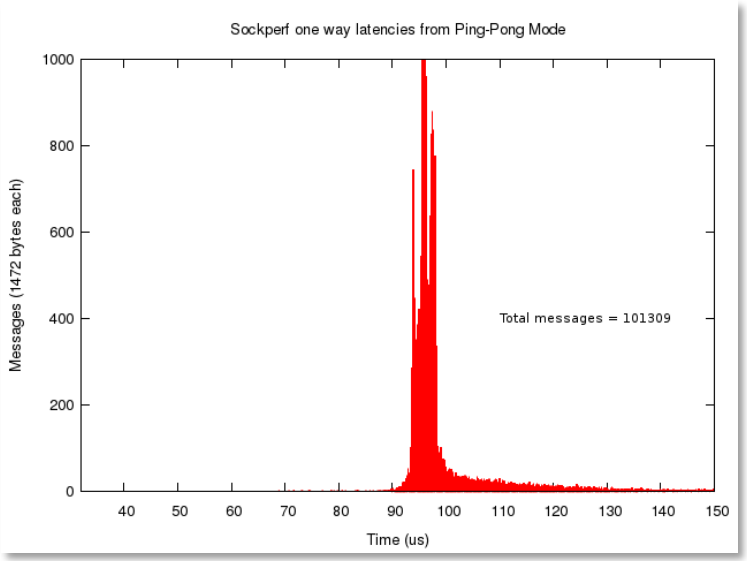


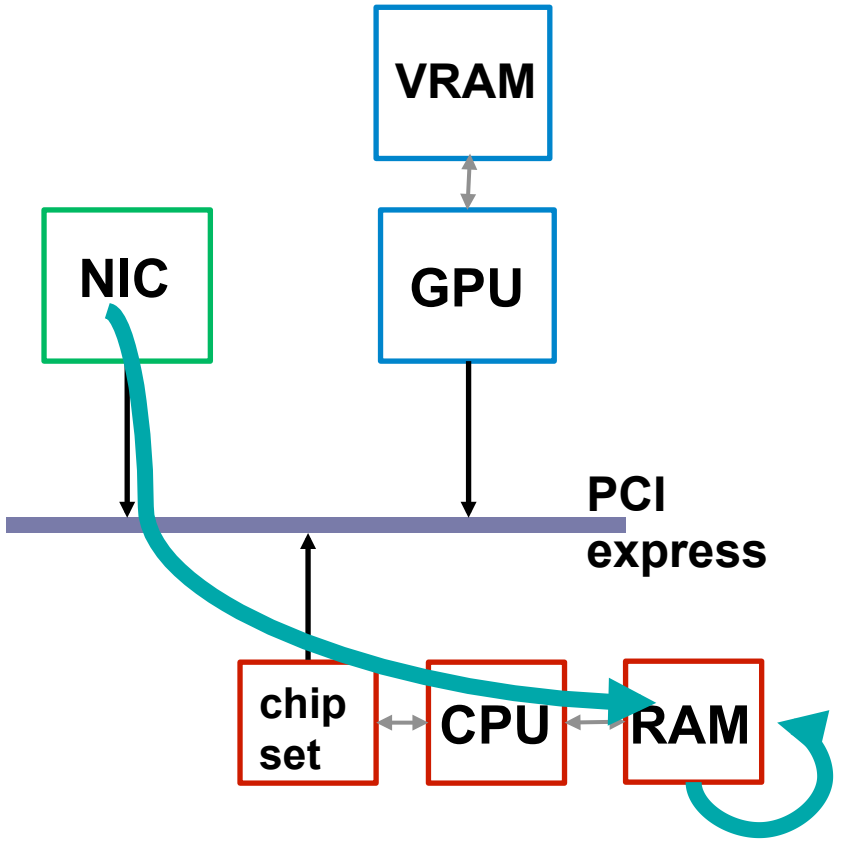
GPU Processing



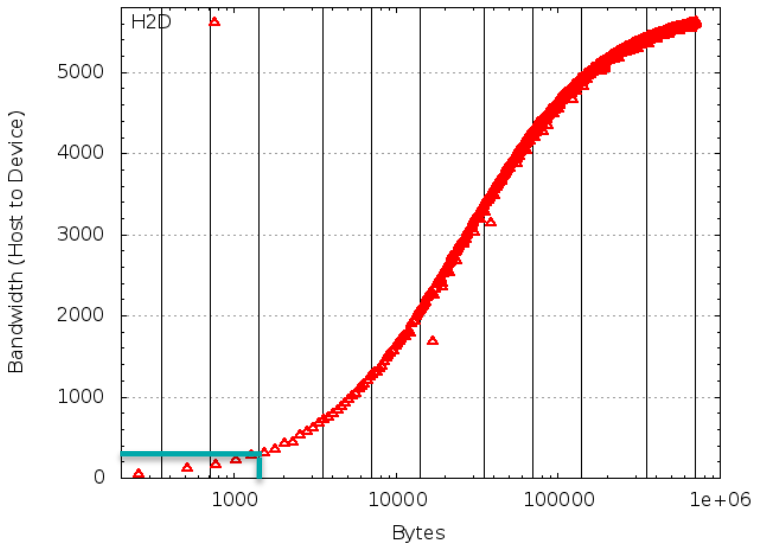


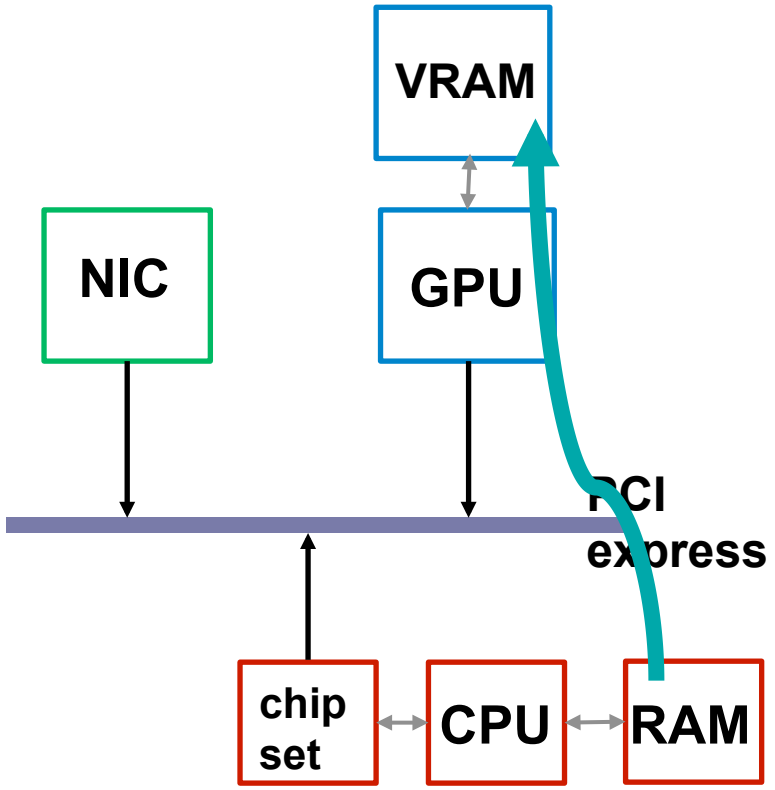
GPU Processing



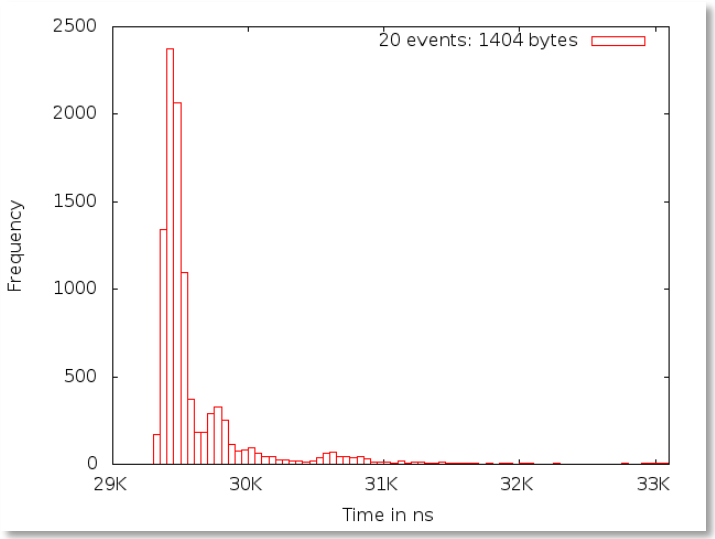


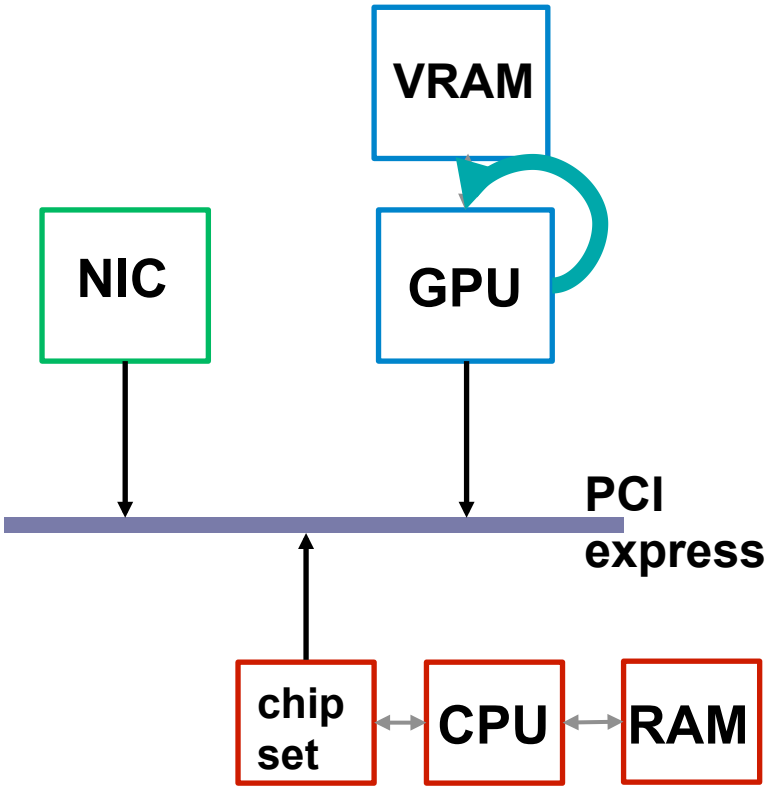
GPU Processing



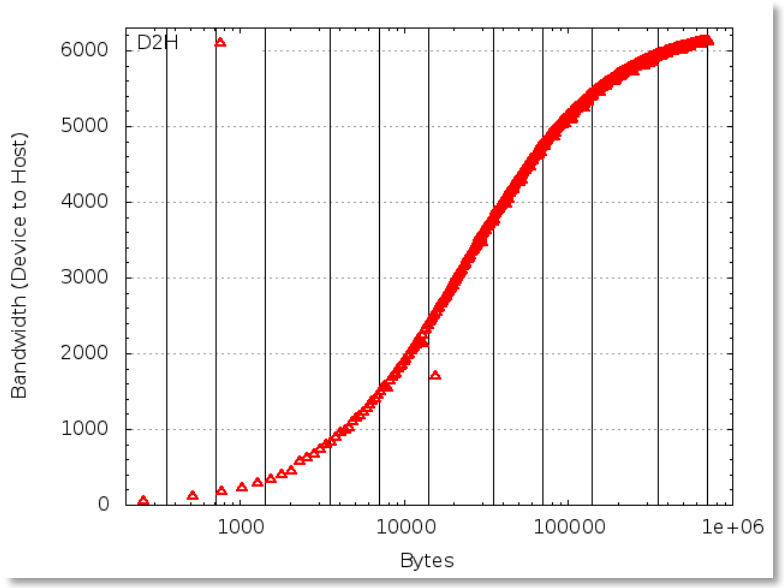


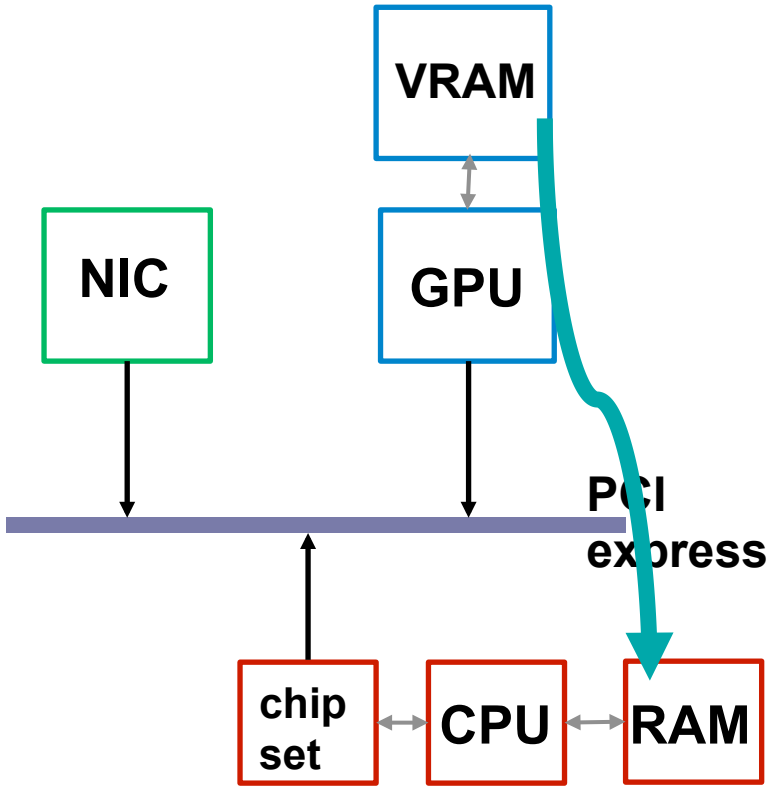
GPU Processing





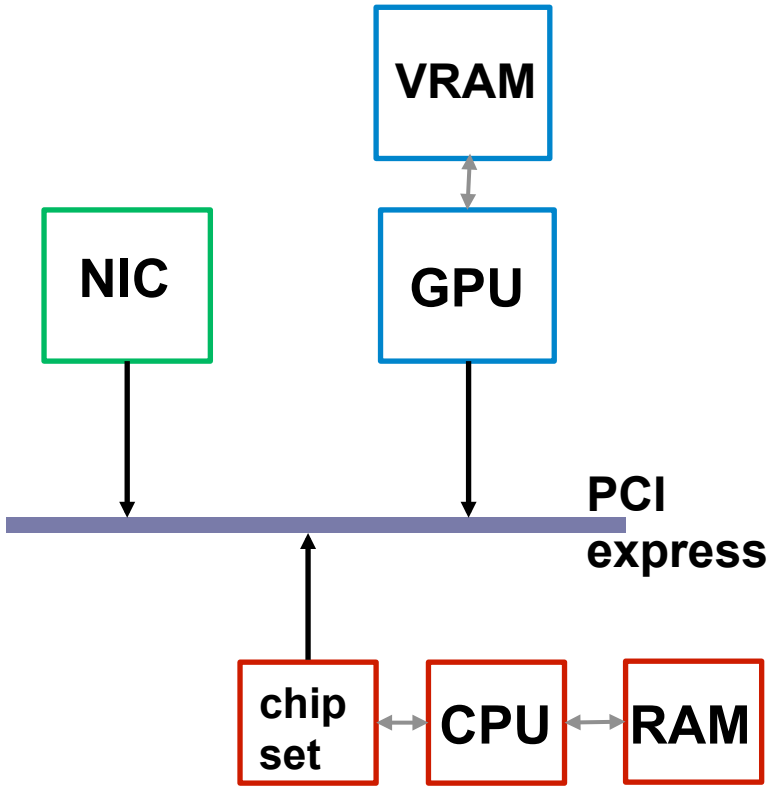
GPU Processing





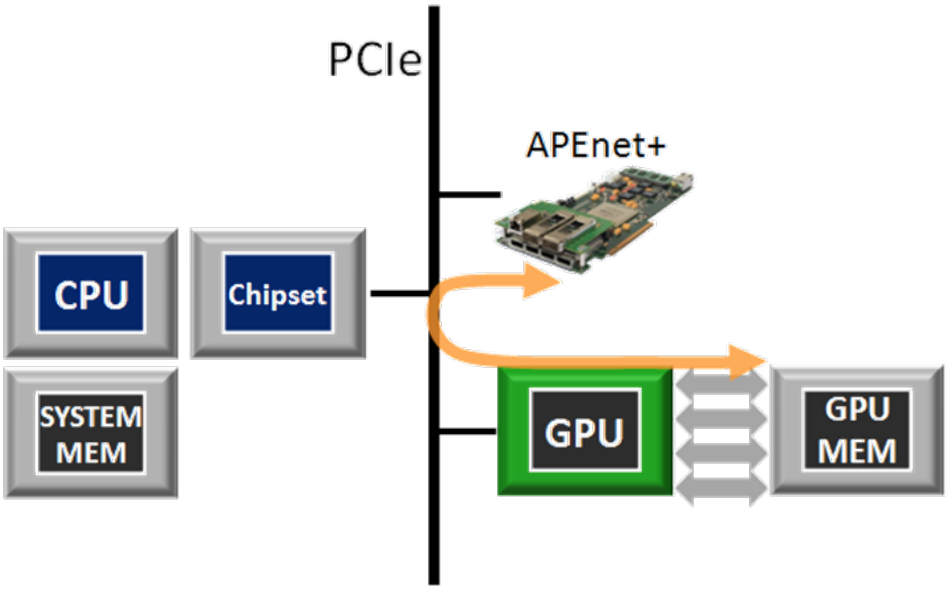
GPU Processing

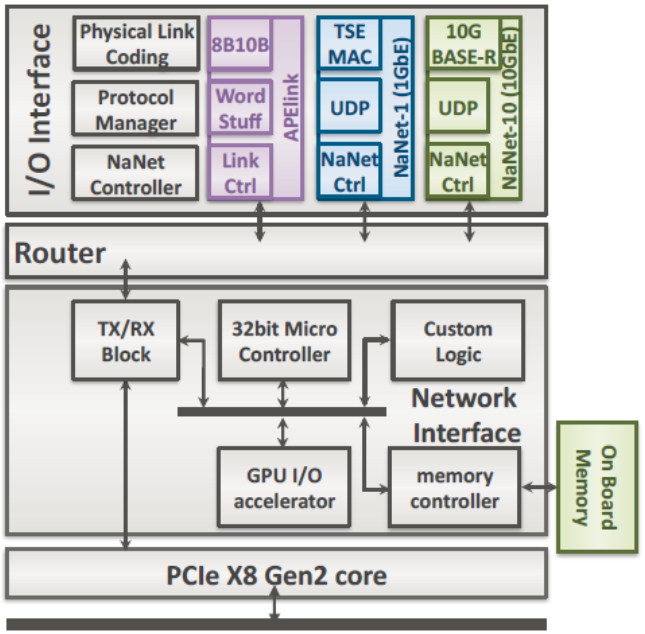
- Latency due to data transfer from the detector to the system is bigger than the latency due to GPU computing
- It scales almost linearly (apart from the overheads) with the data size while the latency due to computing can be hidden exploiting the huge resources
- Communication latency fluctuations quite big

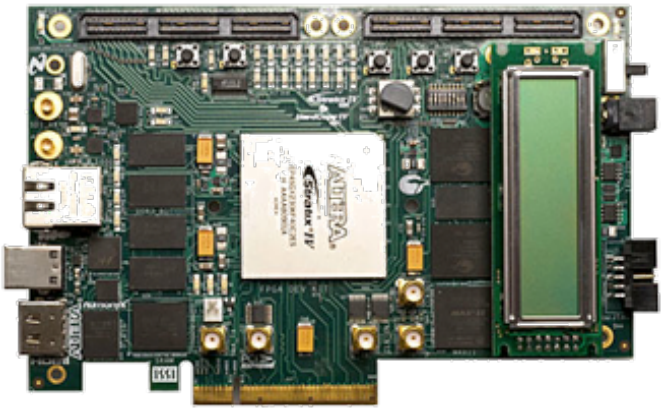


First solution: NANET

- NANET is an FPGA-based NIC that has GPUDirect RDMA capabilities





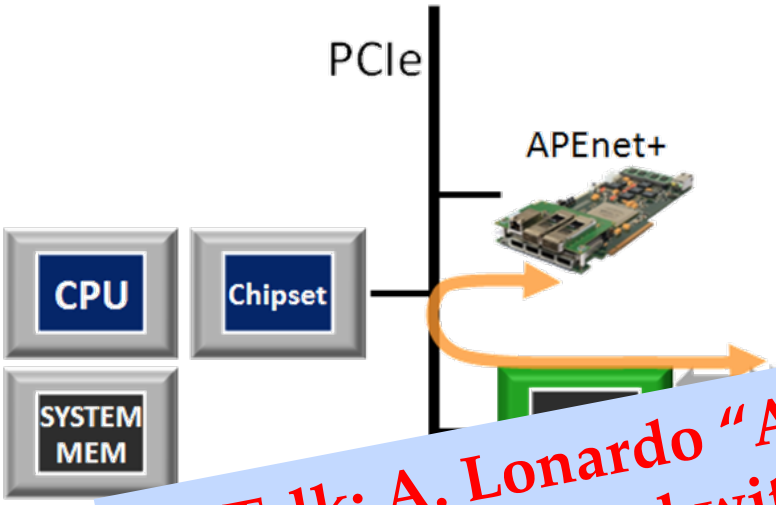


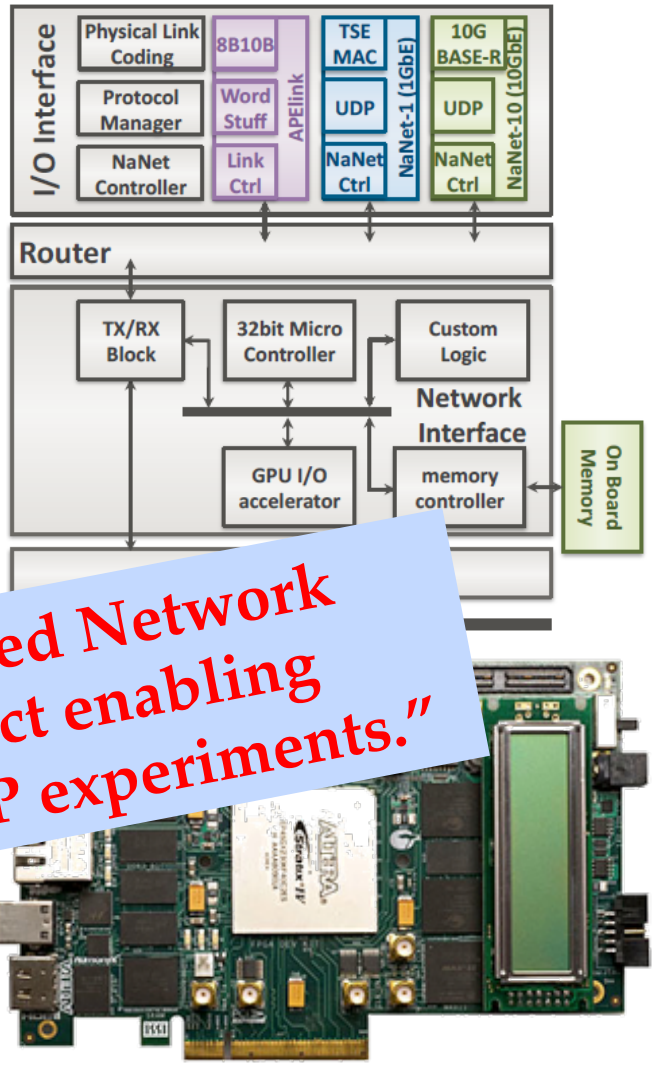
APEnet Rome Group

R. Ammendola *et al.*, *JINST* 9 C02023, 2014

First solution: NANET

- NANET is an FPGA-based NIC that has GPUDirect RDMA capabilities





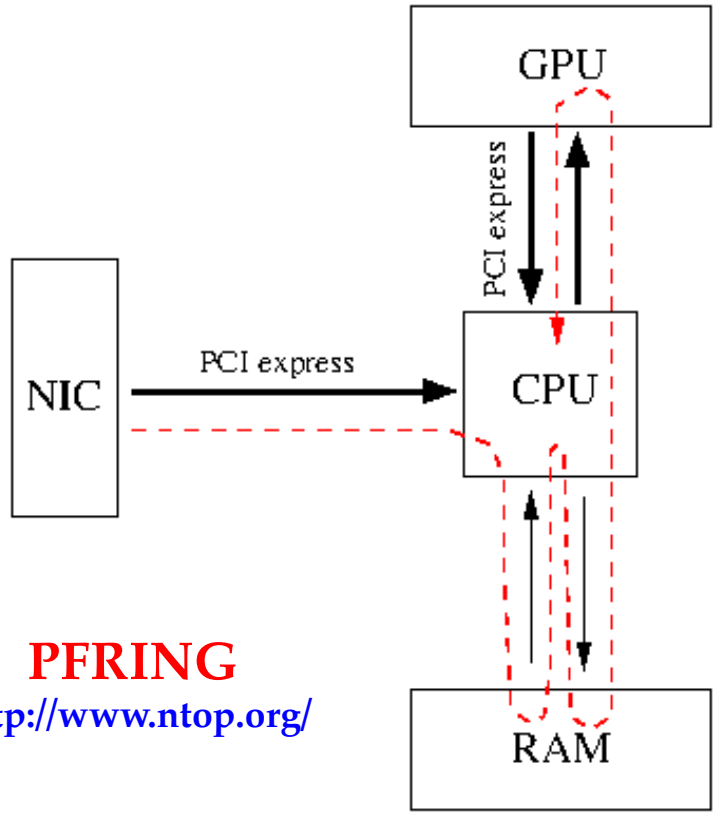
Talk: A. Lonardo "A FPGA-based Network Interface Card with GPUDirect enabling real-time GPU computing in HEP experiments."

...ine Group

R. Ammendola *et al.*, *JINST* 9 C02023, 2014

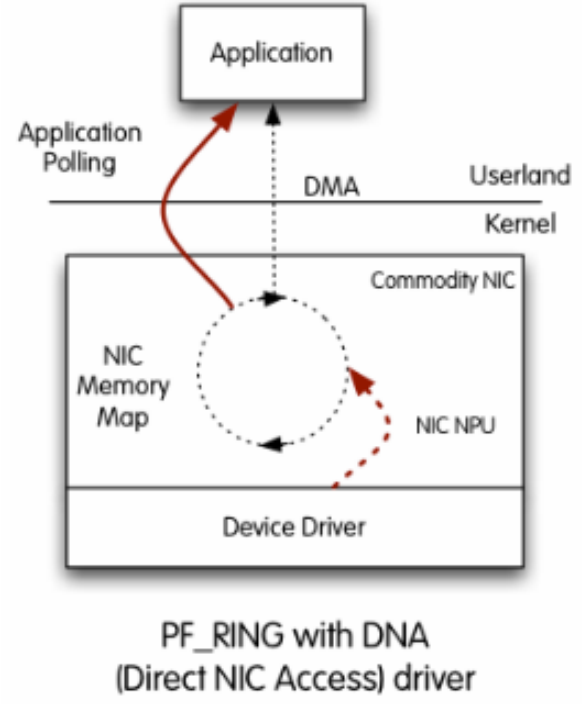
Second solution: PFRING

- PFRING DNA (Direct NIC Access) is a way to map NIC memory to userland so that there is no additional packet copy besides the DMA transfer done by the NIC



PFRING

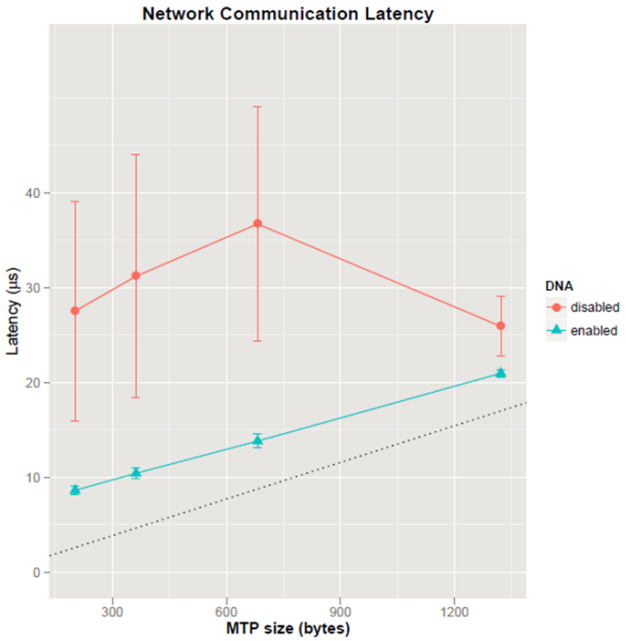
http://www.ntop.org/

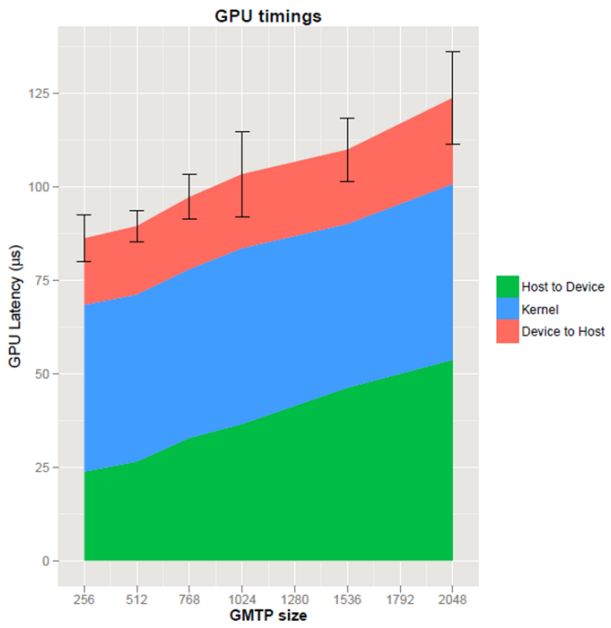


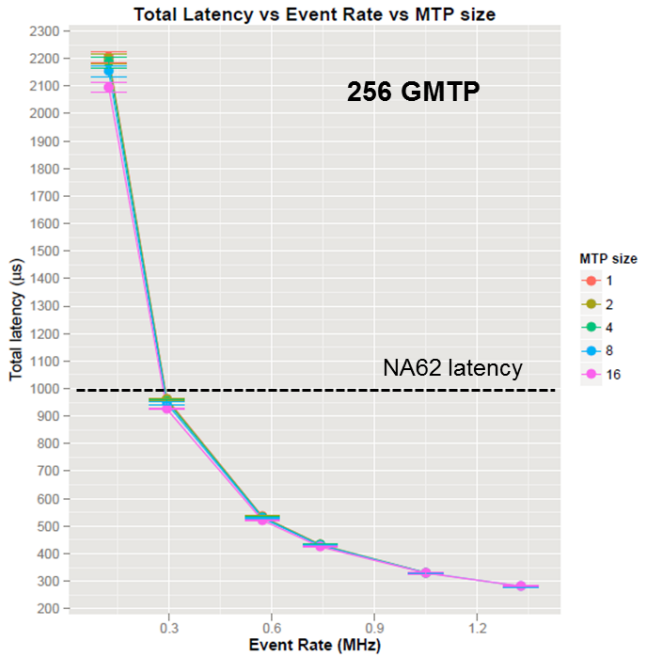
PF_RING with DNA (Direct NIC Access) driver

Results: PFRING

- Latency reduced and negligible fluctuations
- The total latency is given as a function of the number of events to buffer before the start of GPU computation
- For real application the "working point" depends on the events rate and event dimension



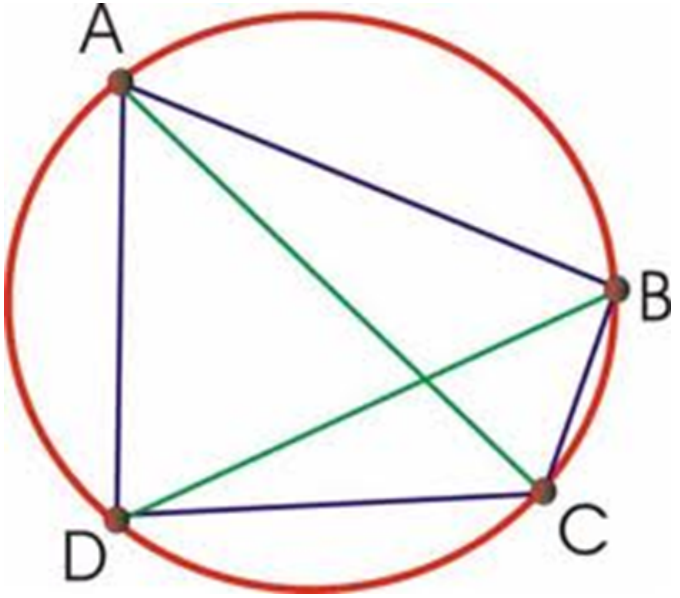




L0 RICH trigger algorithm

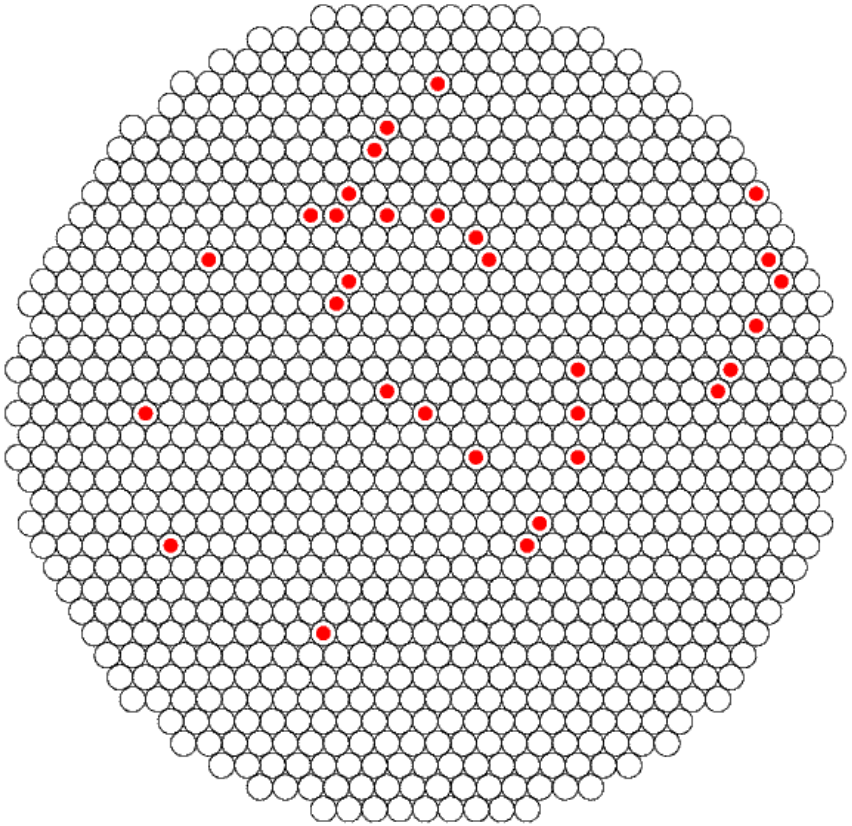
- Requirements for an on-line RICH reconstruction algorithm:
- Trackless
  - No information from the tracker
  - Difficult to merge information from many detectors at L0
- Multi-rings
  - Many-body decay in the RICH acceptance
- Fast
  - Non-iterative procedure
  - Events rate at a level of ~10 MHz
- Low latency
  - Online (synchronous) trigger
- Accurate

Almagest



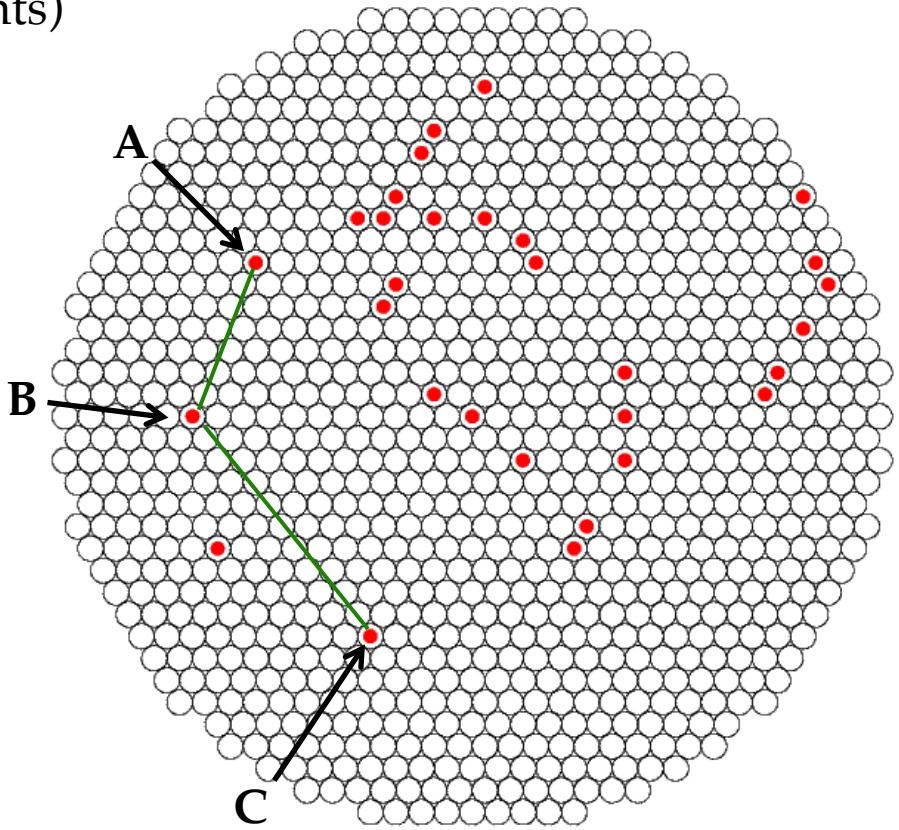
- New algorithm (Almagest) based on Ptolemy's theorem: *"A quadrilater is cyclic (the vertex lie on a circle) if and only if is valid the relation: AD*BC+AB*DC=AC*BD "*
- Design a procedure for parallel implementation

Almagest: example



Almagest: example

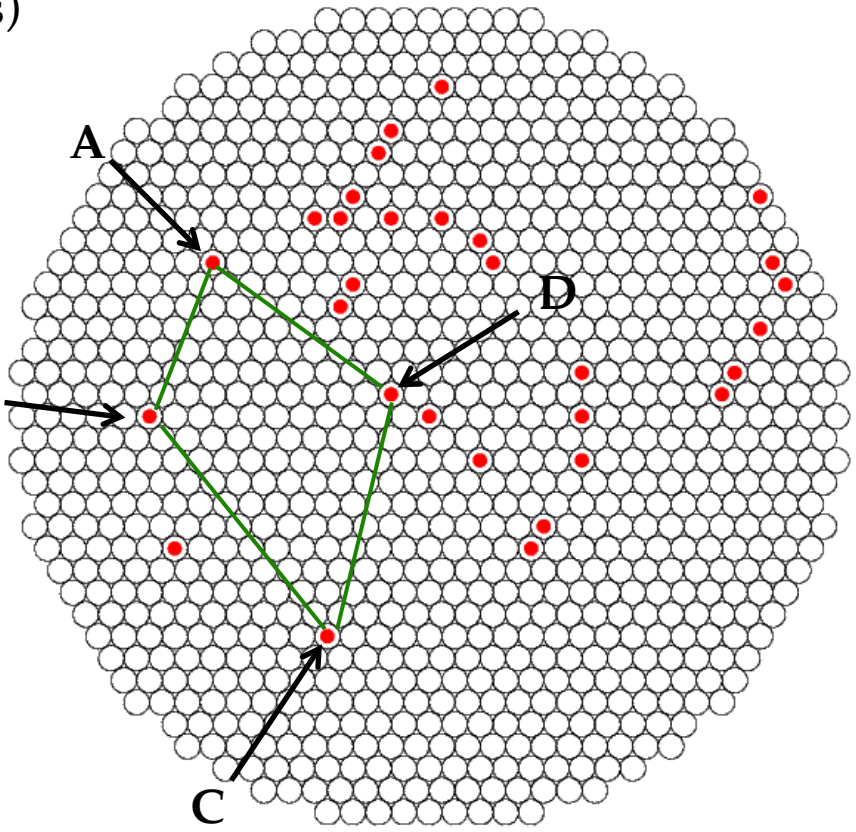
i) Select a *triplet* (3 starting points)



Almagest: example

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

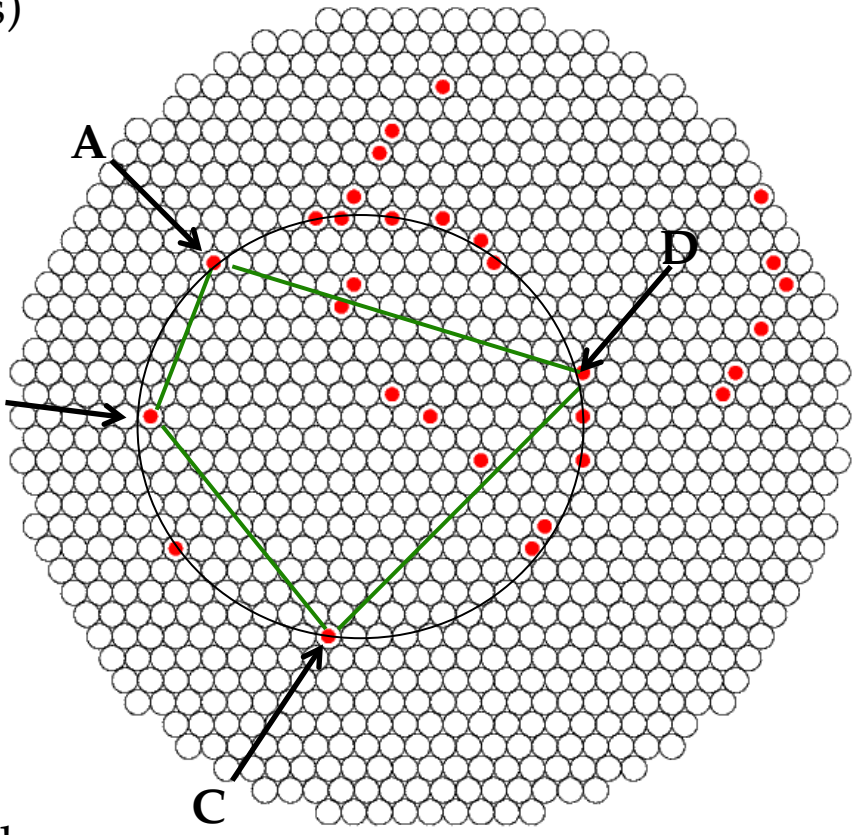


Almagest: example

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

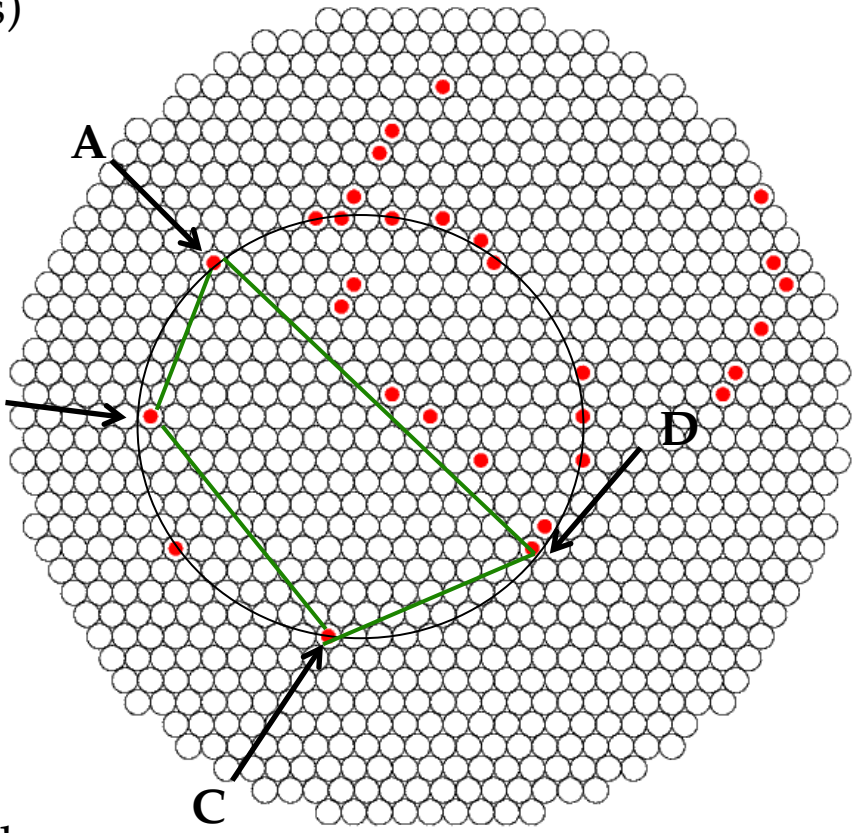


Almagest: example

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



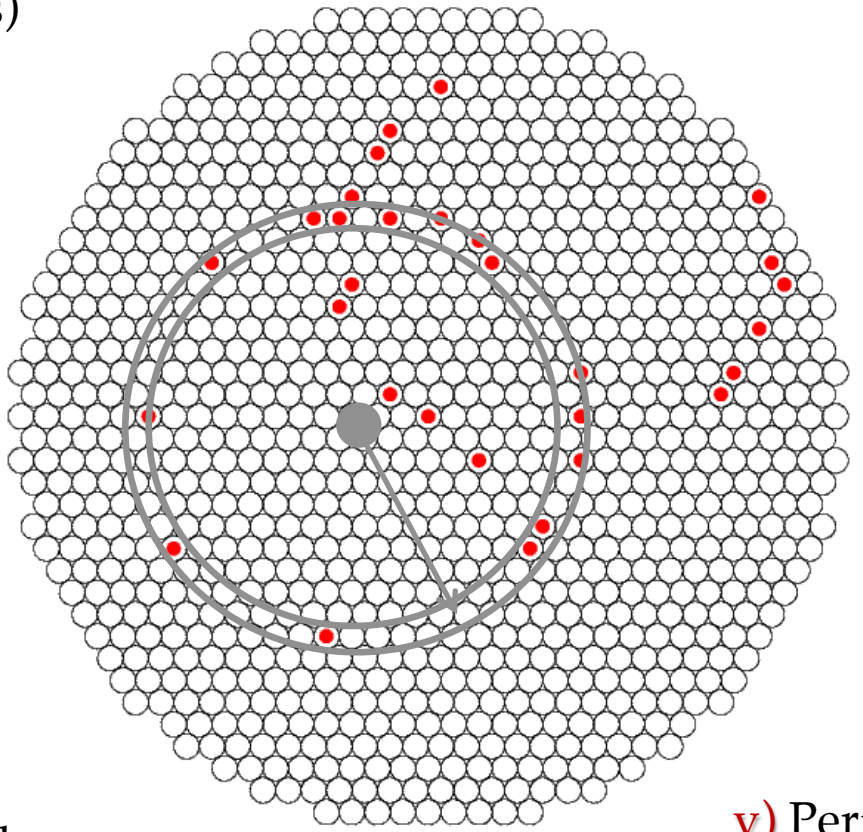
iv) …again…

Almagest: example

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…

v) Perform a single ring fit

Almagest: example

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

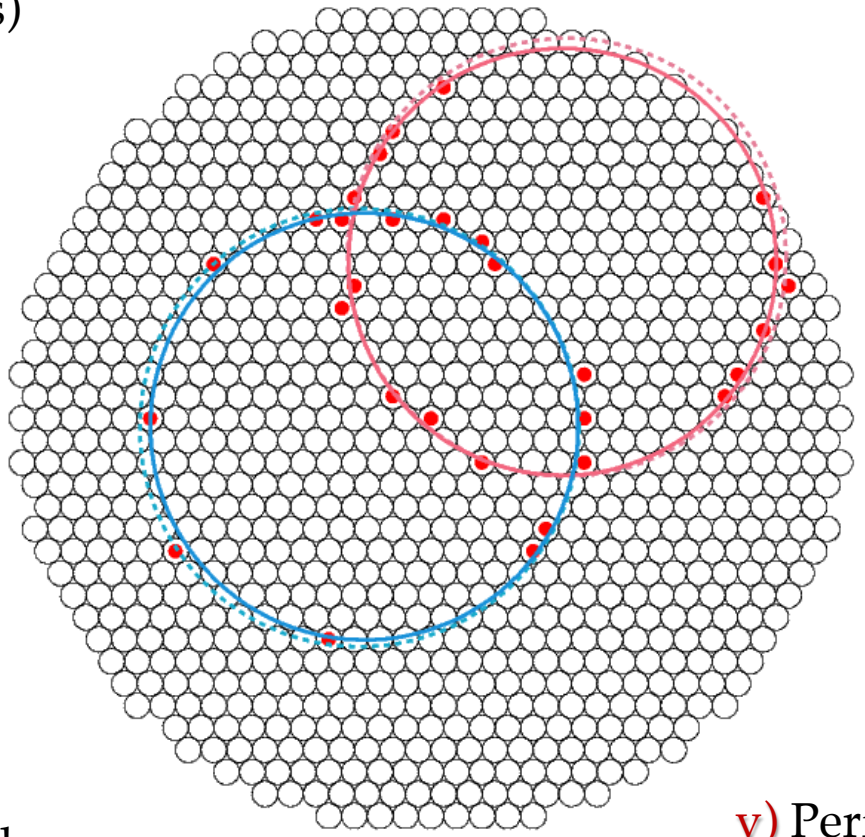
ii) Loop on the remaining points: if the next point does not satisfy the Ptolemy's condition then reject it

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

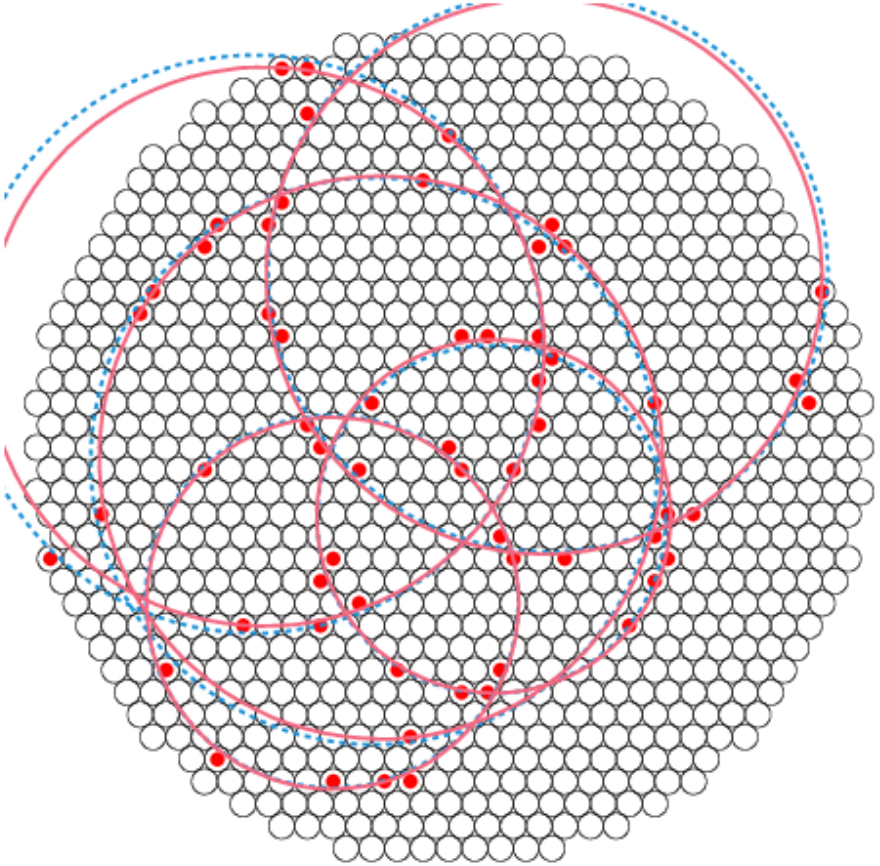
iv) …again…

v) Perform a single ring fit

vi) Repeat by excluding the already used points

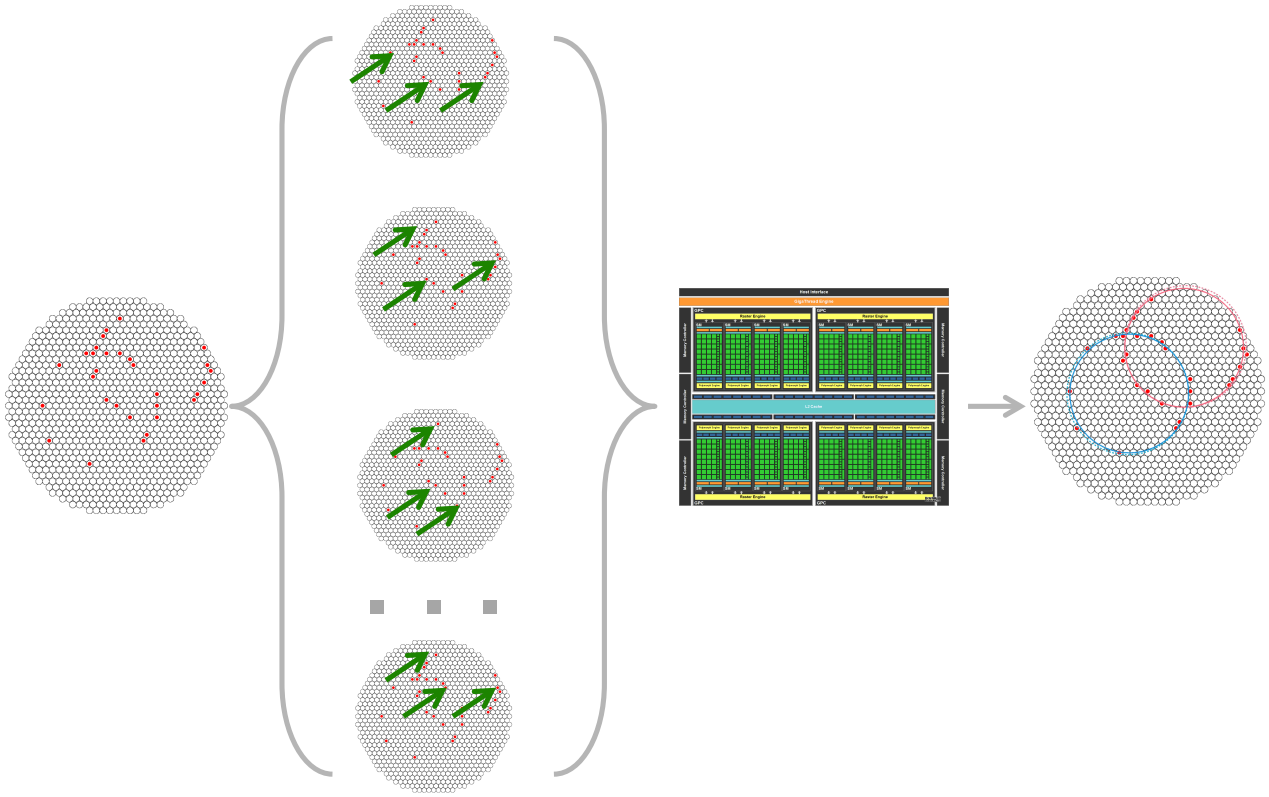


A more complicated example



Almagest on GPU

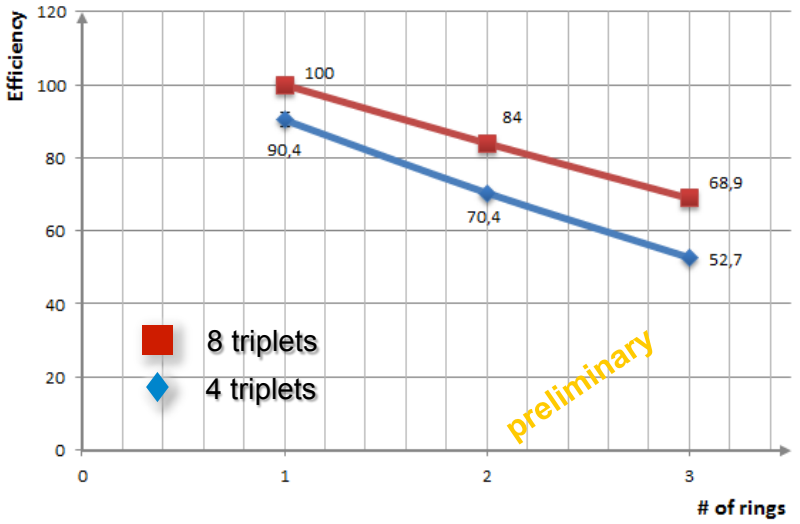
- Very high parallelism
  - Huge number of computing cores (>2000)
  - Huge memory bandwidth

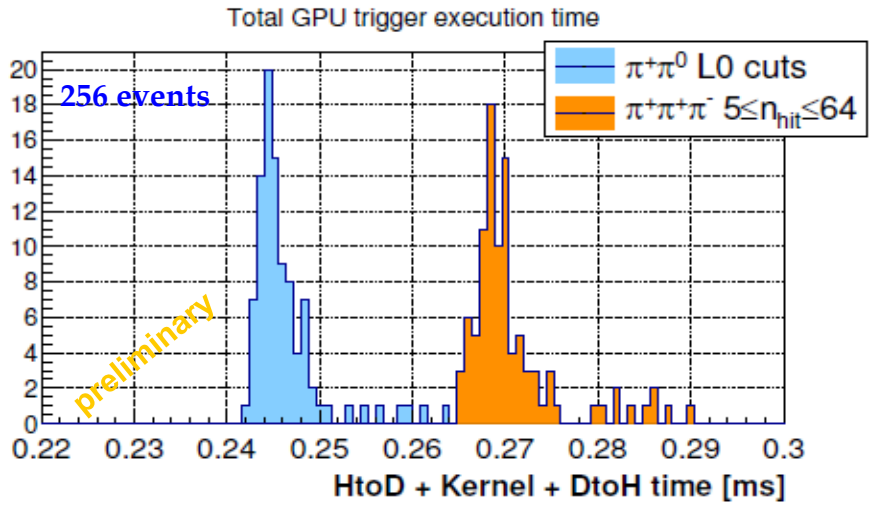


- Two levels of parallelism
  - Several triplets run in parallel
  - Several events at the same time

Almagest: implementation

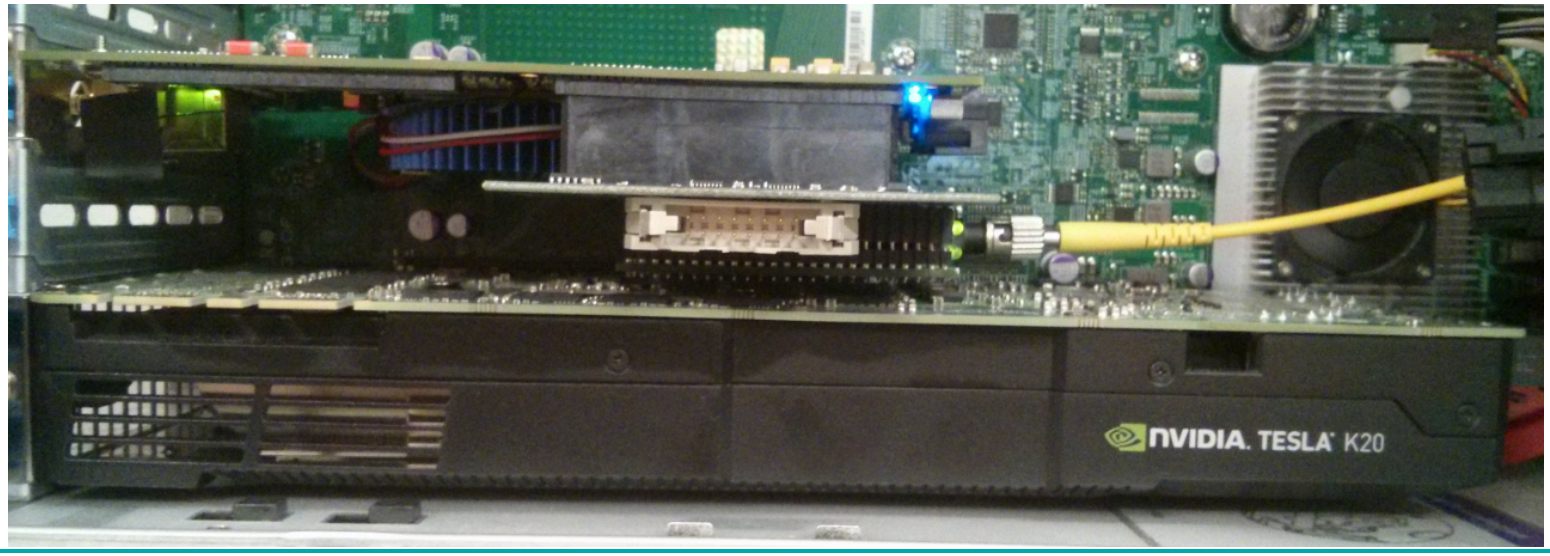
- Tests on NVIDIA Tesla K20 GPU
- Total computing time order **a few μs** per event (on single GPU)
- Good efficiency (using 8 triplets)
  - Room for improvement
- Further tests ongoing to study noise immunity, bias, efficiency a function of the number of hits, etc…





Next steps

- Receive the TTC stream (timing and trigger) from the experiment
  - TTC interface board with HSMC connector
- Integration in the NA62 Trigger and DAQ system
  - First test during dry run in August
  - Parasitic test during NA62 experimental run in October



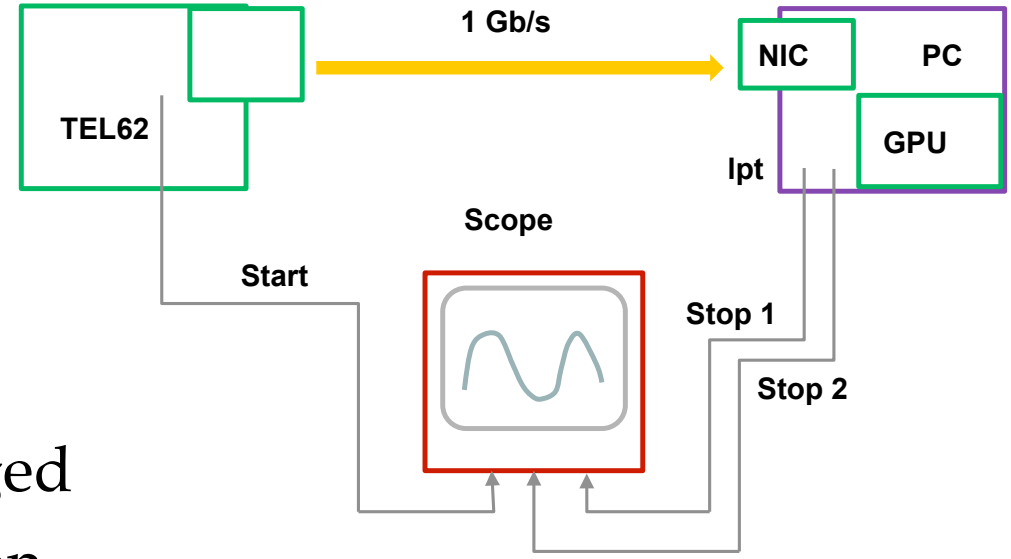
Conclusions

- The use of GPUs in HEP trigger systems could give several advantages, but processing performances and latencies should be carefully studied
  - Data transfer is the dominant contribution
- Construction of a demonstrator L0 processor for the NA62 RICH is under way
  - Cherenkov rings pattern recognition within the total L0 latency of 1 ms seems possible
- Integration with the NA62 Trigger and DAQ system
  - First tests during dry run in August 2014
  - Parasitic data taking during NA62 experimental run starting October 2014

SPARES

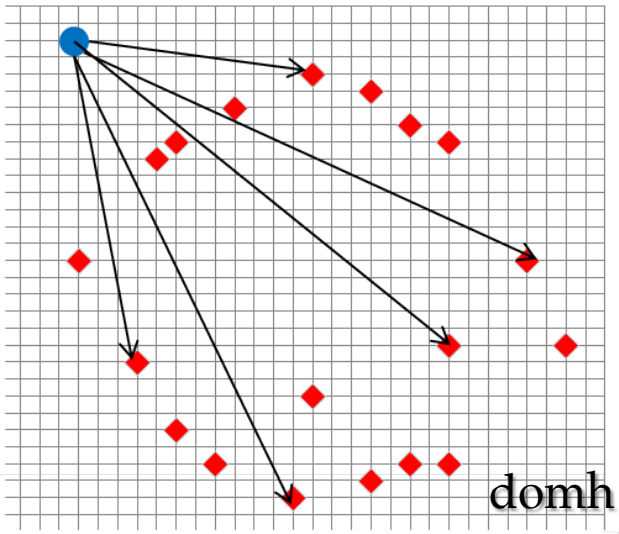
Latency measurement

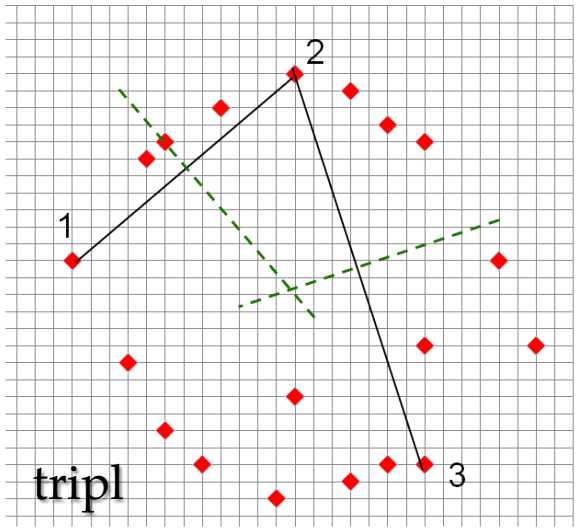
- Events simulated in TEL62
- Grouped in MTP
- Start signal rises with the first event in the MTP
- First stop: packet arrival
- Buffering in the PC RAM: GMTP depth can be changed
- Second stop: after execution on GPU (single ring reconstruction kernel)
- The precision of the method has been evaluated as better than 1 μs

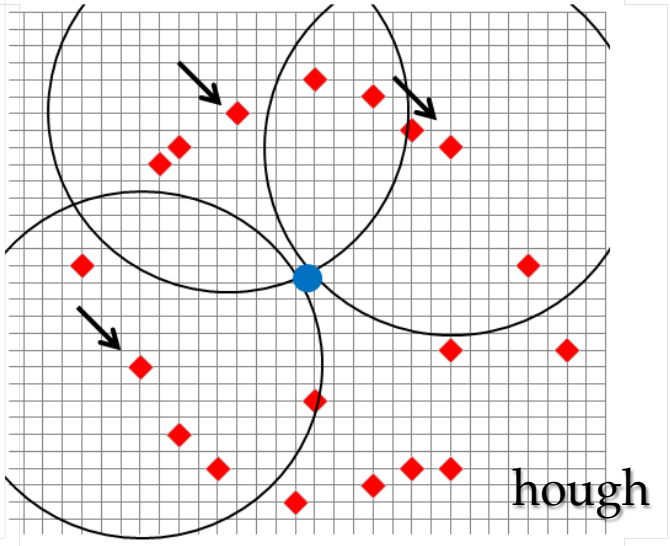


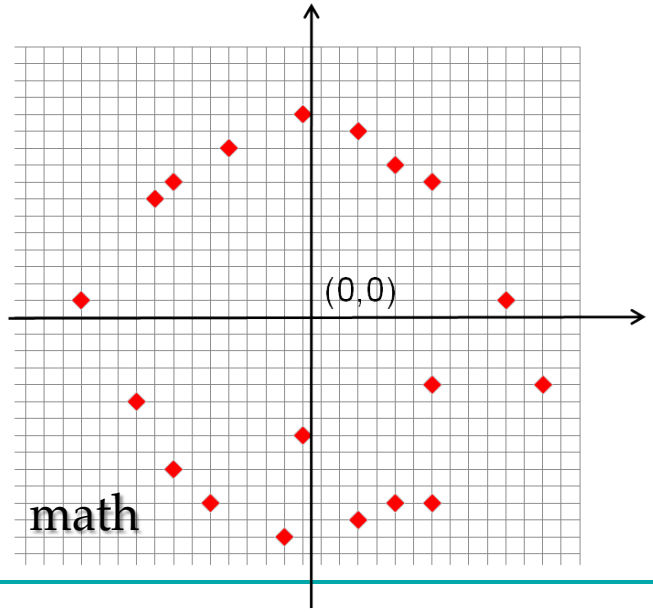
- Dual processor PC:
  - XEON E5-2620 2Ghz
  - I350T2 Gigabit card
  - 32 GB
  - GPU K20c (2496 cores) PCIe v2 x16

Algorithms for single ring



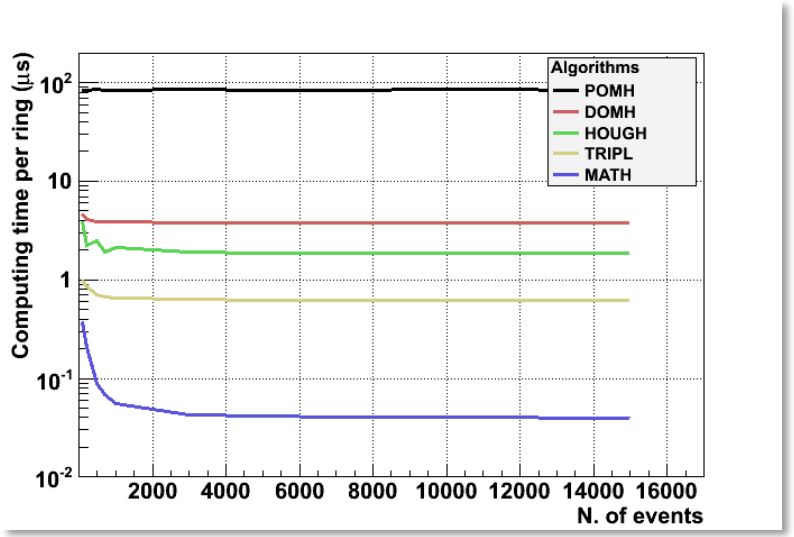


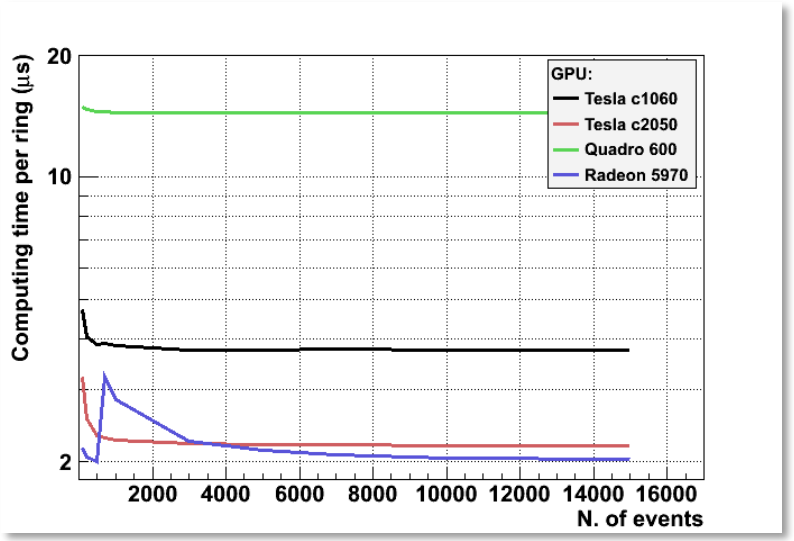




Processing time

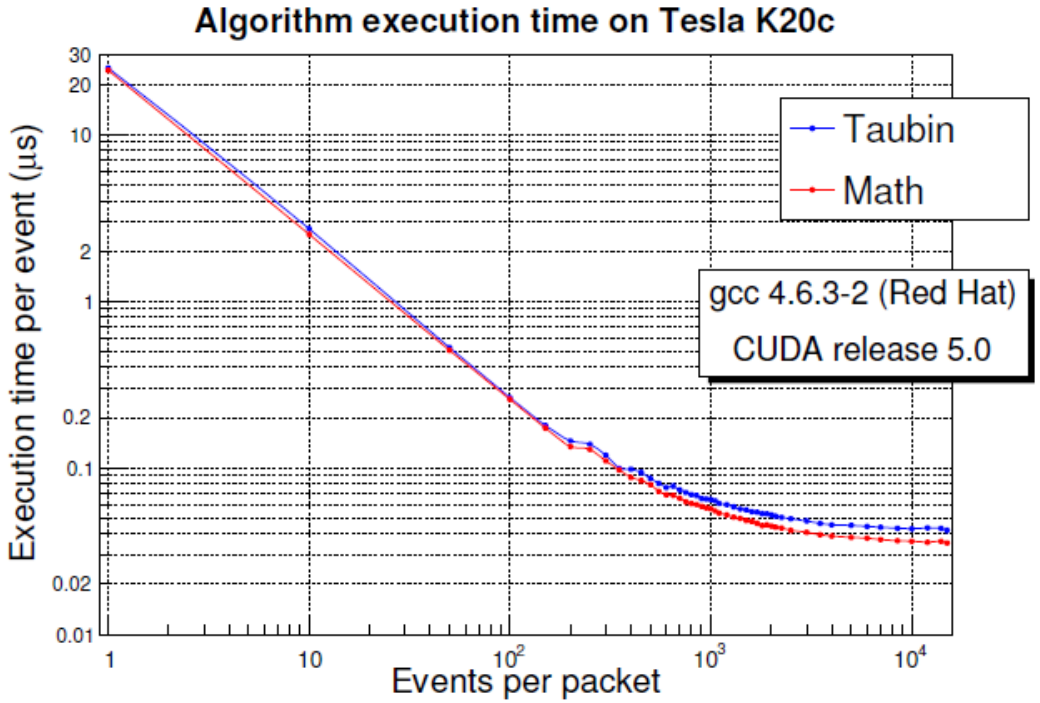
- Using Monte Carlo data, the algorithms are compared on Tesla C1060
- For packets of >1000 events, the MATH algorithm processing time is around 50 ns per event
- The performance on DOMH (the most resource-dependent algorithm) is compared on several GPUs





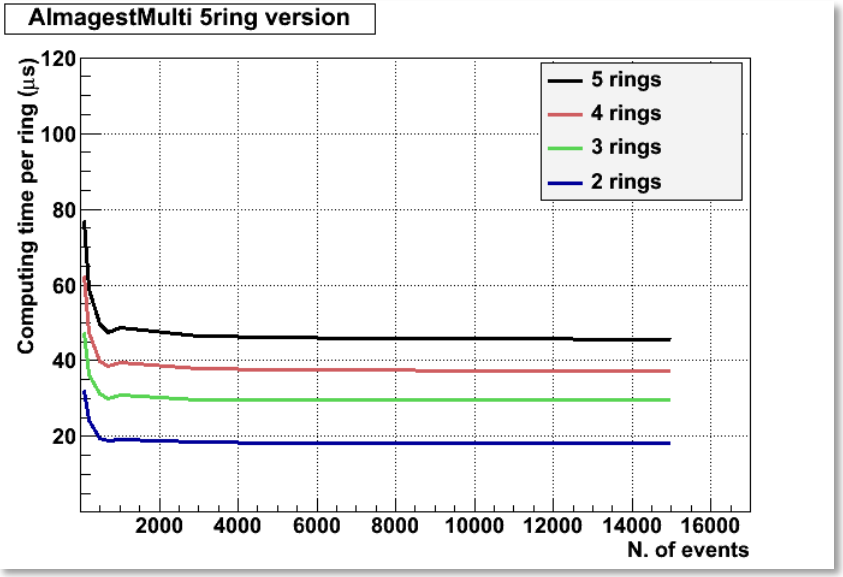
Single ring algorithms

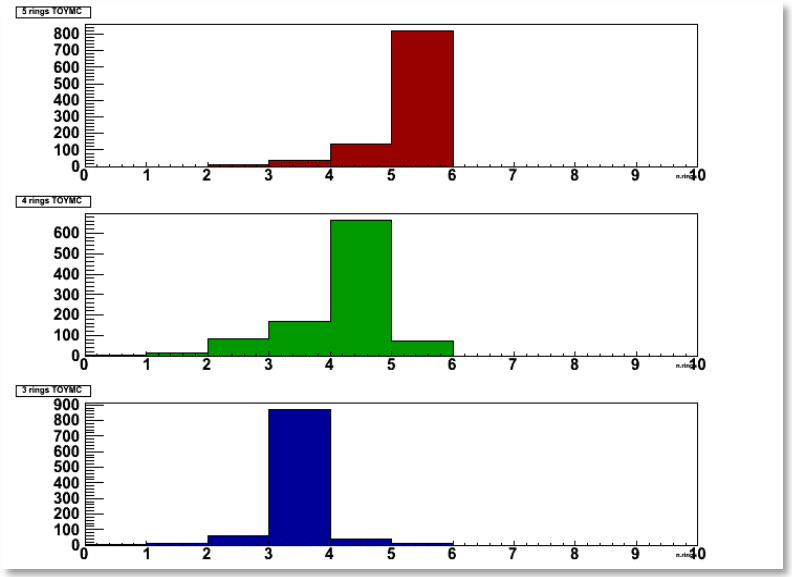
- Crowford method ("math"):
  - Translate in the center of mass
  - Least square minimization → linear
- Taubin method:
  - More efficient: minimize the bias introduced by the Kasa related methods (minimization of simple algebraic distance)
  - Resolution slightly better (on identified rings)
- The difference of computing time on the GPU is at the level of 10 ns per event



N(=hits) triplets

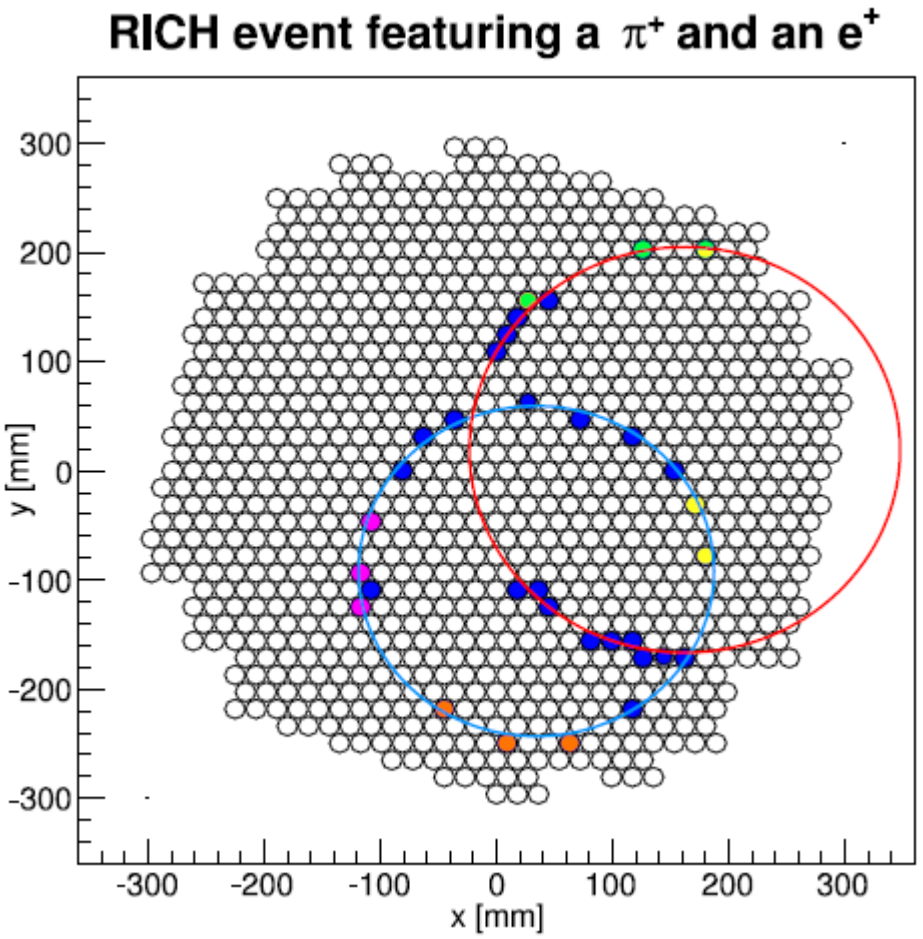
- Number of triplets equal to the number of hits.
- Relatively high efficiency.
- Computing time depends on number of rings (different number of GPU cores per events)
- Results on TESLA C1060 (240 cores, less than 1 Tflops)
- Room for optimization





4 selected triplets

- Only 4 triplets per event are used: left, right, up and down
- Further cuts to avoid too close hits



4 selected triplets

- Stability with small noise (studies are ongoing)
- Inefficiency due to the order in choosing the rings.
- Dependence on the cuts to define the triplets.