GPU Computing and its applications in HEP
*Lecture **2***

Use of GPUs for triggering in HEP experiments

Felice Pantaleo

CERN

Inverted CERN School of Computing, 25-26 February 2013

CONCEPTS

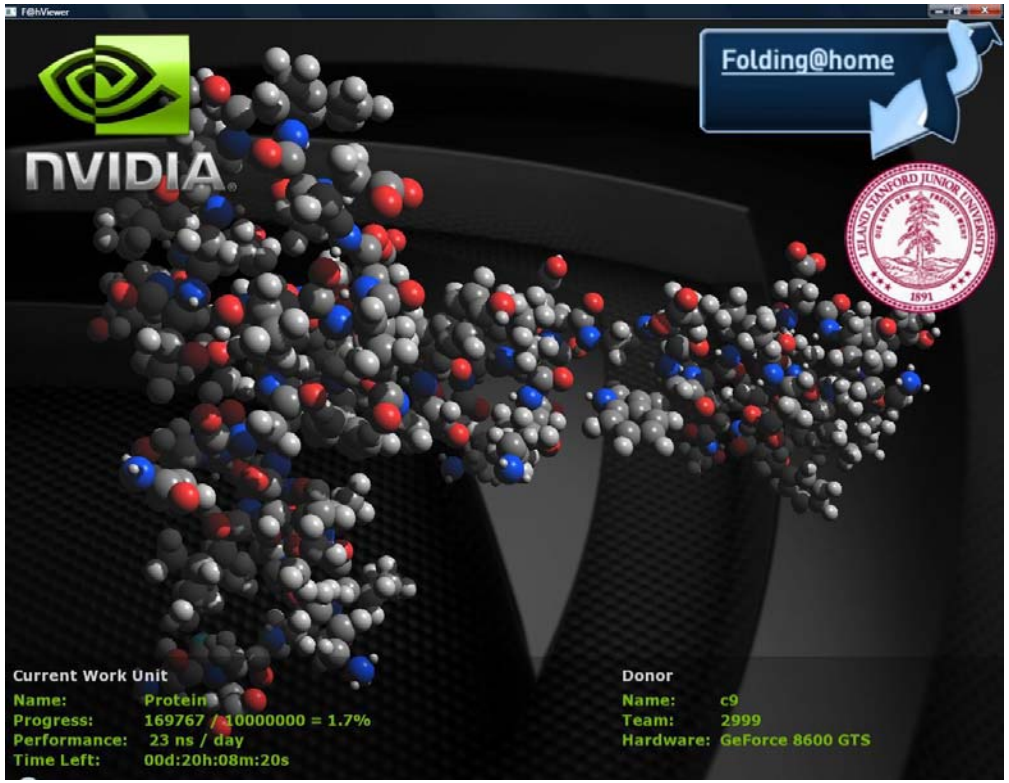
- GPU
- Trigger
- ALICE HLT
- NA62 Low Level Trigger
- Algorithms
- Scheduler
- Tests

GPU

- GPU
- Trigger
- ALICE HLT
- NA62 Low Level Trigger
- Algorithms
- Scheduler
- Tests

Accelerators

- Exceptional raw power wrt CPUs
- Higher energy efficiency
- Plug & Accelerate
- Massively parallel architecture
- Low Memory/core



Accelerators

- GPUs were traditionally used for real-time rendering. NVIDIA & AMD main manufacturers.
- Intel introduced the coprocessor Xeon Phi (MIC)

NVIDIA CUDA

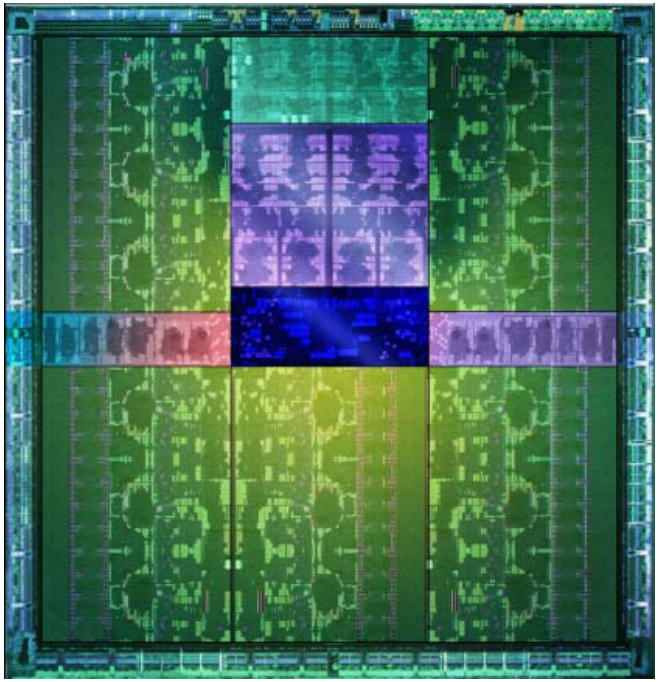
- SMX executes hundreds of threads concurrently.
- SIMT (Single-Instruction, Multiple-Thread)
- Instructions pipelined
- Thread-level parallelism
- Instructions issued in order
- No Branch prediction
- No speculative execution
- Branch predication

What is CUDA?

- **CUDA Architecture**
  - Expose GPU parallelism for general-purpose computing
  - Retain performance
- **CUDA C/C++**
  - Based on industry-standard C/C++
  - Small set of extensions to enable heterogeneous programming
  - Straightforward APIs to manage devices, memory etc.

CUDA - Abstractions

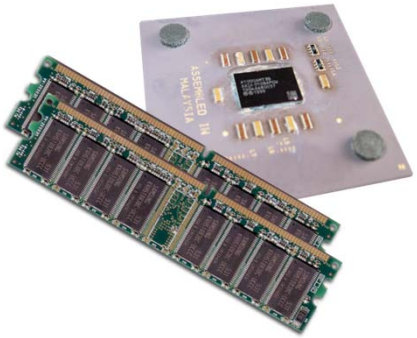
1. Hierarchy of thread groups
2. Shared memory
3. Barrier synchronization



- Fine-grained data/thread parallelism
- Coarse-grained data/task parallelism
- Instruction-level parallelism

Heterogeneous Computing

- **Terminology**
  - Host The CPU and its memory space
  - Device The GPU and its memory space



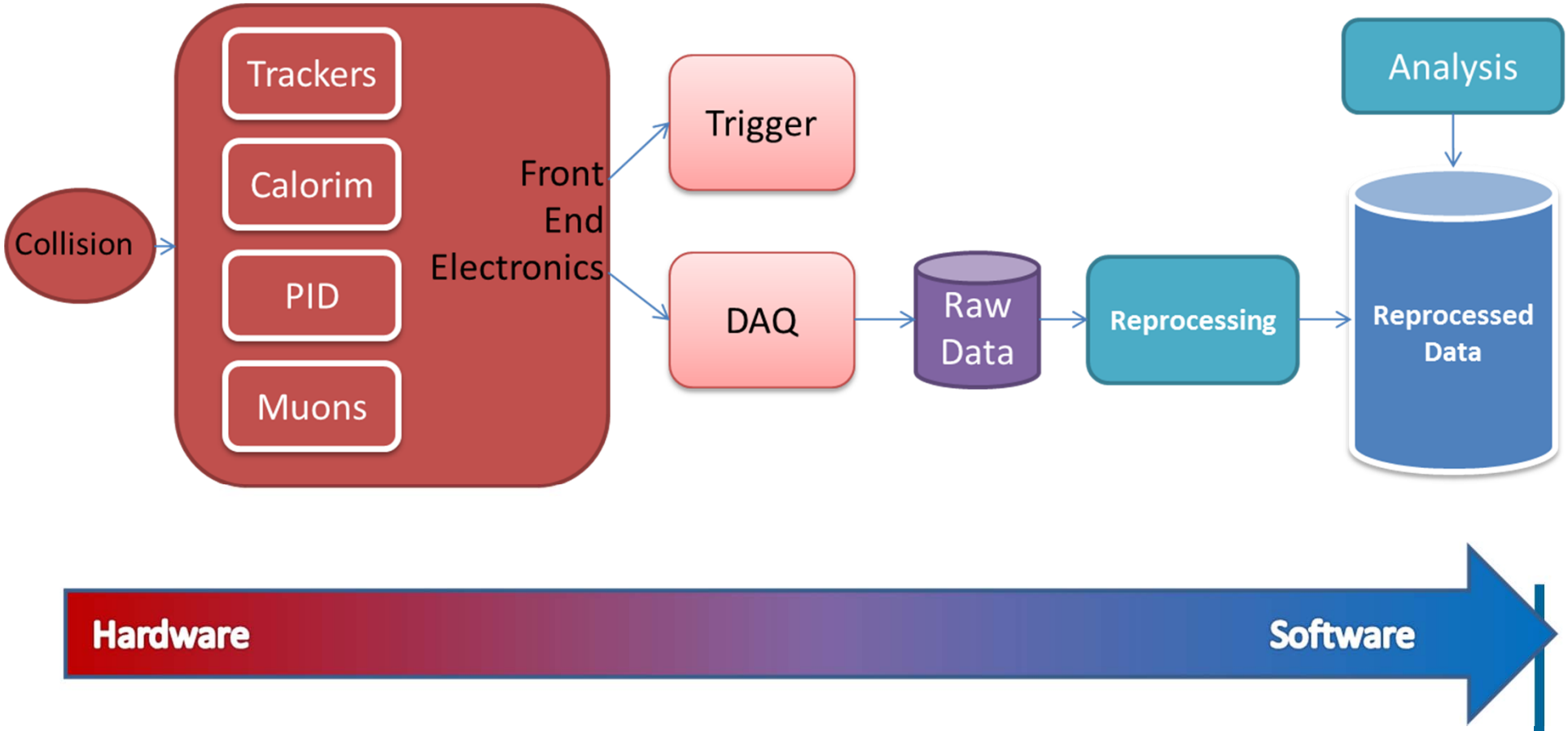
Host

Device

Trigger

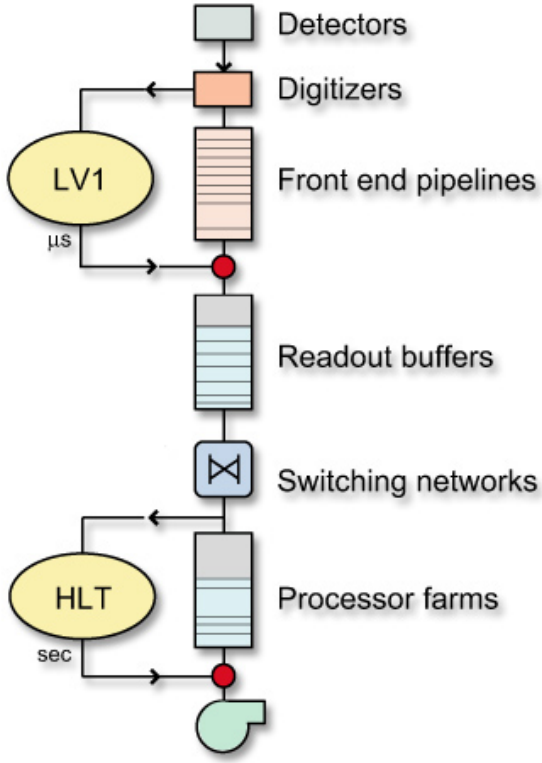
- GPU
- Trigger
- ALICE HLT
- NA62 Low Level Trigger
- Algorithms
- Scheduler
- Tests

Generic Detector structure



Generic Trigger structure

40 MHz
Clock driven
Custom processors

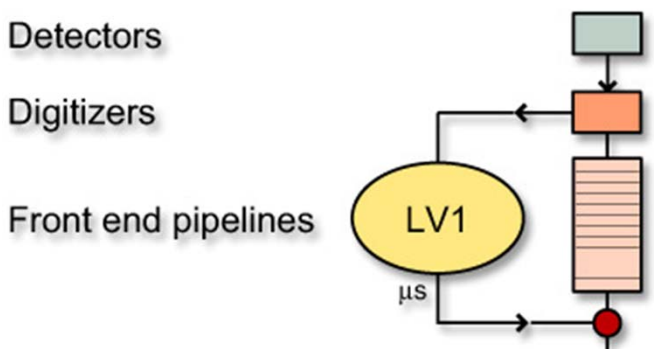


Low Level Trigger
NA62

100 kHz
Event driven
PC network

High-Level Trigger
ALICE

Low Level Trigger



- Time needed for decision $\Delta t_{dec} \approx 1$ ms
- Particle rate ≈ 10MHz
- Need pipelines to hold data
- Need fast response

- Backgrounds are huge
- High rejection factor

- Algorithms run on local, coarse data

- Ultimately, determines the physics

Parallel Opportunities

- **On detector (mostly Cellular Automata)**
  - Trackers
  - Calorimeters
- **Trigger Systems**
  - Event decision: one event per node + dispatcher
- **The Grid**

HEP is a long lived science...

- Many algorithms in HEP have cores dating the seventies
- Mostly thought and devised as single threaded
- Current solution: one event-one core
- This solution shows saturation

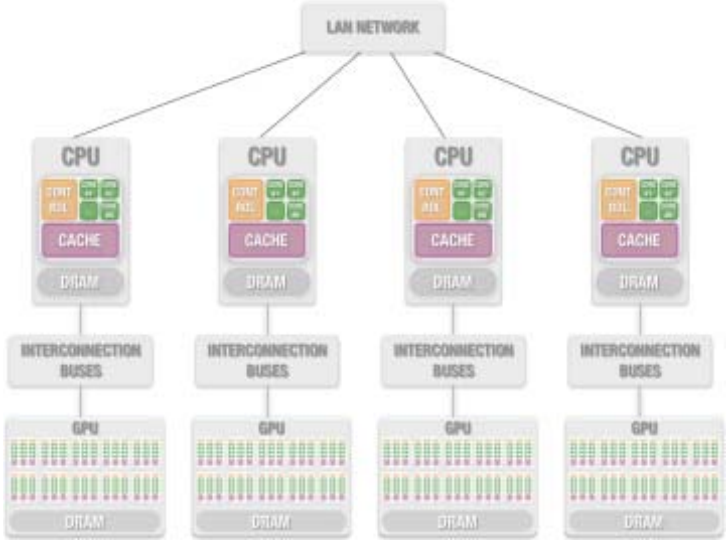
Change of paradigm ?

Opportunities for HEP experiments

- HEP characteristics and needs seem to fit perfectly both
  - Cellular techniques
  - Multi- Many- cores architectures
- Our feeling: a large number of opportunities ahead of us
- Further study is needed to evaluate the real benefits of these solutions.

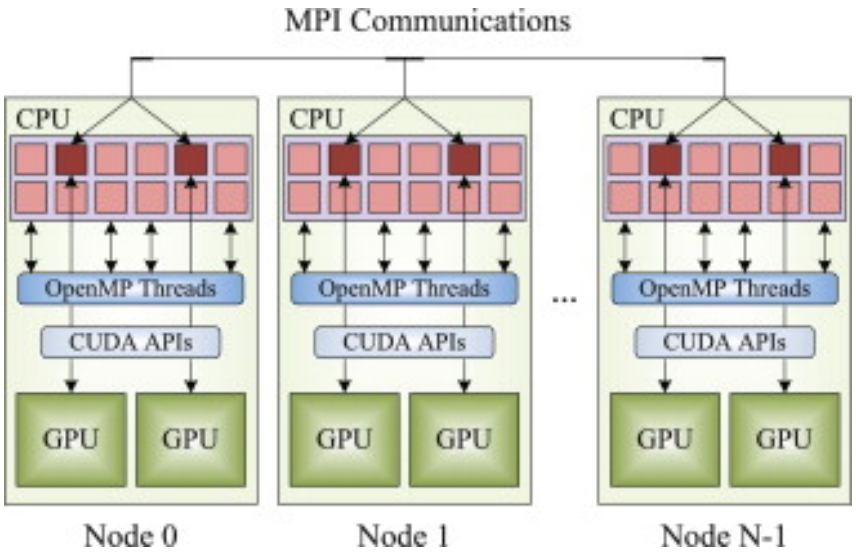
What is happening in HPC…

Use several platforms containing GPUs to solve one single problem



Programming challenges:

- Algorithm parallelization
- Perform computation in GPUs
- Execution in a distributed system where platforms have their own memory
- Network communication.



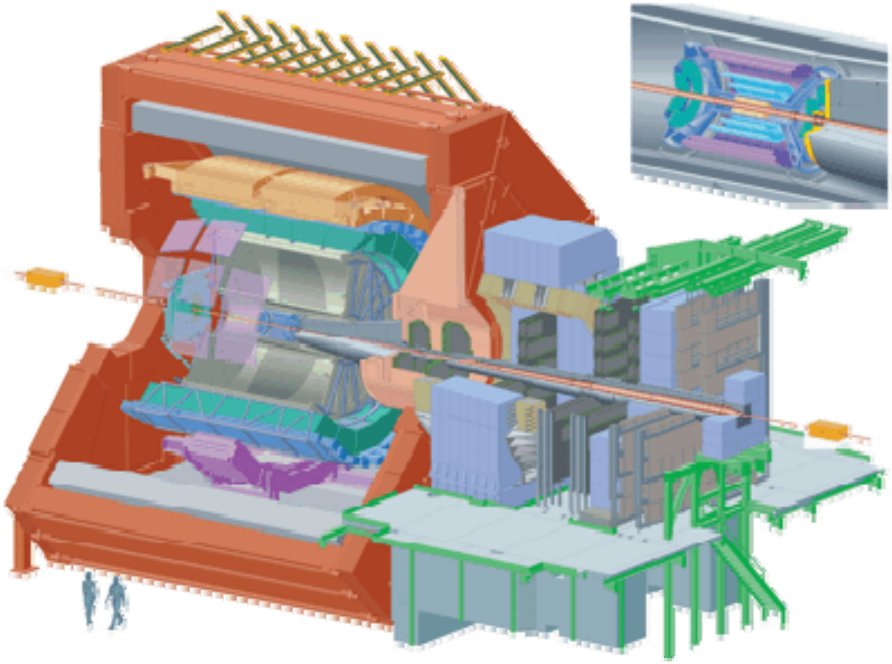
ALICE TPC HLT

- GPU
- Trigger
- ALICE HLT
- NA62 Low Level Trigger
- Algorithms
- Scheduler
- Tests

Thanks to ALICE Collaboration

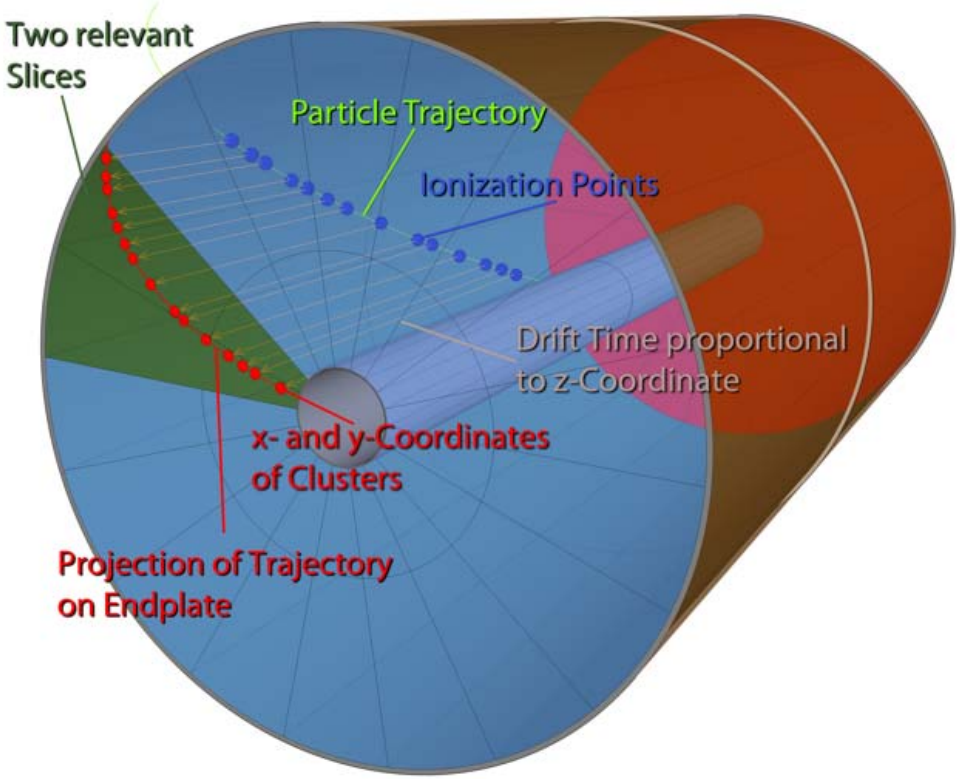
ALICE Detector

ALICE is one of the major four experiments of the Large Hadron Collider at CERN. It was specifically designed to study heavy ion collisions.



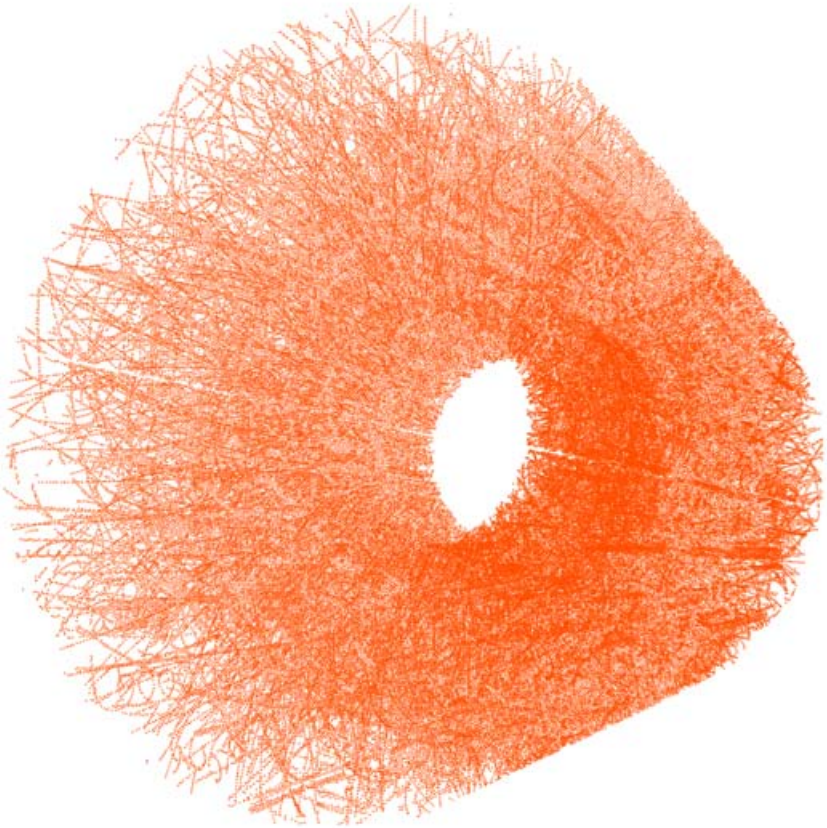
ALICE TPC

A Time Projection Chamber (TPC) is used to measure particle trajectories.



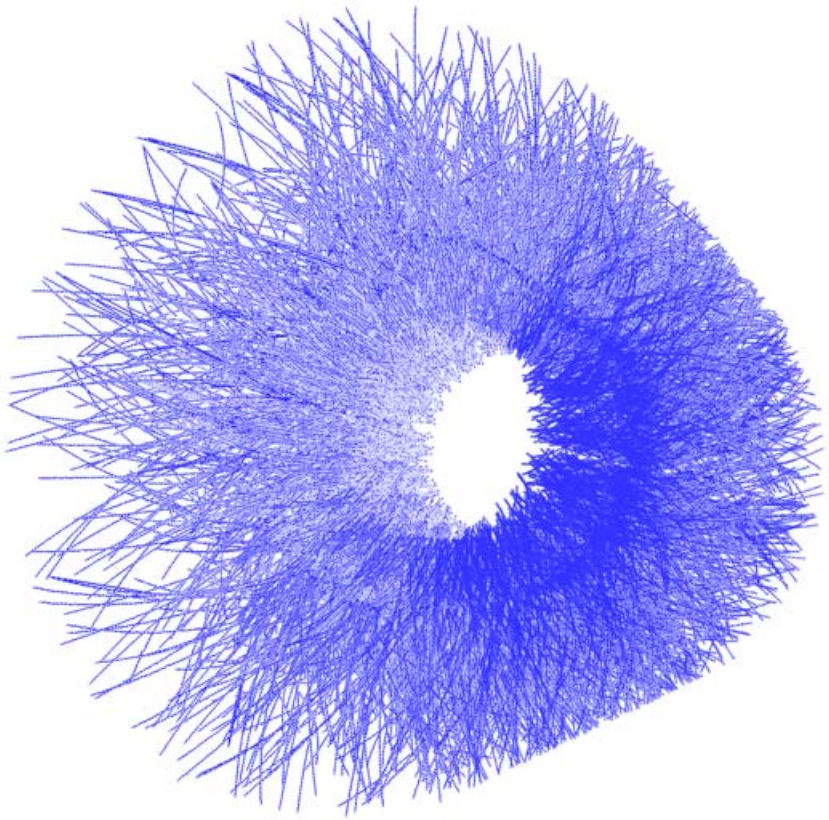
ALICE TPC

- **TPC clusters of a heavy ion event**



ALICE TPC

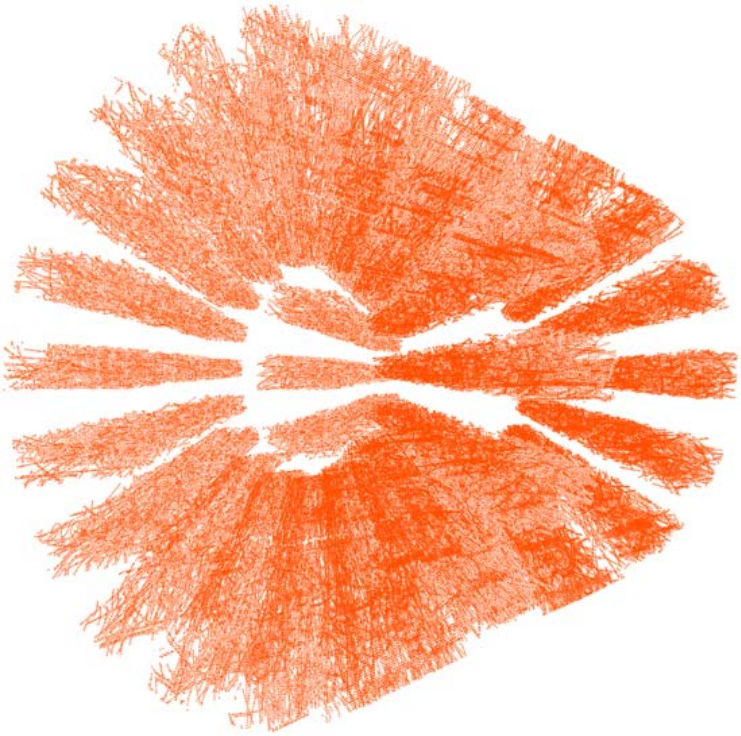
- **Tracks reconstructed from the clusters**



ALICE HLT

- Alice HLT tracker divides the TPC in slices and processes the slices individually.
- Track segments from all the

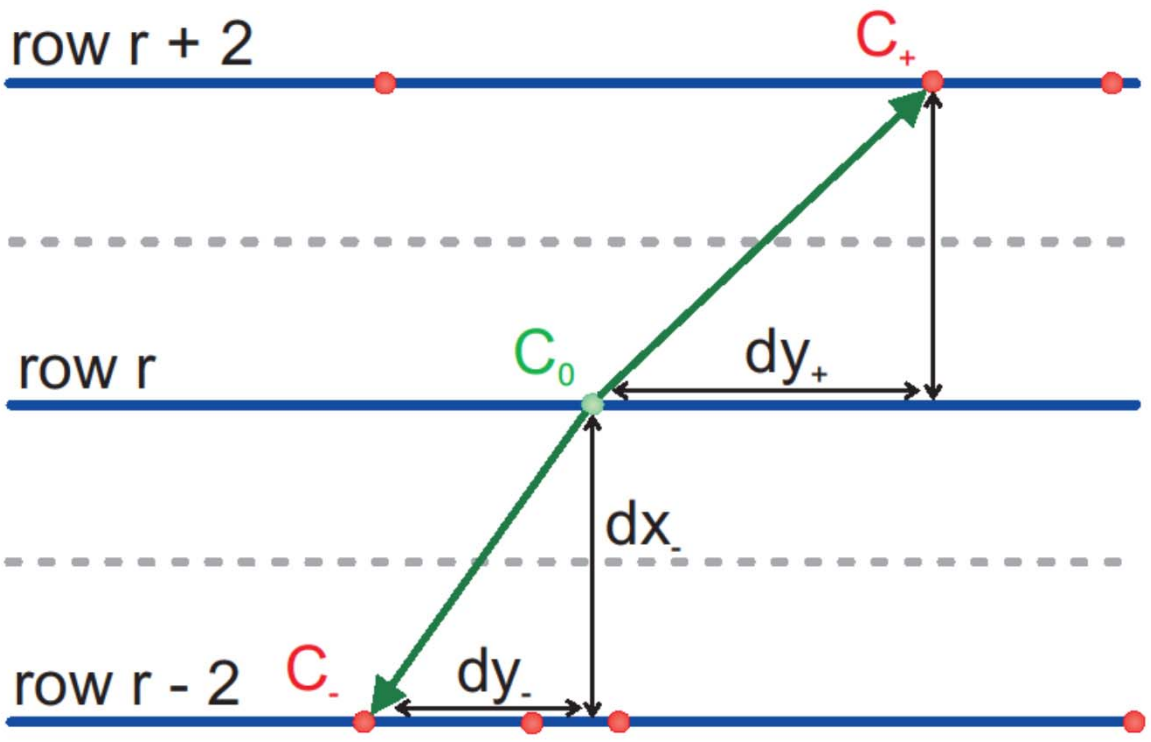
slices are merged later



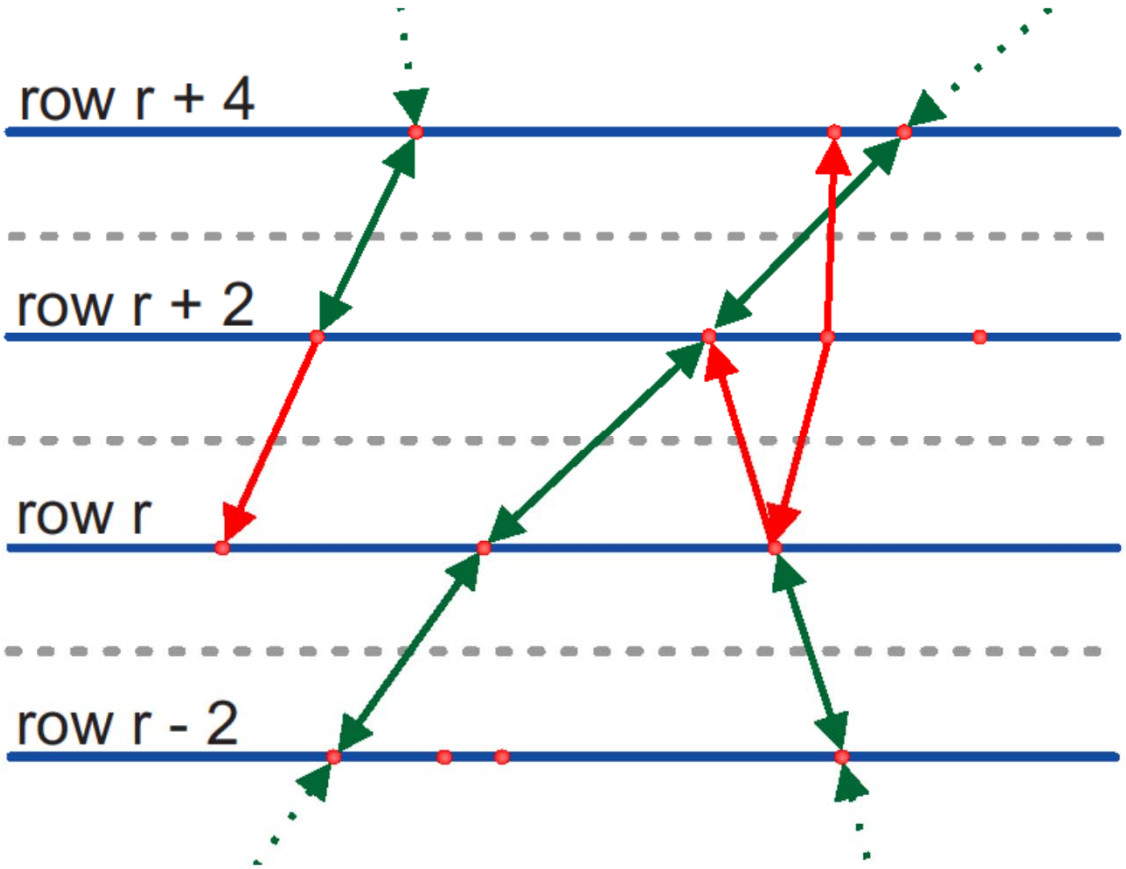
Tracking algorithm

Category of task	Name of task	Description on task
	(Initialization)	
Combinatorial part (Cellular automation)	I: Neighbors finding	Construct seeds (Track candidates)
	II: Evolution	
Kalman filter part	III: Tracklet construction	Fit seed, extrapolate tracklet, find new clusters
	IV: Tracklet selection	Select good tracklets, assign clusters to tracks
	(Tracklet output)	

Neighbor finding

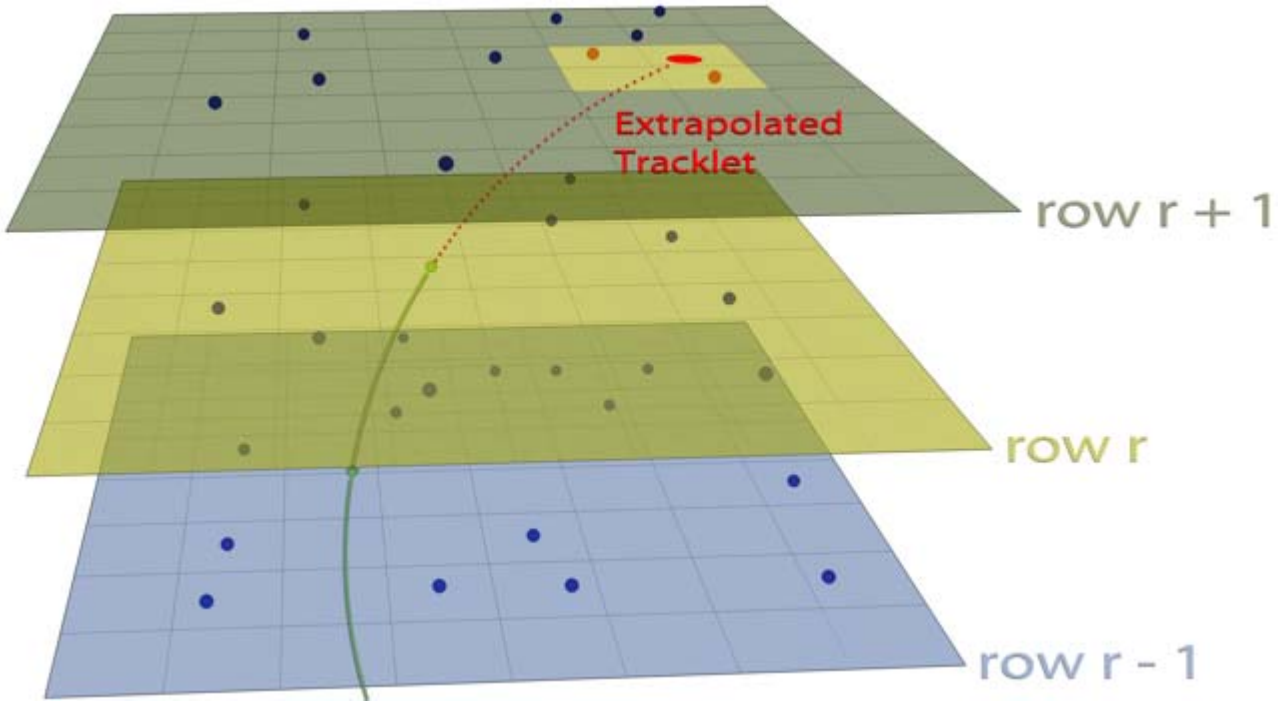


Evolution step

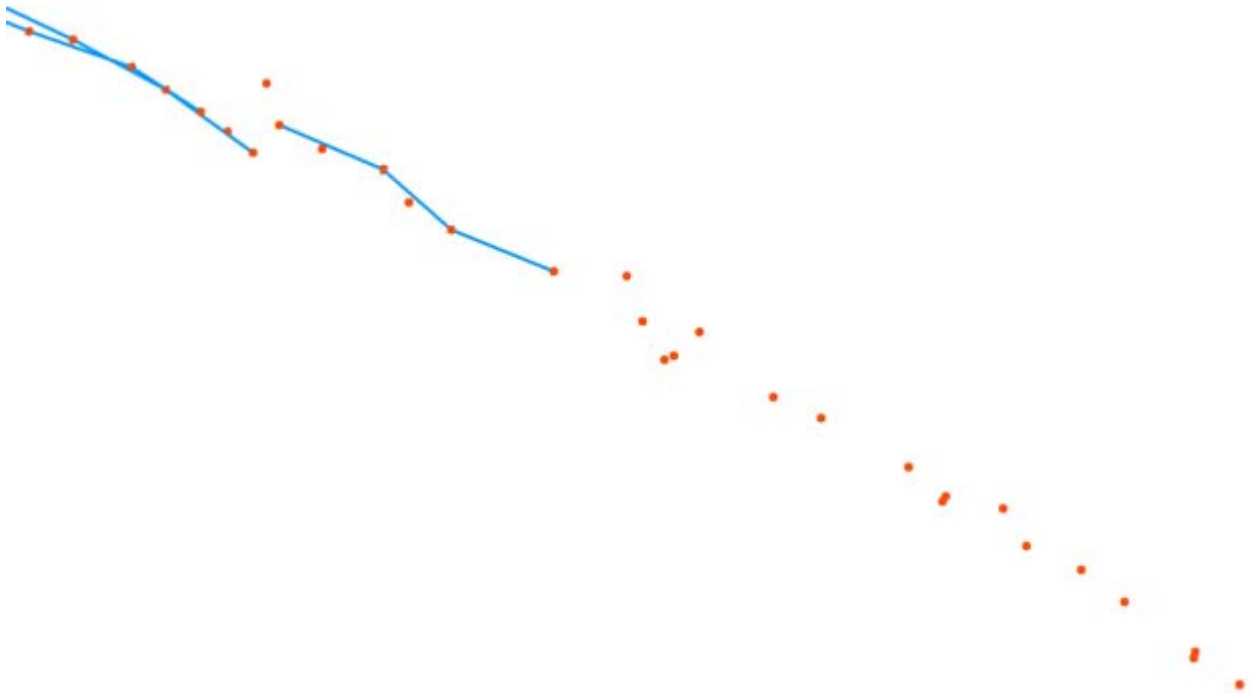


Tracklet reconstruction

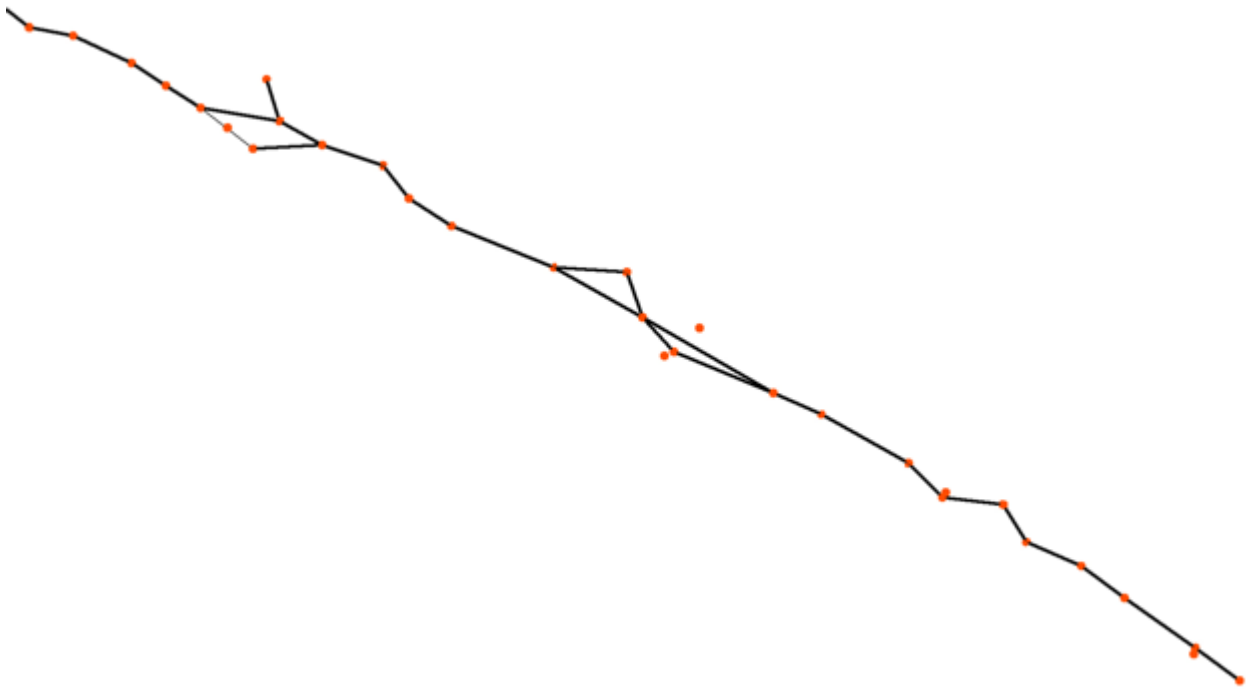
The algorithm looks for clusters close to extrapolation point



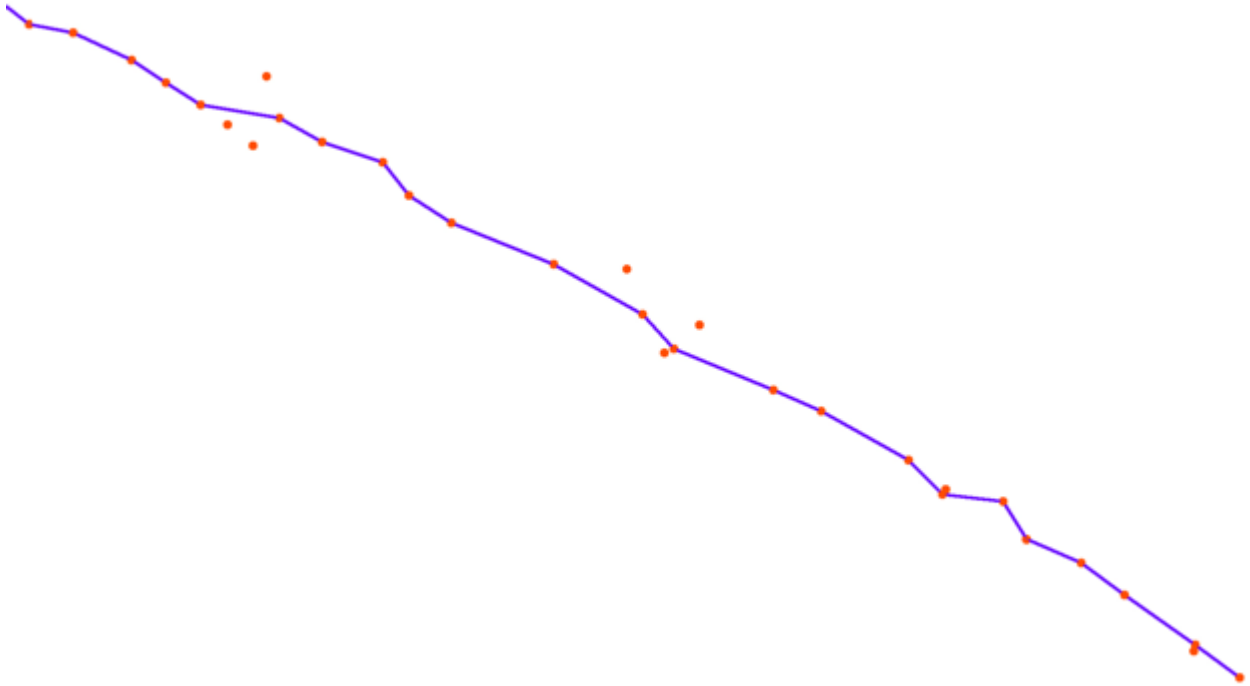
Evolution step



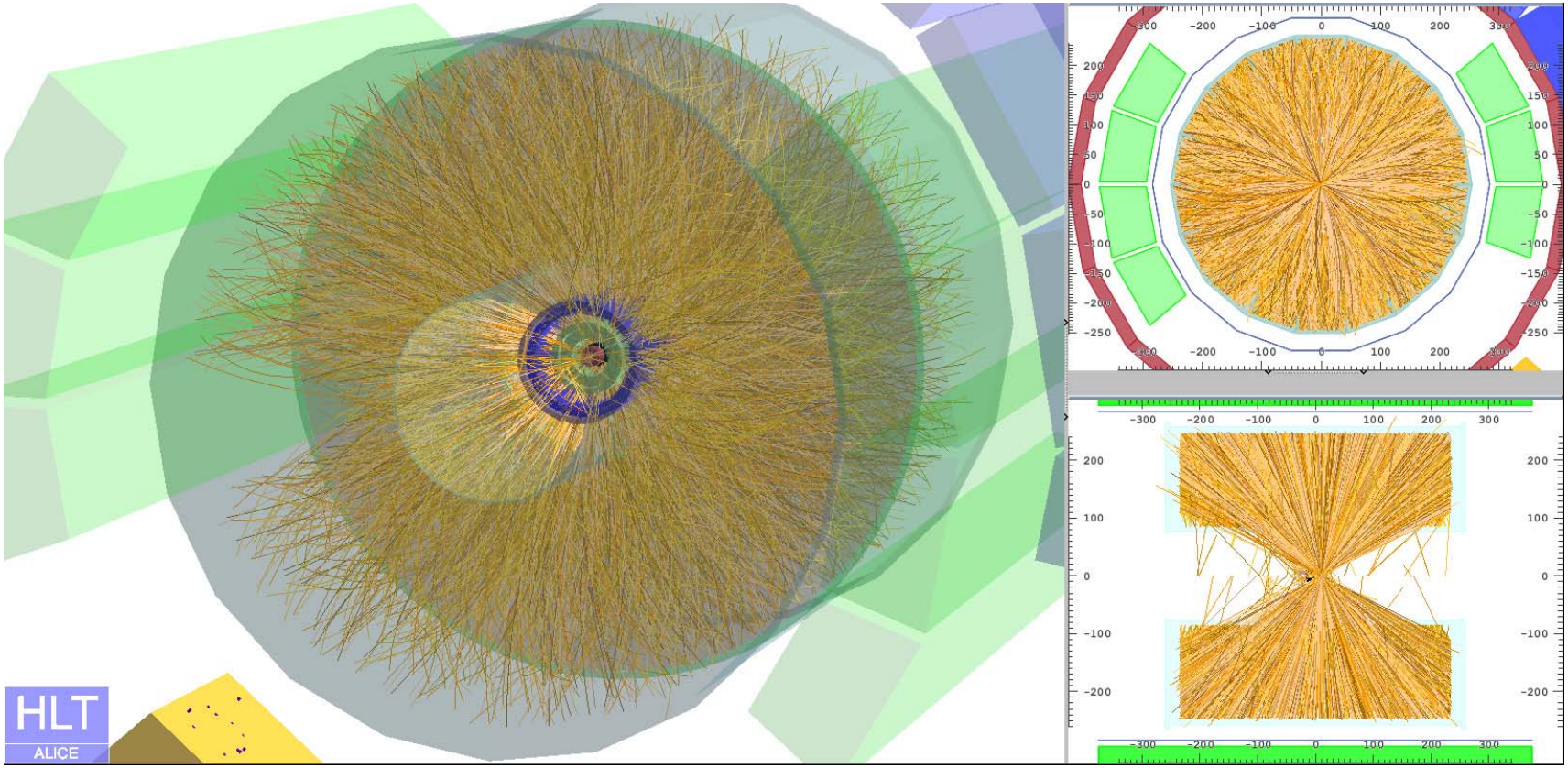
Tracklet construction



Tracklet selection



ALICE GPU Tracker



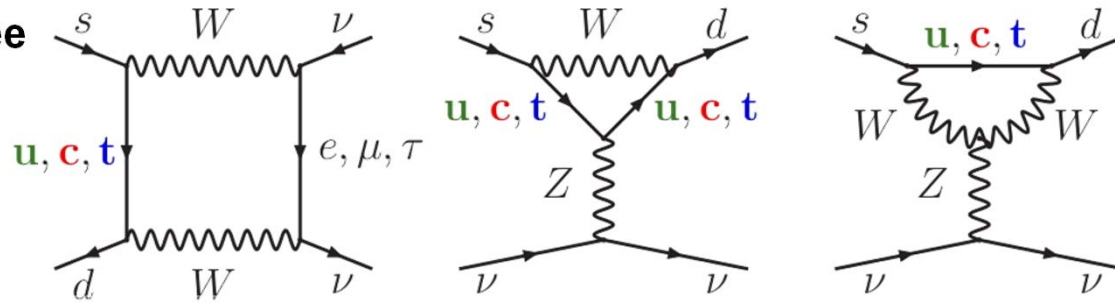
Screenshot of ALICE Online-Event-Display during first physics-fill with active GPU Tracker.

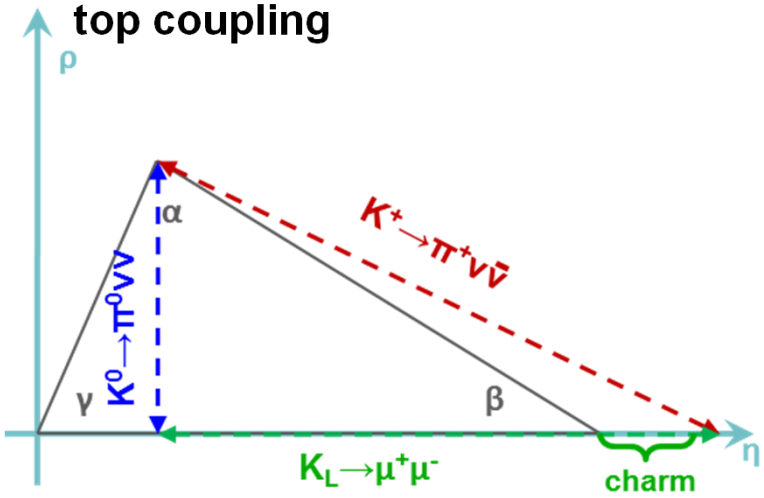
NA62 Low-Level Trigger

- GPU
- Trigger
- ALICE HLT
- NA62 Low Level Trigger
- Algorithms
- Scheduler
- Tests

K+ → π+νν in the Standard Model

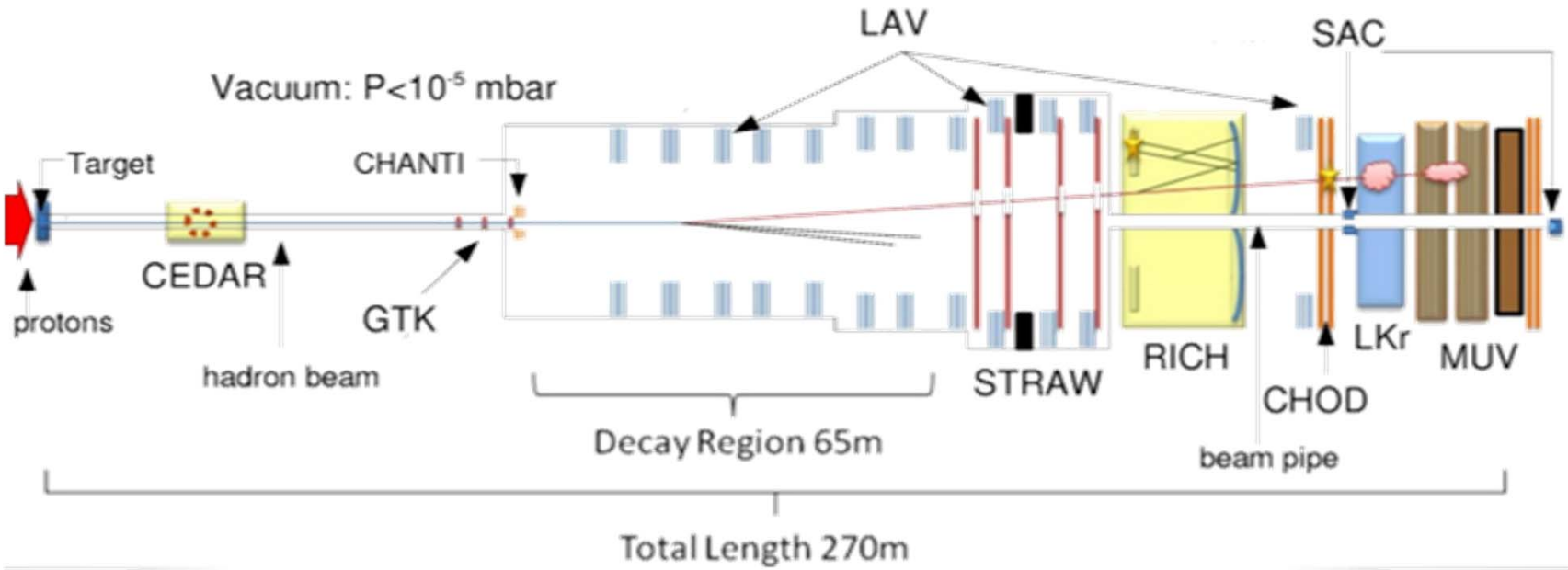
- **FCNC process forbidden at tree level**
- **Short distance contribution dominated by Z penguins and box diagrams**
- **Negligible contribution from u quark, small contribution from c quark**
- ***Very small* BR due to the CKM top coupling**





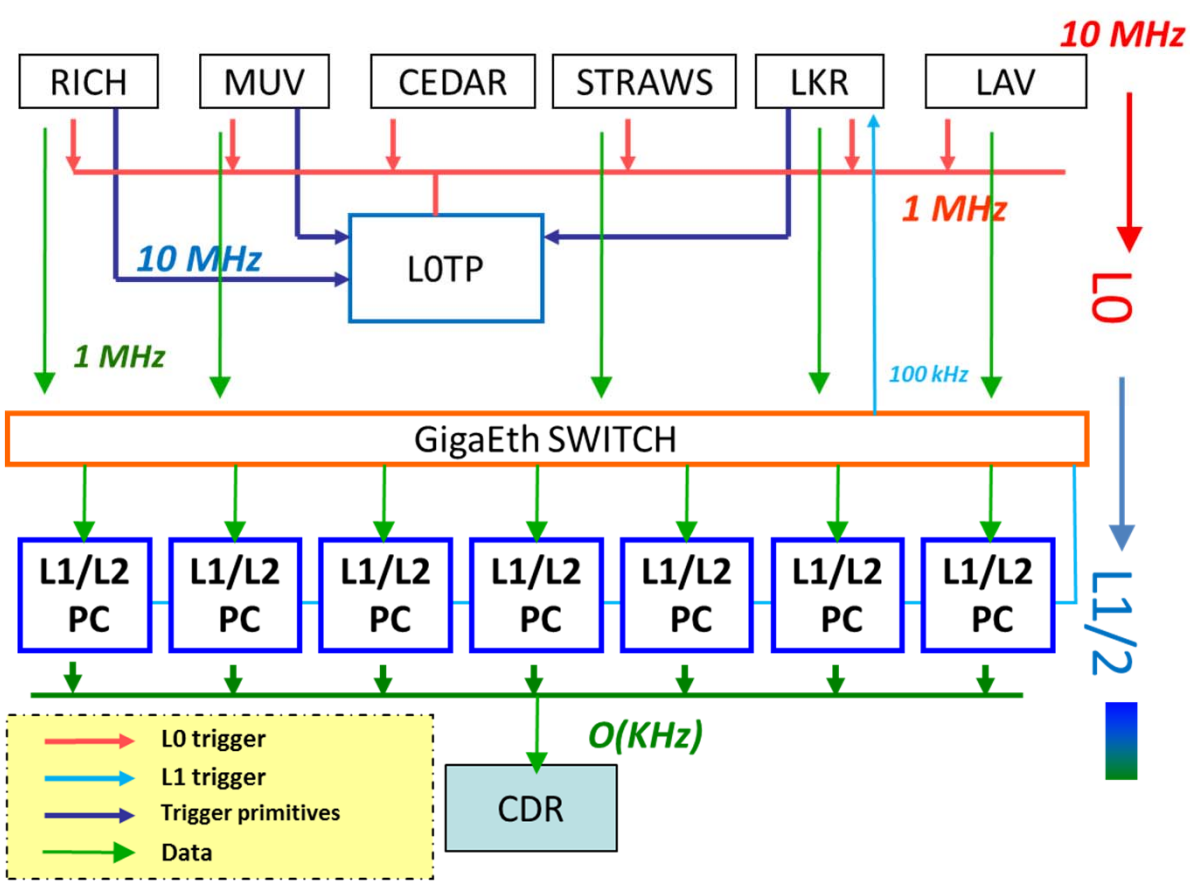
- **Amplitude well predicted in SM (measurement of Vtd) [see E.Stamou]**
- **Residual error in the BR due to parametric uncertainties (mainly due to charm contributions): ~7%**
- **Alternative way to measure the Unitarity Triangle with smaller theoretical uncertainty**

Experimental technique



- Kaon decay in-flight from an unseparated 75 GeV/c hadron beam, produced with 400 GeV/c protons from SPS on a fixed berilium target
- ~800 MHz hadron beam with ~6% kaons
- Goal: measurement of O(100) events in two years of data taking with % level of systematics
- Present result (E787+E949): 7 events, total error of ~65%.

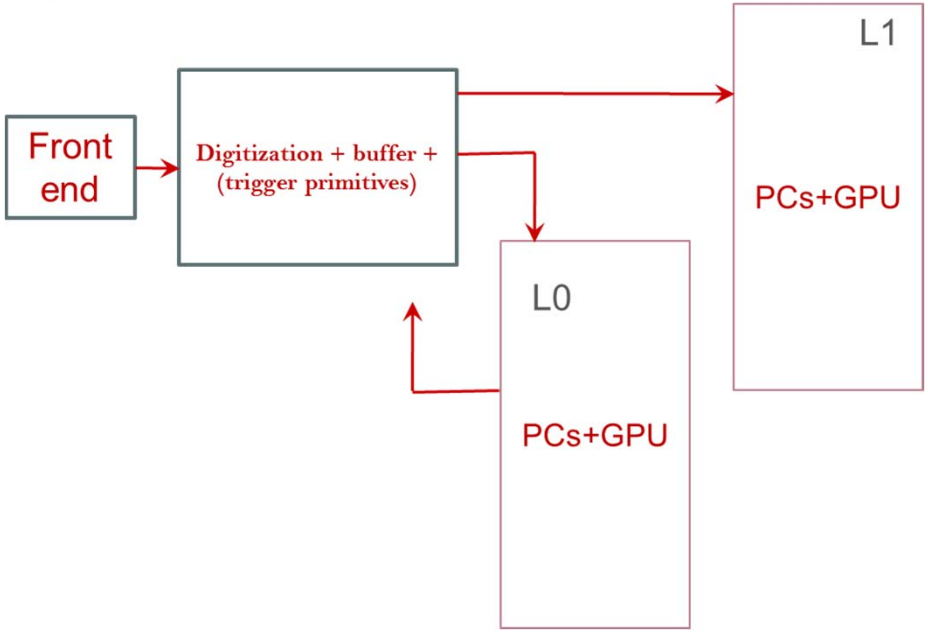
NA62 Trigger



- L0: Hardware synchronous level. 10 MHz to 1 MHz. Max latency 1 ms.
- L1: Software level. “Single detector”. 1 MHz to 100 kHz
- L2: Software level. “Complete information level”. 100 kHz to few kHz.

GPU as Low-Level Trigger

- The idea: exploit GPUs to perform high quality analysis at trigger level
- GPU architecture: massive parallel processor SIMD
- "Easy" at L1/2, challenging at L0
- Real benefits: increase the physics potential of the experiment at very low cost!
- Profit from continuative developments in technology for free (Video Games,…)



Data Flow

Max time O(100us)

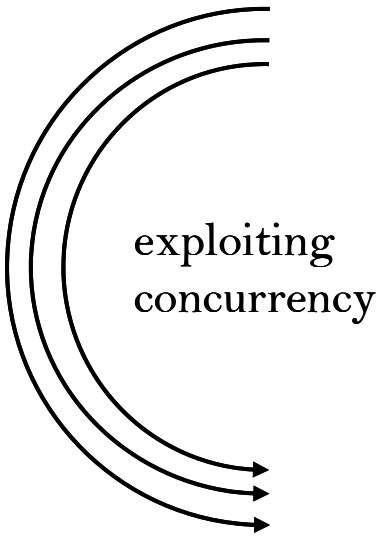
Read data directly from Network Interface buffers

Filling data structures of arrays

Multiple threads transfer this data to GPU Memory on different streams

Multiple threads launch kernels on different streams

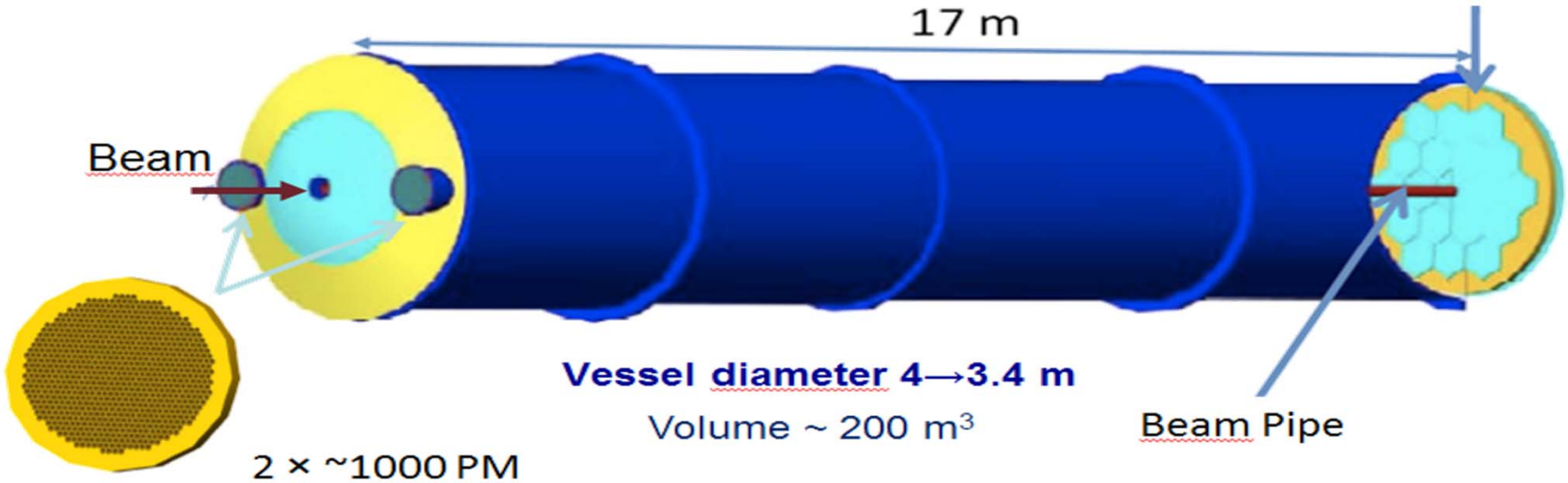
Concurrently transfer the results to the NIC ring buffers and to the frontend electronics



Algorithms

- GPU
- Trigger
- ALICE HLT
- NA62 Low Level Trigger
- Algorithms
- Scheduler
- Tests

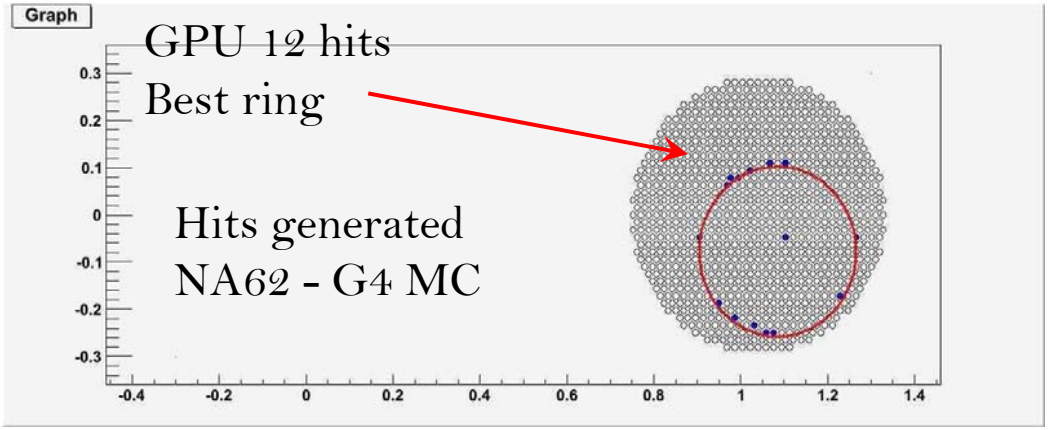
RICH

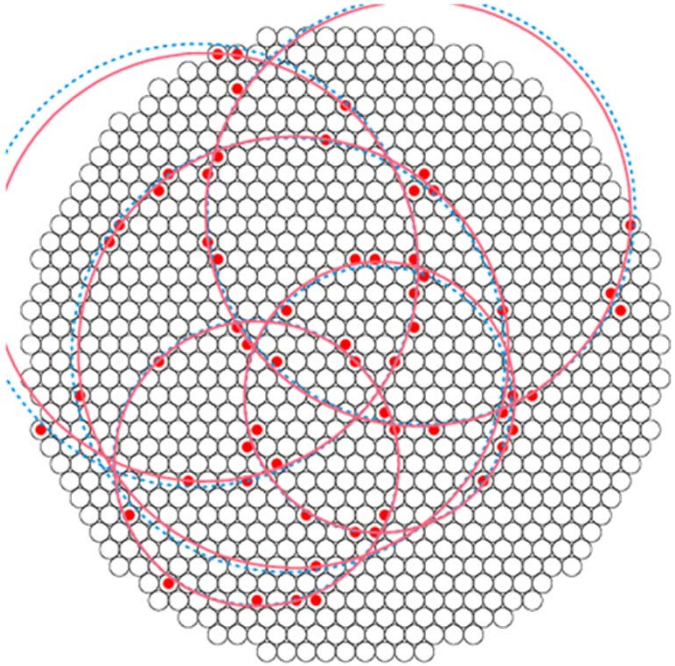


- ~17 m RICH
- 1 atm Neon
- Light focused by two mirrors on two spots equipped with ~1000 PMs each (pixel 18 mm)
- 3s p-m separation in 15-35 GeV/c, ~18 hits per ring in average
- ~100 ps time resolution, ~10 MHz events rate
- Time reference for trigger

Ring Reconstruction

- Natively built for pattern recognition problems
- First attempt: ring reconstruction in RICH detector.





Crawford algorithm

Consider a circle of radius R, centered in (x0, y0)and a list of points (xi, yi).

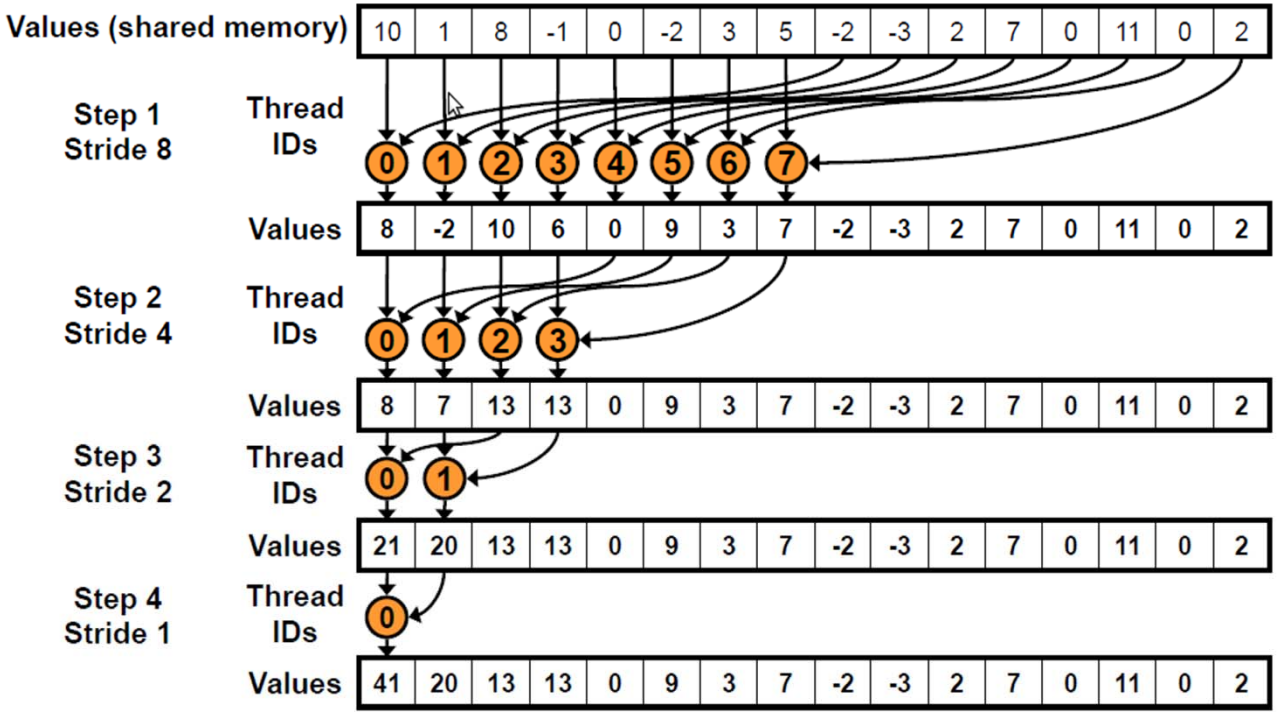
The following relations exist:

$$x_0{}^2 + y_0{}^2 - R^2 = \frac{1}{N}\{2x_0 \sum x_i + 2y_0 \sum y_i - \\ - \sum x_i^2 - \sum y_i^2\}. \qquad (1)$$

$$x_0\{\sum x_i^2 - \frac{(\sum x_i)^2}{N}\} + y_0\{\sum x_i y_i - \\ - \frac{\sum x_i \sum y_i}{N}\} = \frac{1}{2}\{\sum x_i^3 + \sum x_i y_i^2 \\ - \sum x_i \frac{\sum x_i^2 + \sum y_i^2}{N}\}, \qquad (2)$$

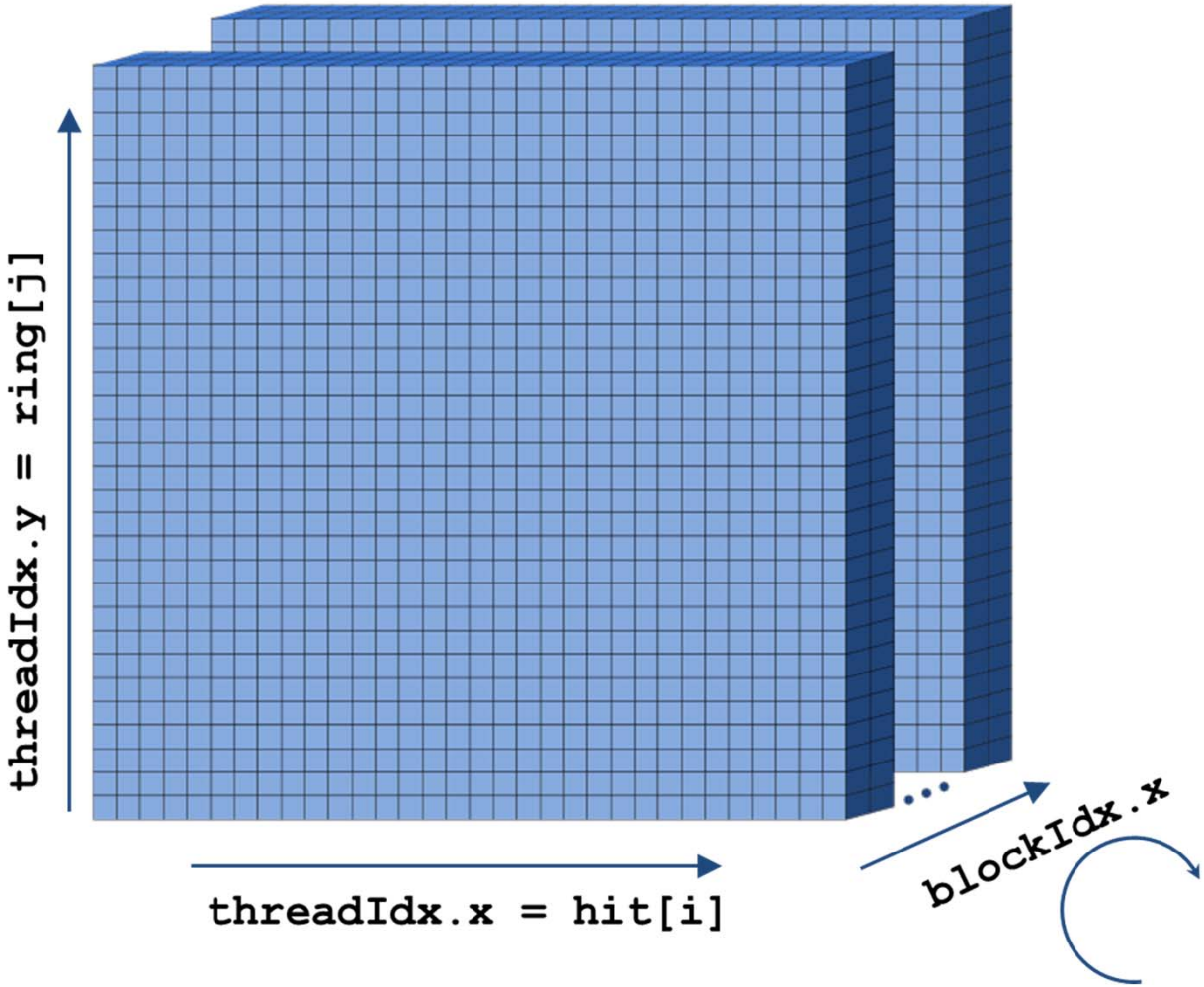
$$x_0\{\sum x_i y_i^2 - \frac{\sum x_i \sum y_i}{N}\} + y_0\{\sum y_i^2 - \\ - \frac{\sum y_i^2}{N}\} = \frac{1}{2}\{\sum x_i^2 y_i + \sum y_i^3 - \\ - \sum y_i \frac{\sum x_i^2 + \sum y_i^2}{N}\}. \qquad (3)$$

Reduction



- One reduction kernel is called per block, giving an array of results (one for each event)
- Must use sequential addressing instead of interleaved addressing to avoid Shared Memory bank conflicts
- Time complexity is O(logN), cost is O(N*logN): not cost efficient
- Brent's theorem (algorithm cascading) suggests O(N/logN) threads:
  - Each thread does O(logN) sequential work
  - All O(N/logN) threads cooperate for O(logN) steps
  - New cost = O(N/logN * logN) = O(N)

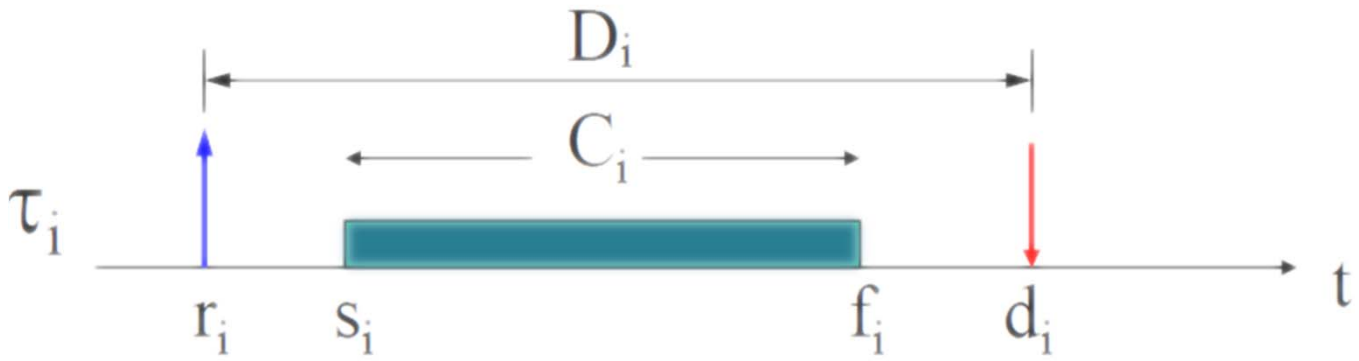
GPU Grid organization



Scheduler

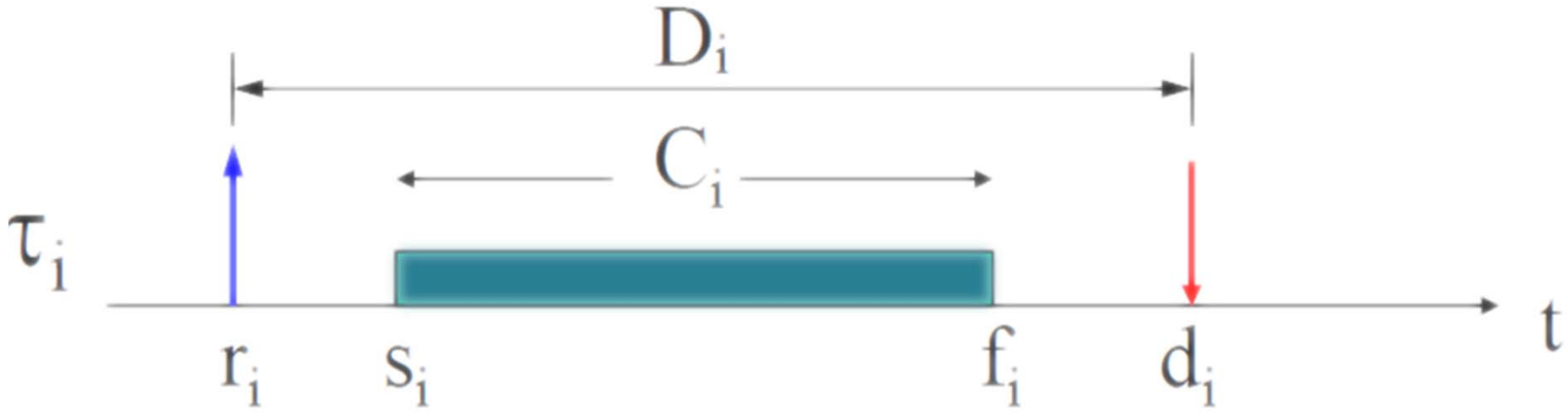
- GPU
- Trigger
- ALICE HLT
- NA62 Low Level Trigger
- Algorithms
- Scheduler
- Tests

Task Parameters



$\mathbf{r_i}$	release time (arrival time $\mathbf{a_i}$)
$\mathbf{s_i}$	start time
$\mathbf{C_i}$	worst-case execution time (wcet)
$\mathbf{d_i}$	absolute deadline
$\mathbf{D_i}$	relative deadline
$\mathbf{f_i}$	finishing time

Lateness



- **The scheduler must be aware of the lateness defined as Li = fi - di**
- **The more precise the time synchronization with the detector, the more precise the lateness.**

Tasks for a GPU trigger

Tasks:

- Receive from the Network Interface
- Wait for a good number of TPs (Trigger Primitives) to sustain the throughput (d1)
- Send to the GPU the whole GMTP (GPU MultiTP)
- Compute the whole GMTP on the GPU
- Copy the results back to the host memory
- Send the results through the NIC (d2)

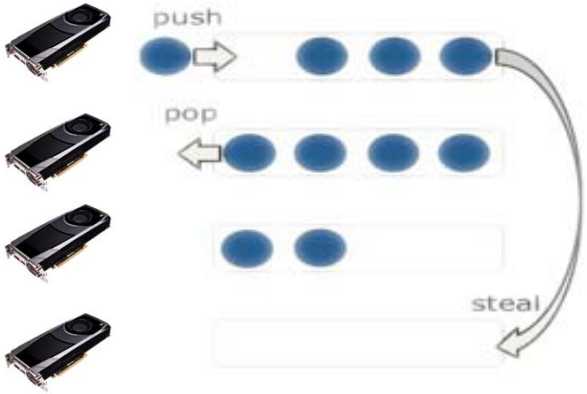
Task Execution

- Each of these tasks run concurrently (for the CPU part) and instruction-level parallelism is exploited to hide latency.
- The scheduling algorithm is partially preemptive (each task always run to completion, but the whole job could be suspended).

Work stealing could be implemented

in the case of multi-GPU setup to

improve the load balancing.



Tests

- GPU
- Trigger
- ALICE HLT
- NA62 Low Level Trigger
- Algorithms
- Scheduler
- Tests

HW configuration (1/2)

- **First Machine**
  - GPU: NVIDIA Tesla C2050
    - 448 CUDA cores @ 1.15GHz
    - 3GB GDDR5 ECC @ 1.5GHz
    - CUDA CC 2.0 (Fermi Architecture)
    - PCIe 2.0 (effective bandwidth up to ~5GB/s)
    - CUDA 5

  - CPU: Intel® Xeon® Processor E5630 (released in Q1'10)
  - 2 CPUs, 8 physical cores (16 HW-threads)

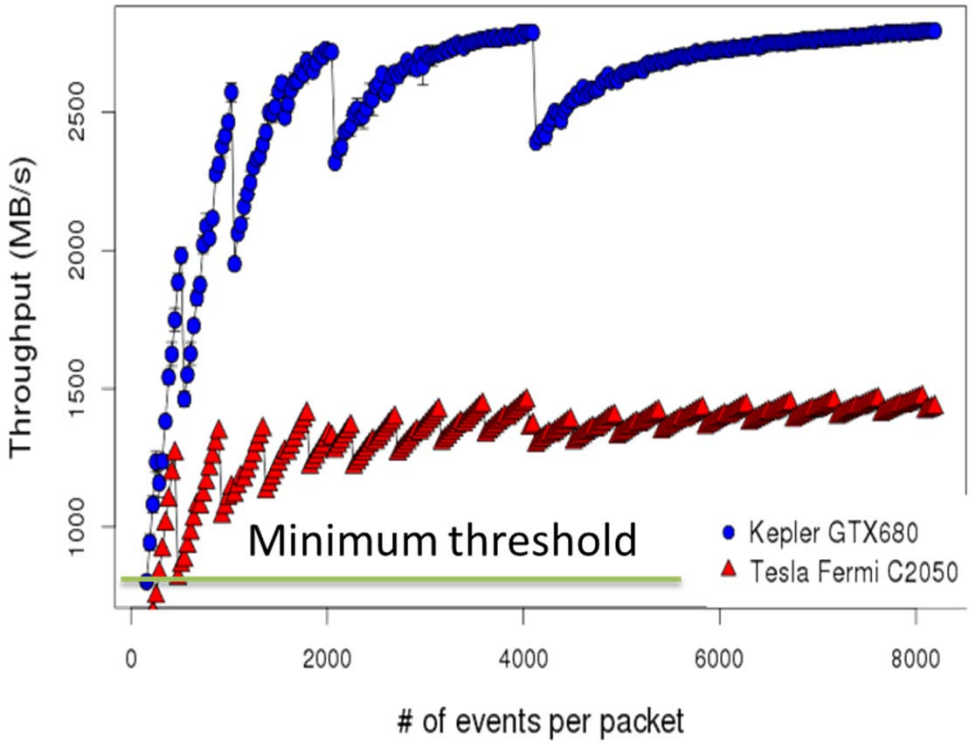
HW configuration (2/2)

- **Second Machine**
  - GPU: NVIDIA GTX680
    - 1536 CUDA cores @ 1.01GHz
    - 2GB GDDR5 ECC @ 1.5GHz
    - CUDA CC 3.0 (Kepler Architecture)
    - PCIe 3.0 (effective bandwidth up to ~11GB/s)
    - CUDA 5
  - CPU: Intel® Ivy Bridge Processor i7-3770 (released in Q2 '12)
  - 1 CPUs, 4 physical cores (8 hw-threads) @3.4GHz

Throughput

The throughput behaviour for a varying number of events inside a packet is a typical many-core device behaviour:

- constant time to process a varying number of events, activating more SMs as the packet size increases
- discrete oscillations due to the discrete nature of the GPU
- saturation plateau (1.4GB/s and 2.7GB/s)

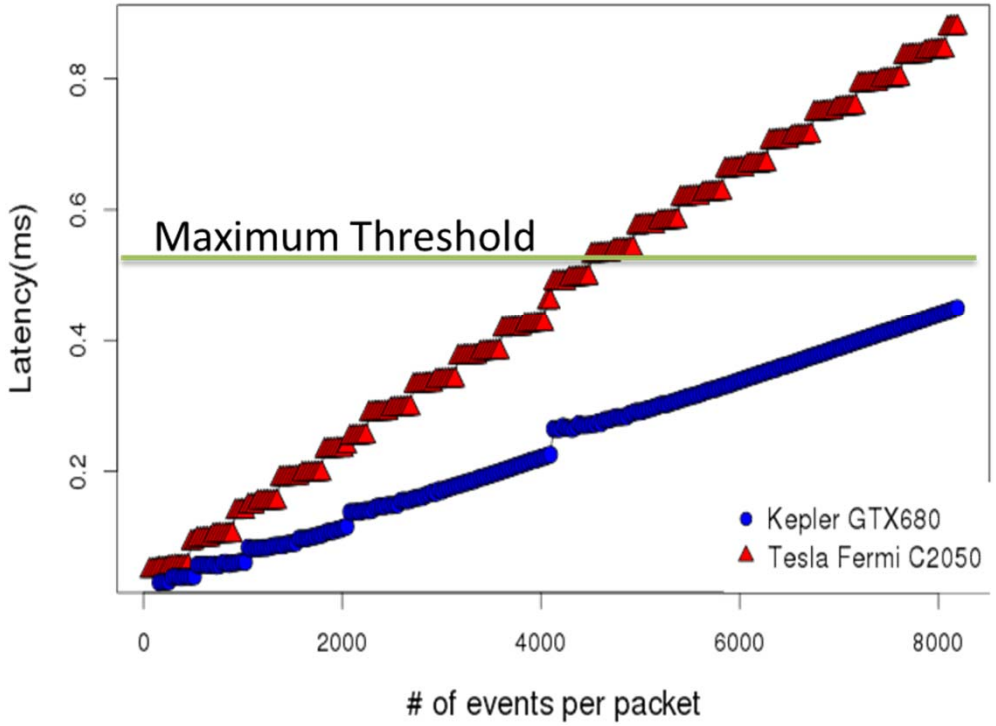


Latency

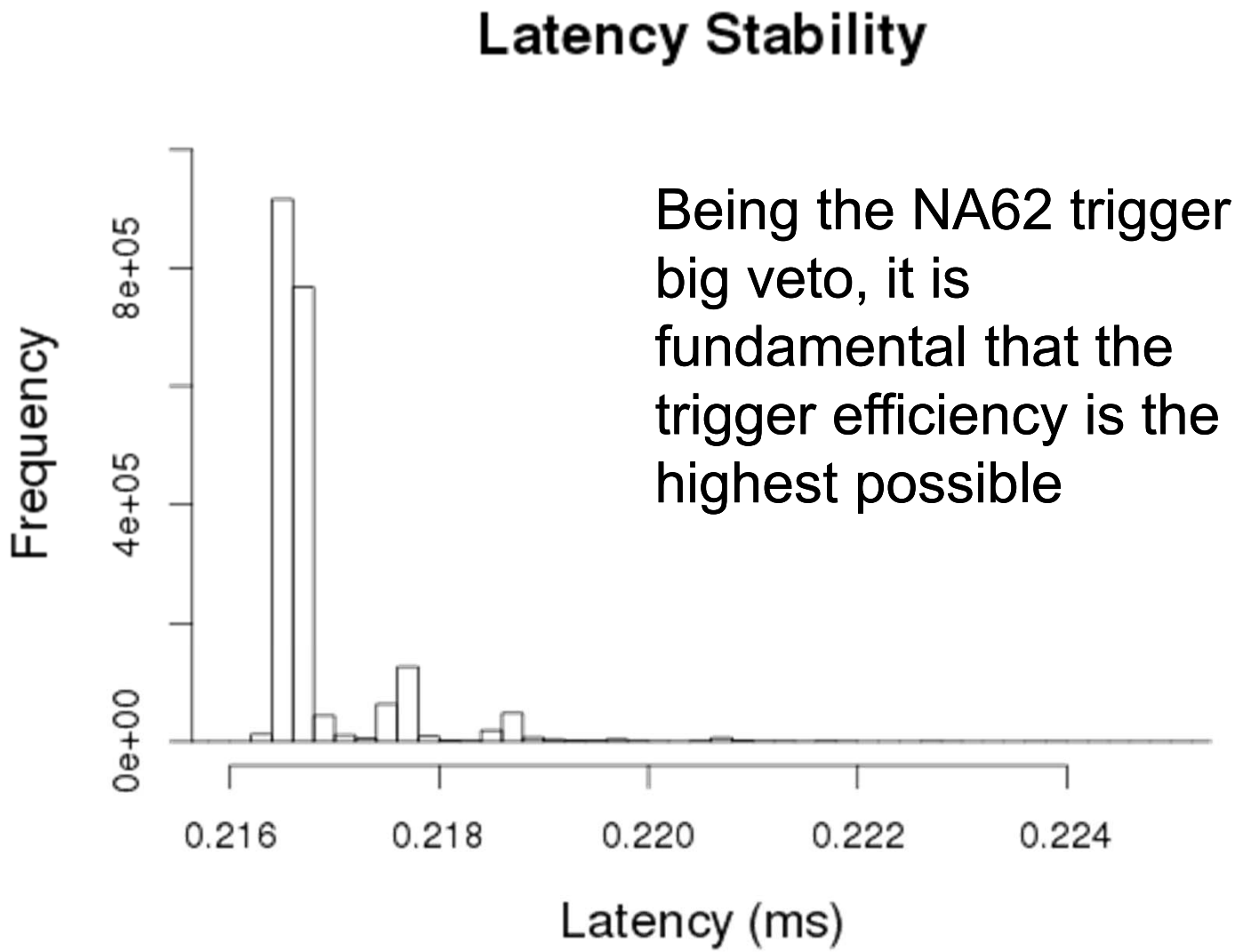
Latency pretty stable wrt event size.

- A lower number of event inside a package is better to achieve a low latency.
- A larger number of event guarantees a better performance and a lower overhead.

The choice of the packet size depends on the technical requirements.



Latency Stability



Being the NA62 trigger a big veto, it is fundamental that the trigger efficiency is the highest possible

Memory Hierarchy - LUT

Investigation on which memory to use to store a LUT:

Global memory (read and write)

- **Slow, but now with cache**
- **L1 cache designed for spatial re-usage, not temporal (similar to coalescing)**
- **It benefits if compiler detects that all threads load same value (LDU PTX ASM instruction, load uniform)**

Texture memory

- **Cache optimized for 2D spatial access pattern**

Constant memory

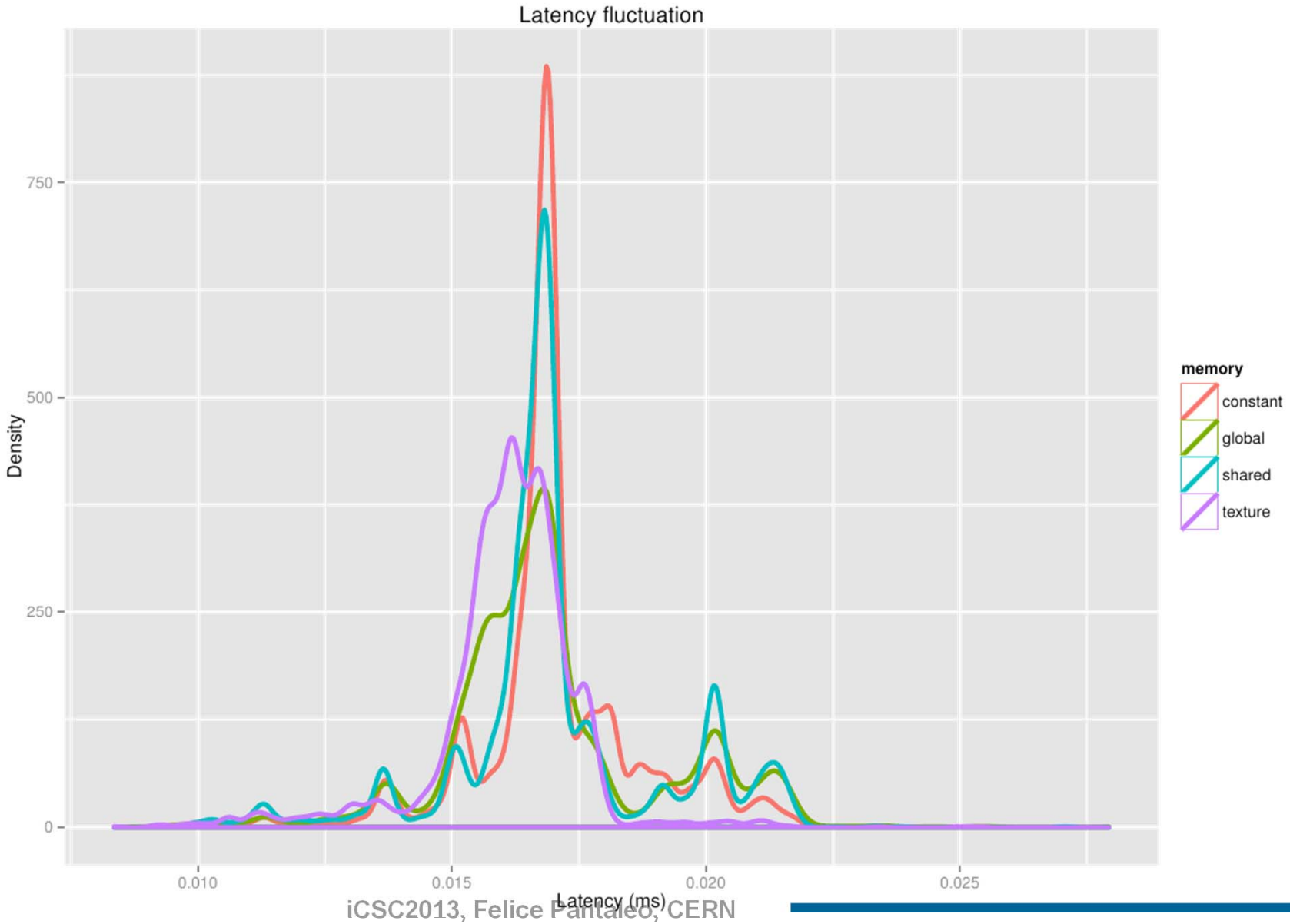
- **Slow, but with cache (8 kb)**

Shared memory (48kB per SMX)

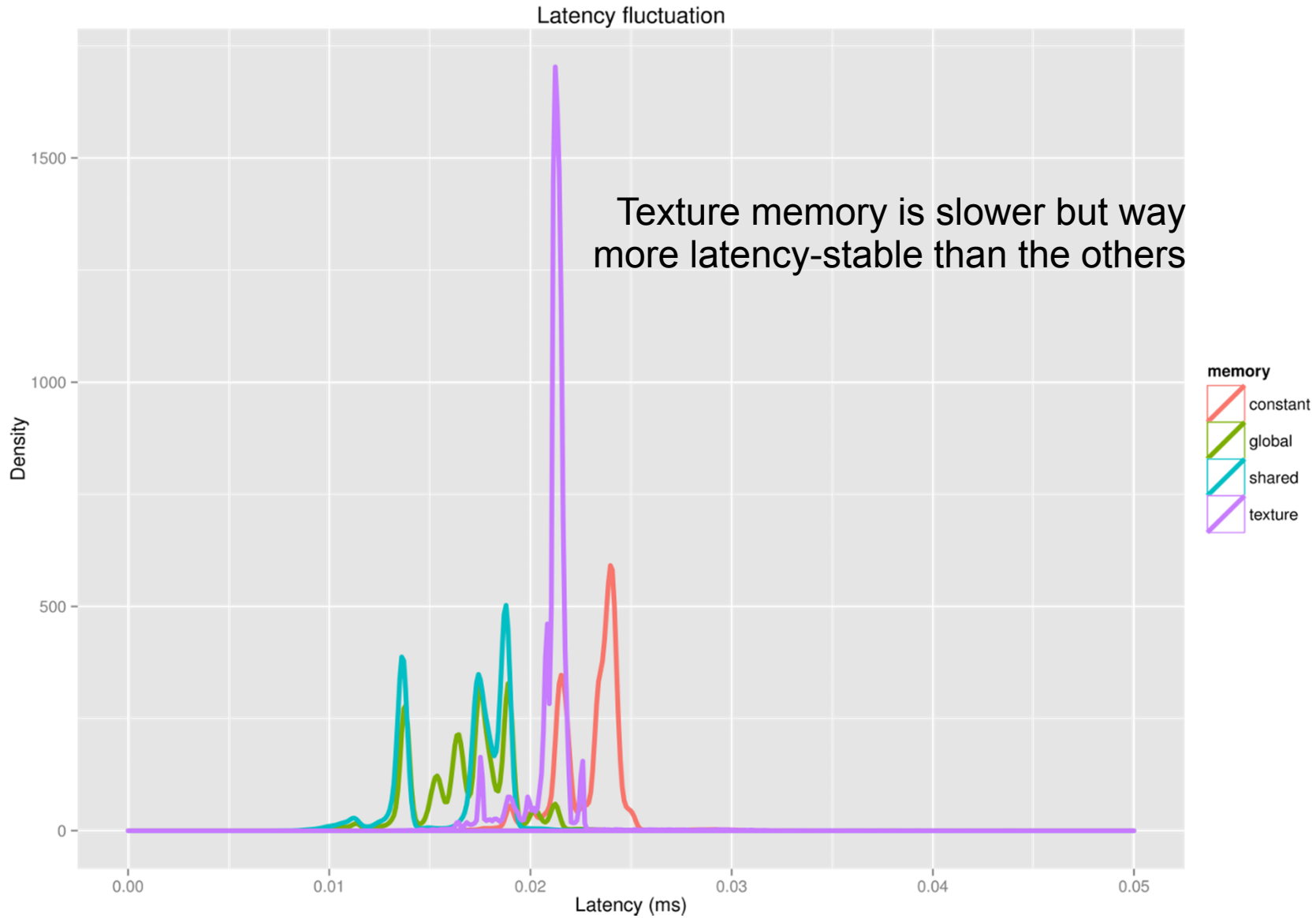
- **Fast, but slightly different rules for bank conflicts now**

Registers (65536 32-bit registers per SMX)

LUT 8x8

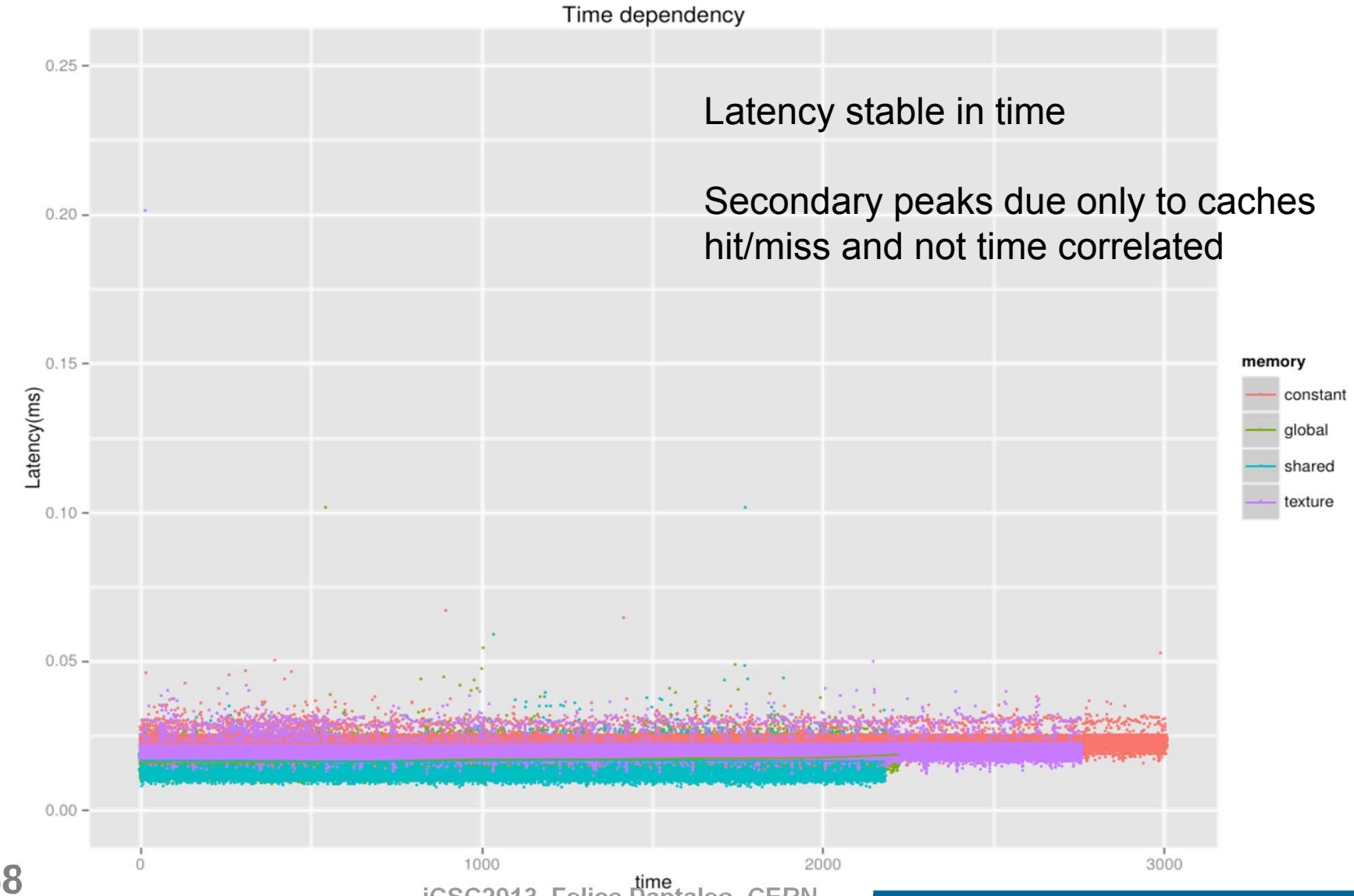

Latency fluctuation
Density
750
500
250
0
0.010
0.015
0.020
0.025
Latency (ms)
memory
constant
global
shared
texture

LUT 64x64



Texture memory is slower but way more latency-stable than the others

Time Dependency



Latency stable in time

Secondary peaks due only to caches hit/miss and not time correlated

Conclusions

- **GPUs seem to represent a good opportunity, not only for analysis and simulation applications, but also for more “hardware” jobs.**
- **Replacing custom electronics with fully programmable processors to provide the maximum possible flexibility is a reality not so far in the future.**

Questions?