D1: Track Reconstruction based on Cellular Automaton

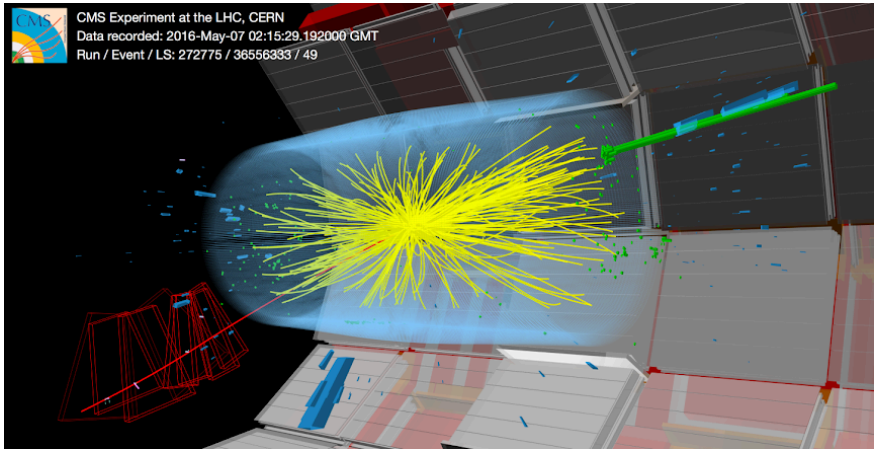
Alexander Schmidt

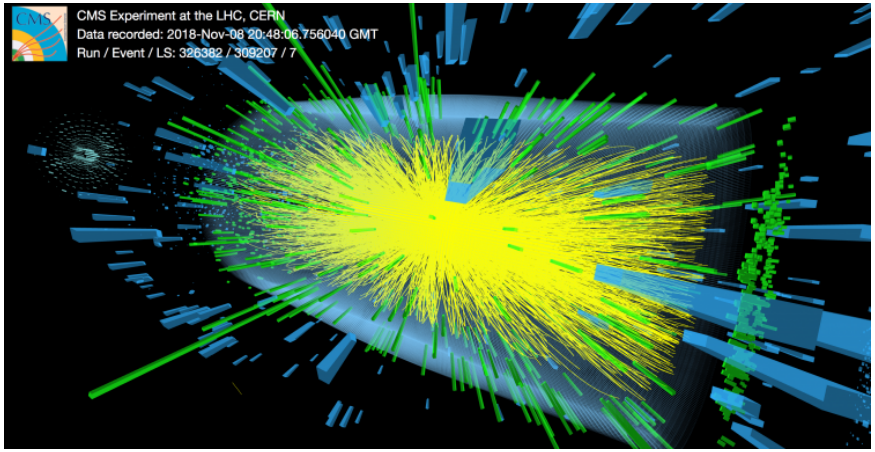
Rheinisch-Westfälische Technische Hochschule Aachen

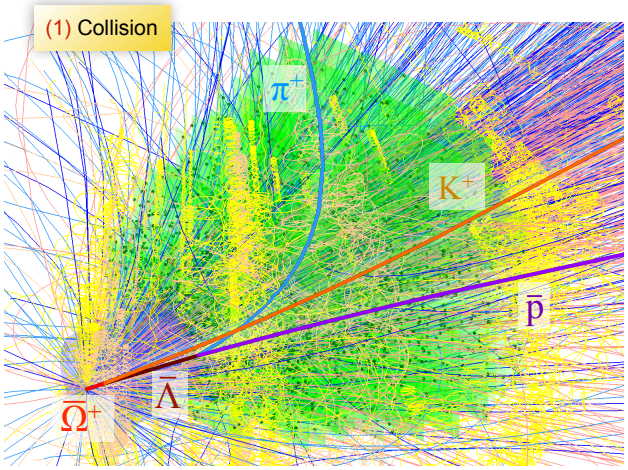
Ivan Kisel

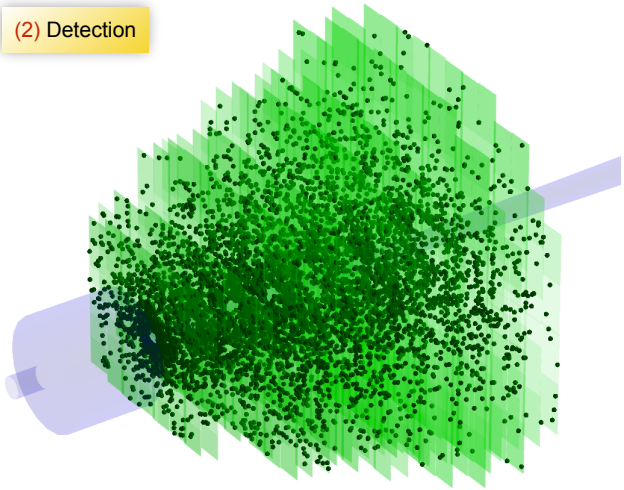
Johann Wolfgang Goethe-Universität Frankfurt am Main

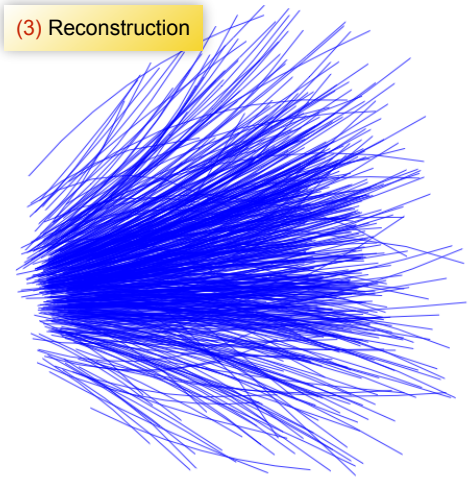
Reconstruction Challenges in CMS and CBM









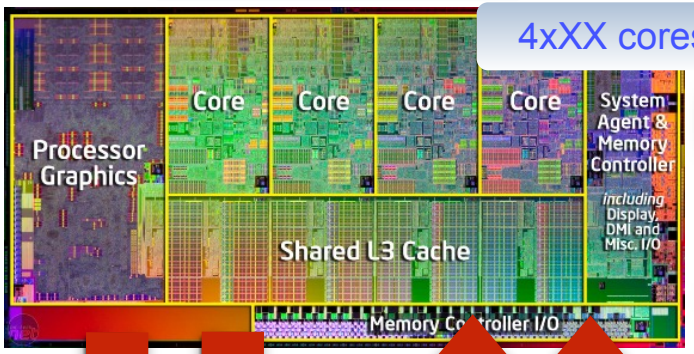


Track finding is a complicated time consuming combinatorial problem!

Many-Core CPU/GPU Architectures

Intel/AMD CPU

4xXX cores



- Optimiz or lo tency access ache ata sets
- Control c for -of-order and cula execution

Nvidia/ATI GPU

XXXX cores

Optimized for data-parallel, throughput computation
lore transistors dedicated to computation

Math

Memory

Parallelism

Math

Memory

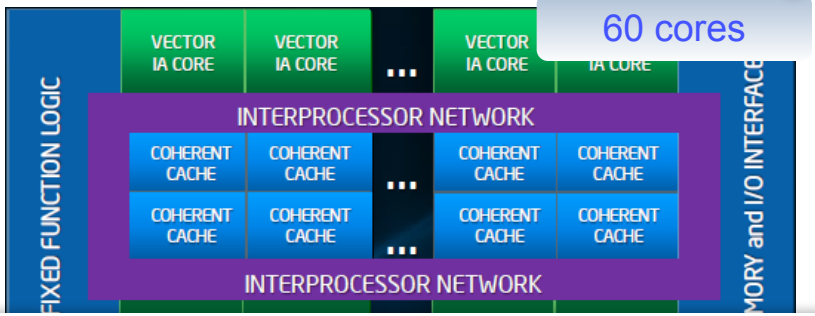
#Cores

Stability

Memory

Intel Phi

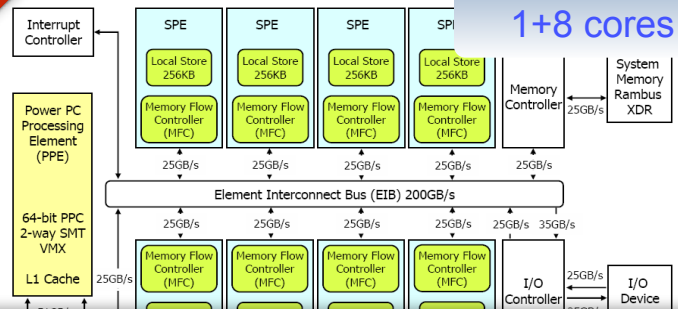
60 cores



- Many Integrated Cores architecture announced at ISC10 (June 2010)
- Based on the x86 architecture
- Many-cores + 4-way multithreaded + 512-bit wide vector unit

IBM Cell

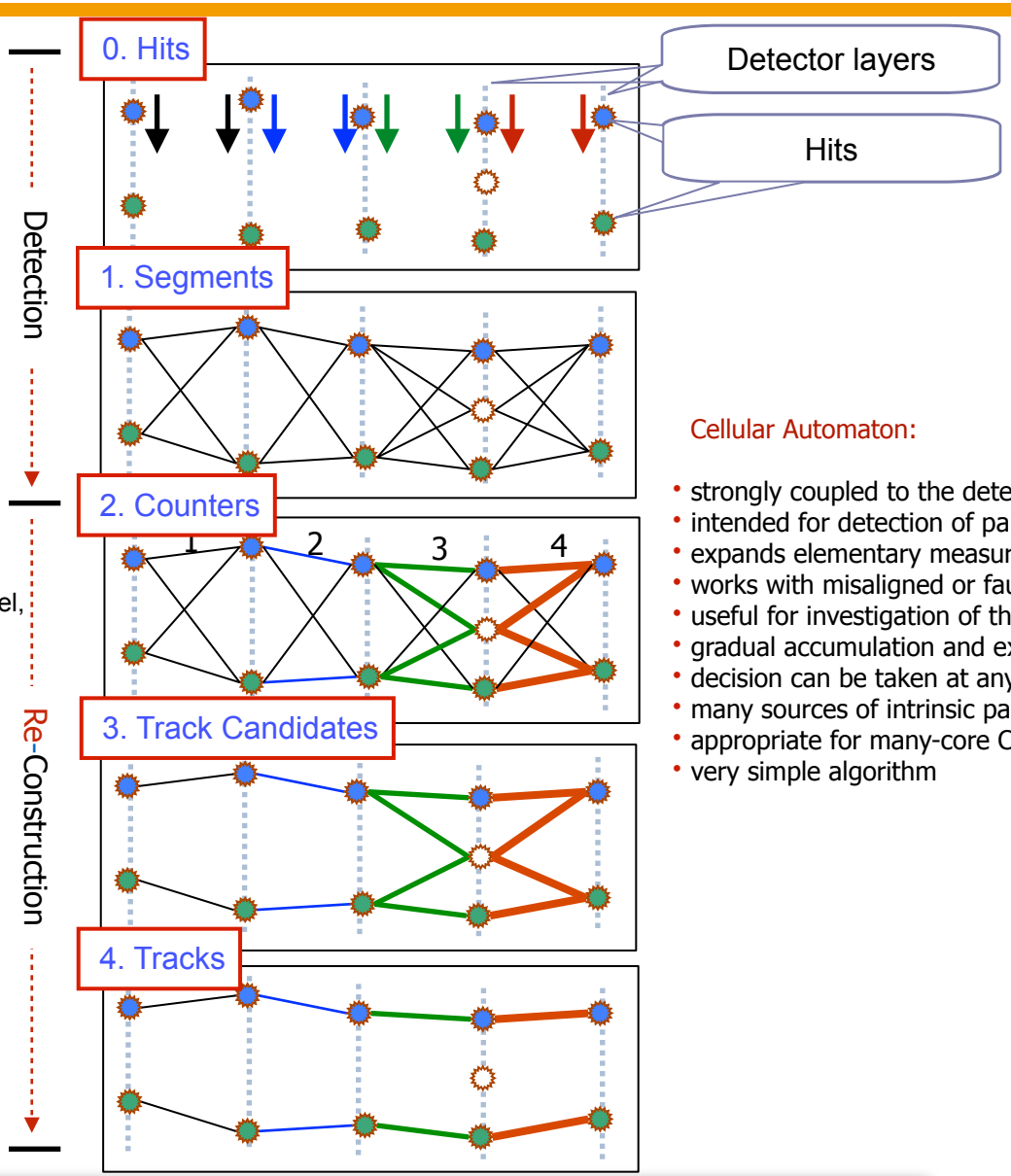
1+8 cores



- General purpose RISC processor (PowerPC)
- 8 co-processors (SPE, Synergistic Processor Elements)
- 128-bit wide SIMD units

Future systems are heterogeneous. Fundamental redesign of traditional approaches to data processing is necessary

Cellular Automaton (CA) Track Finder



Cellular Automaton:

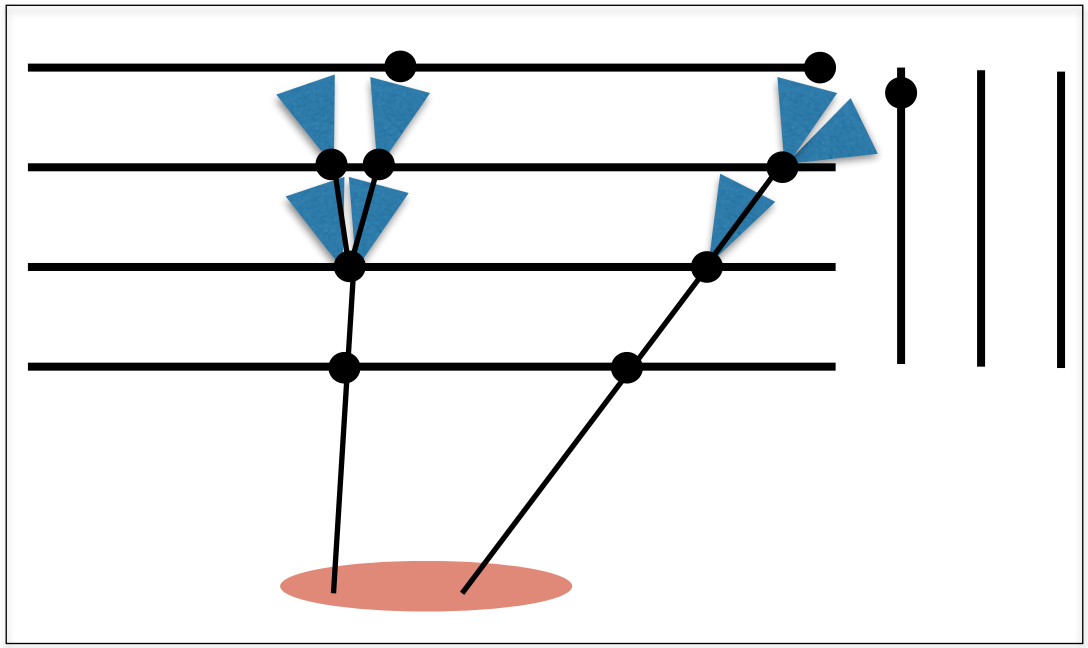
1. Build short track segments.
2. Connect them according to the track model, estimate their location in a track.
3. Tree structures appear, collect segments into track candidates.
4. Select the best track candidates.

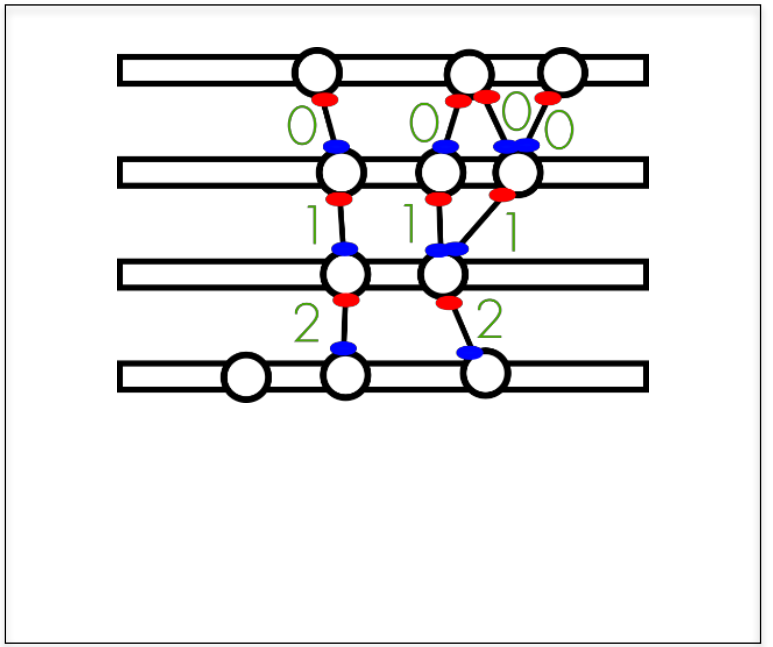
Cellular Automaton:

- strongly coupled to the detector system as its extension
- intended for detection of particle trajectories
- expands elementary measurements over 2-3 detectors
- works with misaligned or faulty detector systems
- useful for investigation of the detector performance
- gradual accumulation and extraction of information
- decision can be taken at any stage
- many sources of intrinsic parallelism
- appropriate for many-core CPU/GPU
- very simple algorithm

Useful for complicated event topologies with heavy combinatorics

CMS Example: Phase II Pixel Detector



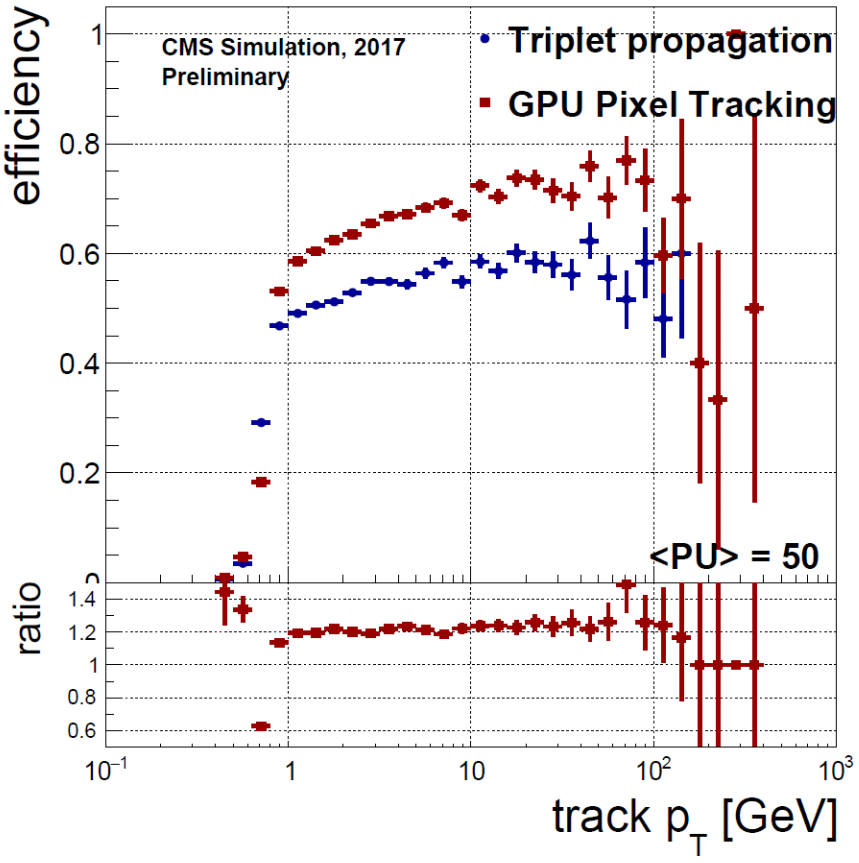


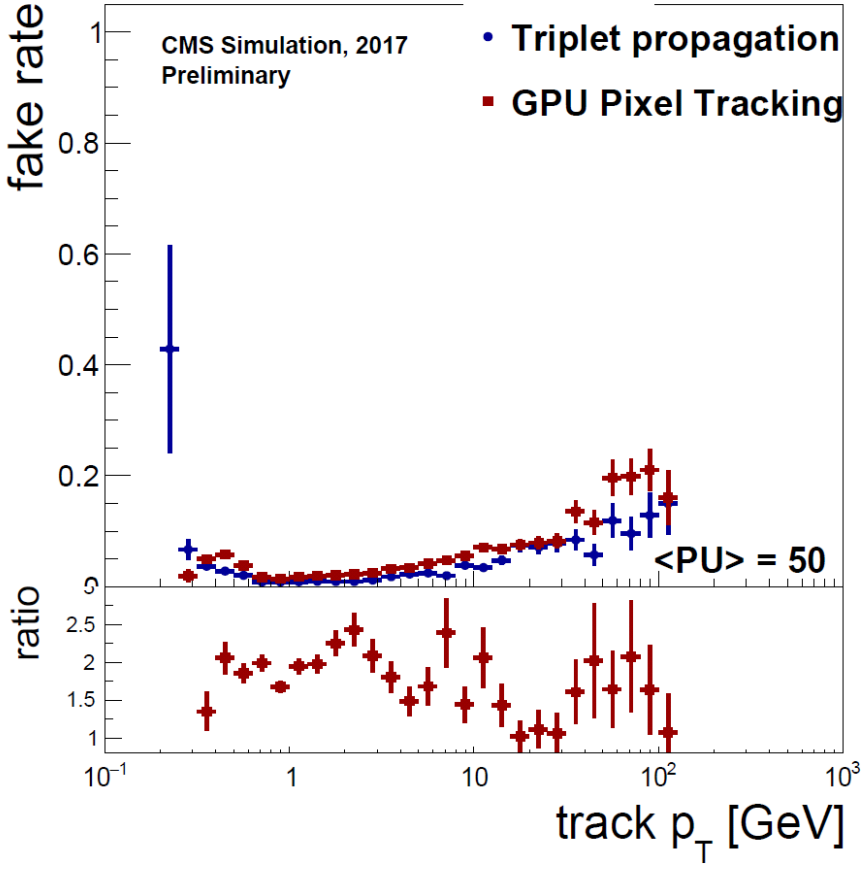
fourth detector layer

- quadruplet seeding: propagate triplets to fourth layer
- natural extension of current algorithm
- computing time grows **exponential** with PU

- new seeding algorithm
- based on parallel-friendly algorithmic structure (Cellular Automaton)
- computing time grows **linear** with PU

CMS Physics Performance





Efficient and fast tracking

CMS Timing Performance

	Time/event **CPU** (ms)	Time/event **GPU** (ms)
Triplet Propagation	66.3	
Cellular Automaton	22.0	1.6

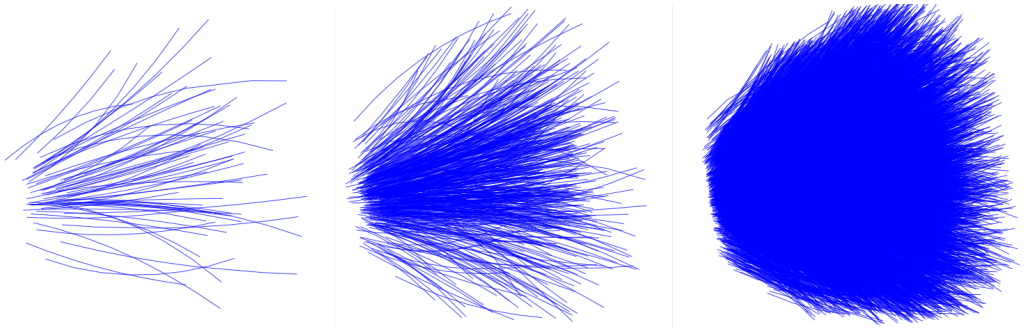
Hardware used:
- CPU Intel 4771K
- GPU NVIDIA K40

- **improve** physics performance at the same cost (easily run tracking on all events in HLT)
- or **save** millions of EUR at same physics performance

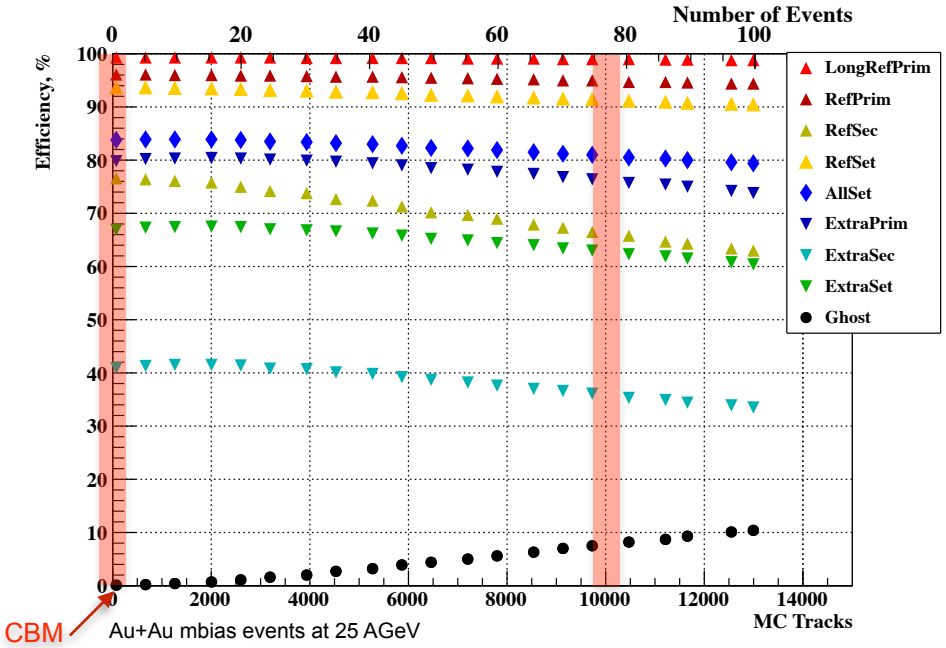
→ Change our approach to algorithm development

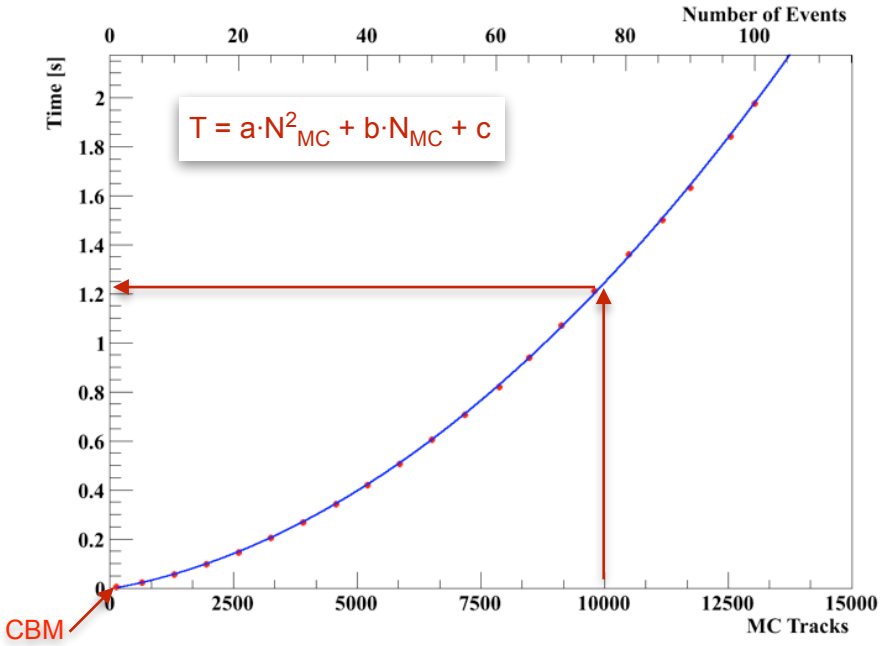
CBM CA Track Finder at High Track Multiplicity

A number of minimum bias events is gathered into a group (super-event), which is then treated by the CA track finder as a single event.



1 mbias event, $\langle N_{reco} \rangle = 109$ | 5 mbias events, $\langle N_{reco} \rangle = 572$ | 100 mbias events, $\langle N_{reco} \rangle = 10340$

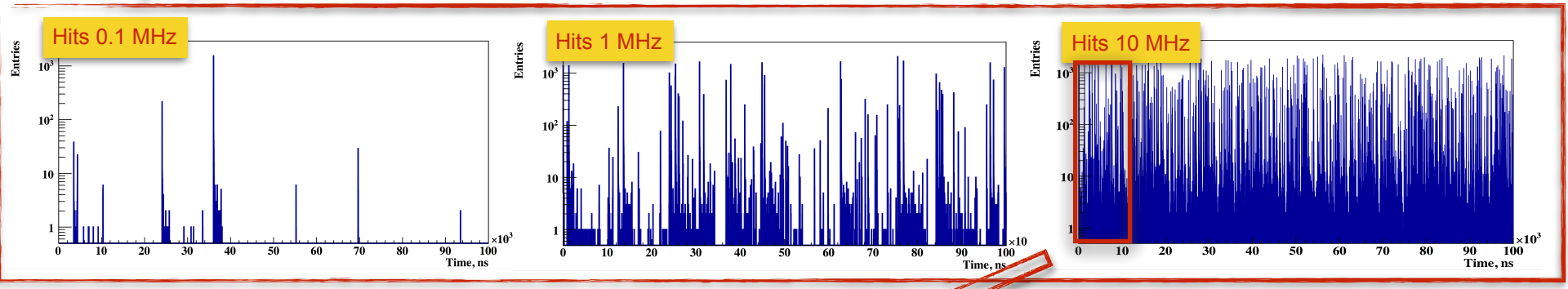




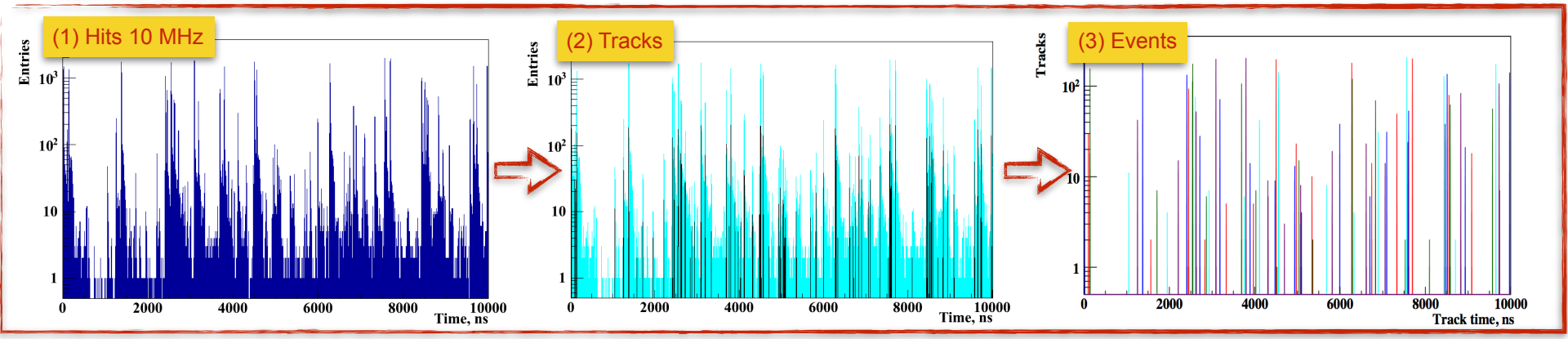
Reliable reconstruction efficiency and time as a second order polynomial w.r.t. to the track multiplicity

CBM 4D Event Building at 10 MHz

Hits at high input rates



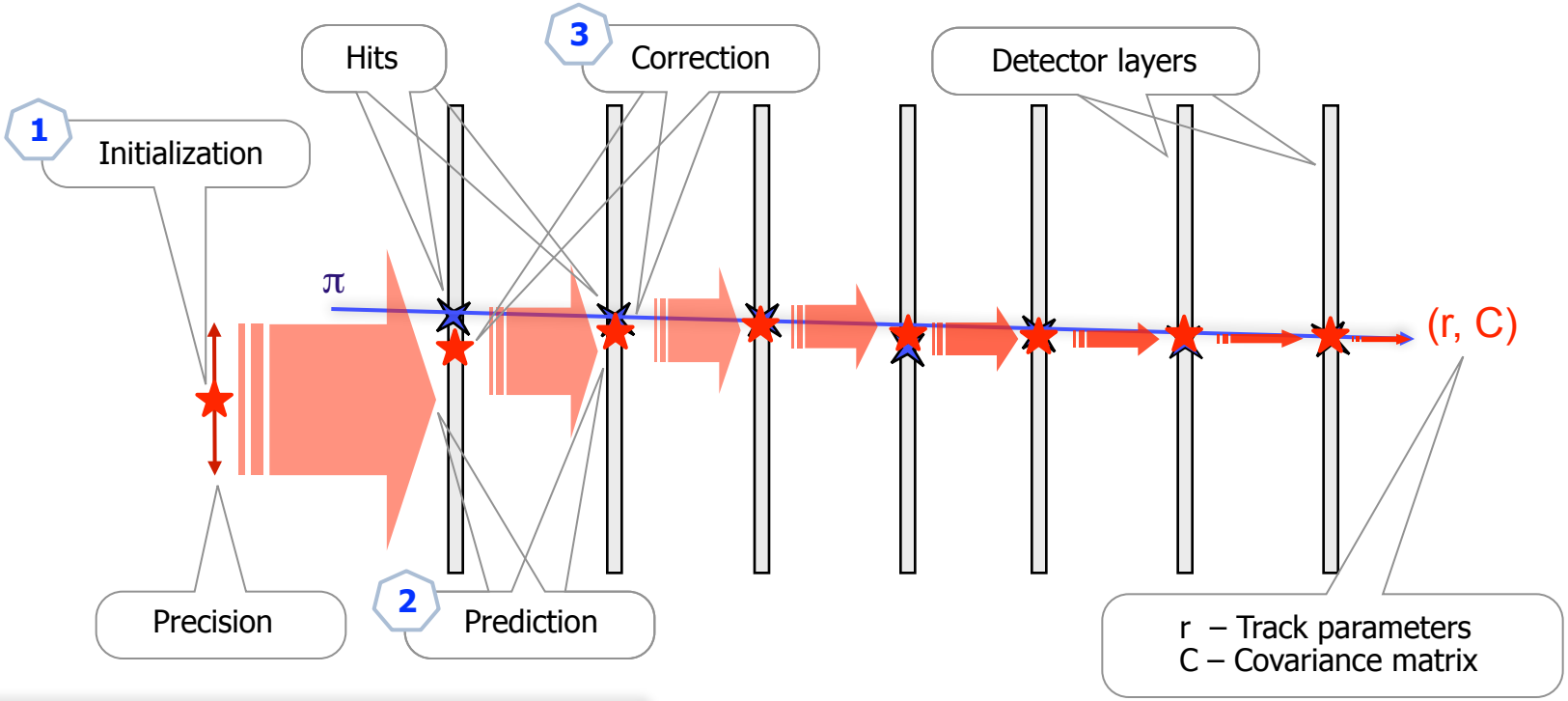
From hits to tracks to events

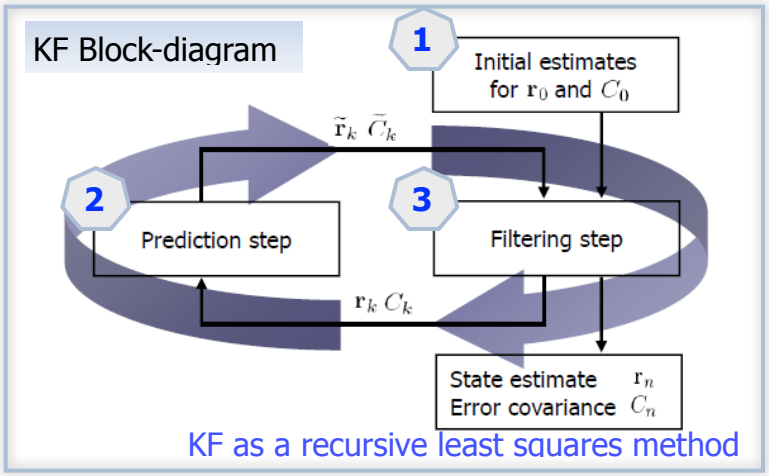


Reconstructed tracks clearly represent groups, which correspond to the original events

Kalman Filter (KF) based Track Fit

Estimation of the track parameters at one or more hits along the track – Kalman Filter (KF)





KF as a recursive least squares method

State vector

Position, direction and momentum

$r = \{ x, y, z, p_x, p_y, p_z \}$

Kalman Filter:

1. Start with an arbitrary initialization.
2. Add one hit after another.
3. Improve the state vector.
4. Get the optimal parameters after the last hit.

Nowadays the Kalman Filter is used in almost all HEP experiments

Kalman Filter Track Fit Library

Kalman Filter Methods

Kalman Filter Tools:
- KF Track Fitter
- KF Track Smoother
- Deterministic Annealing Filter

Kalman Filter Approaches:
- Conventional DP KF
- Conventional SP KF
- Square-Root SP KF
- UD-Filter SP
- Gaussian Sum Filter
- 3D (*x*,*y*,*z*) and 4D (*x*,*y*,*z*,*t*) KF

Track Propagation:
- Runge-Kutta
- Analytic Formula

Detector Types:
- Pixel
- Strip
- Tube
- TPC

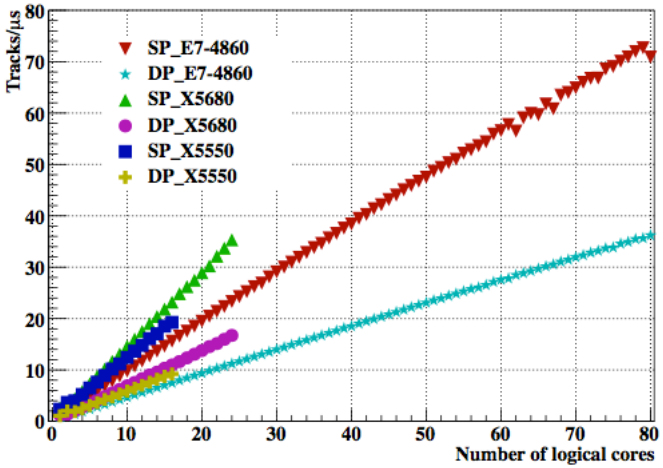
Implementations

Vectorization (SIMD):
- Header Files
- Vc Vector Classes
- ArBB Array Building Blocks
- OpenCL

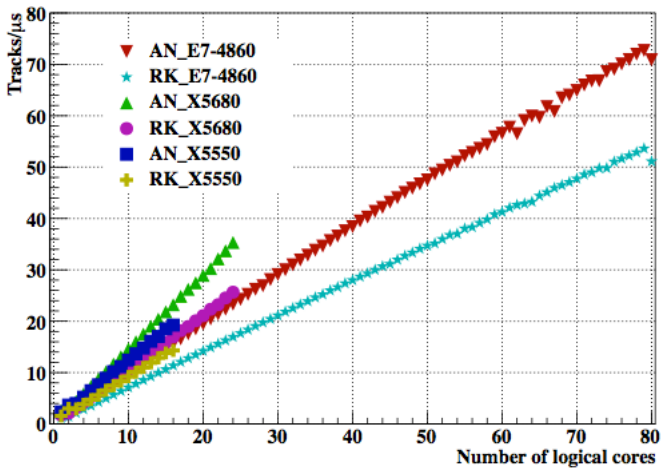
Parallelization (many-cores):
- Open MP
- ITBB
- ArBB
- OpenCL

Precision:
- single precision SP
- double precision DP

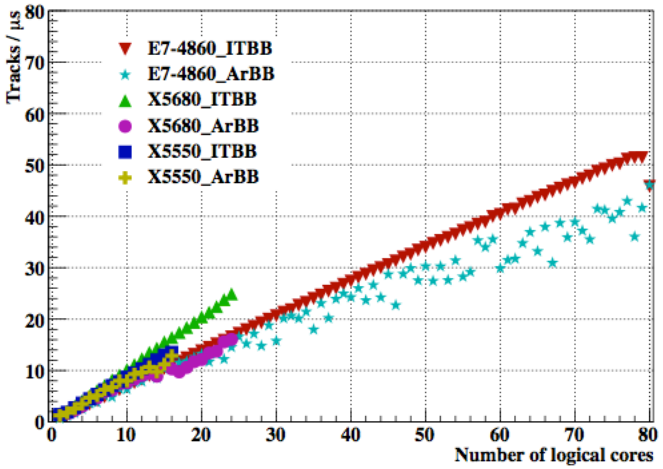
Conventional KF DP vs. SP



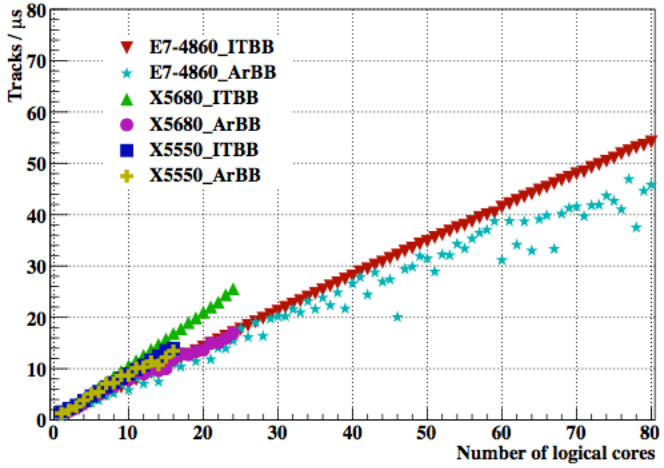
Conventional KF RK4 vs. Analytical



Square-Root KF



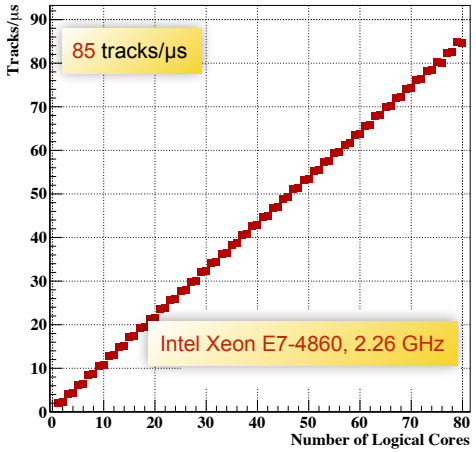
UD KF

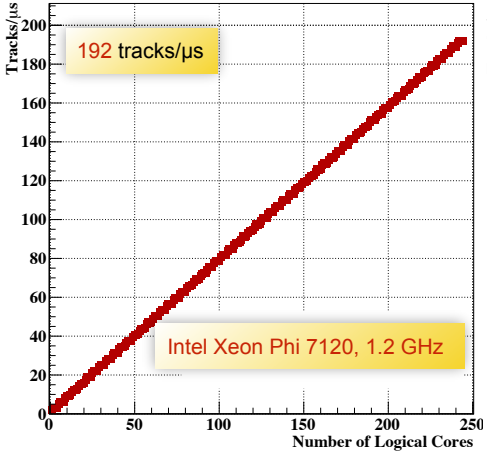


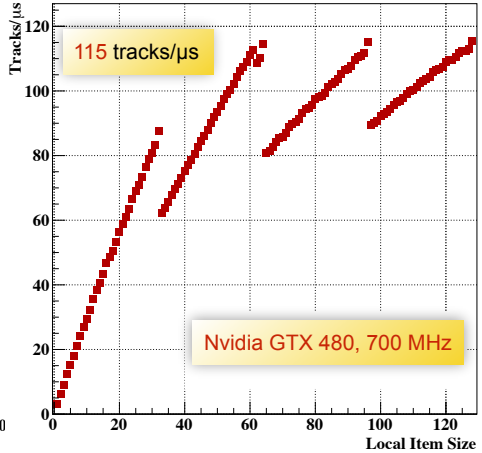
Strong many-core scalability of the Kalman filter library

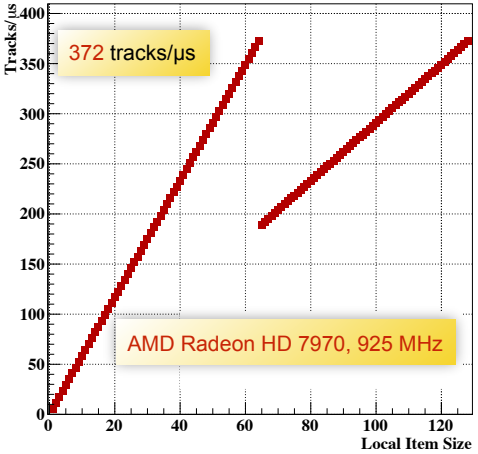
with I. Kulakov, H. Pabst* and M. Zyzak (*Intel)

Kalman Filter Track Fit





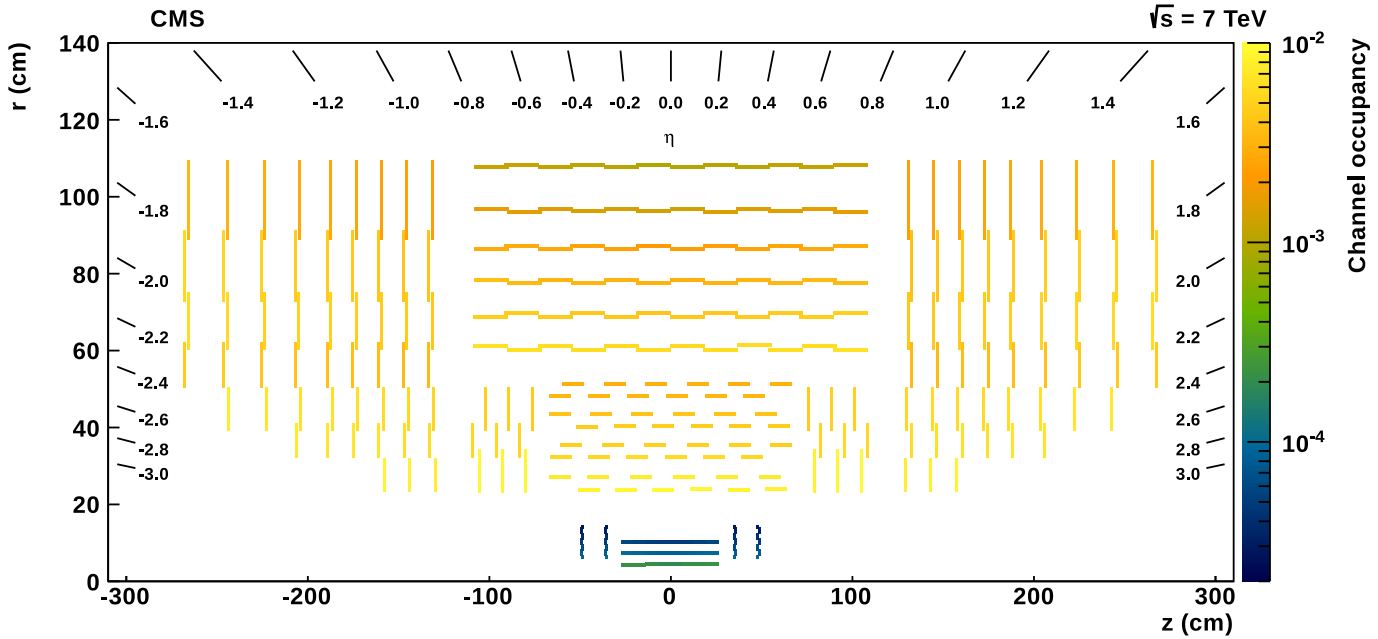




- Precise estimation of the parameters of particle trajectories is the core of the reconstruction procedure.
- Scalability with respect to the number of logical cores in a CPU is one of the most important parameters of the algorithm.
- The scalability on the Intel Xeon Phi coprocessor is similar to the CPU, but running four threads per core instead of two.
- In case of the graphics cards the set of tasks is divided into working groups of size *local item size* and distributed among compute units (or streaming multiprocessors) and the load of each compute unit is of the particular importance.
- The track fit performance on a single node: 2*CPU+2*GPU = 10^9 tracks/s = (100 tracks/event)* 10^7 events/s = 10^7 events/s.
- A single compute node is enough to estimate parameters of all particles produced at the maximum 10^7 interaction rate!

The fastest implementation of the Kalman filter in the world

Summary

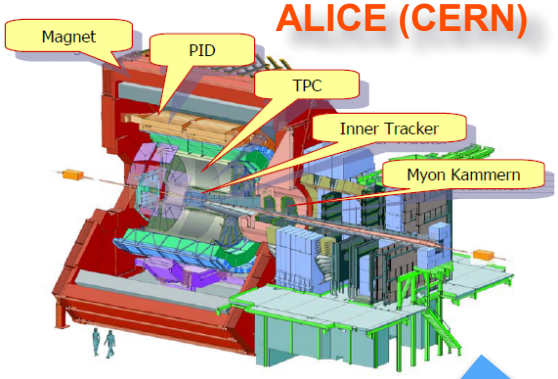


- Concrete next step for CMS: extend pixel tracking to the whole cylinder
- Concrete next step for CMS: port CBM tracking to the endcap

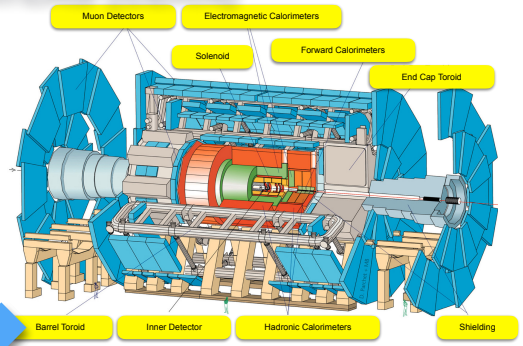
- Frankfurt and Aachen groups have started collaborative effort (e.g. try to run CBM tracking on CMS endcap)
- Need a common cross-experiment pool of expertise, as the problems are of highest complexity
- Possible vision: Experiment independent approach

Consolidate Efforts: A Common Reconstruction Package

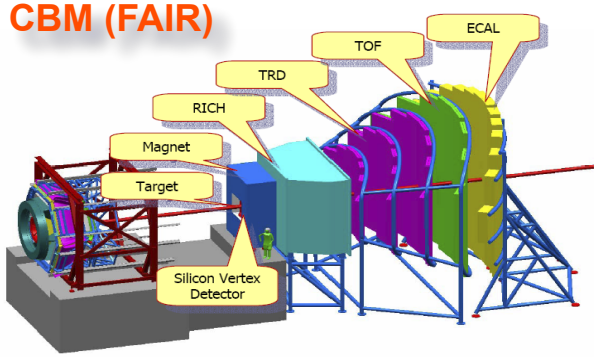
ALICE (CERN)



ATLAS (CERN)

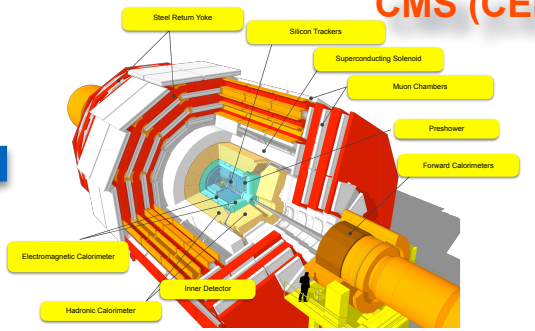


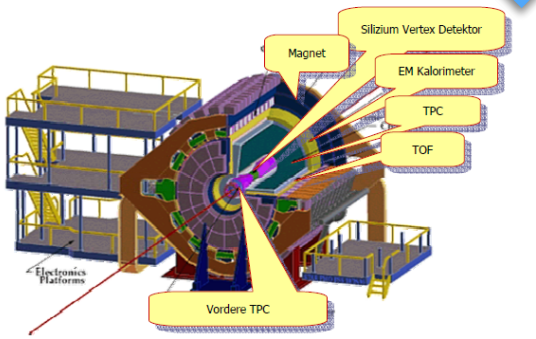
CBM (FAIR)



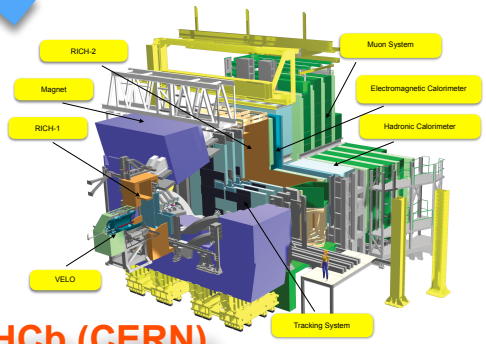
CA

CMS (CERN)





STAR (BNL)



LHCb (CERN)