“GPU for triggering in HEP esperiments”

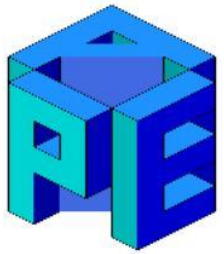
20th Real Time Conference (RT2016)

Padova, 10.06.2016

Michele Martinelli

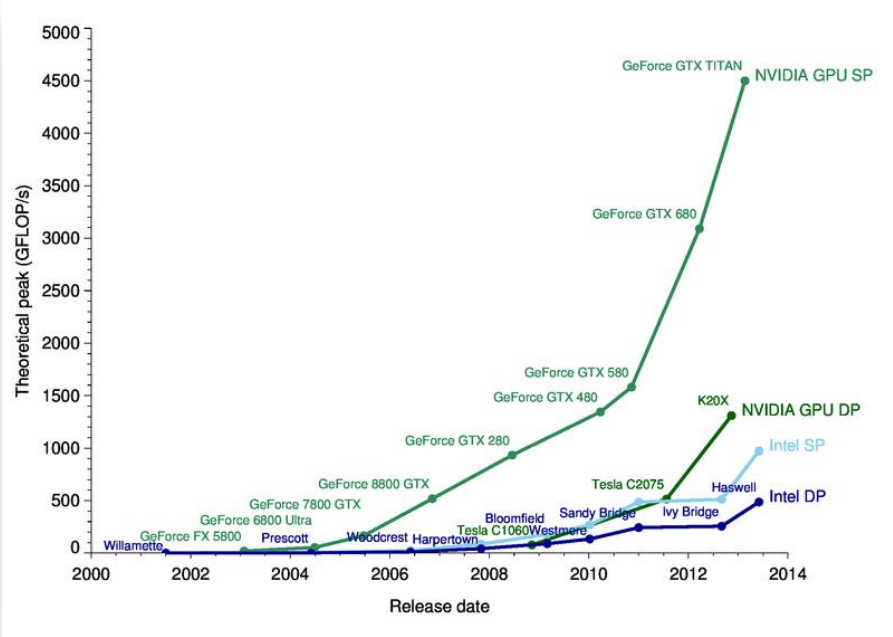
PhD Student

on behalf of the GAP and NaNet collaborations

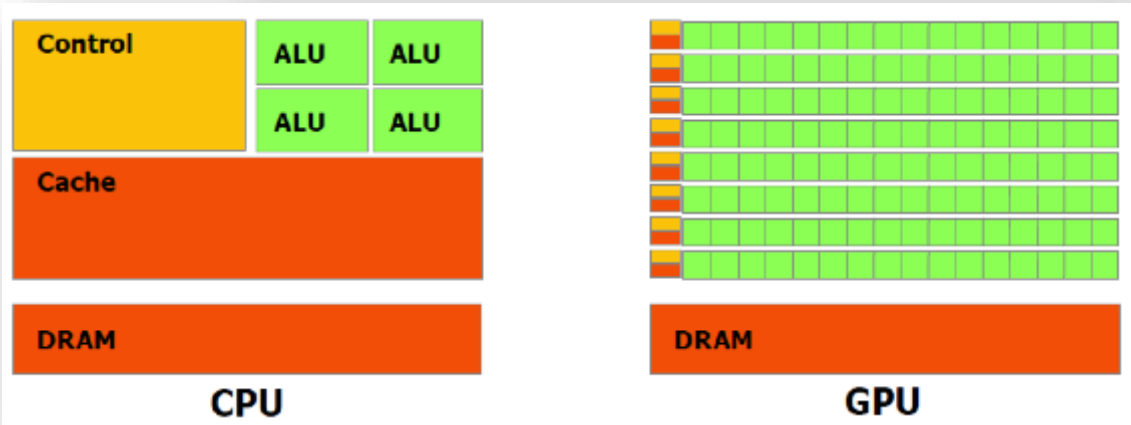


Graphics Processing Unit (GPU)

- 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



- Less control units, many more ARITHMETIC LOGIC units.

GPUs in High Energy Physics: Multi Level Trigger systems

- **HEP Trigger systems offer a complex environment in terms of rate, bandwidth and latency**
  - **A lot of data comes from the detectors, we need to separate the interesting data from the (huge) background (uninteresting events, noise,…).**
- **This task is typically accomplished in a series of stages (levels), for simplicity:**
  - **Low-level trigger: high event rate, synchronous, tight time budget, custom electronics**
  - **High-level trigger: lower event rate, larger time budget, software tasks on commodity processors**
- **GPUs can be exploited in the high-level trigger to speed-up processing and/or reduce the number of nodes in the farm (e.g. ATLAS and CMS)**
- **The topic of this talk is on the use of GPUs in low-level trigger systems.**

GPUs in High Energy Physics: Low Level Trigger systems

Their implementation is less straightforward:
hard realtime
online computing

High event rate (tens of MHz)

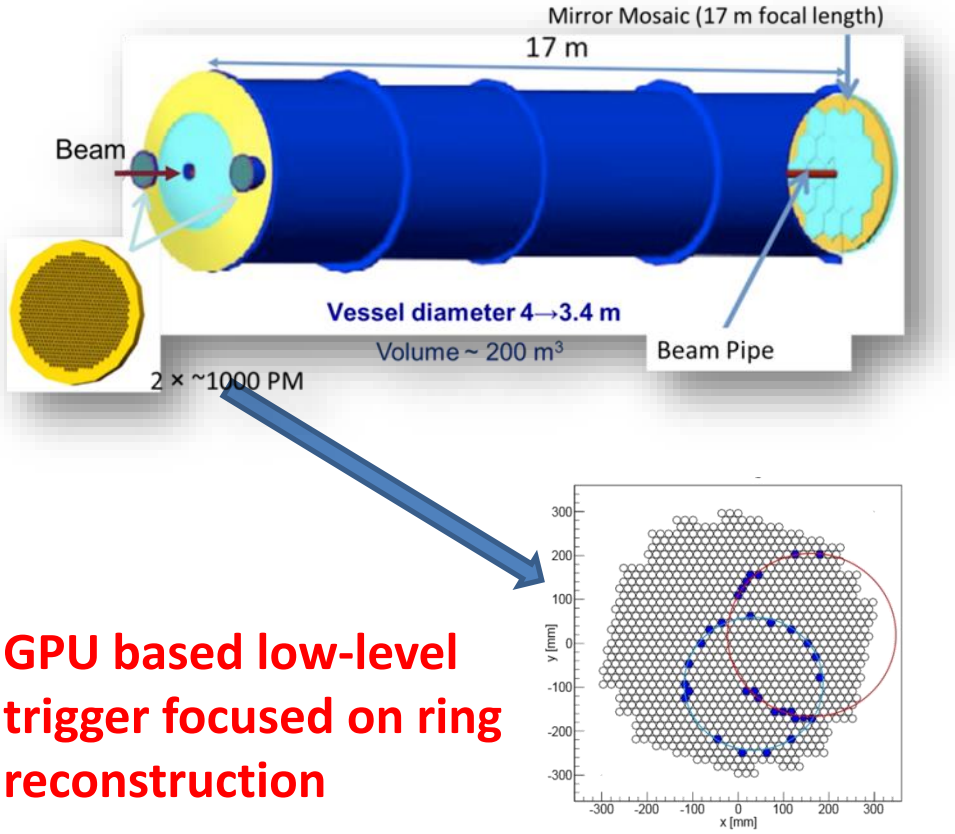
are GPUs fast enough in taking decisions?

Tiny latency

is GPU processing latency small and stable enough for the given task?

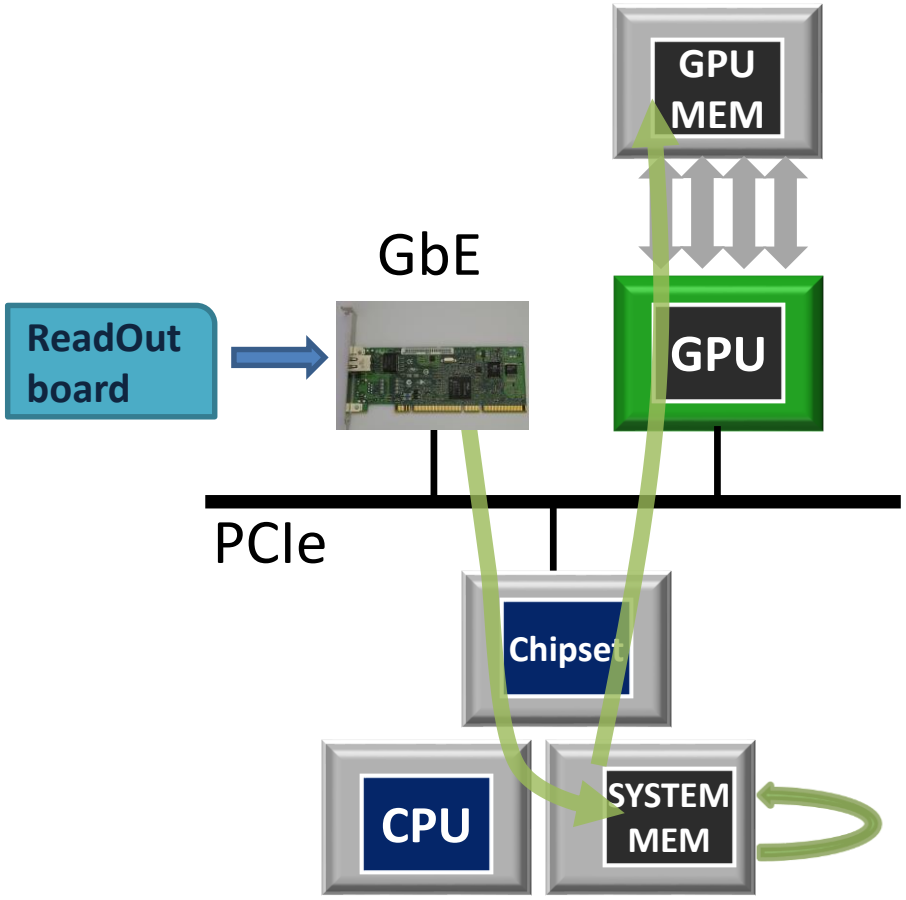
Physics case: Low Level Trigger system in NA62

- **NA62 experiment: measurement of ultra-rare decay of particles.**
- **Low-level trigger: synchronous level must reduce rate from 10 MHz to 1 MHz in real-time, 1 ms time budget.**
- **Real-time production of refined RICH primitives (circles centers and radii) with GPU.**



GPU based low-level trigger focused on ring reconstruction

Communication latency: traditional data flow



Total latency dominated by double copy in Host RAM:

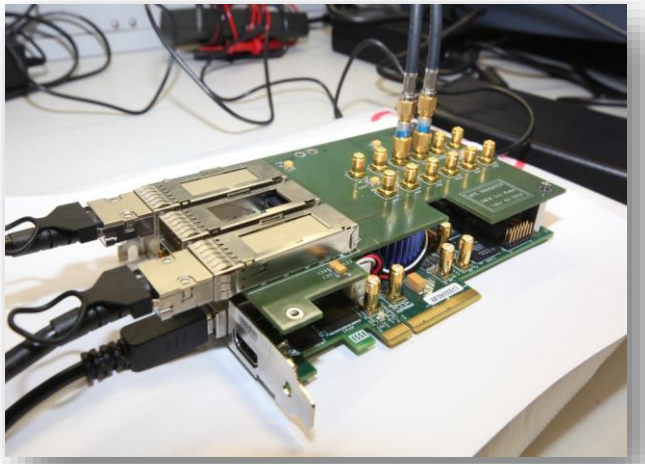
- **Data are copied from kernel buffer to destination buffer in user space**
- **Data are copied from CPU memory to GPU memory**

How to reduce data transfer time:

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

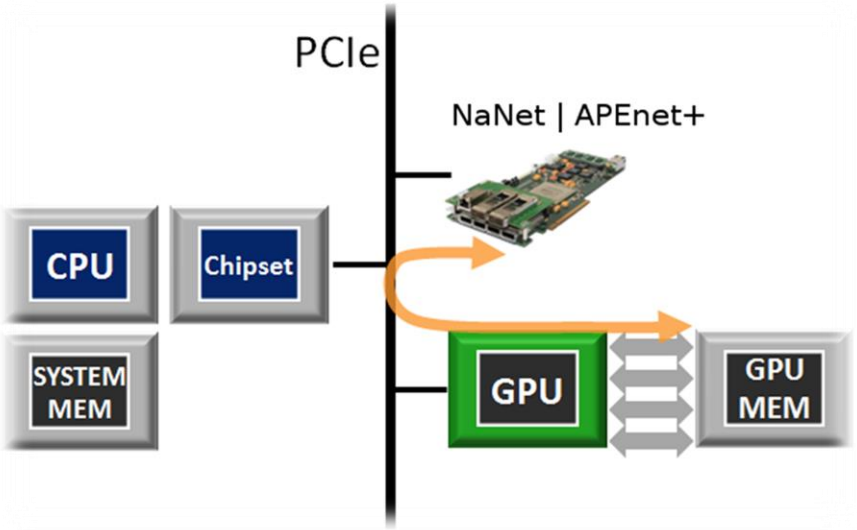
NaNet: a PCIe FPGA-based NIC family for HEP

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



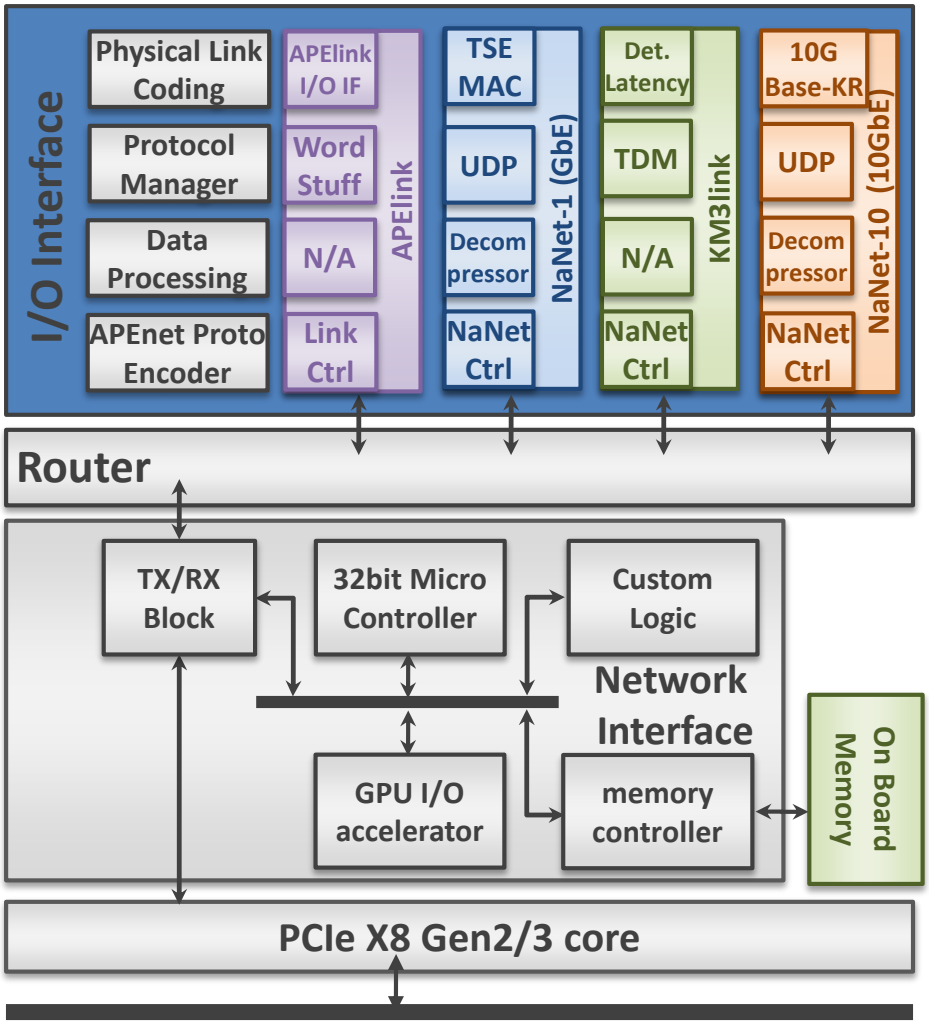
Developed at INFN Roma APE Lab

NaNet-1 is based on Altera Stratix IV dev board



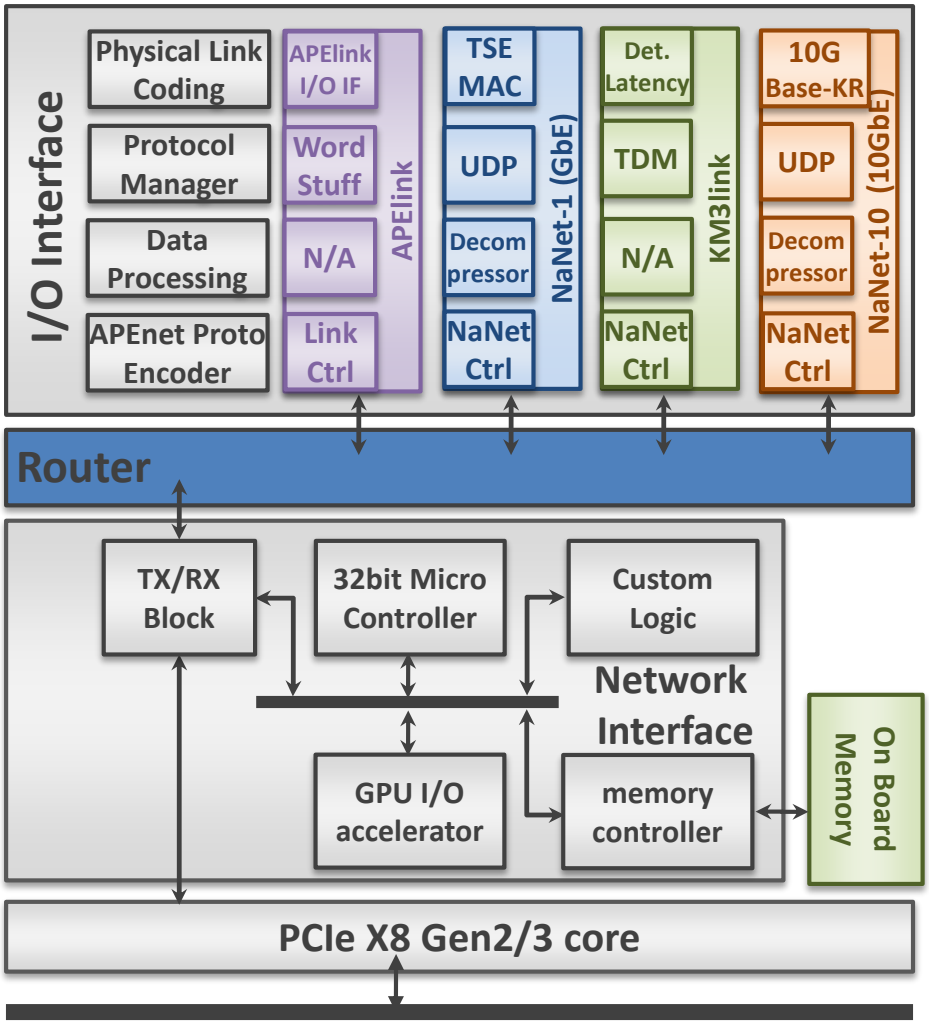
- GPUDirect allows direct data exchange on the PCIe bus with no CPU involvement
- No bounce buffers on host memory
- Zero copy I/O
- Buffers on GPU arranged in a circular list of persistent receiving buffers (CLOP)
- nVIDIA Fermi/Kepler/Maxwell/Pascal

NaNet Design – I/O interface



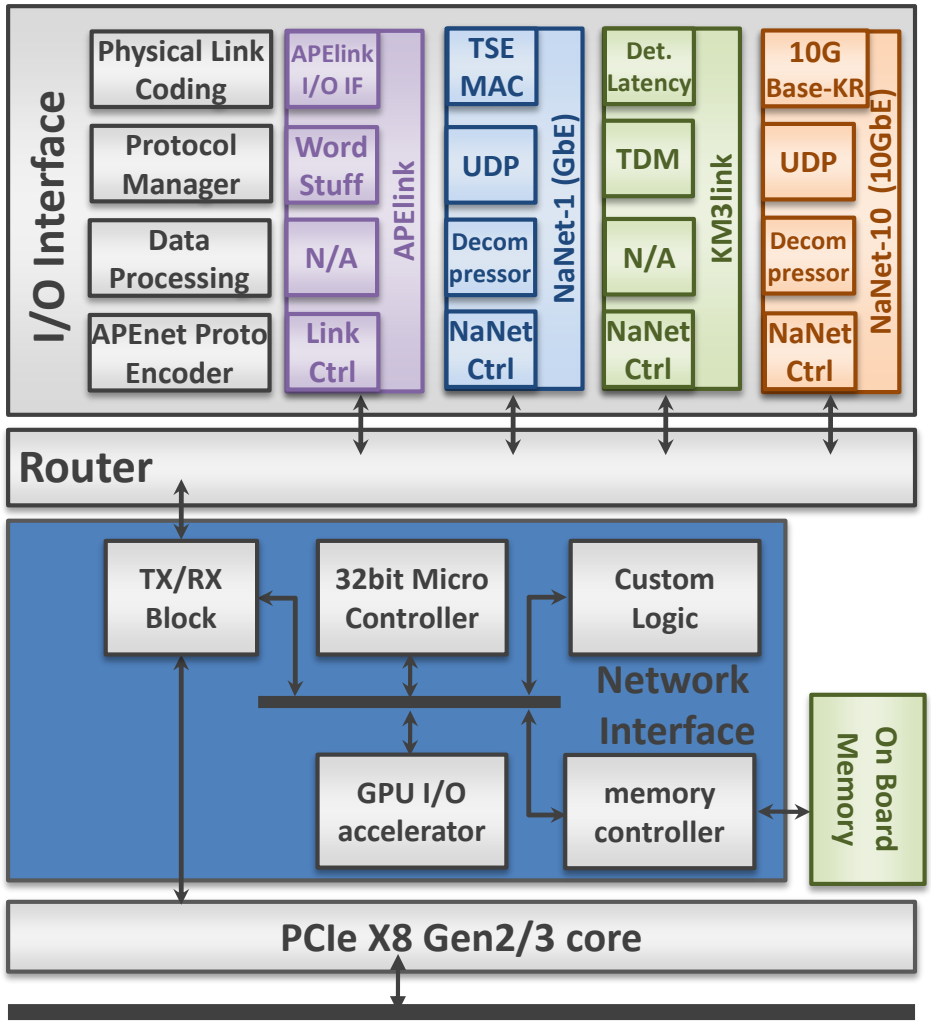
- Physical Link Coding
  - Standard: **1GbE** (1000Base-T), **10GbE** (10Base-KR).
  - Custom: **APElink** (34 Gb/s QSFP), **KM3link** Det. Lat. (2.5 Gb/s optical).
- Protocol Manager
  - Data/Network/Transport layers off-load (**UDP, TDM**)
  - Minimize latency fluctuations.
- Data Processing
  - Application-specific processing on data stream.
  - **e.g. NA62 Decompressor&Merger:** re-format event data, coalesce event fragments before forwarding packets to Network Interface.
- APEnet Protocol Encoder
  - Protocol adaptation between on-board and off-board protocol.

NaNet Design – router



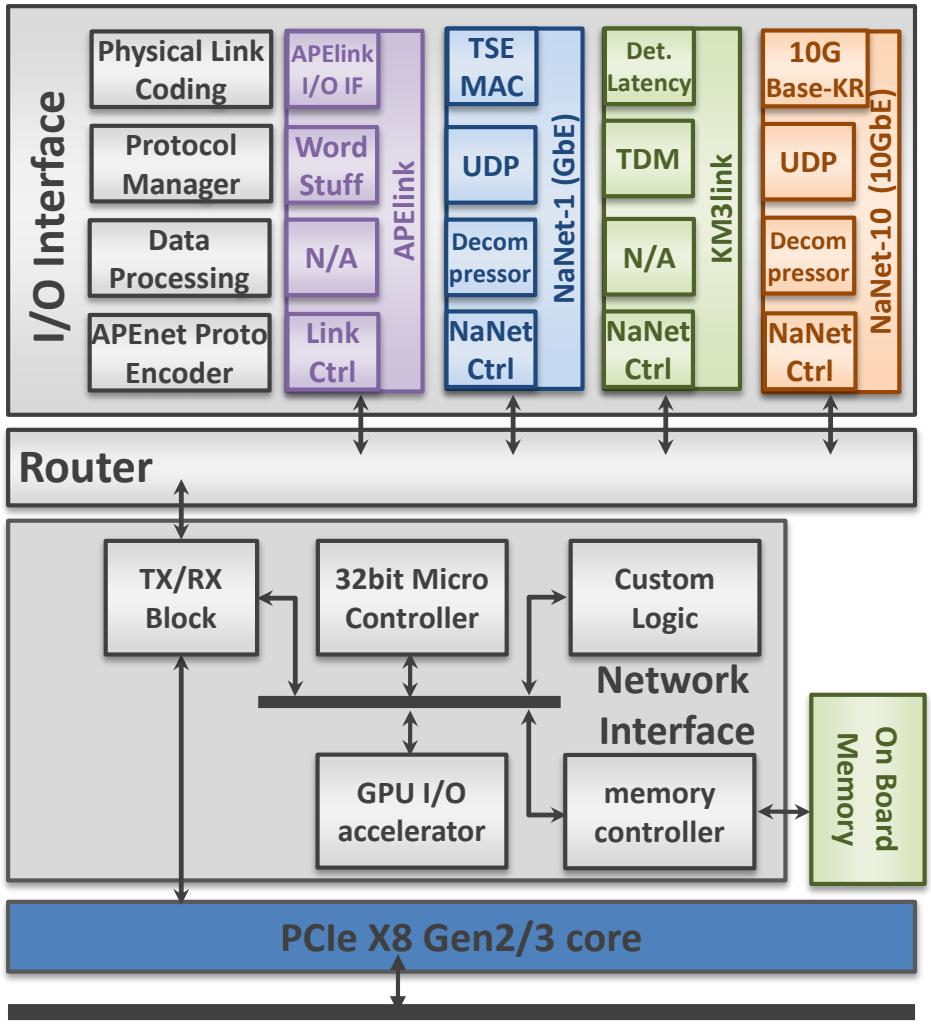
- 5 ports bidir full crossbar switch.
- Router: dynamically interconnects ports.
- Arbitration: resolve contention on ports requests (static, round robin).
- Supports up to 10 simultaneous 2.8 GB/s data streams. (x2 in next PCIe Gen 3 designs)
- Re-configurable number of ports.

NaNet Design – network interface



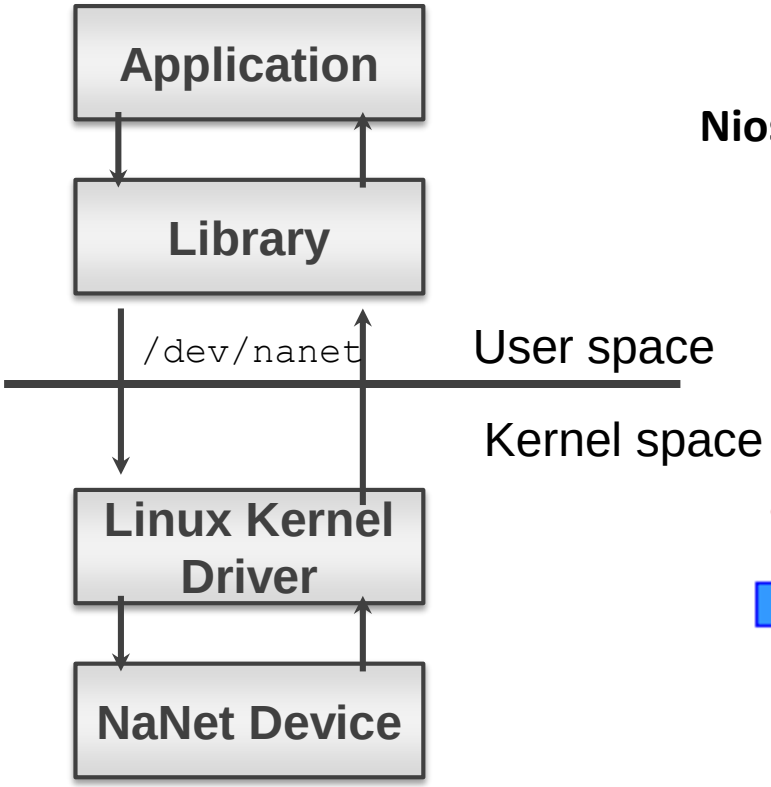
- In TX gathers data from PCIe interface and forwards them to destination port.
- In RX performs **zero-copy RDMA** receive operation managing CPU/GPU Virtual to Physical address translation.
  - Translation Lookaside Buffer (associative cache).
  - Microcontroller in case of miss.
- GPU I/O accelerator (**GPUDIRECT P2P/RDMA**)
- TX/RX Block
  - Multiple DMA engines instantiating concurrent PCIe transactions.
- Altera NIOS II microcontroller.

NaNet Design – PCIe cores



- PCIe X8 Gen2 Core (stable)
  - Based on PLDA EZDMA Gen2 hard IP core.
  - CPU BW: 2.8 GB/s Write, 2.5 GB/s Read.
  - GPU BW: 2.8 GB/s Write, 2.5 GB/s Read.
- PCIe X8 Gen3 Core (testing)
  - Based on PLDA XpressRICH Gen3 soft IP core.
  - CPU BW: 4.2GB/s Write, 4.8 GB/s Read.
  - GPU BW: 3.0 GB/s Write, 3.0 GB/s Read.
- Home-made PCIe X8 Gen3 Core (developing)
  - Based on the Altera PCIe hard-IP
- Test setup: Ivy Bridge, Kepler K20c

NaNet Design – software stack

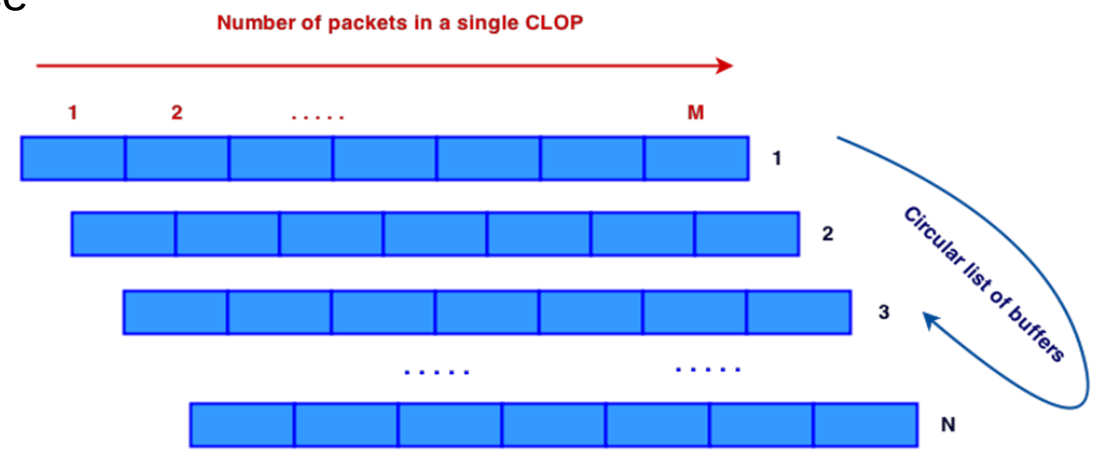


Host

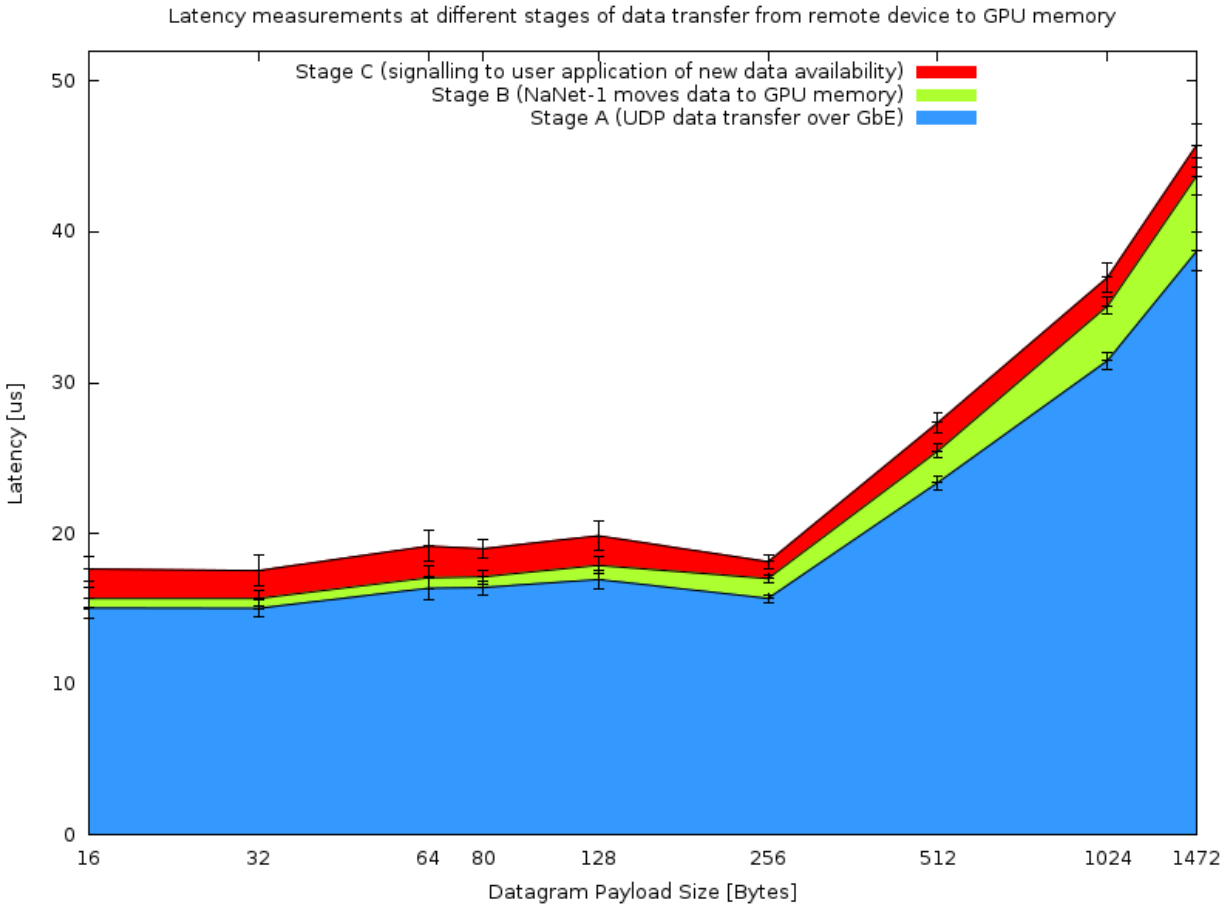
- **Linux Kernel Driver**
- **User space Library (open/close, buf reg, wait recv evts, …)**

Nios II Microcontroller

- **Single process program performing System Configuration & Initialization tasks**



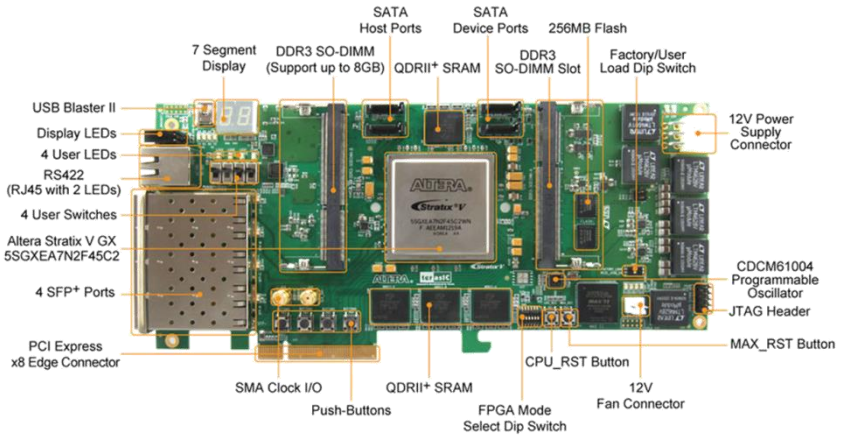
NaNet-1: latency characterization

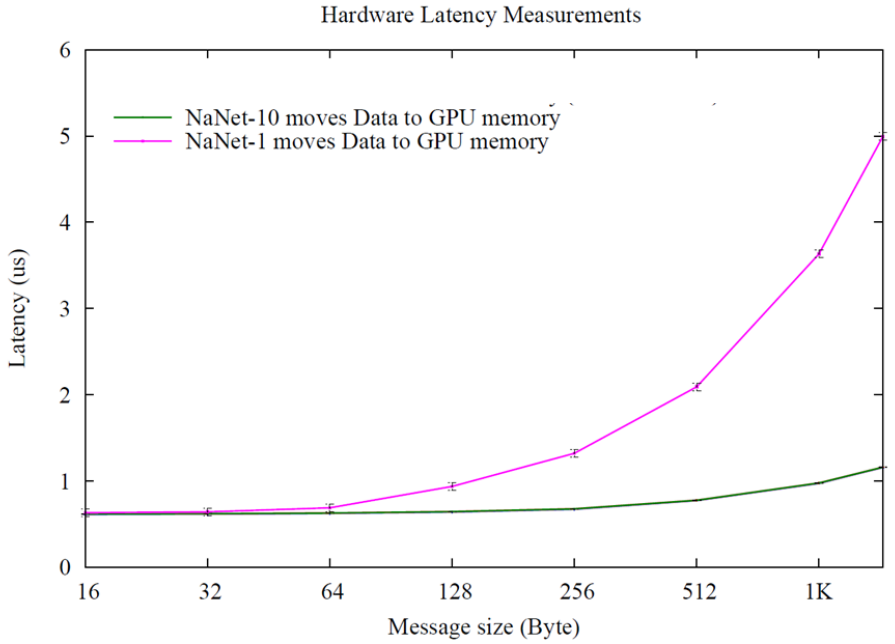


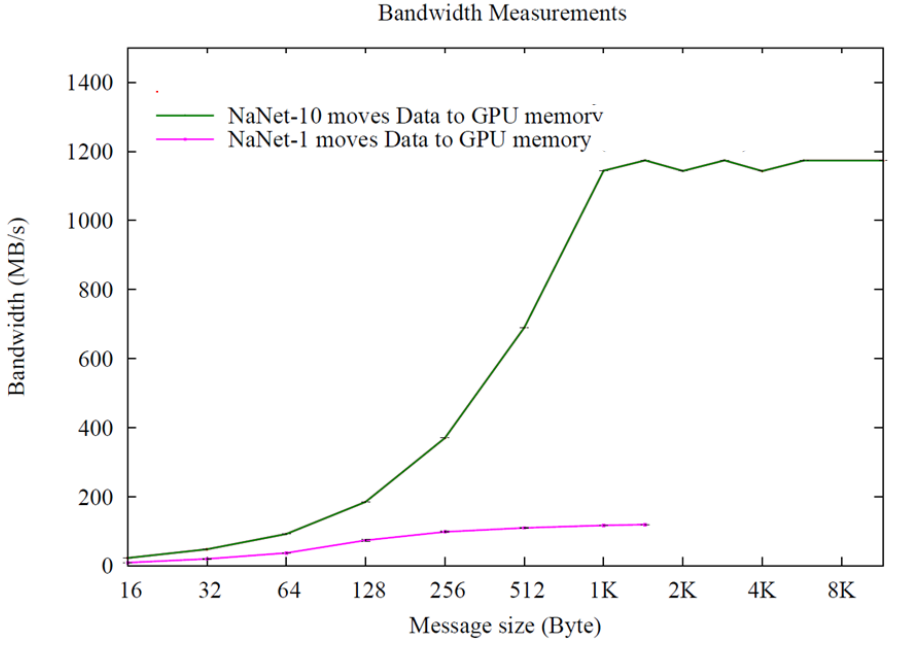
Latency measurements of data transfer to GPU memory

NaNet-10: the next generation

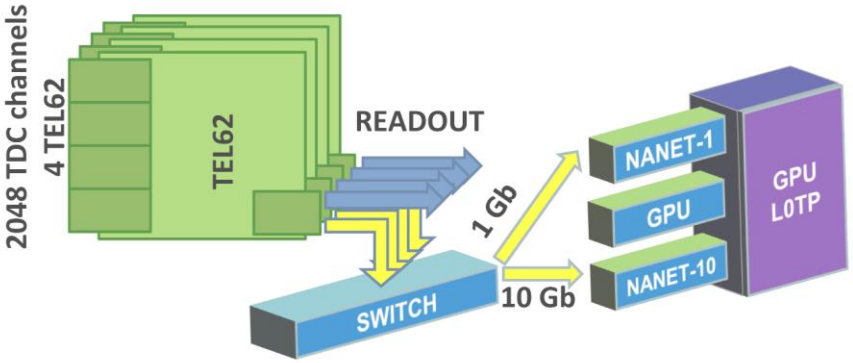
- ALTERA Stratix V dev board
- **PCIe x8 Gen3** (8 GB/s)
- 4 SFP+ ports (Link speed up to 10Gb/s)
- Implemented on Terasic DE5-NET board
- GPUDirect /RDMA capability
- UDP offloads supports





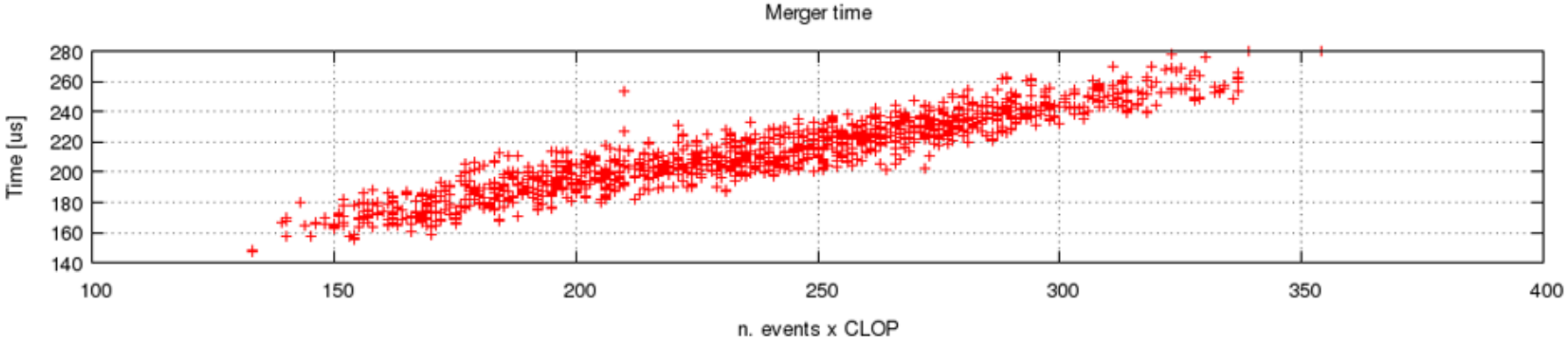


Merging events…



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

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



Merging will be performed in HW (network processing stage) (NaNet-10 default)

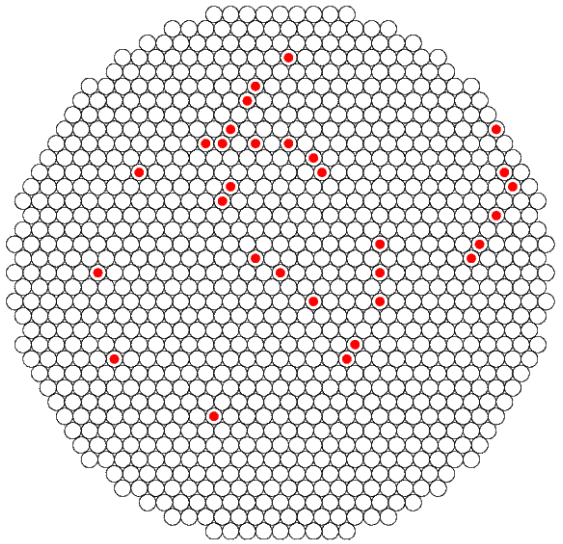
Events are arranged in CLOPs with a format more suitable for GPU's thread memory access Multi Merged Event GPU Packet (M^2EGP)

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

Low-level RICH trigger algorithm

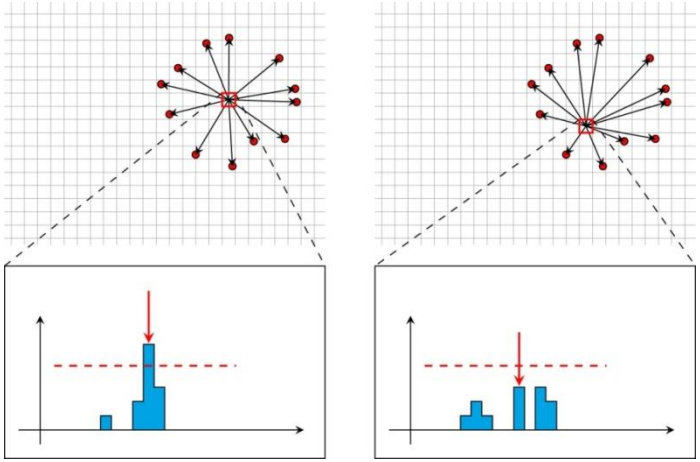
Requirements for an on-line RICH reconstruction algorithm:

- **Trackless**
  - **No information from the tracker**
  - **-> Difficult to merge information from many detectors at low-level**
- **Multi-rings**
  - **Many-body decay in the RICH acceptance**
- **Fast**
  - **Events rate at ~10 MHz**
- **Real-time (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

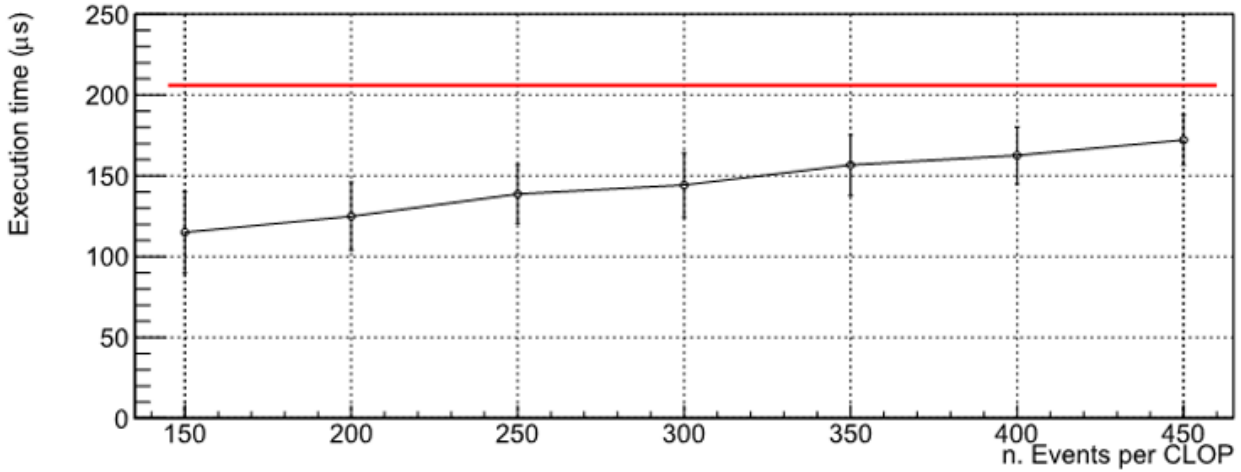


Pros: naturally mapped on the GPU threads grid

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

Cons: memory limited, performances depending on number of hits

Histogram: a pattern recognition algorithm

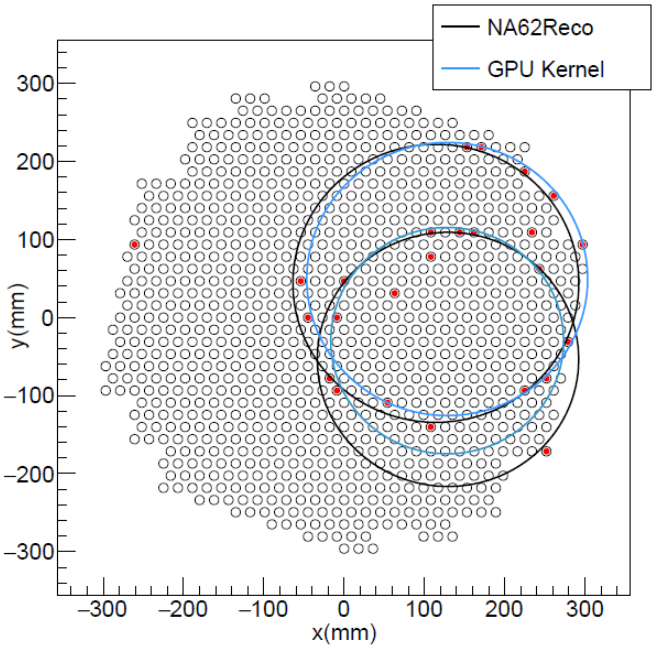


Multicircle fit for ~10% of events and single circle for ~90%

Testing histogram kernel with MC data

With a single GPU we can't handle 10Mhz:

- Use two GPUs
- next generation of GPUs (Maxwell, Pascal)

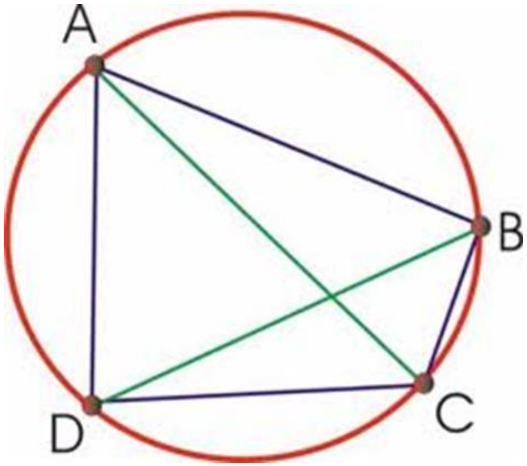


Good quality of online ring fitting

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"



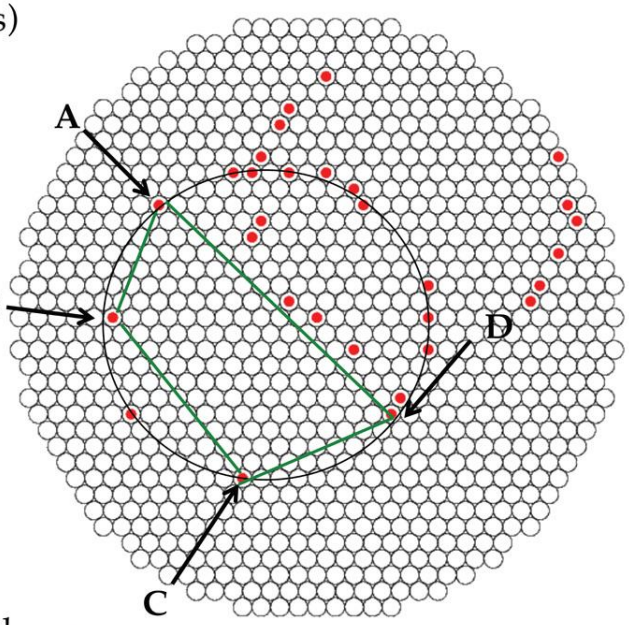
Almagest: a new multi-ring algorithm

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

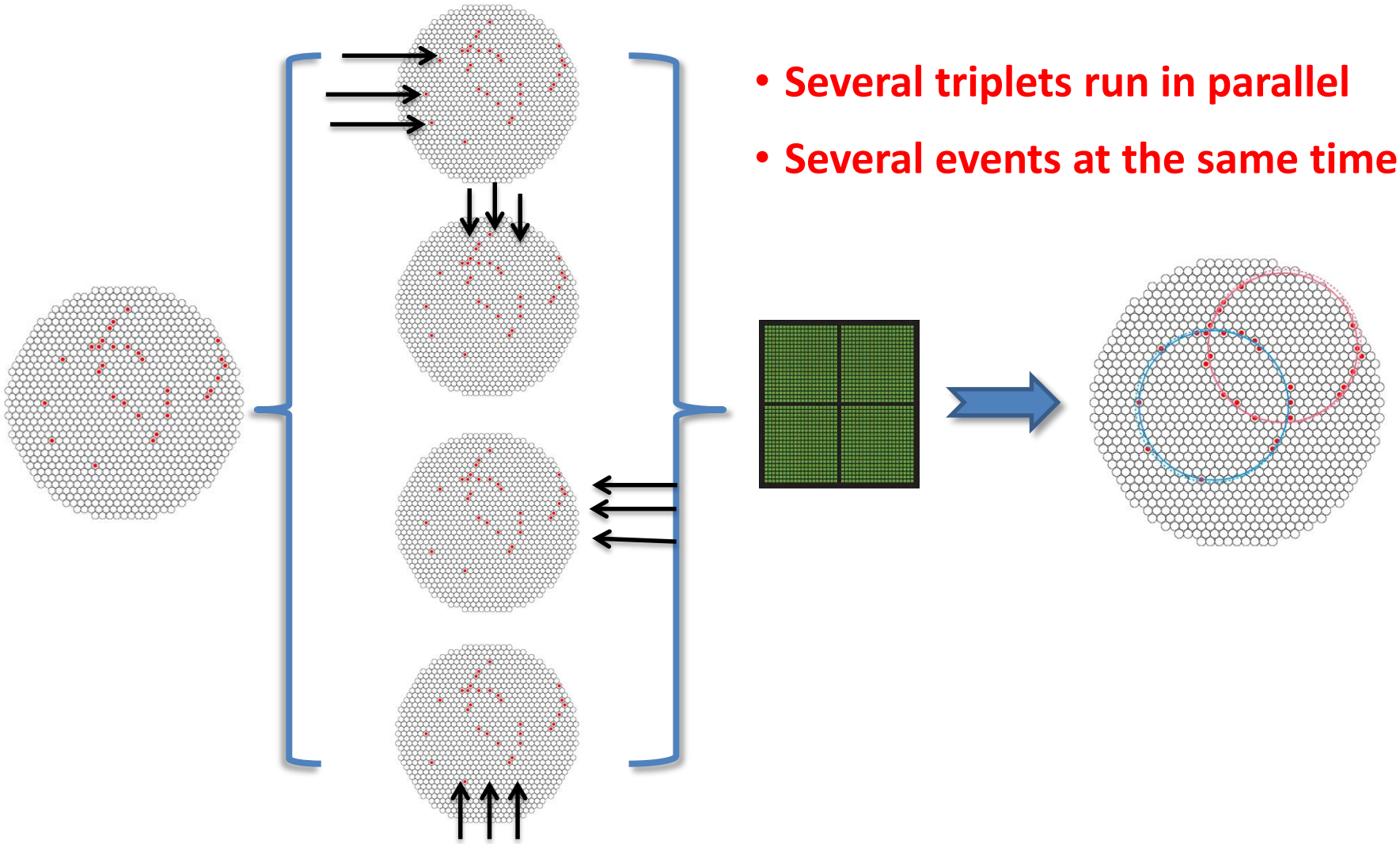
iii) If the point satisfy the Ptolemy's condition then consider it for the fit

iv) …again…



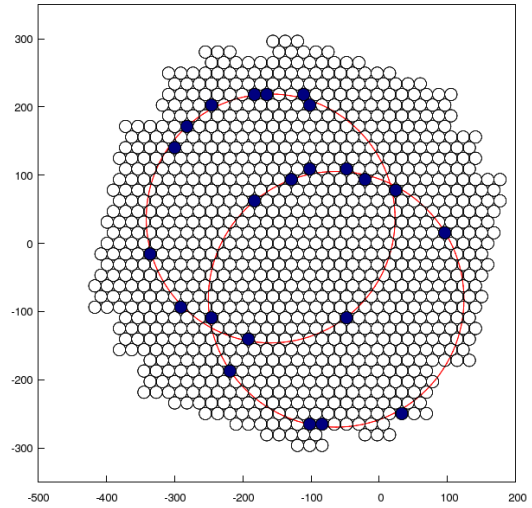
This algorithm exposes two levels of parallelism…

Almagest: a new multi-ring algorithm

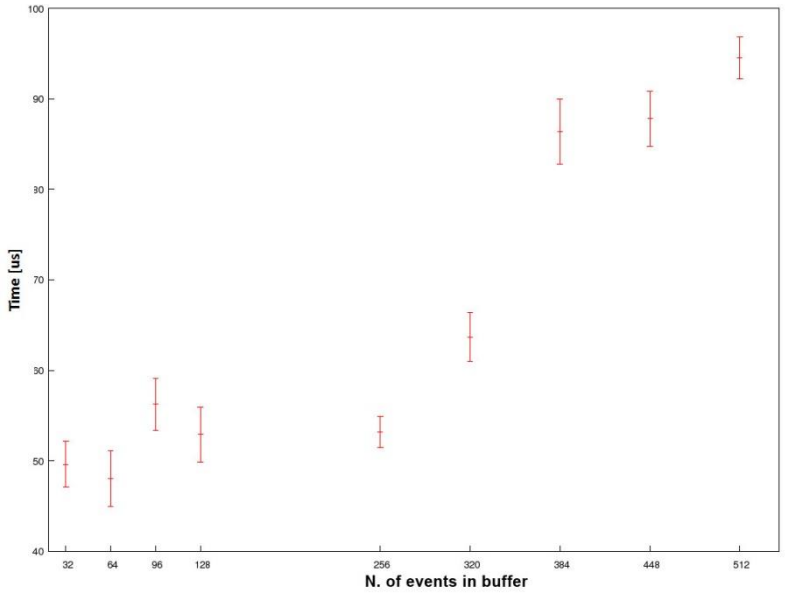


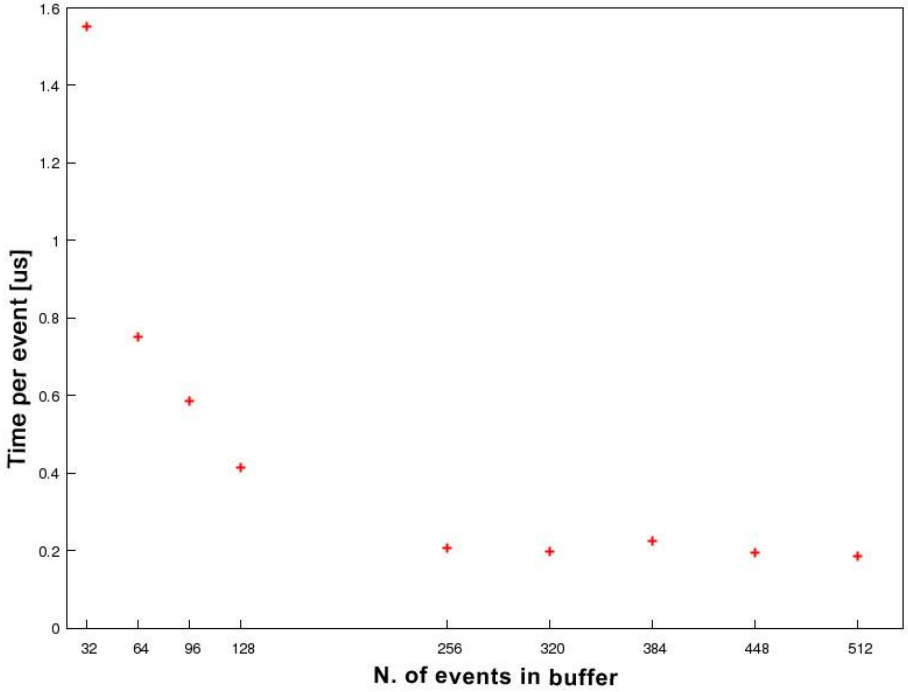
- Several triplets run in parallel
- Several events at the same time

Almagest



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





Concluding remarks

- **Key enabling features for a data transport system for the real-time GPU-based trigger have proven to be:**
  - **Zero-copy DMA to the GPU memory**
  - **Network protocol offloding and data processing stages in the FPGA**
  - **Interface to the experiment TTC system (global clock, trigger signals)**

- **Tested a working solution with the NaNet-1 board during NA62 2015 Run**

- **Multi-ring algorithms such as Histogram and Almagest are implemented on GPU**
  - **There is still room for improvement -> Optimize computing kernels**

- **The GPU-based low-level trigger with the new board NaNet-10 is installed and being currently tested during the the NA62 2016 run (with HW merger)**

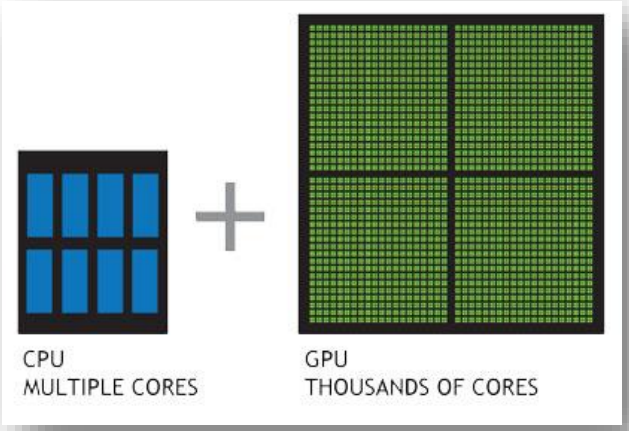
R. Ammendola [(b)], M. Bauce[(a)],
A. Biagioni[(a)], S. Di Lorenzo[(f,g)],
R. Fantechi[(f)], M. Fiorini[(d,e)],
O. Frezza[(a)], G. Lamanna[(c)],
F. Lo Cicero[(a)], A. Lonardo [(a)],
M. Martinelli[(a)], A. Messina,
I. Neri[(d)], P.S. Paolucci[(a)],
R. Piandani[(f)], L. Pontisso[(f)],
M. Sozzi[(f,g)],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

NaNet-10 at CERN

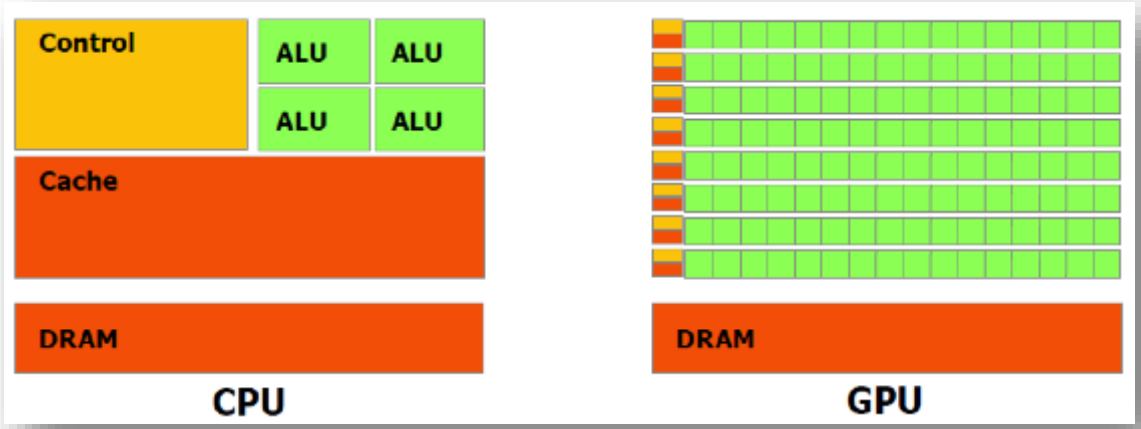
SPARE SLIDES

Graphics Processing Unit (GPU)

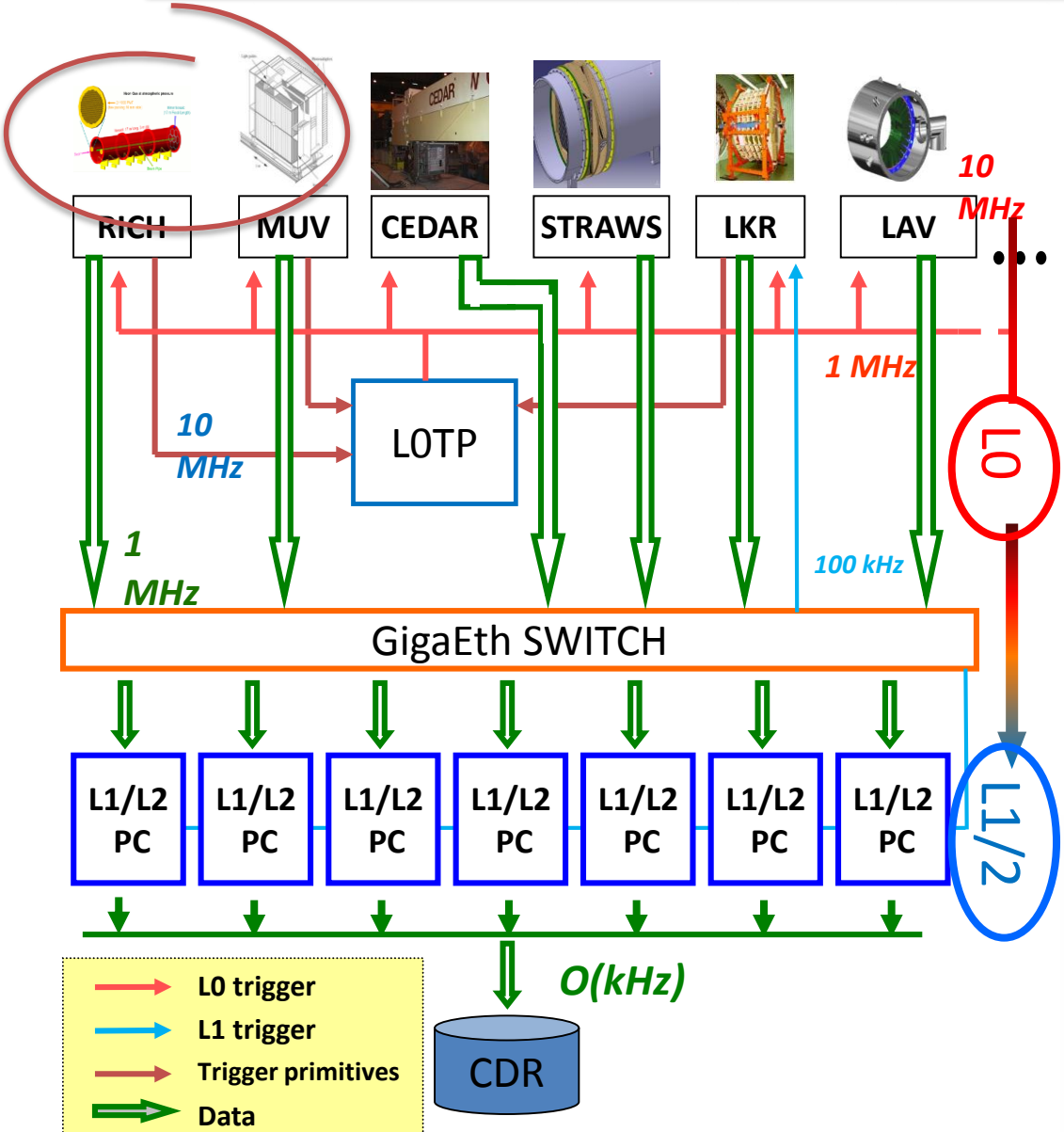


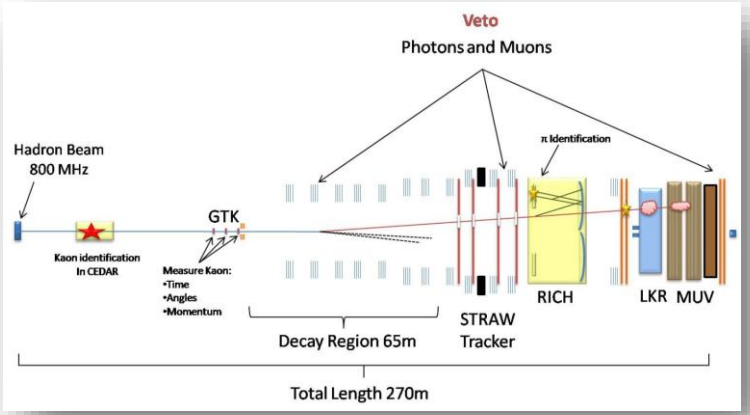
- Based on a massively parallel architecture
- Thousands of cores to process parallel workloads efficiently

- Less control units, many more ARITHMETIC LOGIC units.



NA62 DAQ and Trigger





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

Almagest: selecting triplets on GPU

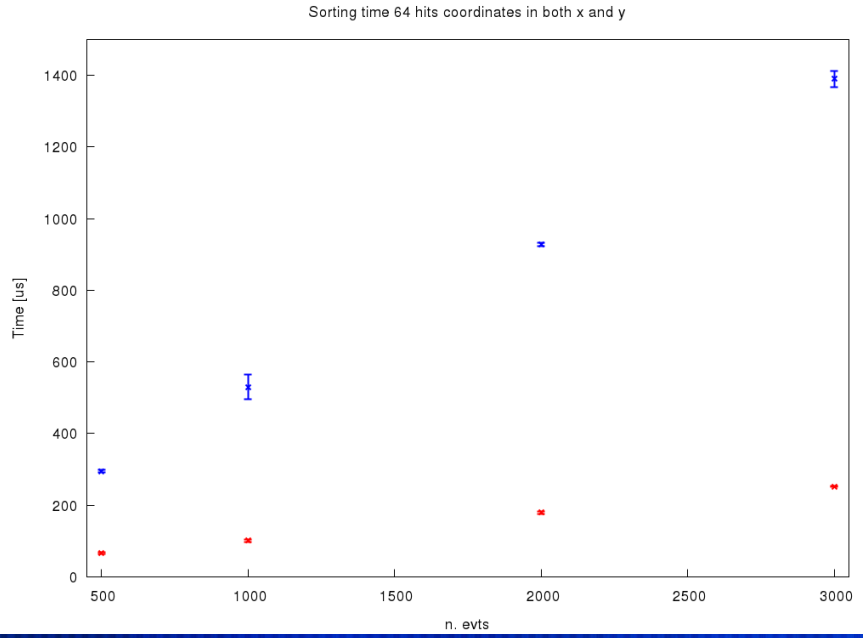
64 hits per events
Sort in both x and y

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

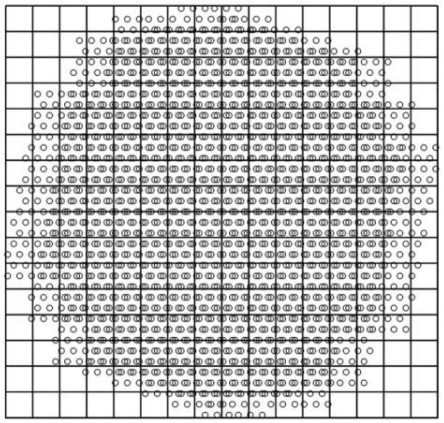
Every thread checks if Tolomeo's condition is satisfied starting with one of the triplets previously found

Vote function __ballot()
Results propagated through the warp

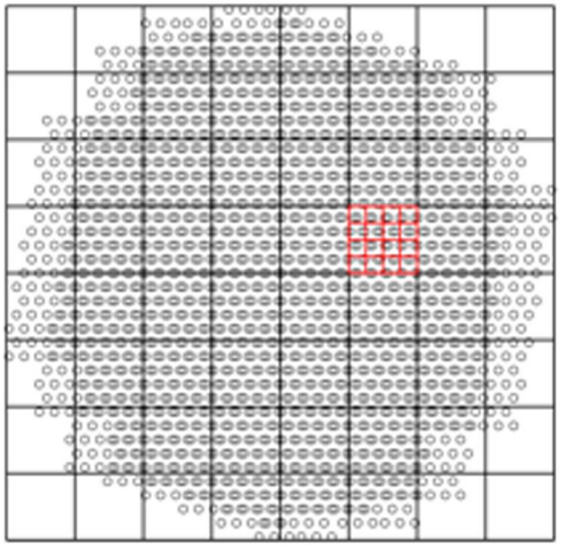
Shared memory and barrier synchronization because of working with 2 warps

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**

Other experiments

Several other experiments are studying GPUs for online data selection.
Not exhaustive list:

- **Alice: A part of the online TPC reconstruction is already running on GPU**
- **LHCb: is considering GPU (and other accelerators)**
- **CMS: GPU to help in cluster splitting, for calorimetric jet trigger.**
- **Panda, CBM, Star, Mu3e, KM3, …**

GPGPU deployment in ATLAS High Level Trigger

ATLAS Trigger and TDAQ

Three upgrade phases expected in the future of LHC: the **detector occupancy** for the ATLAS experiment is expected to be **2-3 times higher**.
Trigger system needs to be upgraded to maintain or improve current performances.

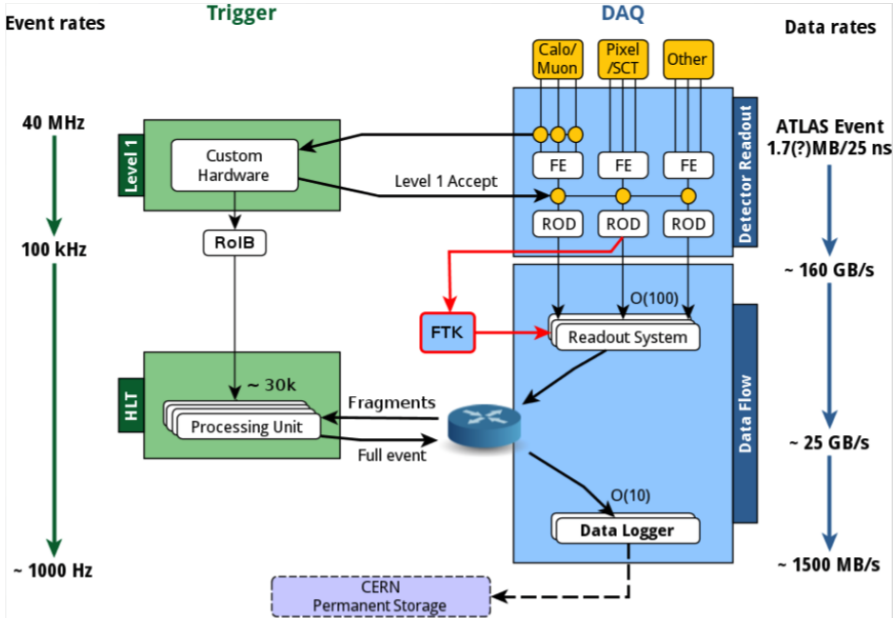
	Run 1		LS1 - Phase-0	Run 2	LS2 - Phase-I	Run 3	LS3 - Phase-II	Run 4
	2011	2012		2015-17		2019-21		2023-
Center of mass Energy $\sqrt{s}$ (TeV)	7	8		13-14		14		14
Luminosity ($cm^{-2}s^{-1}$)	8×10^{33}			1×10^{34}		2×10^{34}		5×10^{34}
Bunch spacing (ns)	50			25		25		25
Number of interaction/event, $< \mu >$	10	20		~ 27		$\sim 55 - 80$		~ 140
Total Integrated luminosity (fb^{-1})	25			~ 100		~ 300		~ 3000

ATLAS Trigger system selects 1 interesting event out of 40000 at a rate of 40MHz.
Organized in 2 levels:

- **L1 - hardware based**: coarse granularity, simplified calibrations, provides seeds for next level
- **HLT - software based**: runs on a farm of CPUs under a customized offline framework, can access full detector information

Investigating the **deployment of GPGPUs in the ATLAS HLT**, optimizing throughput per cost unit and power consumption.
Several **algorithm** and **sub-detector** under study:

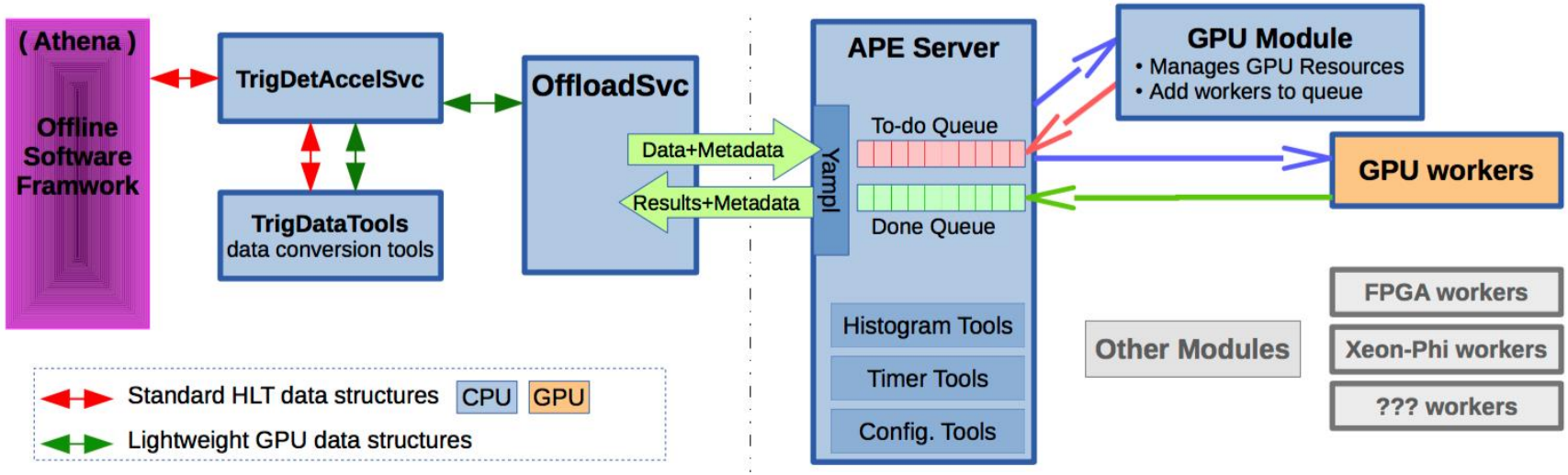
- Inner Detector tracking
- Calorimetric jet clustering
- Muon track finding



GPGPU deployment in ATLAS High Level Trigger

GPU acceleration framework

Implemented a **client-server architecture** to offload and process HLT data:
different modules for each algorithm/subdetector, accepts multiple clients, includes monitoring tools



Client side

- HLT **algorithm requires offloading** to AccelSvc
- AccelSvc:
  - **converts data** through TrigDataTools
  - adds metadata and requests offload (OffloadSvc)
  - **returns the result** to requesting algorithms

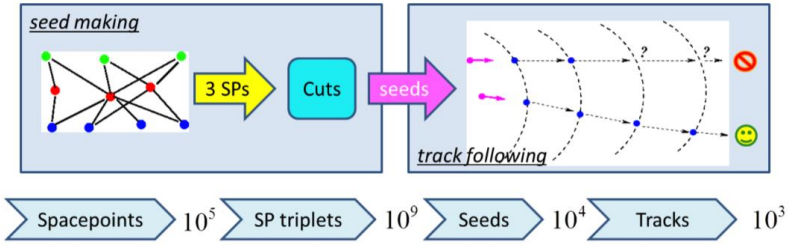
Server side

- Manager handles **communication** and **scheduling**:
  - links requests to corresponding modules
  - executes items in the queue and send results back
- Module: **manages GPU resources** and create work items requested
- Worker: **runs algorithms** over data and **prepares results**

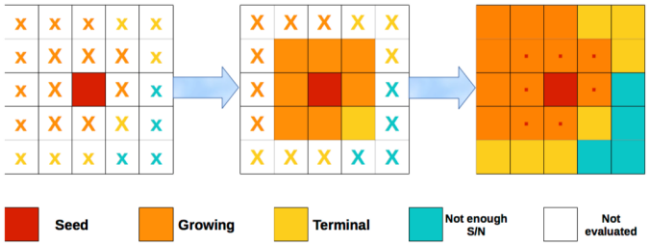
GPGPU deployment in ATLAS High Level Trigger

Algorithm performances

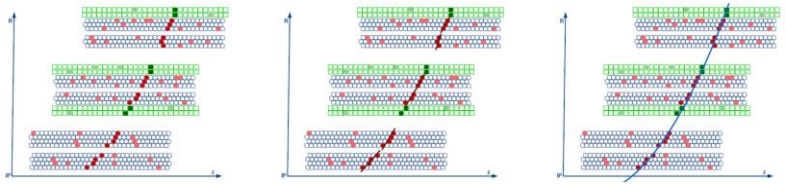
Inner Detector tracking - most time consuming part, dependent on detector occupancy.
- raw-data decoding and hits clustering
- hit triplets formation based on hit-pair seeds
- track following through the detector

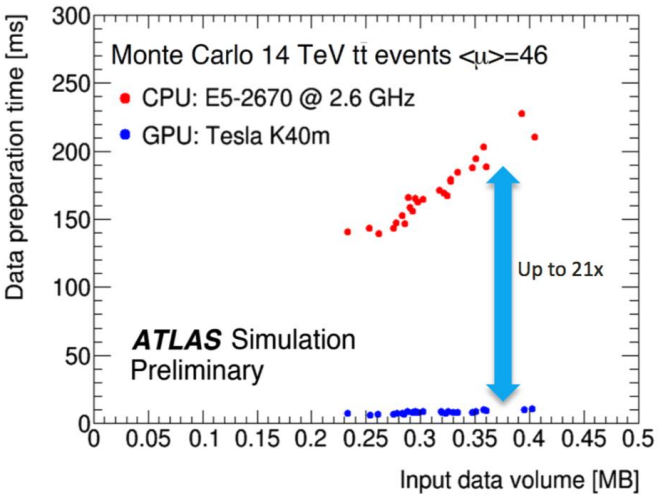


Calorimeter clustering - based on cellular automaton algorithm.
- starts from high S/N seed cells and growing cells
- evolve clusters until system is stable



Muon track finding - based on Hough Transform.
- voting kernels translate hit information to Hough Space
- maxima detection to determine track parameters





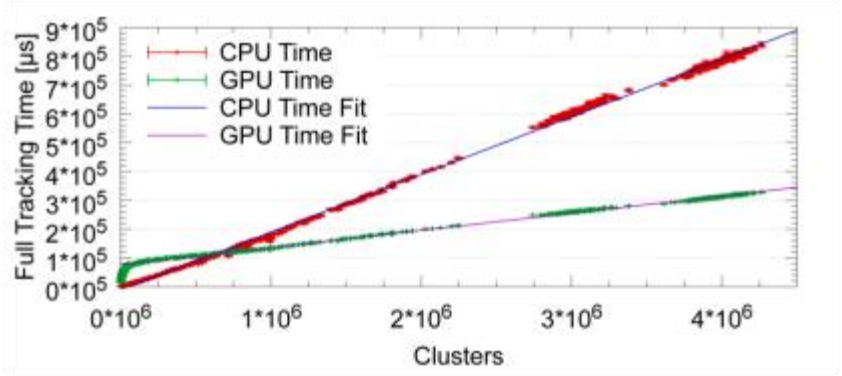
Preliminary result: **up to 20x faster** for **Inner Detector tracking.**

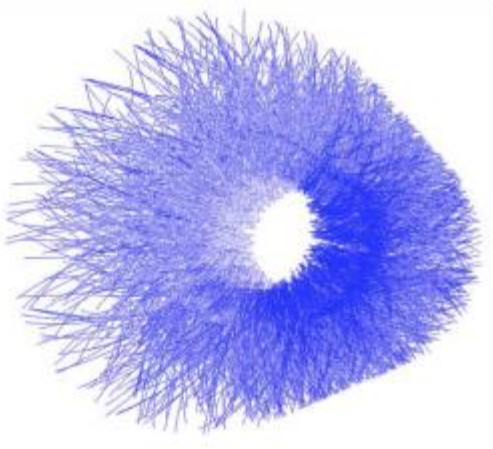
Setup working smoothly and **ready for different hardware configuration**.

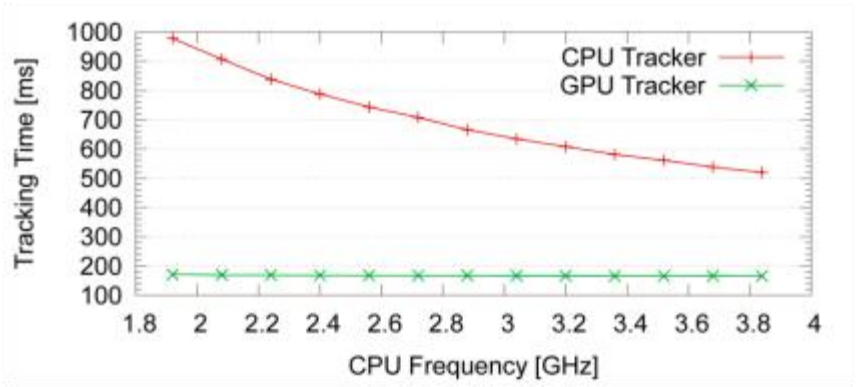
Measurements ongoing, **further results expected soon**.

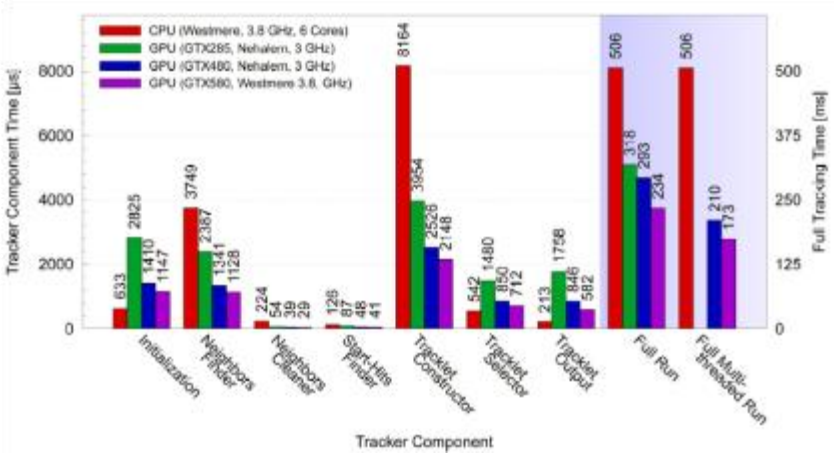
Alice: HLT TPC online Tracking

- 2 kHz input at HLT, $5x10^7$ B/event, 25 GB/s, 20000 tracks/event
- Cellular automaton + Kalman filter
- GTX 580



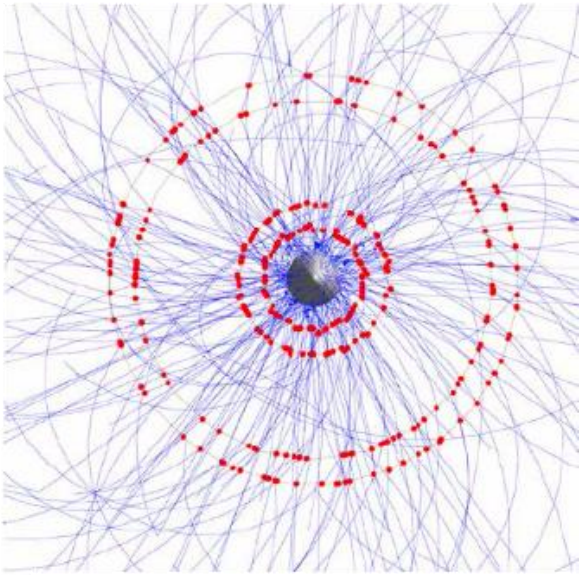


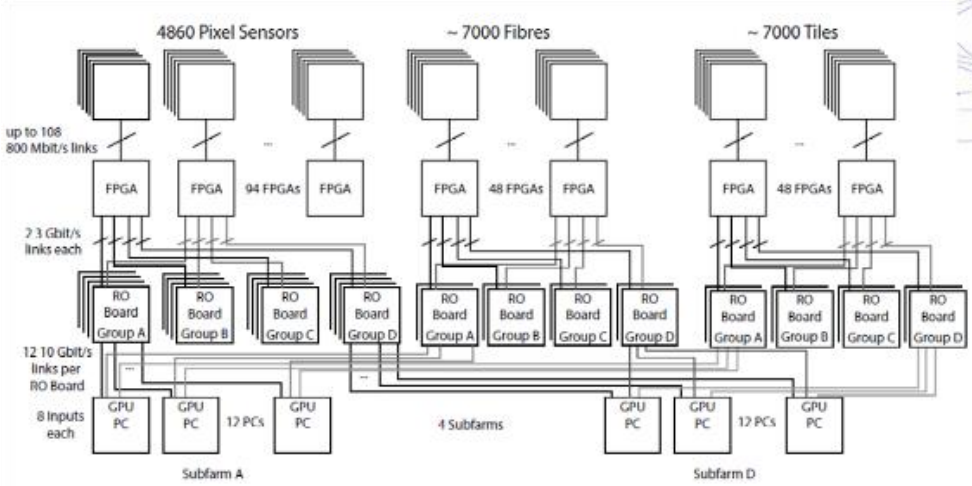


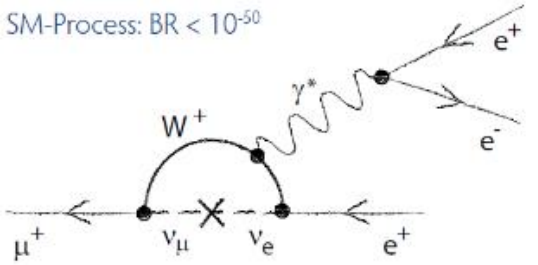


Mu3e

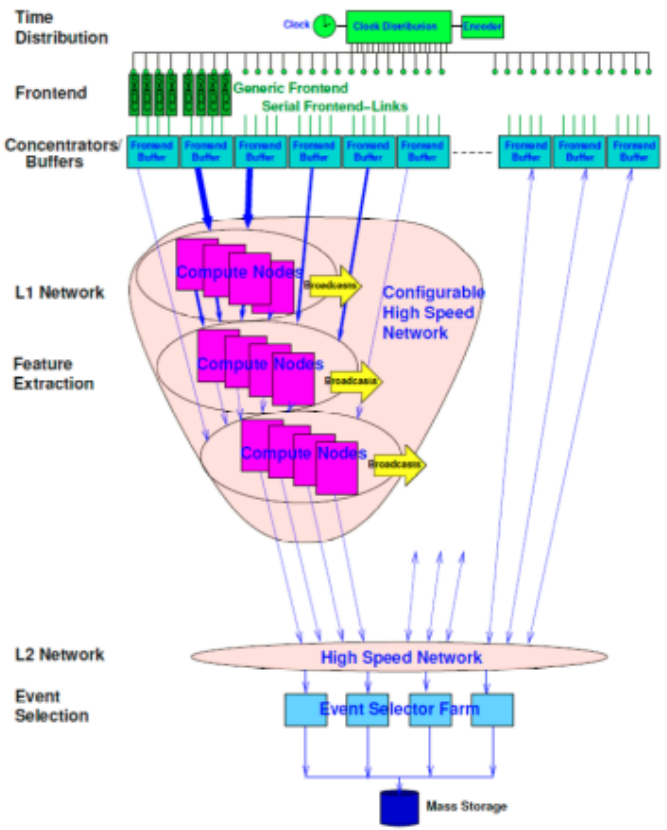
- Possibly a "trigger-less" approach
- High rate: 2×10^9 tracks/s
- >100 GB/s data rate
- Data taking will start >2016







Panda



- 10^7 events/s
- Full reconstruction for online selection: assuming 1-10 ms → 10000 – 100000 CPU cores
- Tracking, EMC, PID,…
- First exercise: online tracking
- Comparison between the same code on FPGA and on GPU: the GPUs are 30% faster for this application (a factor 200 with respect to CPU)

	CPU (ms)	GPU (ms)	Improvement	Occupancy	Notes
total runtime (without Z-Analysis)	117138	590	199		
startUp()	0.25	0.0122	20	2%	runs (num_points) times
setOrigin()	0.25	0.0119	21	25%	runs (num_points) times
clear Hough and Peaks (memset on GPU)	3	0.0463	65	100%	runs (num_points) times
conformalAndHough()	73	0.8363	87	25%	runs (num_points) times
findPeaksInHoughSpace()	51	0.497	103	100%	runs (num_points) times
findDoublePointPeaksInHoughSpace()	4	0.0645	62	100%	runs (num_points) times
collectPeaks()	4	0.066	61	100%	runs (num_points) times
sortPeaks()	0.25	0.0368	7	2%	runs (num_points) times
resetOrigin()	0.25	0.0121	21	25%	runs (num_points) times
countPointsCloseToTrackAndTrackParams()	22444	0.9581	23426	33%	runs once
collectSimilarTracks()	4	2.3506	2	67%	runs once
collectSimilarTracks2()				2%	runs once
getPointsOnTrack()	0.25	0.0187	13	33%	runs (num_tracks) times
nullifyPointsOfThisTrack()	0.25	0.0106	24	33%	runs (num_tracks) times
clear Hough space (memset on GPU)	2	0.0024	833	100%	runs (num_tracks) times
secondHough()	0.25	0.0734	3	4%	runs (num_tracks) times
findPeaksInHoughSpaceAgain()	290	0.2373	1222	66%	runs (num_tracks) times
collectTracks()	0.25	0.0368	7	2%	runs (num_tracks) times