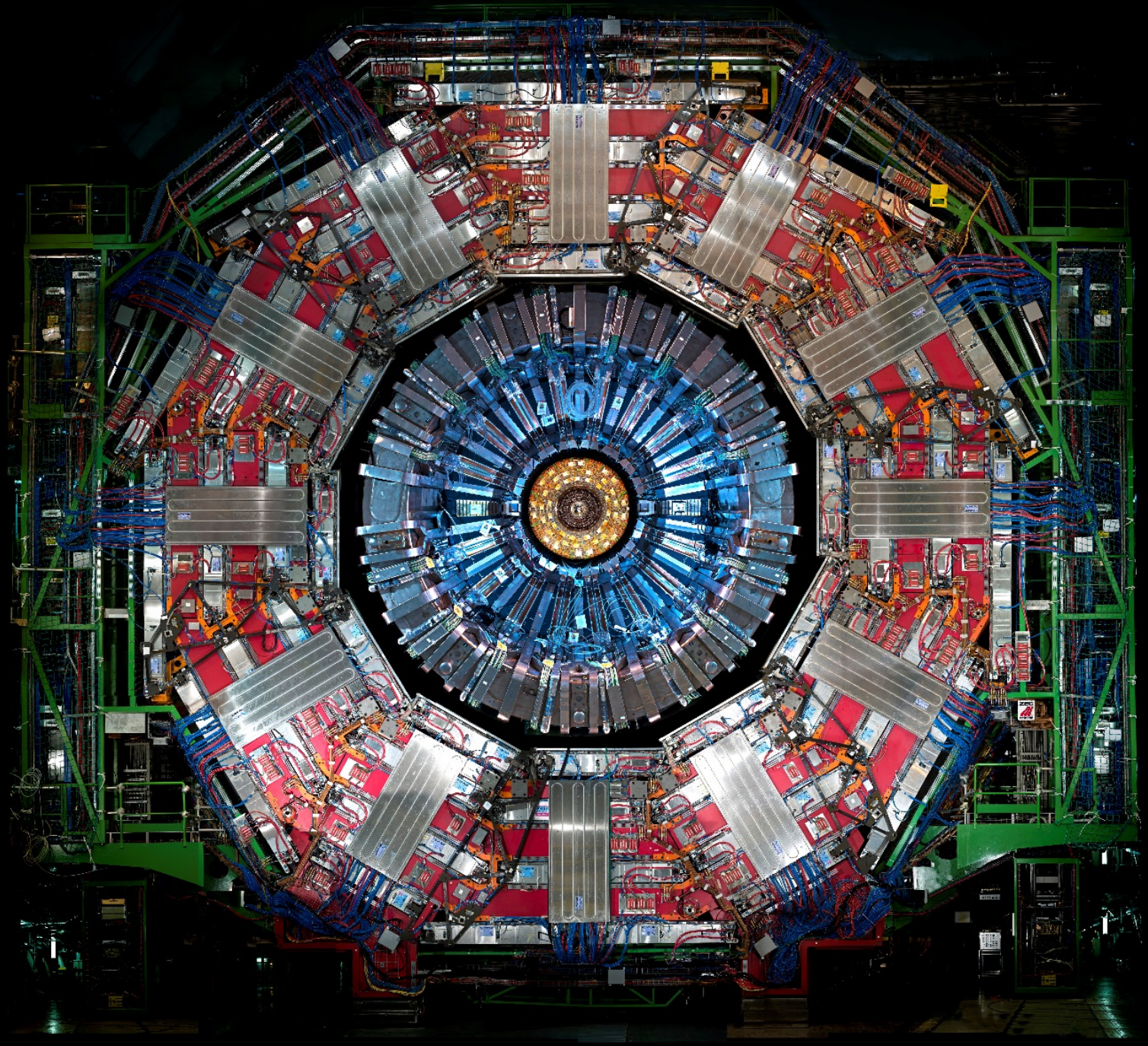
Heterogeneous Event Selection at the CMS experiment at CERN

Felice Pantaleo

CERN, Experimental Physics Department

felice@cern.ch

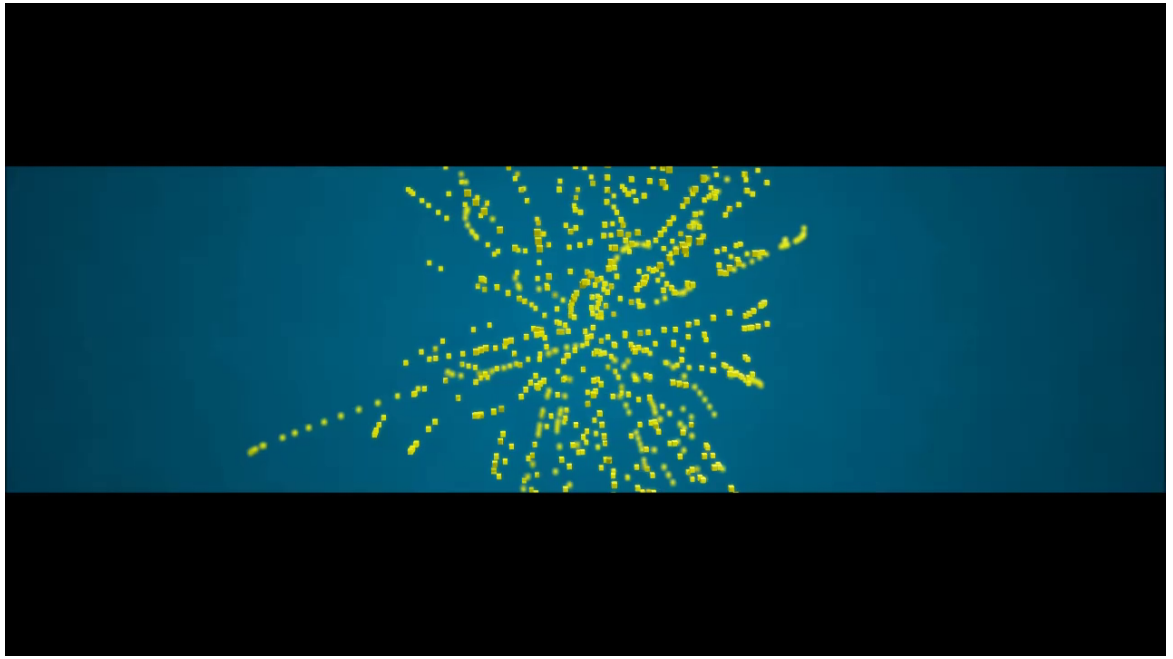


Tracking

- Particles produced in the collisions leave traces (hits) as they fly through the detector
- The innermost detector of CMS is called **Tracker**
- **Tracking**: the art of associate each hit to the particle that left it
- In a solenoidal magnetic field, trajectories are helices

Tracking

- Particles produced in the collisions leave traces (hits) as they fly through the detector
- The innermost detector of CMS is called **Silicon Tracker**
- **Tracking**: the art of associate each hit to the particle that left it
- In a solenoidal magnetic field, trajectories are helices



Two-stages event selection strategy

Trigger System

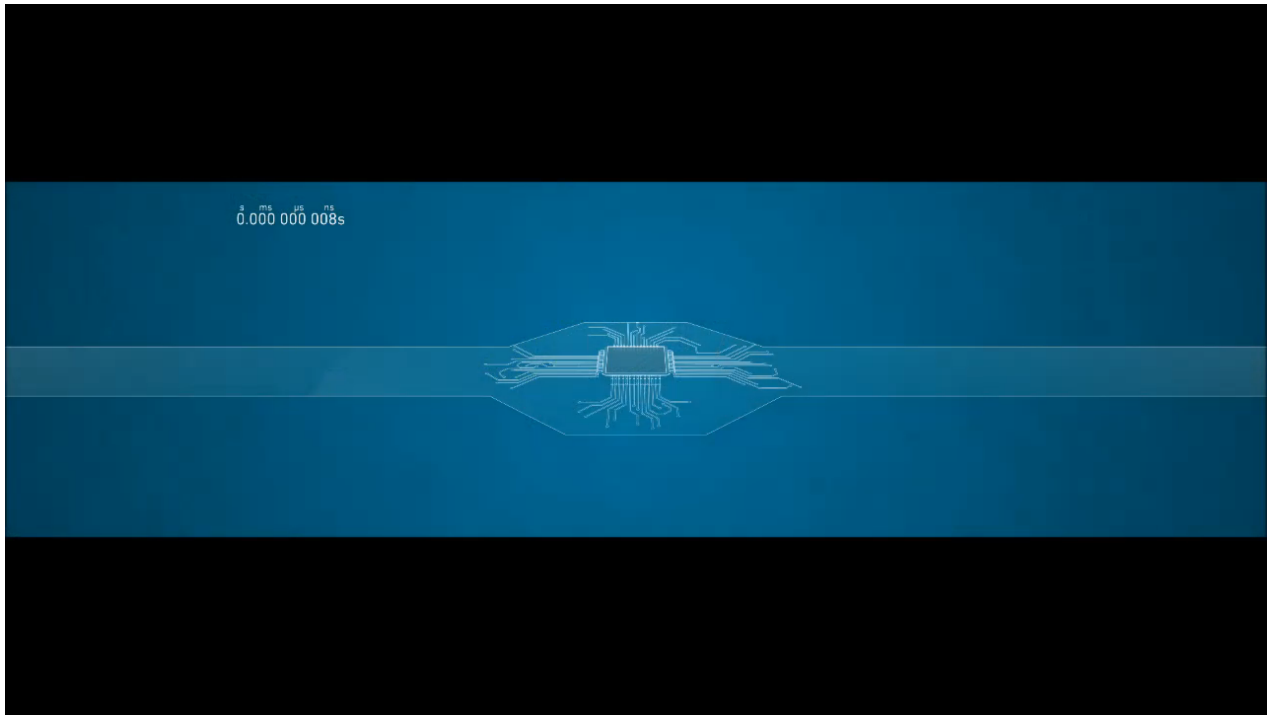
- Reduce input rate (**40 MHz**) to a data rate (**~1 kHz**) that can be stored, reconstructed and analyzed Offline maximizing the physics reach of the experiment

Level 1 Trigger

- coarse readout of the Calorimeters and Muon detectors
- implemented in custom electronics, ASICs and FPGAs
- output rate limited to **100 kHz** by the readout electronics

High Level Trigger

- readout of the whole detector with full granularity
- based on the CMSSW software, running on 22,000 Xeon cores
- organized in O(2500) modules, O(400) trigger paths, O(10) streams
- output rate limited to an average of **~1 kHz** by the Offline resources



Two-stages event selection strategy

Trigger System

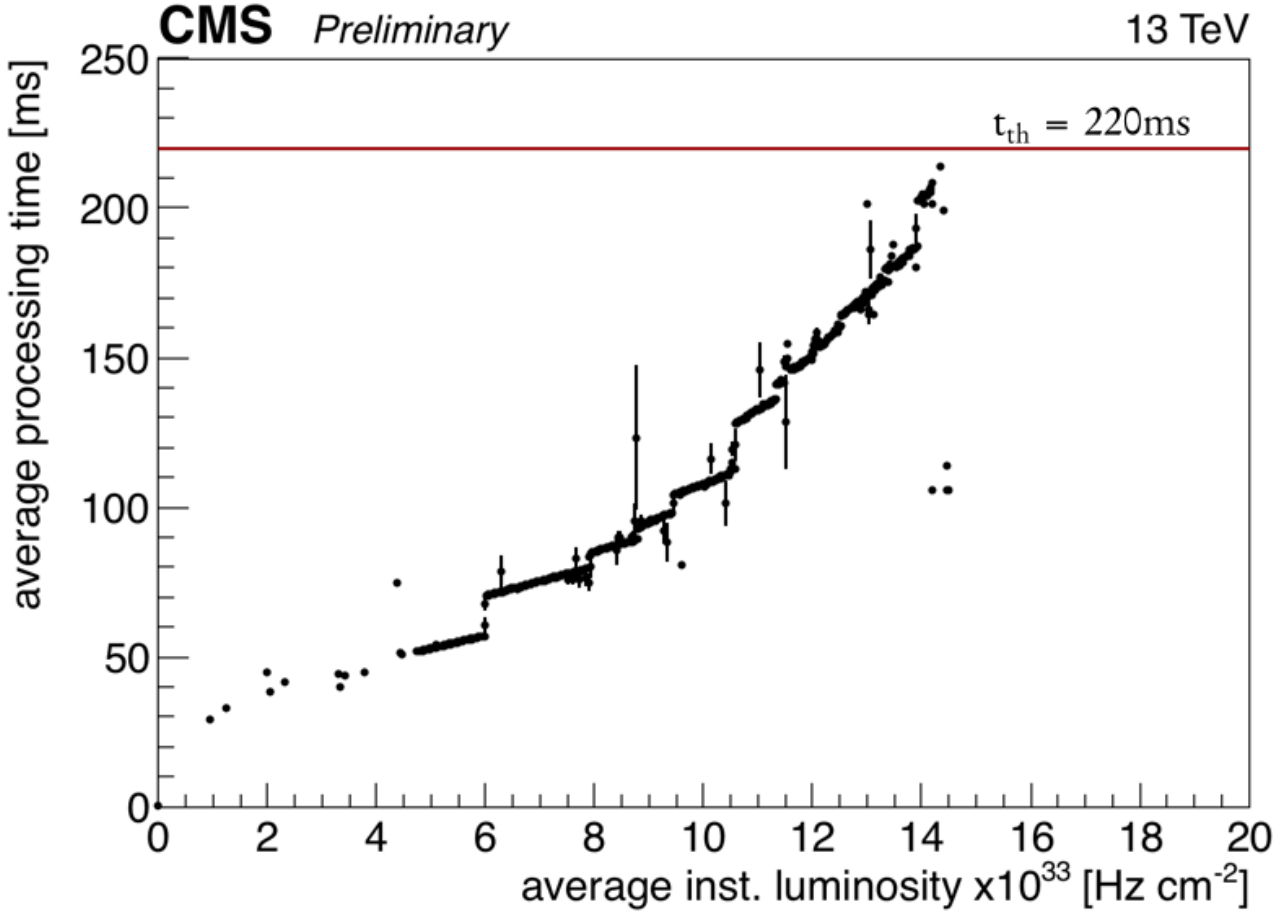
- Reduce input rate (**40 MHz**) to a data rate (**~1 kHz**) that can be stored, reconstructed and analyzed Offline maximizing the physics reach of the experiment

Level 1 Trigger

- coarse readout of the Calorimeters and Muon detectors
- implemented in custom electronics, ASICs and FPGAs
- output rate limited to **100 kHz** by the readout electronics

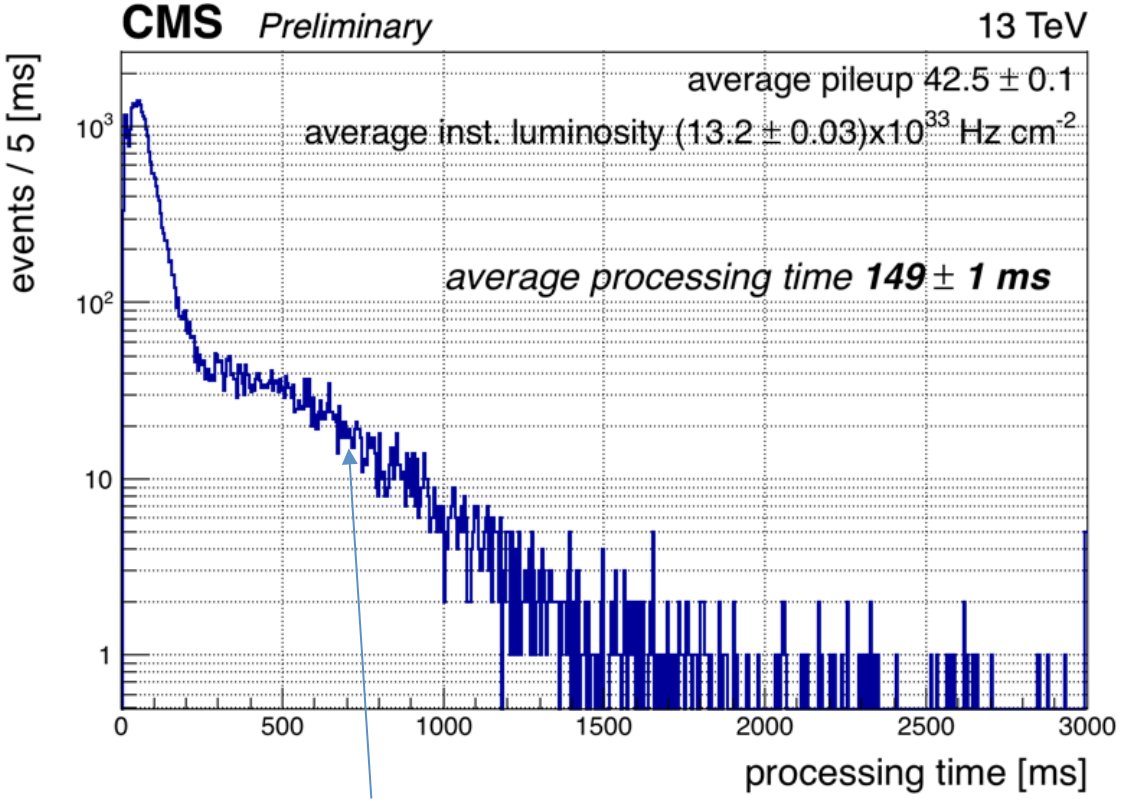
High Level Trigger

- readout of the whole detector with full granularity
- based on the CMSSW software, running on 22,000 Xeon cores
- organized in O(2500) modules, O(400) trigger paths, O(10) streams
- output rate limited to an average of **~1 kHz** by the Offline resources



Tracking at the CMS High-Level Trigger

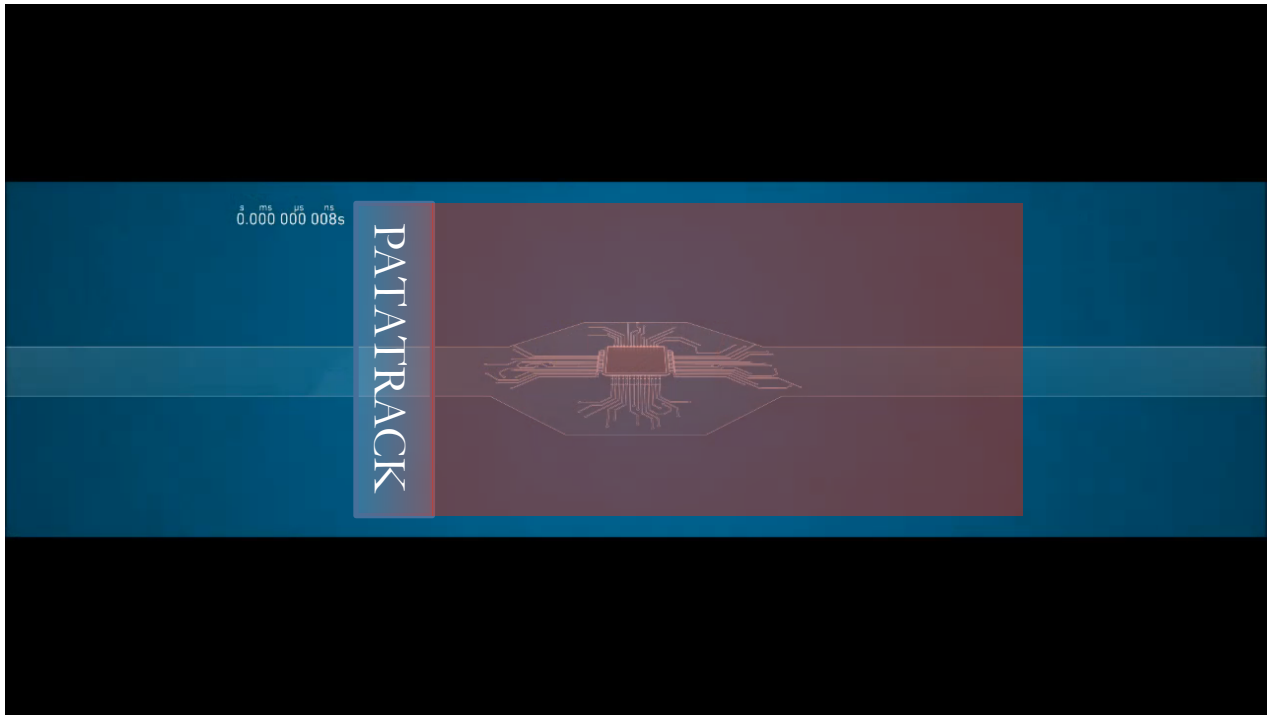
- Today the CMS online farm consists of ~22k Intel Xeon cores
  - The current approach: one event per logical core
- Pixel Tracks cannot be reconstructed for all the events at the HLT
- This will be even more difficult at higher pile-up
  - Combinatorial time in pixel seeding O(pileup!) in worst case
  - More memory/event



full track reconstruction

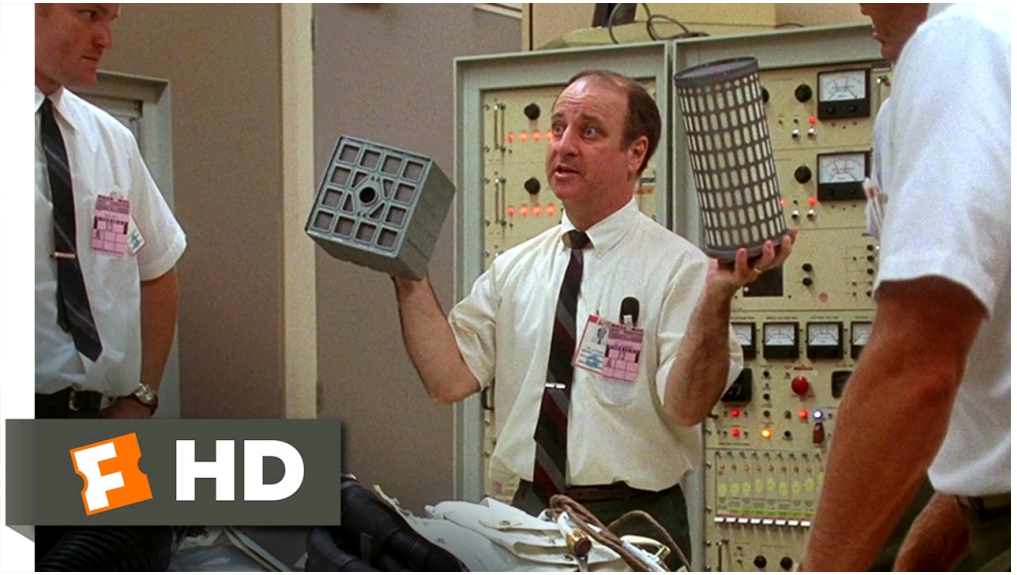
PATATRACK: From RAW data to Tracks

- Objective:
  - A hybrid CPU-GPU application that takes RAW data coming from the pixel detector and gives Tracks as result
- Trigger avg latency should stay within 220ms
- GPU memory transfers are hidden exploiting instruction level parallelism (function executing while non related transfer happens)
  - increased throughput
  - no impact on total latency
- Ingredients:
  - Massive parallelism within the event
  - Avoid useless data transfers and transformations
  - Simple data formats optimized for parallel memory access
  - Renovation at algorithmic level



Discriminants

- PATATRACK should:
  - Increase the throughput of the HLT farm
  - Substantially decrease the average trigger response time of the HLT farm while keeping the rate of input tracking events constant
  - or Substantially increase the rate of input tracking events while keeping the average trigger response time of the HLT constant
  - Cost less (initial cost + energy consumption)
  - Occupy the same volume

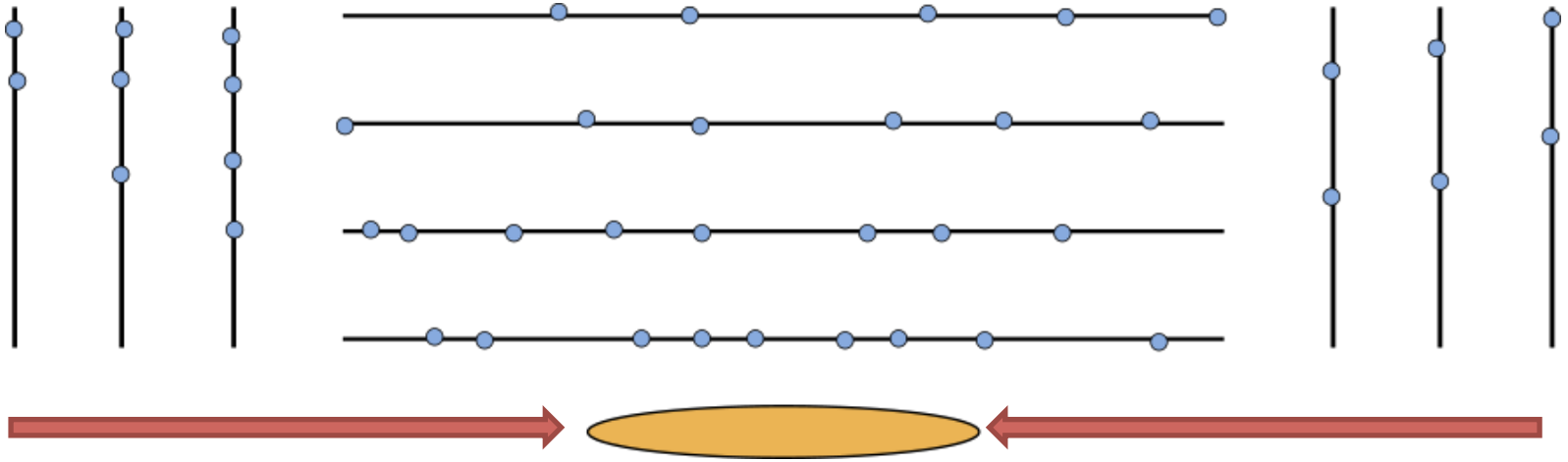


Credits: Apollo 13: Square peg in a round hole


μ = 500 GeV/c²
H,A → ττ → two τ jets + X, 60 fb⁻¹

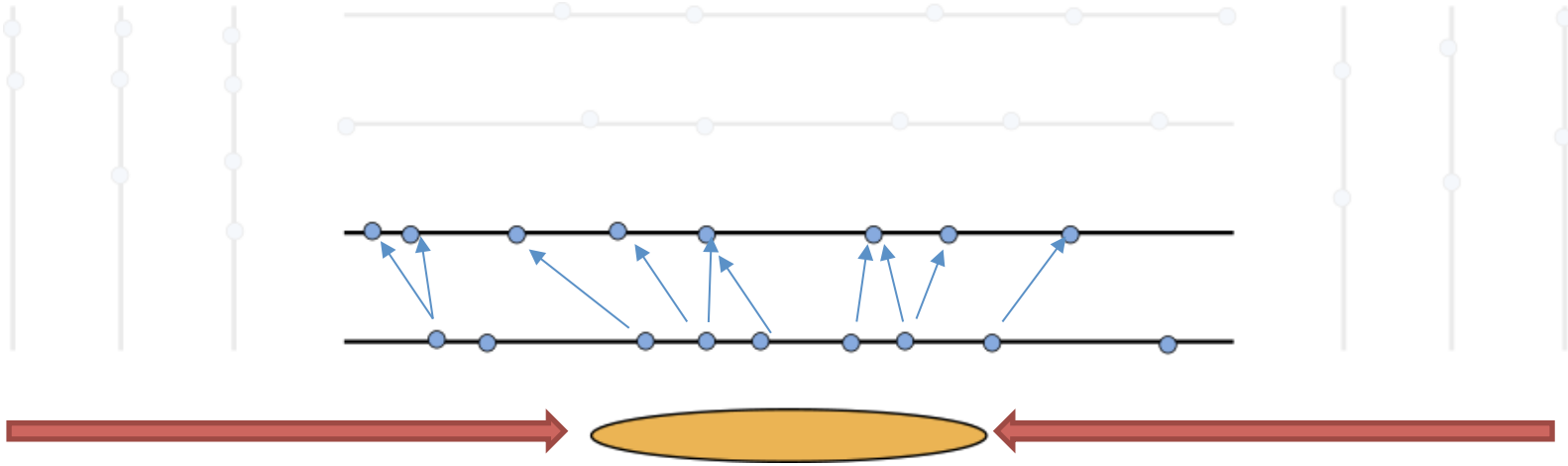
RMS HEP Algorithm

- Hits on different layers
- Need to match them and create quadruplets
- Create a modular pattern and reapply it iteratively



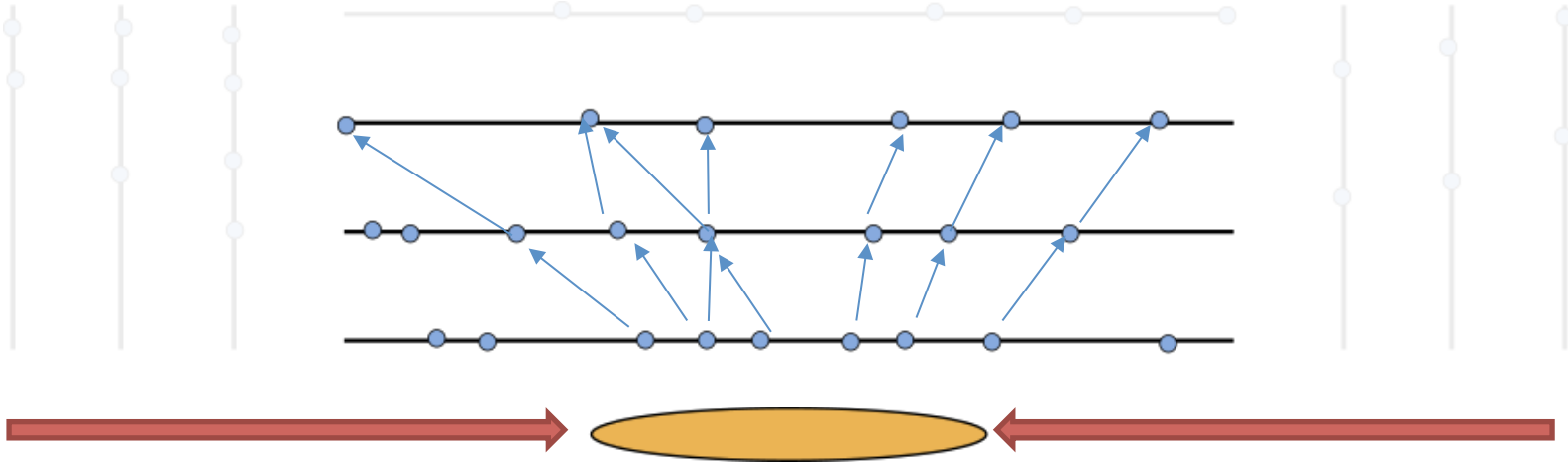
RMS HEP Algorithm

- First create doublets from hits of pairs



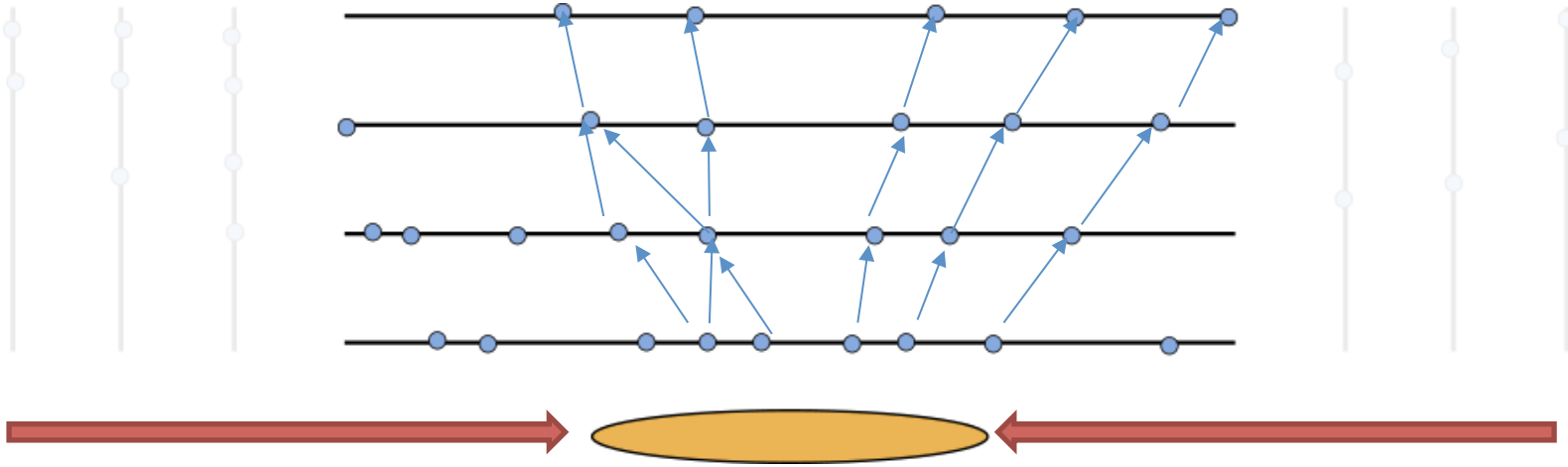
RMS HEP Algorithm

- First create doublets from hits of pairs
- Take a third layer and propagate only the generated doublets



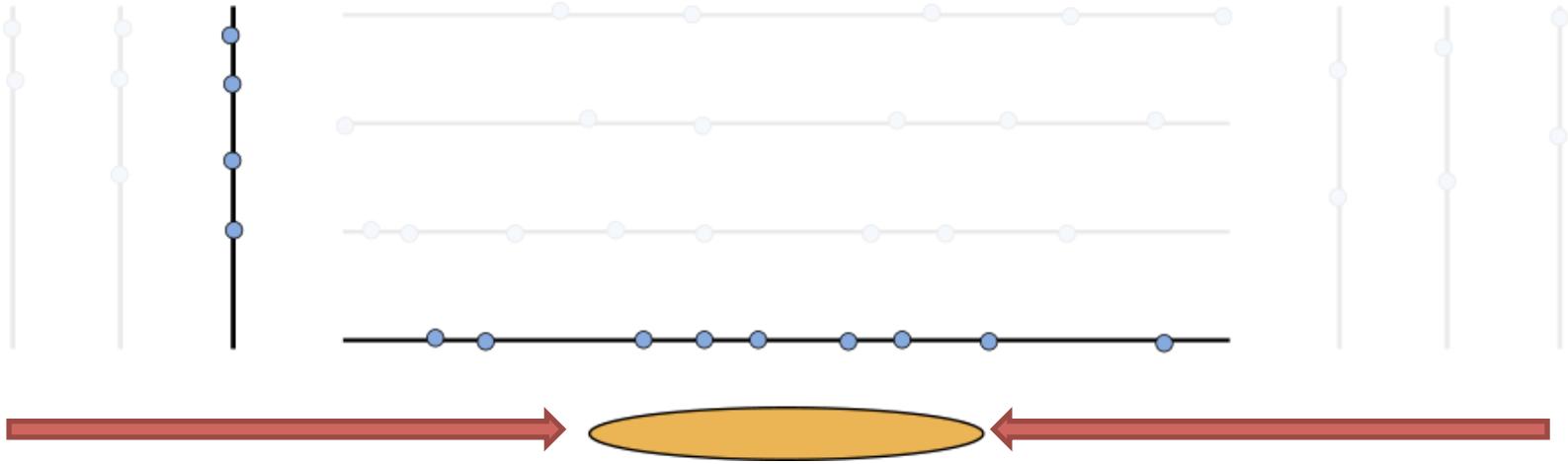
RMS HEP Algorithm

- First create doublets from hits of pairs
- Take a third layer and propagate only the generated doublets
- Consider a fourth layer and propagate triplets
- Store found quadruplets and start from another pair of layers



RMS HEP Algorithm

- First create doublets from hits of pairs
- Take a third layer and propagate only the generated doublets
- Consider a fourth layer and propagate triplets
- Store found quadruplets and start from another pair of layers
- Repeat until happy…
- Does this fit the idea of massively parallel computation? I don't really think so…



RMS HEP Algorithm

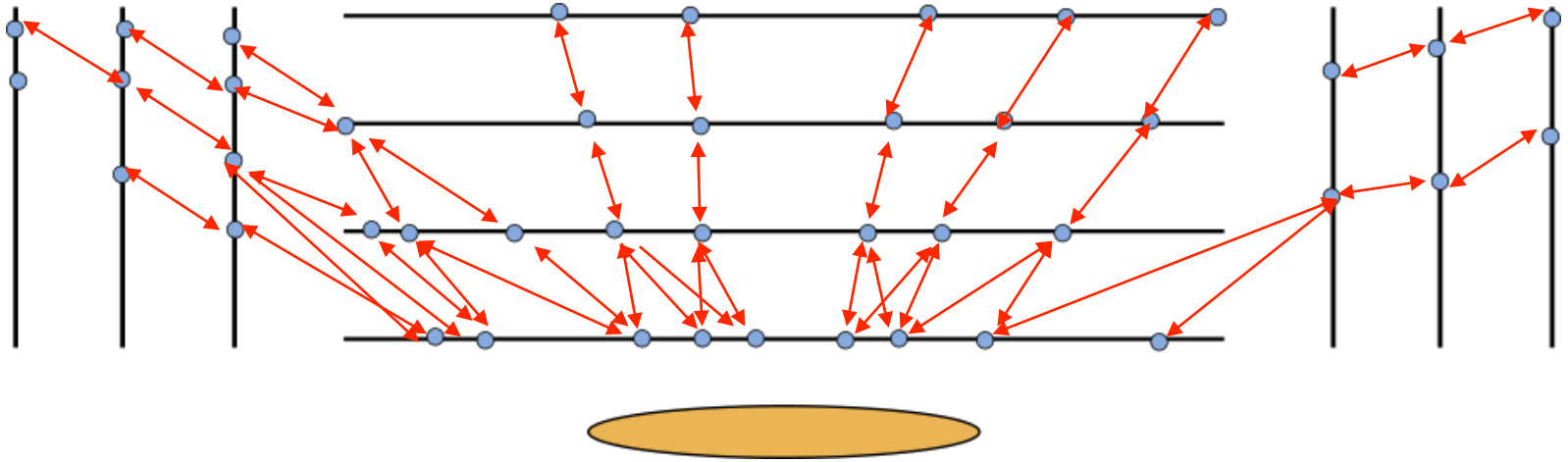
This kind of algorithm is not very suitable for GPUs:

- Absence of massive parallelism
- Poor data locality
- Synchronizations due to iterative process
- Very Sparse and dynamic problem (that's the hardest part, still unsolved)
- Parallelization does not mean making a sequential algorithm run in parallel
  - It requires a deep understanding of the problem, renovation at algorithmic level, understanding of the computation and dependencies

The algorithm was redesigned from scratch getting inspiration from Conway's Game of Life

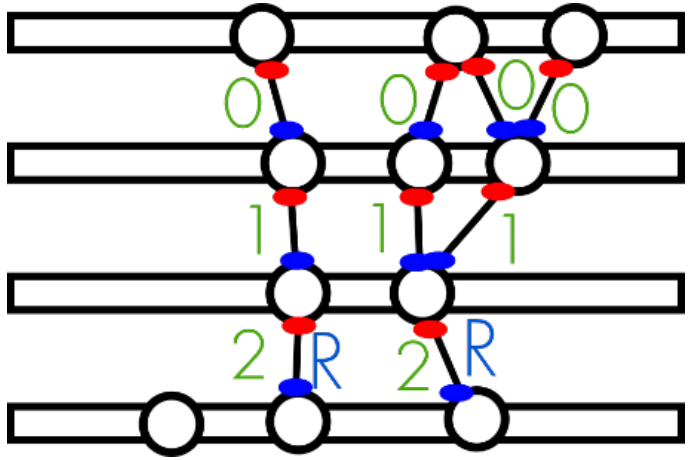
Cellular Automaton (CA)

- The CA is a track seeding algorithm designed for parallel architectures
- It requires a list of layers and their pairings
  - A graph of all the possible connections between layers is created
  - Doublets aka Cells are created for each pair of layers (compatible with a region hypothesis)
  - Fast computation of the compatibility between two connected cells
  - No knowledge of the world outside adjacent neighboring cells required, making it easy to parallelize
- However this is not a static problem, not at all…

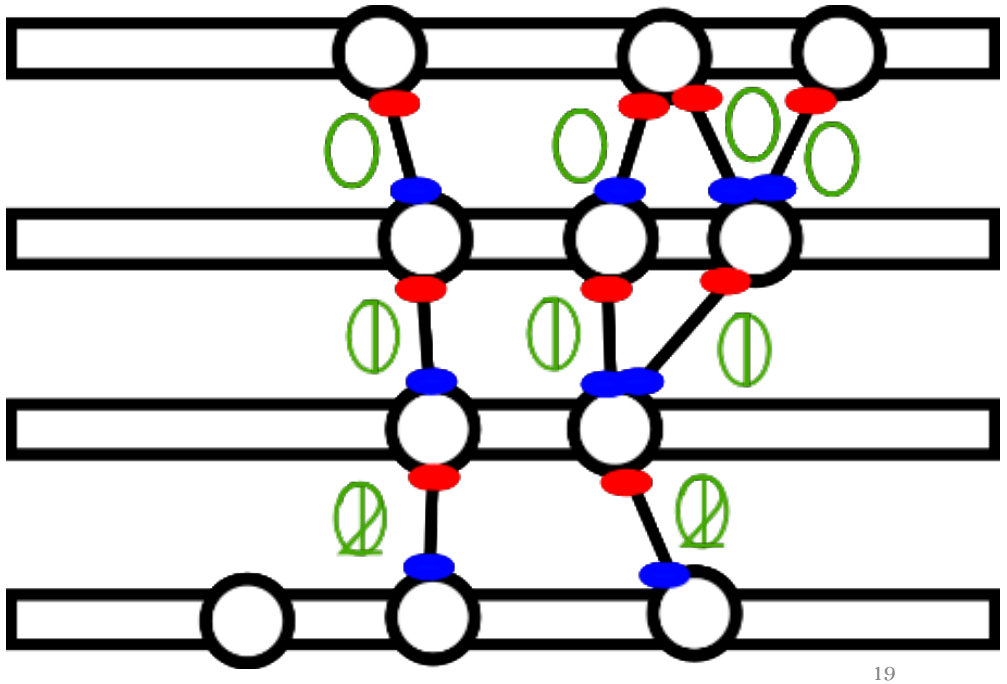


Evolution

- If two cells satisfy all the compatibility requirements they are said to be neighbors and their state is set to 0
- In the evolution stage, their state increases in discrete generations if there is an outer neighbor with the same state
- At the end of the evolution stage the state of the cells will contain the information about the length
- If one is interested in quadruplets, there will be surely one starting from a state 2 cell, pentuplets state 3, etc.



$T=\emptyset$



Tests

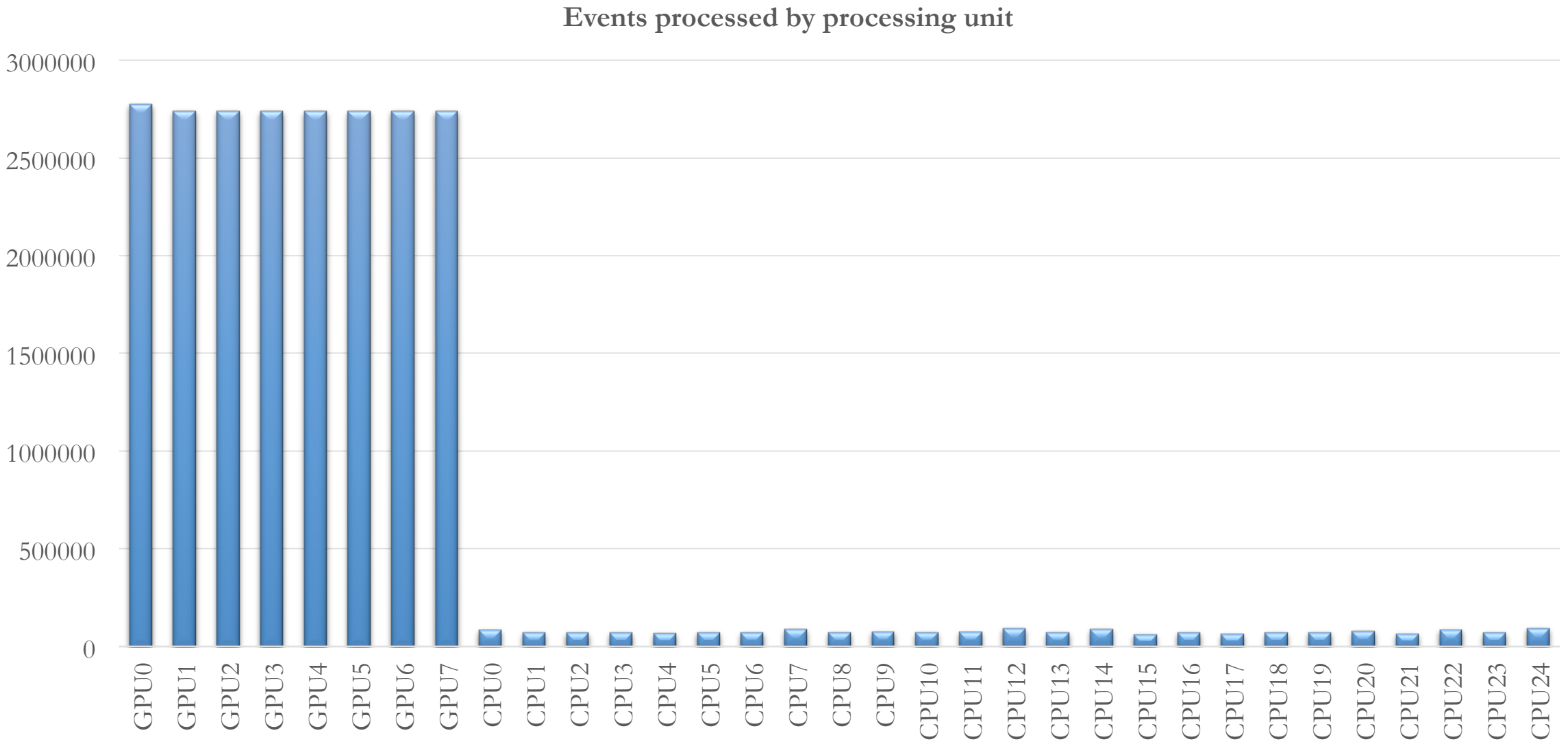
Hardware on the bench

- We acquired small machine for development and testing:
  - 2 sockets x Intel(R) Xeon(R) CPU E5-2650 v4 @ 2.20GHz (12 physical cores)
  - 256GB system memory
  - 8x GPUs NVIDIA GTX 1080Ti
  - Total cost: 5x 🍌

Rate test

- The rate test consists in:
  - preloading in host memory few hundreds events
  - Assigning a host thread to a host core
  - Assigning a host thread to a GPU
  - Preallocating memory for each GPU for each of 8 cuda streams
  - Filling a concurrent queue with event indices
  - During the test, when a thread is idle it tries to pop from the queue a new event index:
    - Data for that event are copied to the GPU (if the thread is associated to a GPU)
    - processes the event (exactly same code executing on GPUs and CPUs)
    - Copy back the result
  - The test ran for approximately one hour
  - At the end of the test the number of processed events per thread is measured, and the total rate can be estimated

Rate test


Events processed by processing unit
3000000
2500000
2000000
1500000
1000000
500000
0
GPU0
GPU1
GPU2
GPU3
GPU4
GPU5
GPU6
GPU7
CPU0
CPU1
CPU2
CPU3
CPU4
CPU5
CPU6
CPU7
CPU8
CPU9
CPU10
CPU11
CPU12
CPU13
CPU14
CPU15
CPU16
CPU17
CPU18
CPU19
CPU20
CPU21
CPU22
CPU23
CPU24

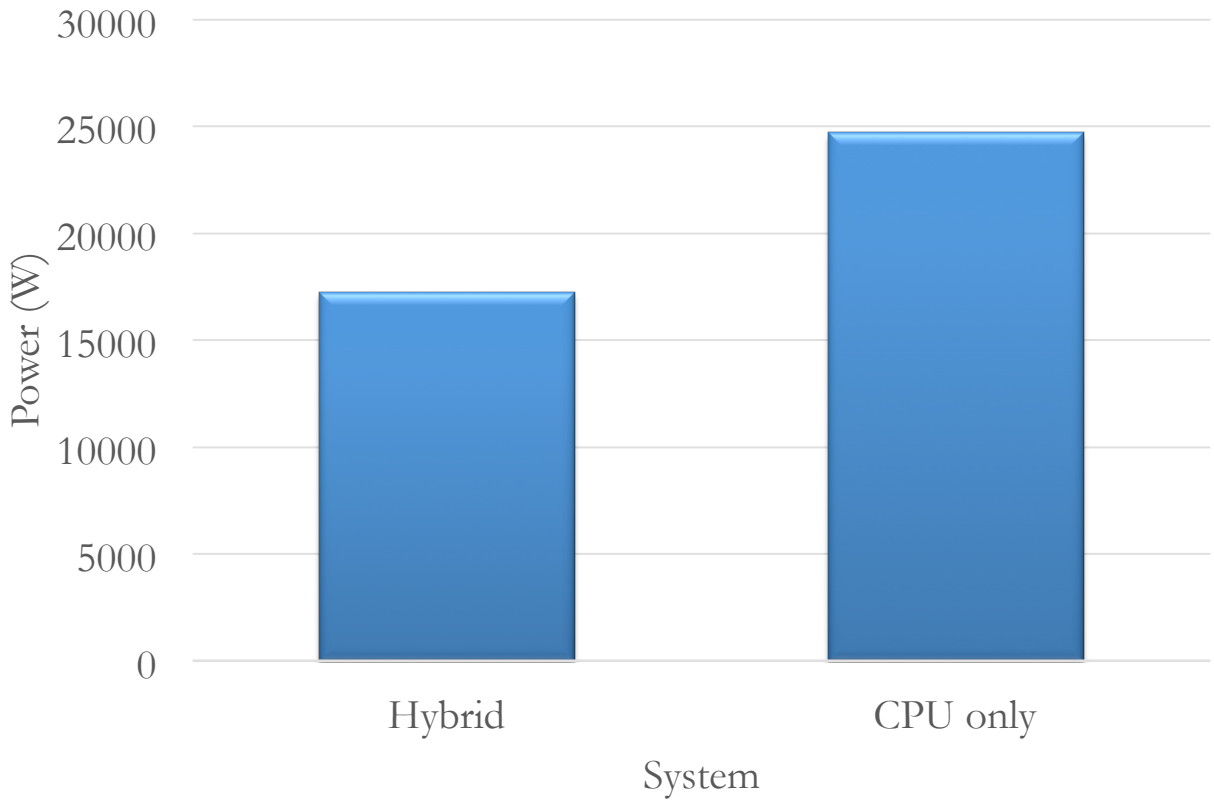
Rate test

- Total rate measured:
  - 8xGPU: 6527 Hz
  - 24xCPUs: 613 Hz
- Number of nodes to reach 100kHz: ~14
- Total Price: 70x 🍌

- When running with only 24xCPUs
  - Rate with 24xCPUs: 777 Hz
- Number of nodes to reach 100kHz: ~128
- Total Price: 320x 🍌
- Assuming an initial cost of 2.5🍌 per node

Energy efficiency

- During the rate test power dissipated by CPUs and GPUs was measured every second
  - Nvidia-smi for GPUs
  - Turbostat for CPUs
- 8 GPUs: 1037W
  - 6.29 Events per Joule
  - 0.78 Events per Joule per GPU
- 24 CPUs in hybrid mode: 191W
  - 3.2 Events per Joule
  - 0.13 Events per Joule per core
- 24 CPUs in CPU-only test: 191W
  - 4.05 Events per Joule
  - 0.17 Events per Joule per core



- That is 1/3 more 🍌s in the energy bill when processing 100kHz input

Conclusion

- Dynamic problems are very hard to fit in HPC-like paradigms
  - Good news*: we have to fit them
- Results demonstrated
  - Challenges in the integration (see talk by Vincenzo this afternoon)
  - Still need to put all the components together and choose the platform
- At CERN, 🍌 are extremely expensive!
- Same codebase running on CPUs and GPUs and producing same results bit-by-bit

* For you