GPU applications in real time event selections

S.Amerio

INFN Padova

Workshop congiunto INFN CCR-GARR 2012
Napoli, 14-17 maggio 2012

GPUs for real time event selections?

GPU

- A lot of computing power for highly parallelizable tasks;
- High level programming (CUDA, OpenCL);
- Commercial device → less expensive than dedicated hardware, continuous improvement of performance;
- NOT designed for low latency response

Real time events selection

- It is usually based on algorithms well suited for parallelization;
- A trigger system needs to be flexible, to be adapted to experiments changing conditions;
- It needs fixed (and low) latencies: from few μs to few tens of μs.

In this talk a brief overview of the current efforts in:

- *NA62 (G.Collazuol, G.Lamanna, M.Sozzi)*
- *ALICE (T.Kollegger et al)*
- *ATLAS (Atlas Edinburgh group, P.Clark et al.)*
- *CDF (D.Lucchesi et al)*

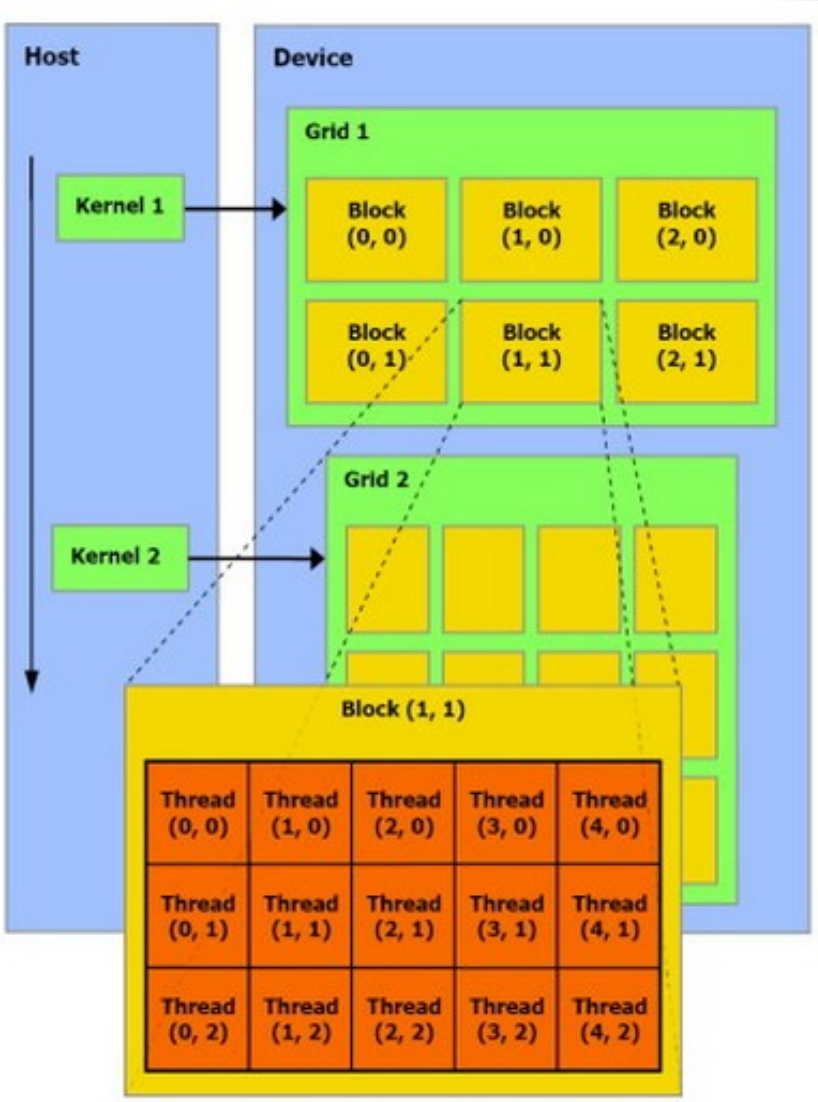
In one slide: GPU programming model ...

A GPU has N **multiprocessors**, each with M **cores**

Instructions are executed in parallel by ***threads.***
Threads are organized into ***blocks***
Blocks are organized into a ***grid.***

A multiprocessor executes a block at a time.

A **warp** is the set of threads executed in parallel.
32 threads in a warp, *they can only execute one particular common instruction.*



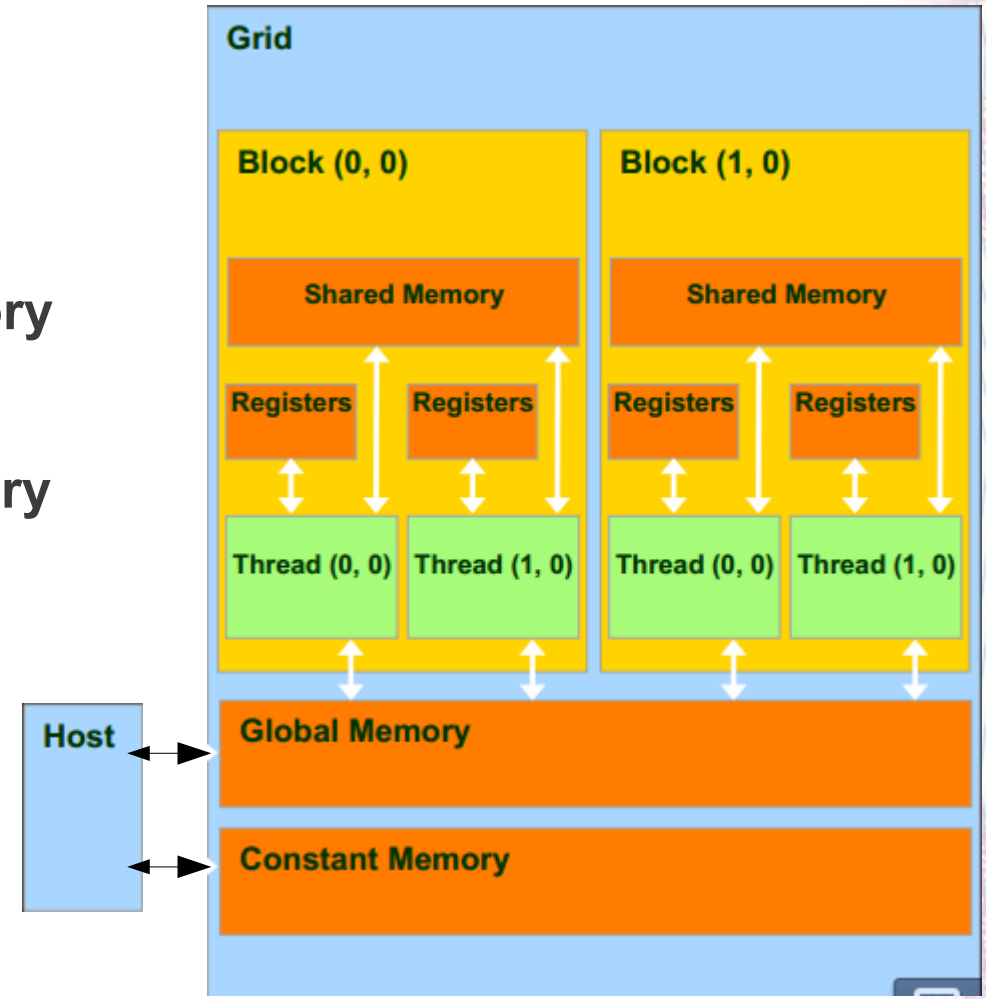
Careful organization of the code to reduce latency and fully exploit GPU computing power.

… and memory organization

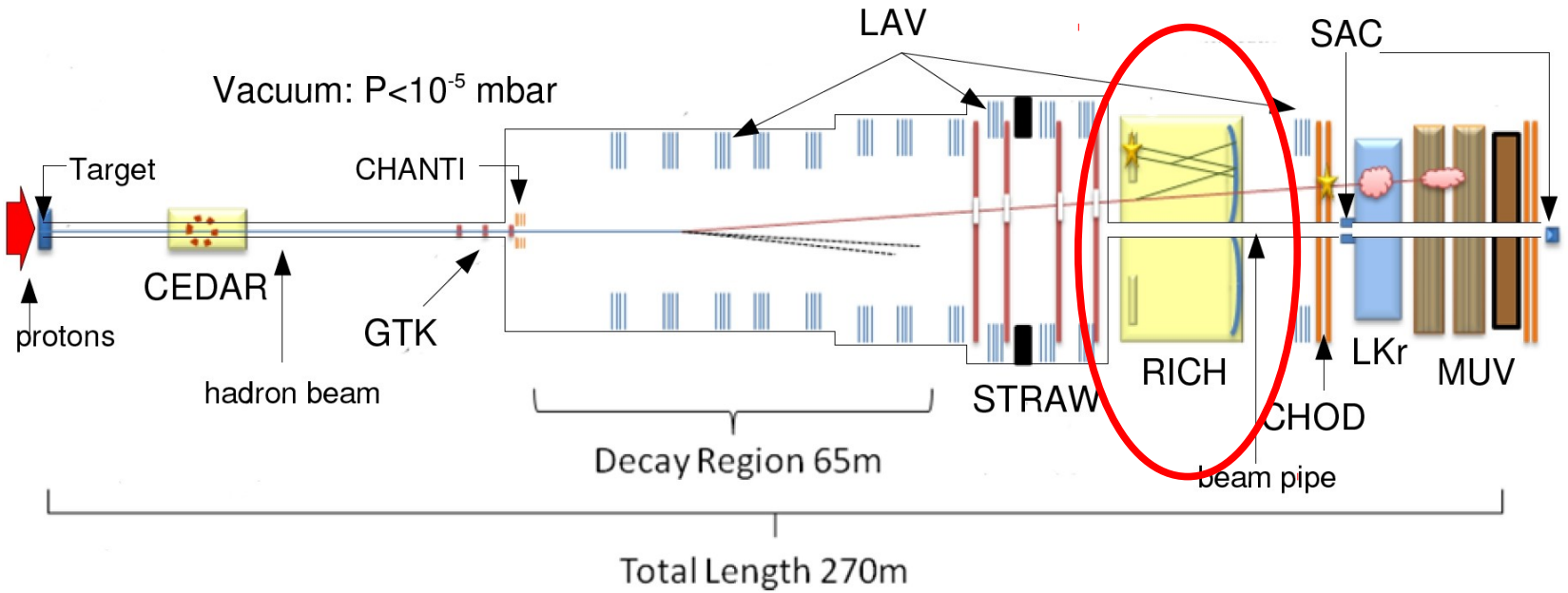
Each thread can:

- Read/Write per-thread **registers**
- Read/Write per-block **shared memory**
- Read/Write per-grid **global memory**
- Read Only per-grid **constant memory**

Different memory types with different access speed → significant impact on total latency.



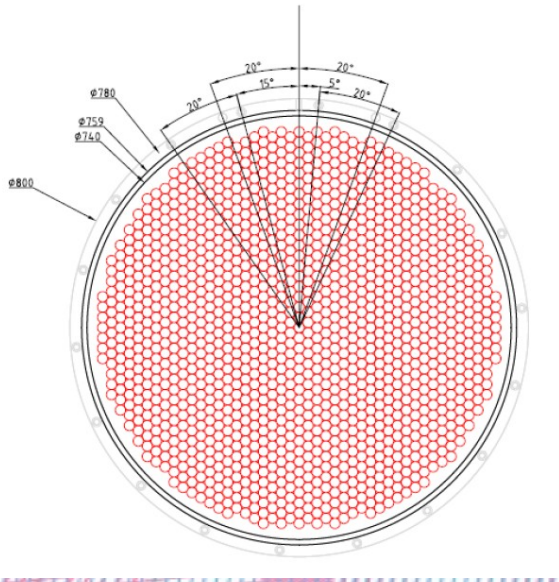
NA62



Main goal: BR measurement of the ultrarare $K \to \pi\nu\nu$ ($BR_{SM}=(8.5\pm0.7)\cdot10^{-11}$).

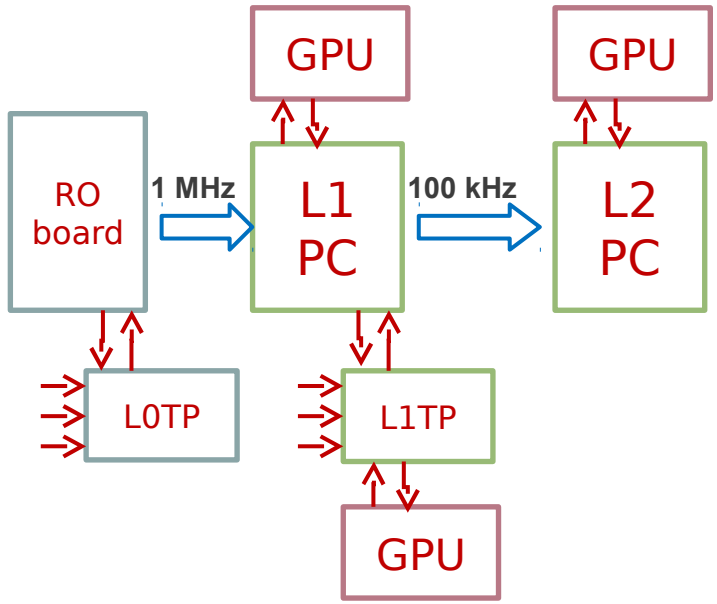
High particle rates required → strong trigger rejection

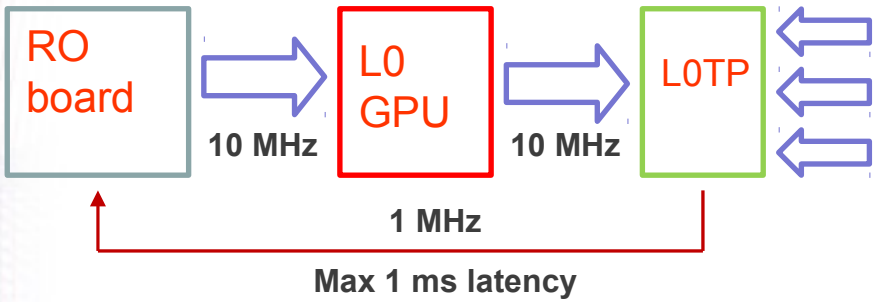
GPUs for RICH trigger → single ring identification and fitting.
- rate of tracks : *10 MHz*
- average number of hits/track: ~ *20*



GPUs in the NA62 TDaq system

- The use of the GPU at the software levels (L1/2) is “straightforward”: put the video card in the PC.
- No particular changes to the hardware are needed
- The main advantages is to exploit the power of GPUs to reduce the number of PCs in the L1 farms



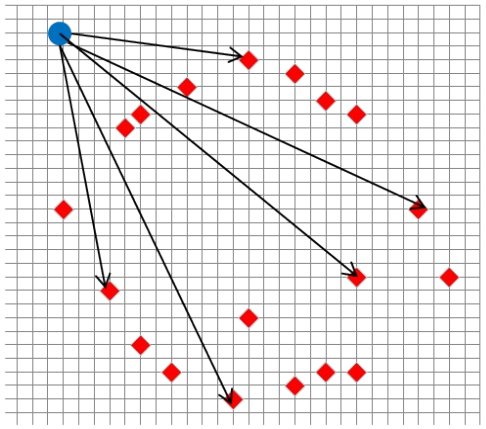


- The use of GPU at L0 is more challenging:
  - Fixed and small latency (dimension of the L0 buffers)
  - Deterministic behavior (synchronous trigger)
  - Very fast algorithms (high rate)

Algorithms for ring searches

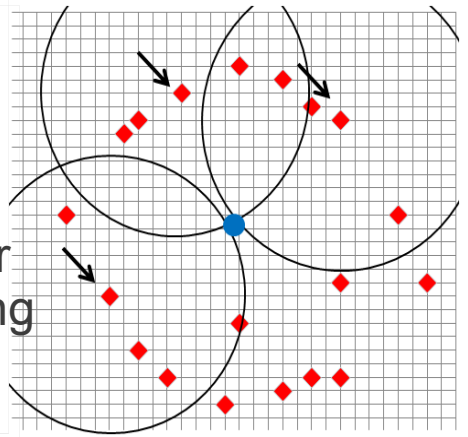
POHM/DOHM:

- Histograms of distances between PM (1000) and hits
- POHM: GPU core ↔ PM
- DOHM: GPU multiproc ↔ event
- Large number of in-memory histograms.



HOUGH:

- Each hit is the center of a test circle with a given radius. Ring center is the best matching point of the test circles

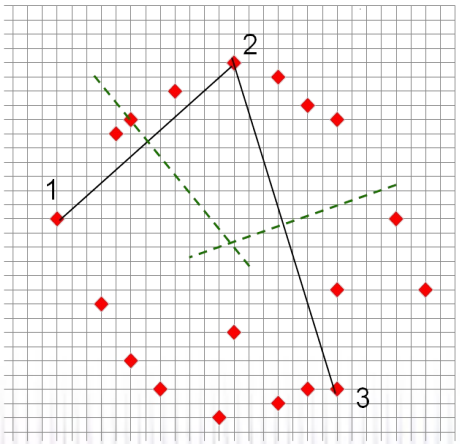


Fast access memory needed

Great computation power needed

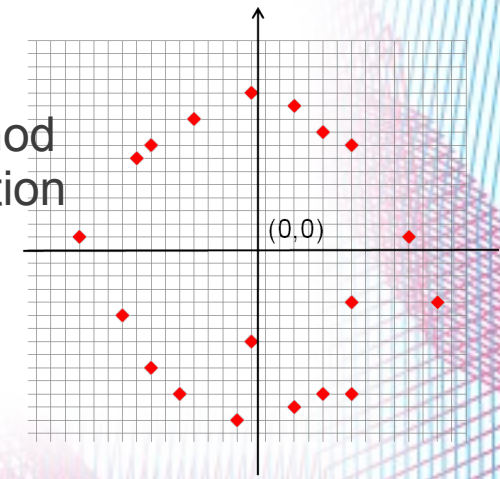
TRIPL:

- Each GPU thread computes the circle center using 3 points → Final ring center is the average.



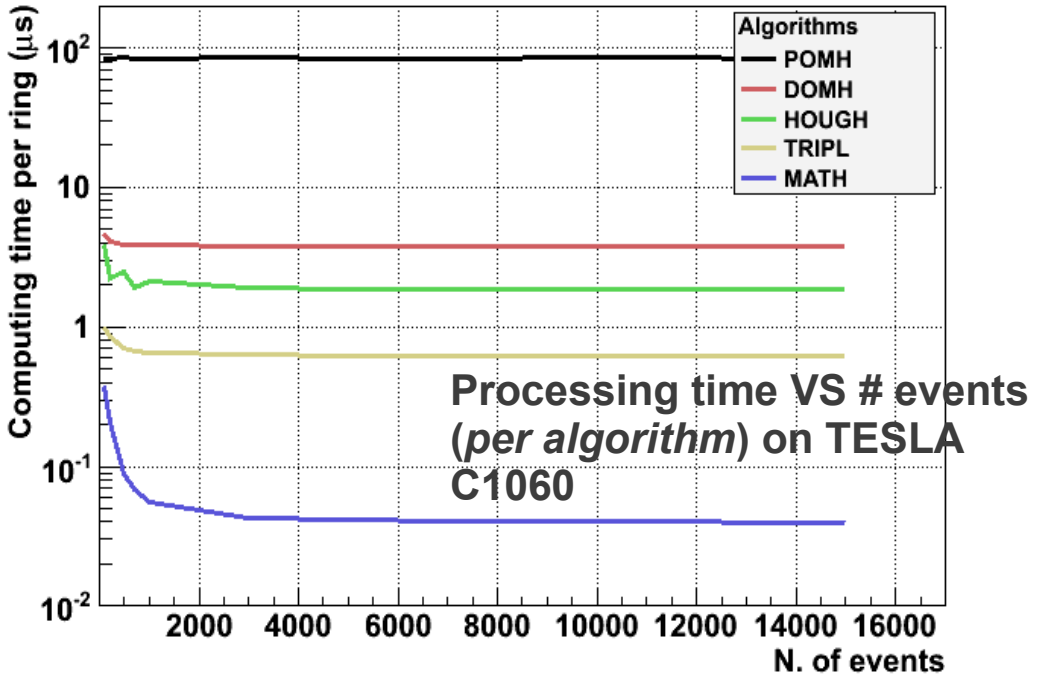
MATH:

- Least squares method applied to the equation of a circle

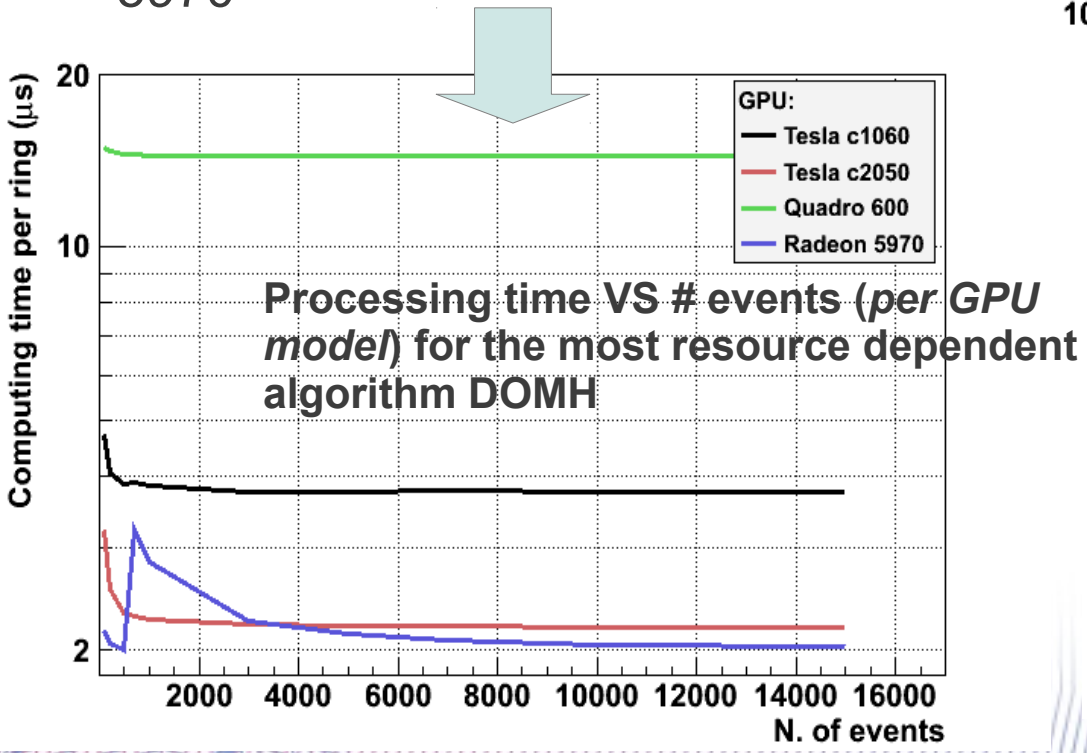


Results: GPU processing time

MATH algorithm is the fastest.
For packets of > 1000 events, processing time is about *50 ns/ring.*



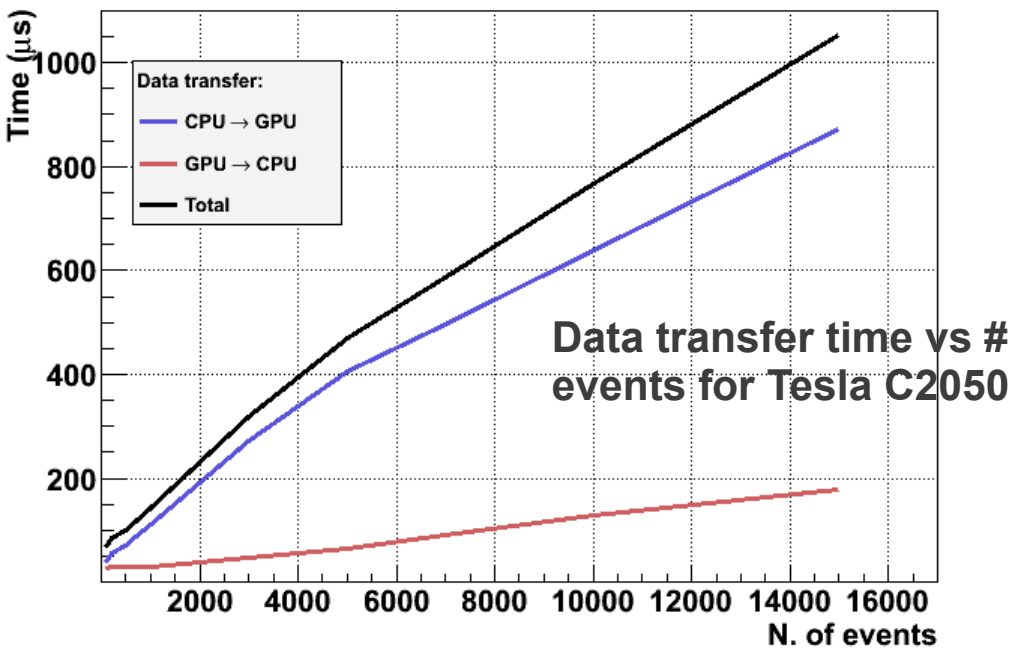
Best results on AMD Radeon HD 5970



	NVIDIA Quadro 600	NVIDIA Tesla C1060	NVIDIA Tesla C2050	AMD Radeon HD 5970
Number of multi-processors	2	30	14	20
Total number of cores	96	240	448	3200[a]
Core frequency (GHz)	0.64	1.3	1.15	0.725
Main mem. (GB)	1	4	3	2[a]
Main mem. bandwidth (GB/s)	25.6	102	144	256[a]
Shared multi-processor mem. (KB)	48	16	48	32
Computing power (TFLOPS)	0.246	0.93	1.03	4.6[a]
Max power consumption (W)	40	188	247	294

Results: data transfer and total latency

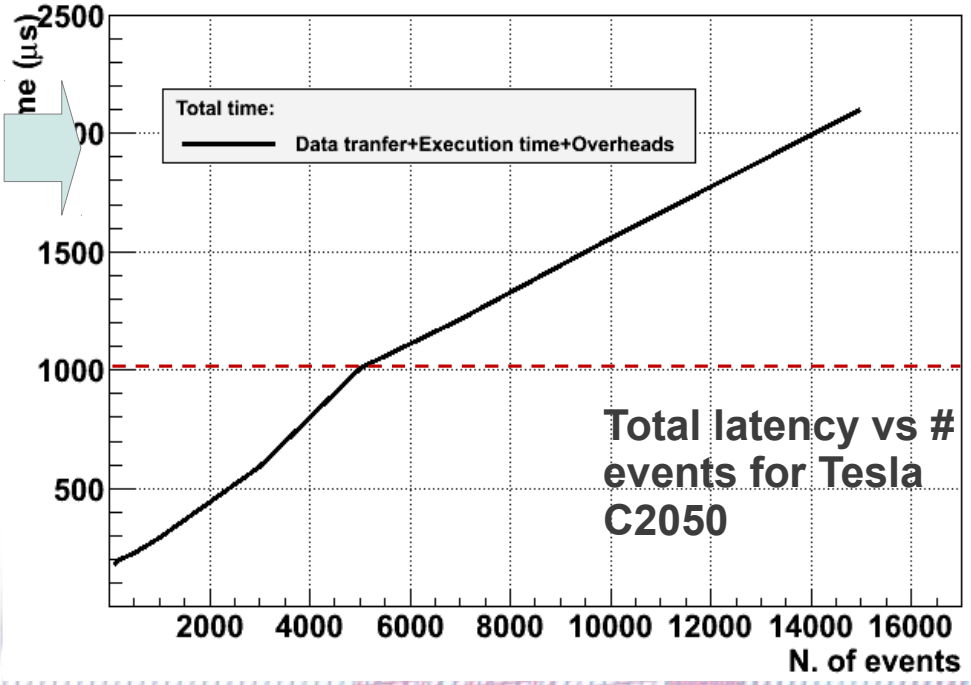
Data transfer time to copy data from the host PC onto the GPU and results back significant.
It depends on the number of events.
For 1000 events, about 175 µs



Total latency comprises:

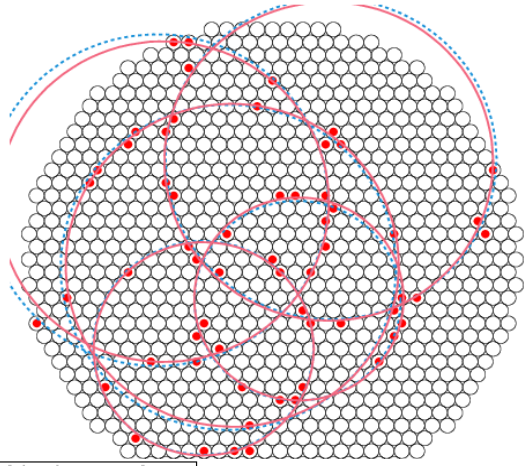
- processing time
- data transfer time
- overheads of the GPU operations.

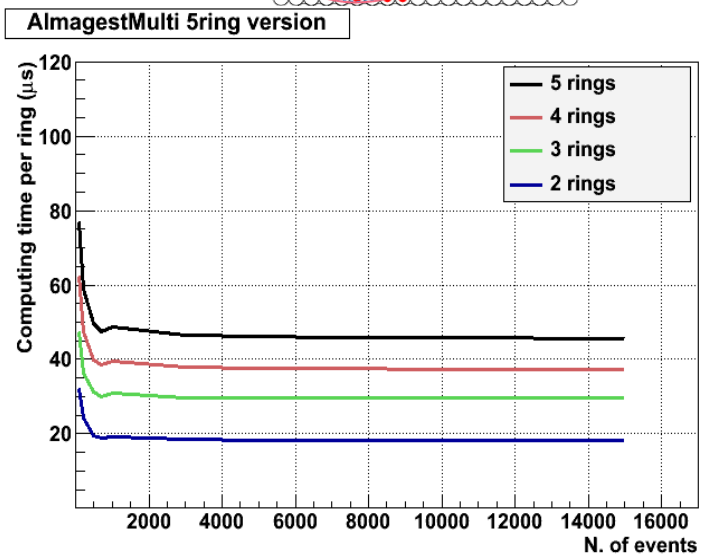
For 1000 evts, total latency is 300 µs (MATH algorithm)



NA62 future perspectives

Summer 2012: integration of a **prototype** in parasitic mode in NA62 trigger system.
October 2012: **tests with beam!**





Multiple ring finding algorithm already implemented and tested.
Optimization in progress.

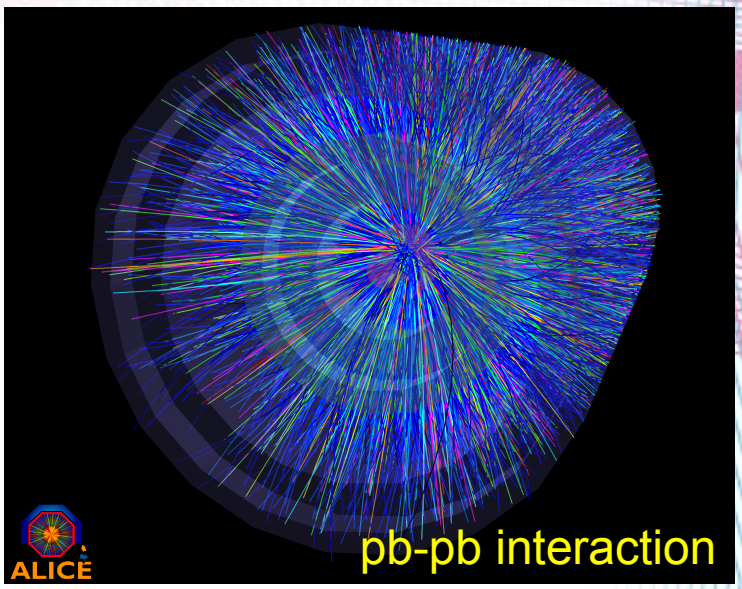
Usage of GPUs for **Online tracking at L1** ongoing.

References:

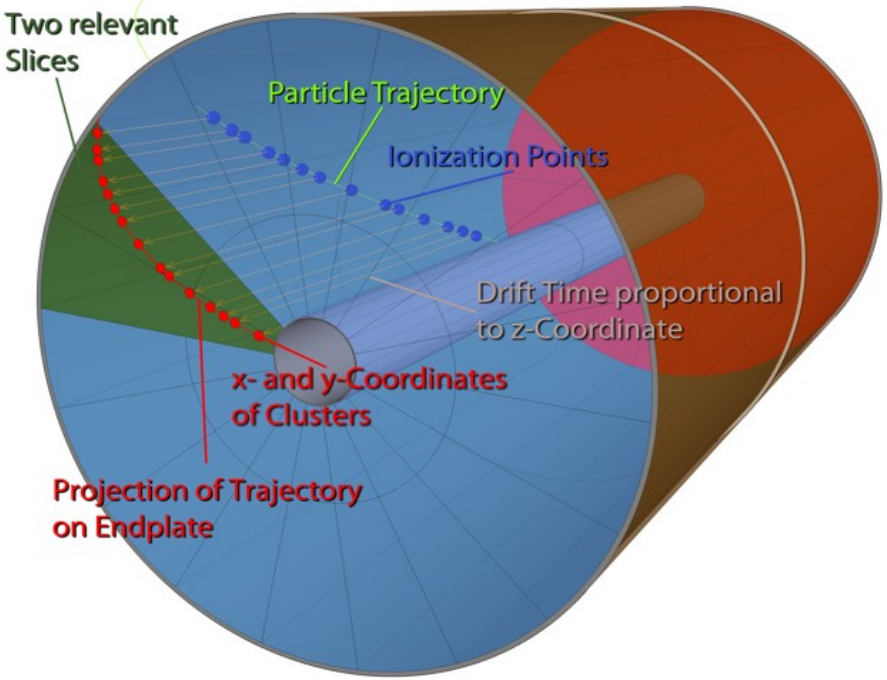
i) IEEE-NSS CR 10/2009: 195-198
ii) Nucl.Instrum.Meth.A628:457-460,2011
iii) Nucl.Instrum.Meth.A639:267-270,2011
iv) "Fast online triggering in high-energy physics experiments using GPUs" Nucl.Instrum.Meth.A662:49-54,2012

ALICE HLT TPC tracking

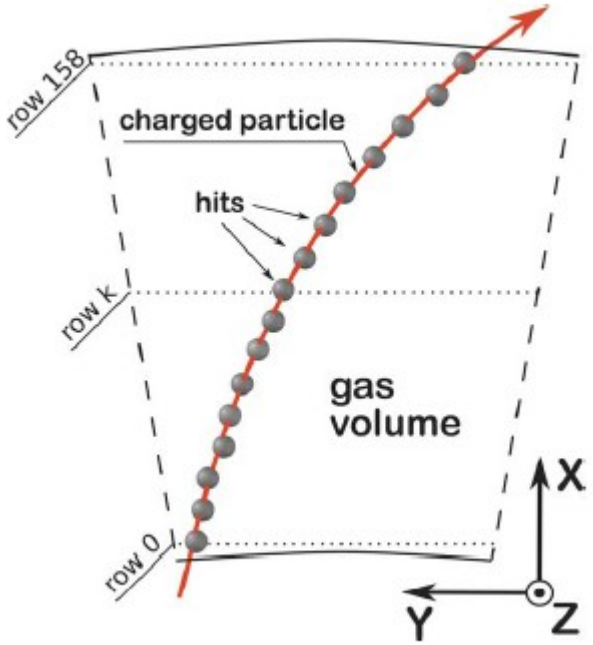
The Time Projector Chamber @ ALICE
- main tracking detector of ALICE
- high occupancy (*20 ktracks/interaction*)
- *2000 evts/s (pp)*
- *300 evts/s (pb-pb)*



TPC divided into 2 cylinders in z, 18 sectors each



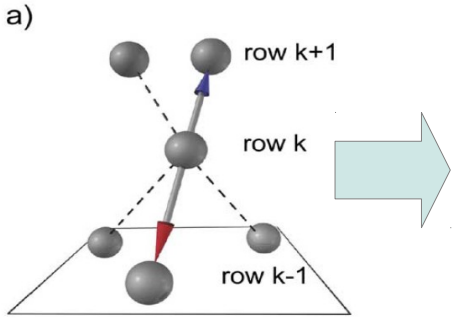
Each sector, 159 rows



ALICE tracking algorithm

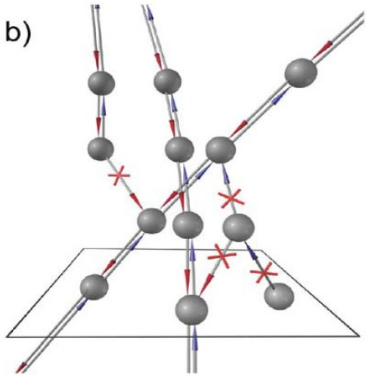
High degree of parallelism

1) Neighbor finder
For each hit at row k, the best pair of neighboring hits from row k+1 and k-1is found (best = straight line)

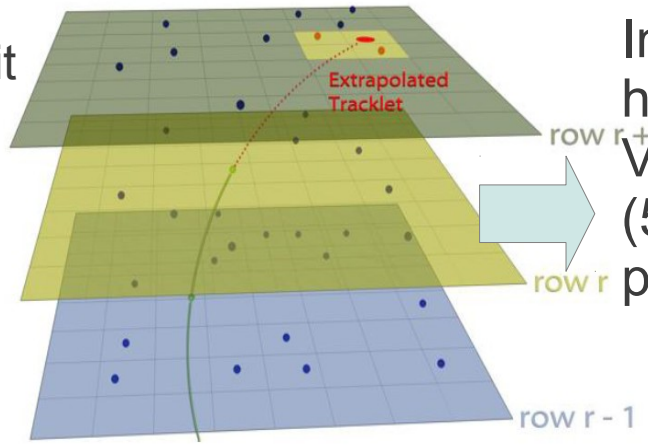


In parallel for every hit

2) Evolution
Reciprocal links are determined and saved



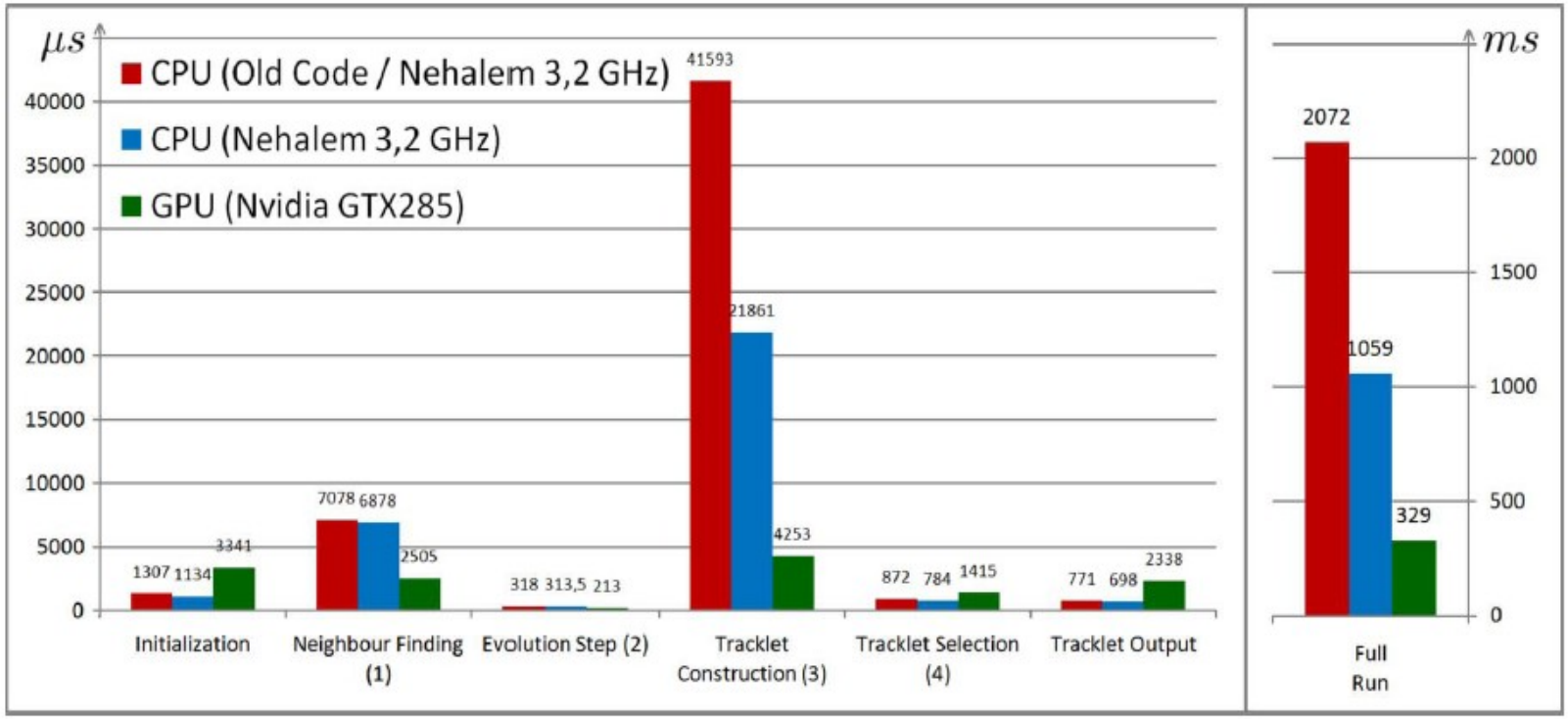
3) Tracklet construction
Tracklets are created following hit-to-hit links;
Kalman filter to fit geometrical trajectories



In parallel for every hit-to-hit link.
Very time consuming (50% of total processing time)

4) Tracklet selection
In case of tracks with intersected parts, the longest is kept.
Quality checks are performed

Results

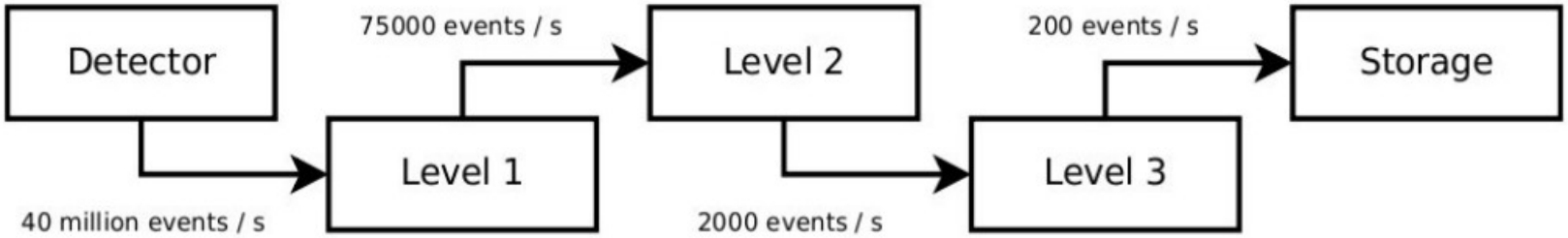


- Impressive reduction of processing time for Tracklet Construction on GPU;
- Significant impact on Neighbor Finding;
- Inizialization, Tracklet Selection and Tracklet Output better on CPU.
- Overall total processing time from 1 s to 300 ms.

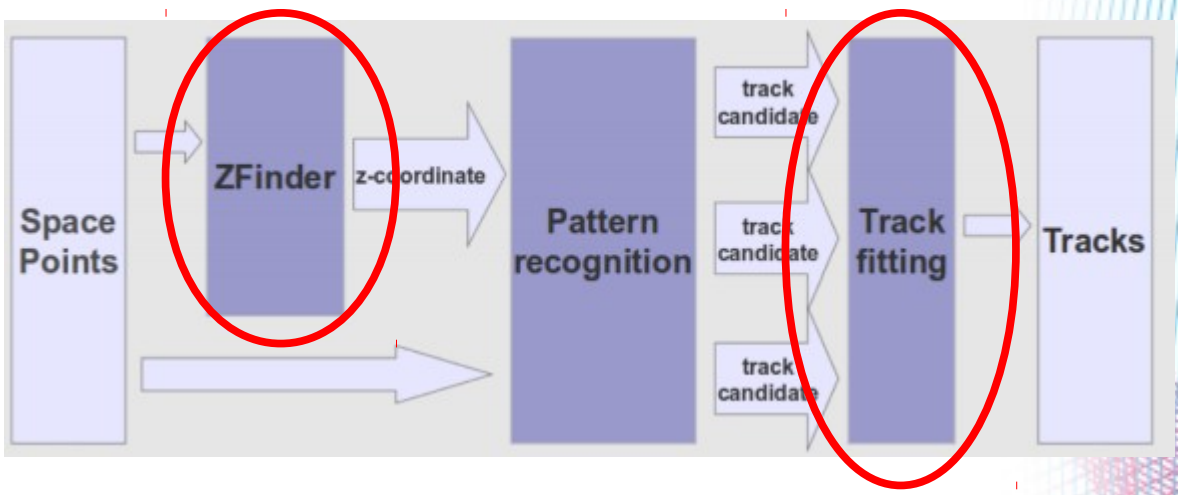
References:

Nuclear Science, IEEE Transactions on, Volume: 58 , Issue: 4 , Part: 1
Publication Year: 2011 , Page(s): 1845 - 1851

GPUs in ATLAS trigger system



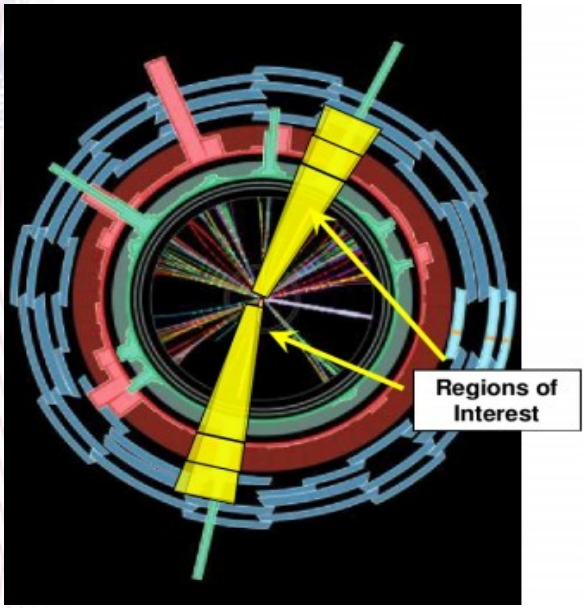
- Level 1: Custom built hardware with special processor units.
- **Level 2**: Software trigger operating independently on detector regions of interest (RoIs). **Ideal for GPGPUs**
- Event filter (Level 3): Software trigger analysing whole event signatures.

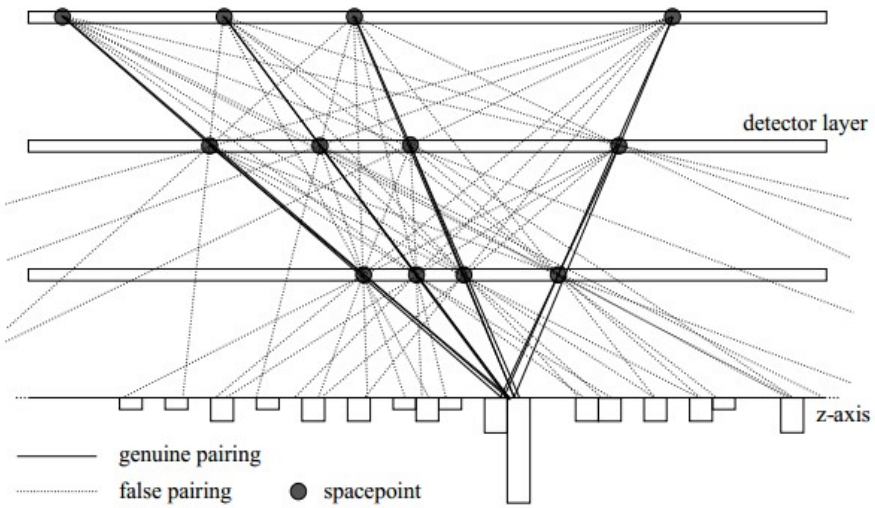


References:

https://twiki.cern.ch/twiki/bin/view/Main/AtlasEdinburghGPUComputing

Z finder algorithm

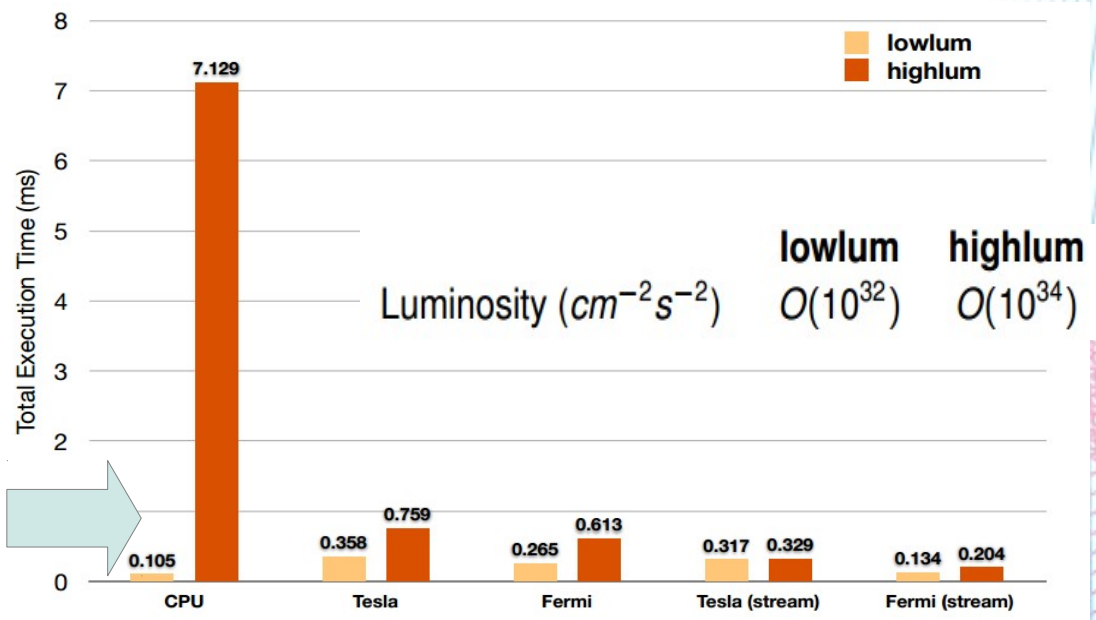




It processes each combination of spacepoints and extrapolates to the beamline.
The histogram peak is the chosen interaction point.

- One GPU block per phi slice
- Histogram per block in shared memory
- Phi slices processed independently

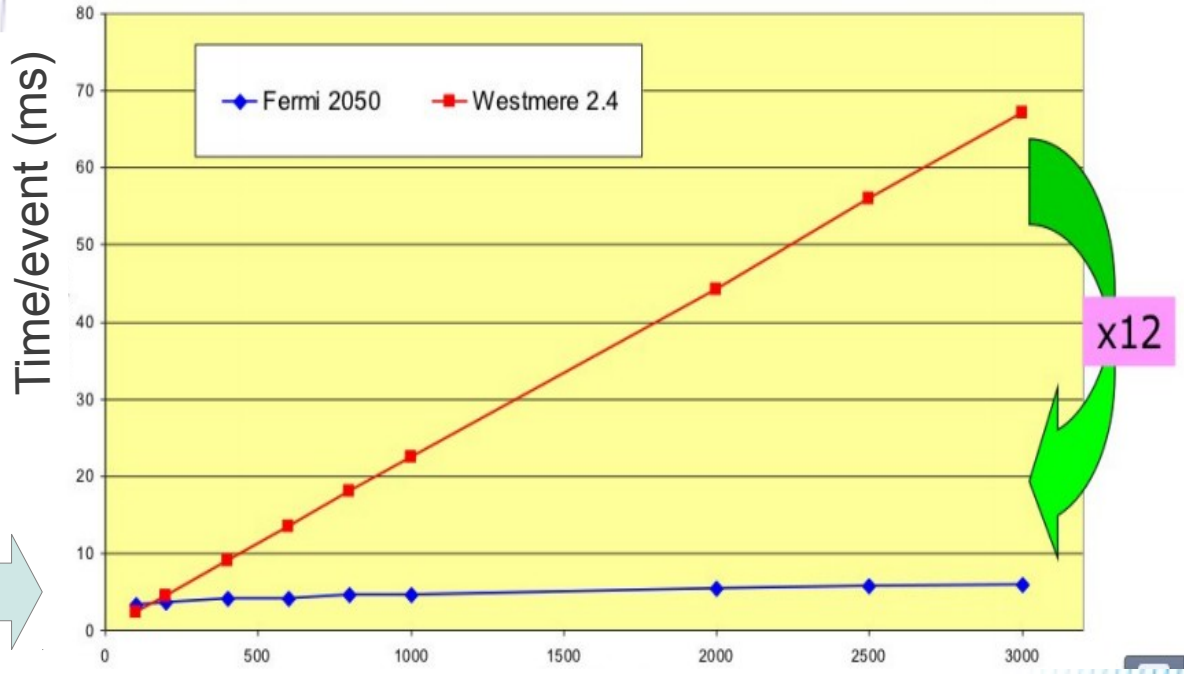
Up to 35x speed-up improvement (on Fermi)



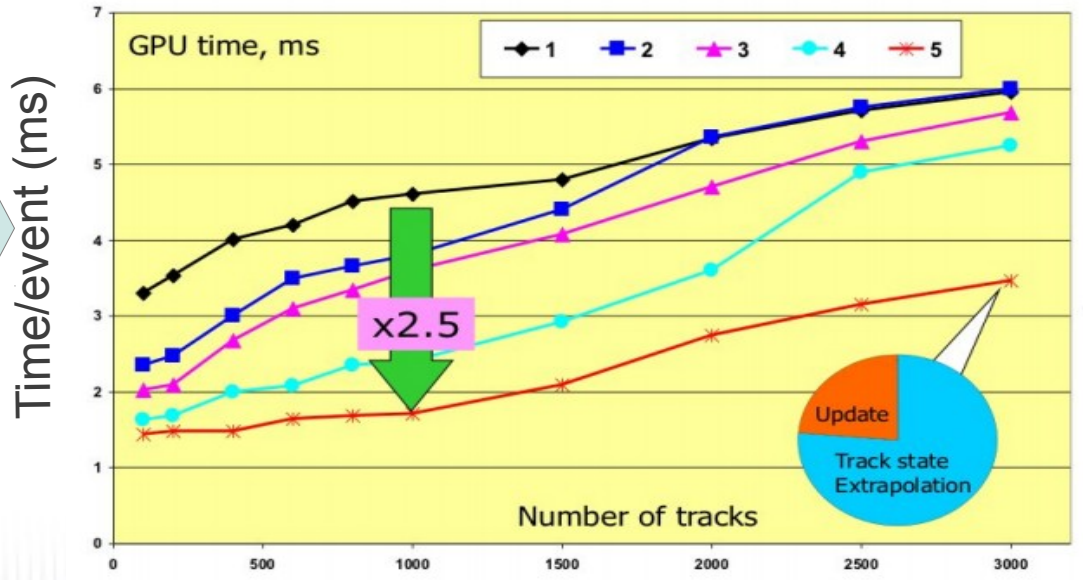
Kalman filter

In ATLAS HLT tracks are reconstructed using the Kalman filter method. Track trajectory predicted using detector hits.

CPU: Intel Westemere 2.4 GHz
GPU: NVIDIA Tesla C2050
Muon tracks, pt = 10 GeV/c, up to 3k tracks/event

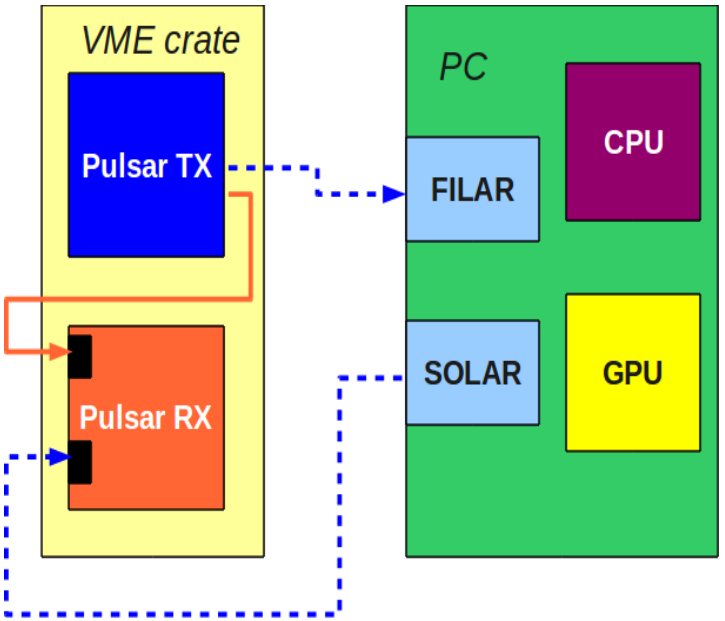


A set of optimizations applied:
1) Original code
2) 32 threads/block
3) Reduced memory usage
4) Track state stored in shared memory
5) Jacobian in shared memory



Generic R&D studies @ CDF

Goal: investigate the potential and limitations of GPUs for low latency (few tens of μs) real-time applications in a realistic HEP trigger environment (CDF Level-2 trigger test stand).



Pulsar : general purpose 9U VME boards.
GPU GTX 285
(30 microprocessors, 240 cores)

Measurements:

- Latency of data transfers between CPU and GPU
- Latency of a real time trigger algorithm implemented on GPU

Our benchmark algorithm is CDF Silicon Vertex Trigger linearized track fitting

Track parameters

Precalculated constants

Hit positions

$$p_i = \vec{f_i}\,\vec{x_i} + q_i$$

References:

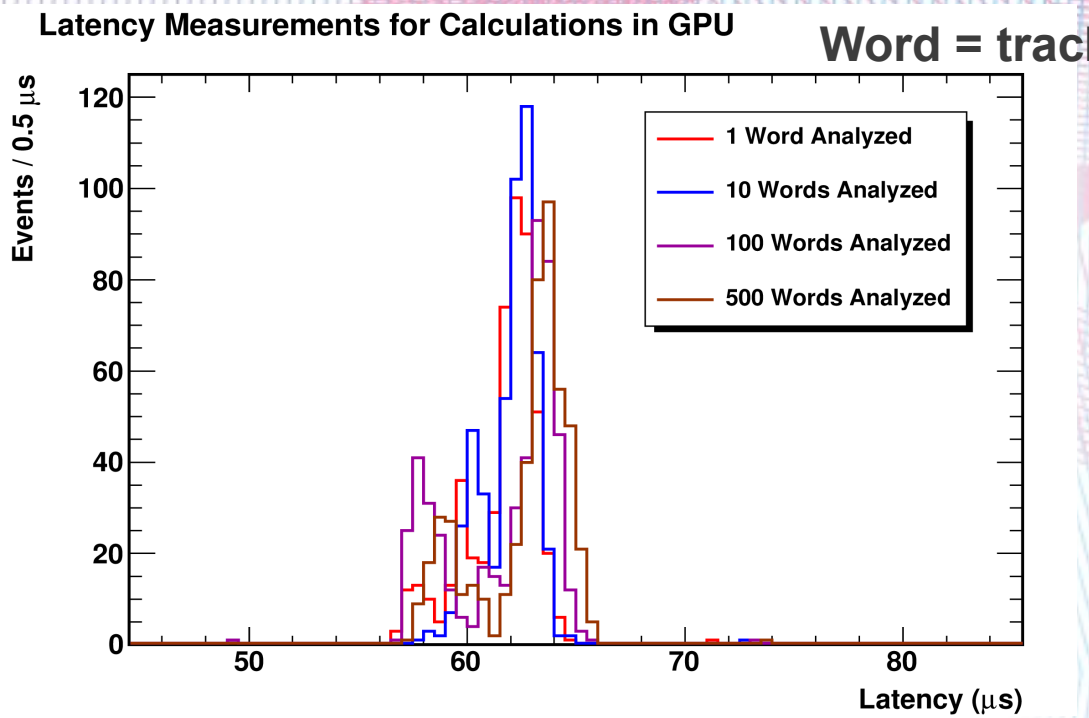
http://indico.cern.ch/contributionDisplay.py?contribId=205&sessionId=19&confId=102998

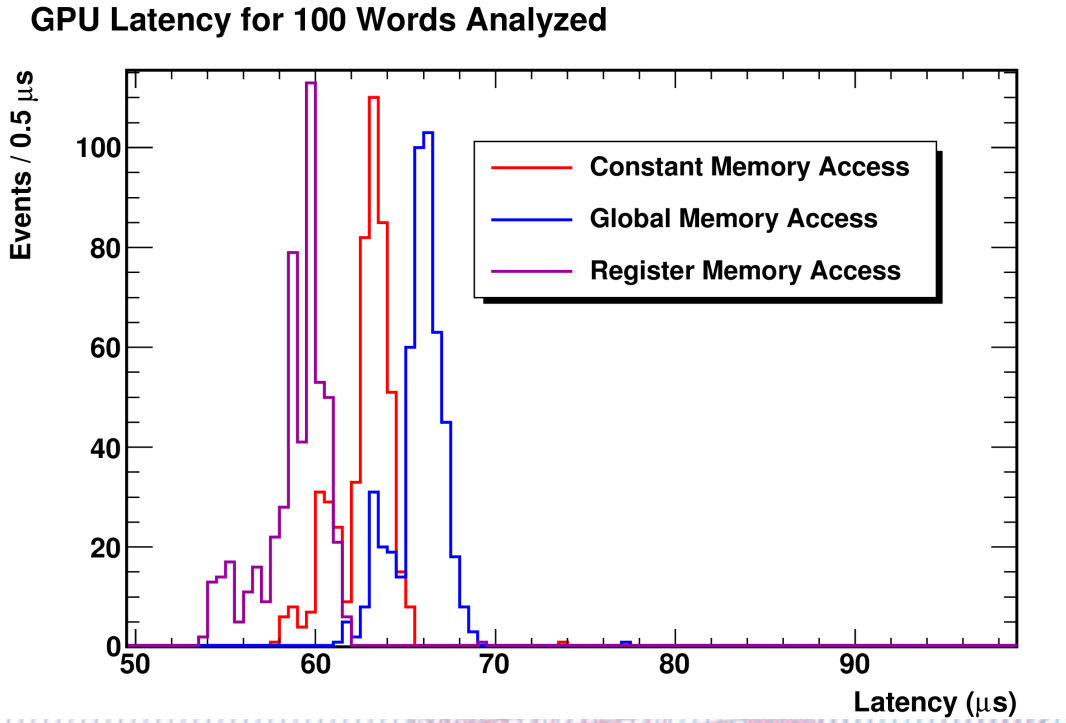
Results

Total latency to fit 500 tracks is about 60 μs.
About half of the time is due to data transfer between CPU and GPU and back → *Room for improvement (e.g. GPUDirect)*

Significant latency reduction (10 μs) using Register memory for track fitting constants.

Further studies ongoing...
- Kalman filter
- Comparison of different GPU cards





Other developments: CARMA @ FermiLAT

The CUDA on ARM Development Kit

A high performance, energy efficient development kit featuring a NVIDIA® Tegra ARM CPU, NVIDIA® CUDA® GPU and hardware developed by SECO.

The CUDA on ARM devKit, codename CARMA, is an ARM-based GPU computing development kit created to support the growing demand for energy-efficient computing initiatives around the world.

Technical Specs for the Development Kit:

CPU	NVIDIA Tegra 3 Quad-Core ARM A9
GPU	NVIDIA QuadroTM 1000M with 96 CUDA Cores

Fast processing of images (1500 pixels, 30 slices/pixel, 300 Hz ≈ 15 MB/s)

Applications:

Calibration: correcting offsets and scales

Cleaning: removing Night Sky Background

Data reduction: Hillas analysis (Computation of image moments related to energy, direction, *hadronness, …)*

References:

D.Bastieri (PD), L.A.Antonelli (INAF-OAR)

Summary

The usage of GPUs in real time event selections is being pursued by several HEP experiments.

In *High Level Triggers*, where there is no latency restriction, their computing power can help reduce the size of PC farms.

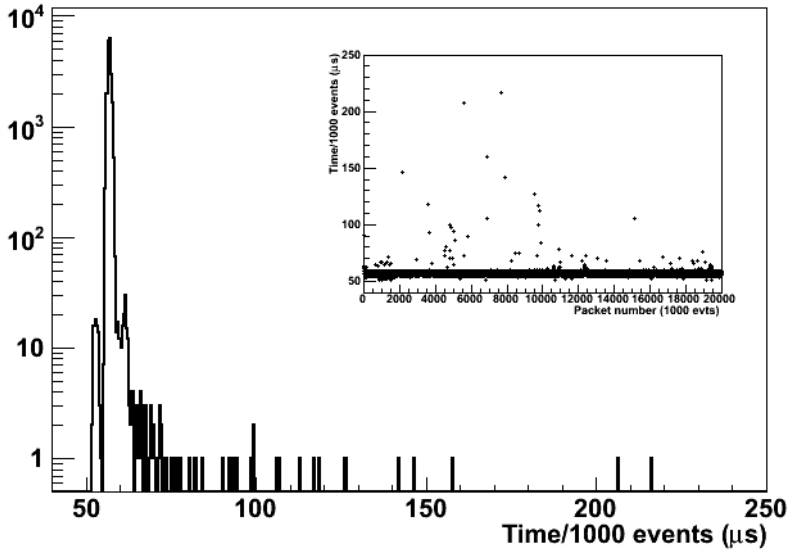
Applications to *Low Level Trigger* looks still very challenging, but first results are promising.

The interest of the Italian community is increasing.
Different experiences that can be shared.

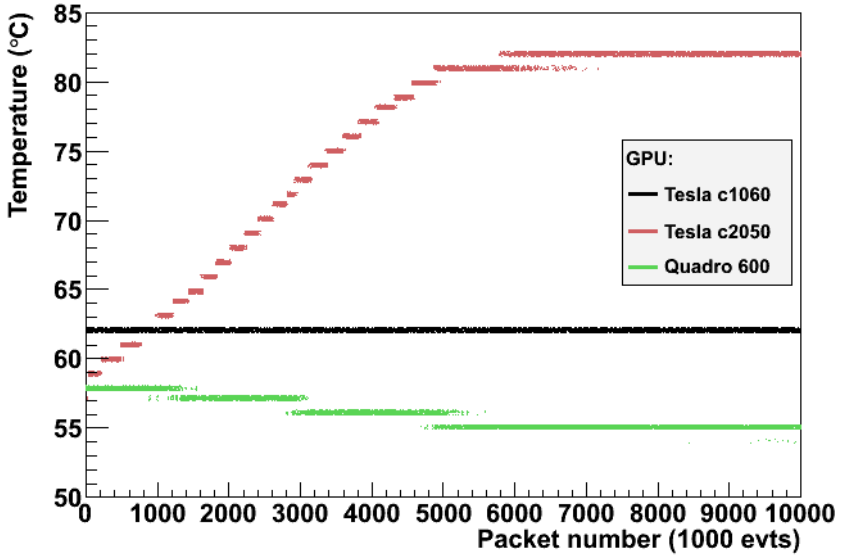
BACKUP

NA62: processing time stability

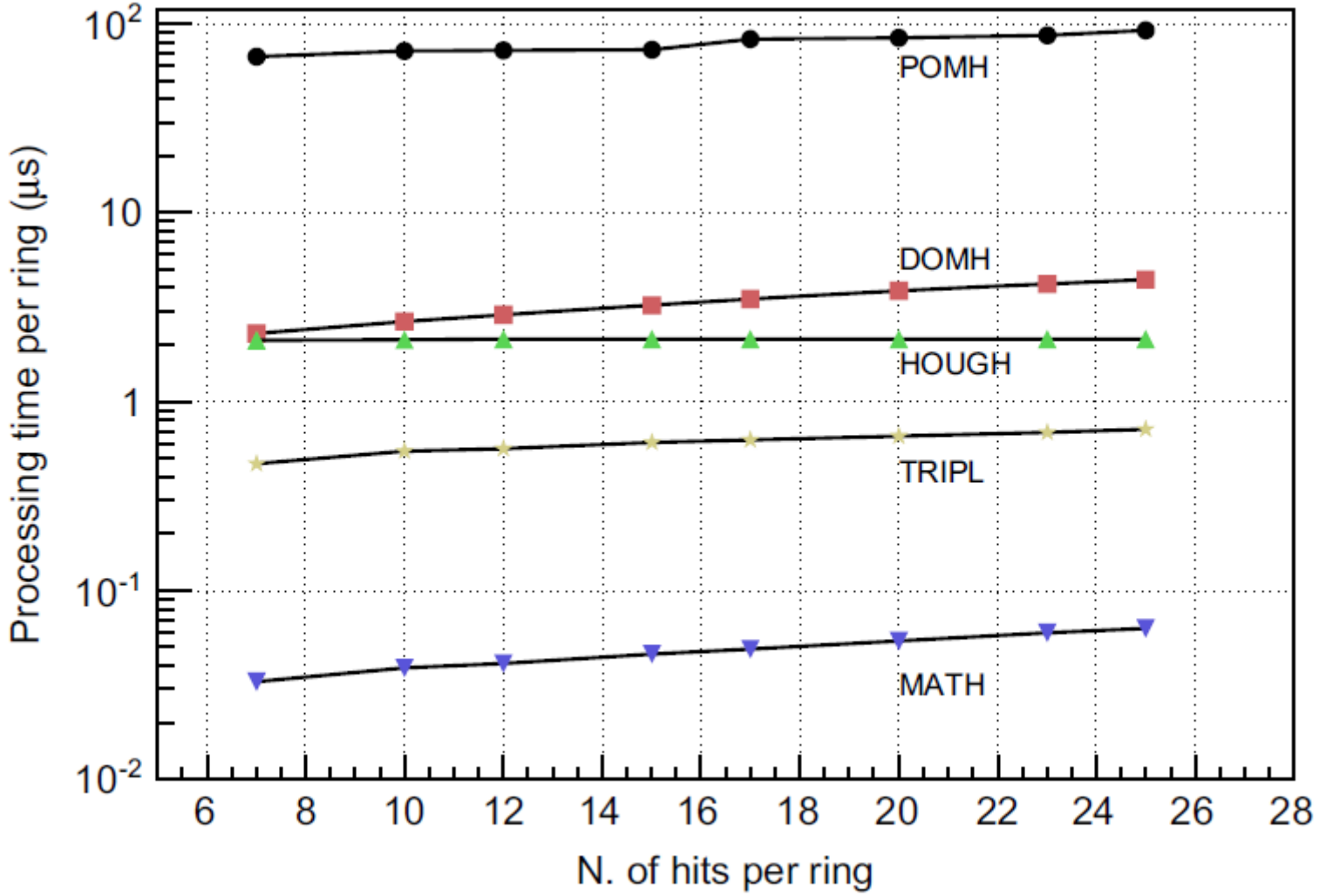
- The stability of the execution time is an important parameter in a synchronous system
- The GPU (Tesla C1060, MATH algorithm) shows a "quasi deterministic" behavior with very small tails.



- The GPU temperature, during long runs, rises in different way on the different chips, but the computing performances aren't affected.



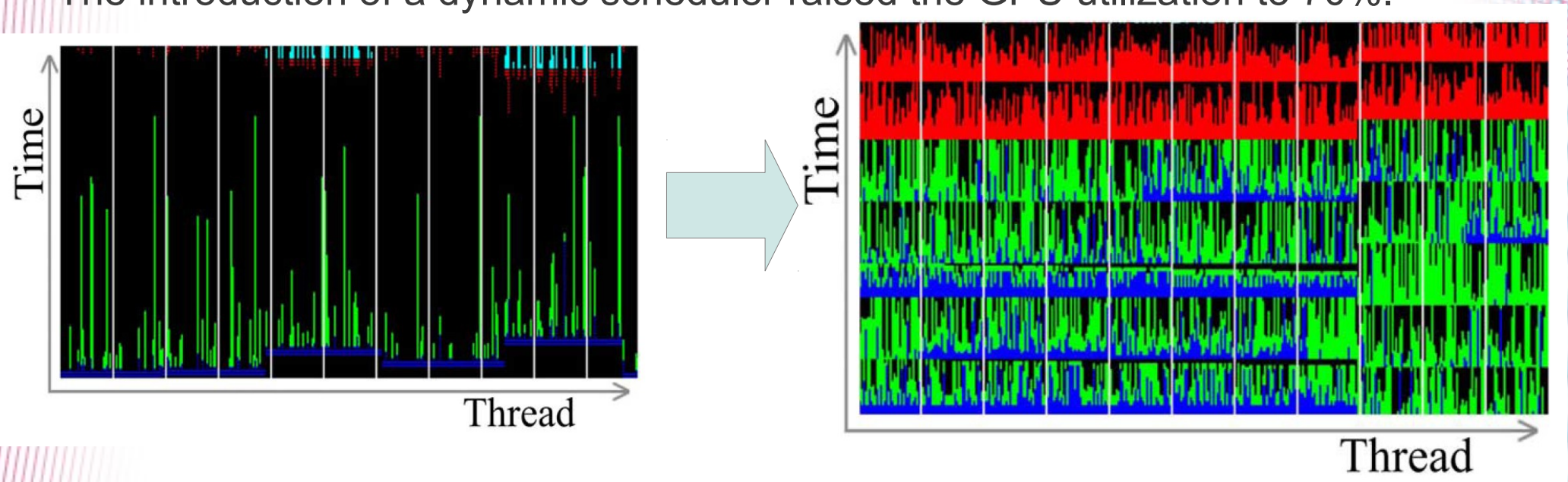
NA62: processing time vs # hits



ALICE track fitting: implementation on GPU

All threads within one *warp* must execute a common instruction → they have to wait for the one thread processing the longest tracklet → GPU Utilization below 20%!

The introduction of a dynamic scheduler raised the GPU utilization to 70%.



To allow efficient scheduling the steps are pipelined asynchronously using both, CPU and GPU, while data is transferred via DMA