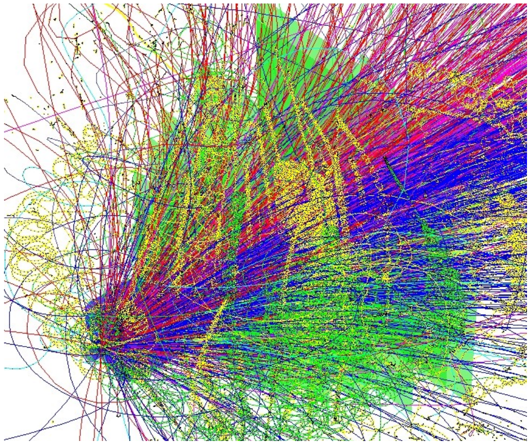
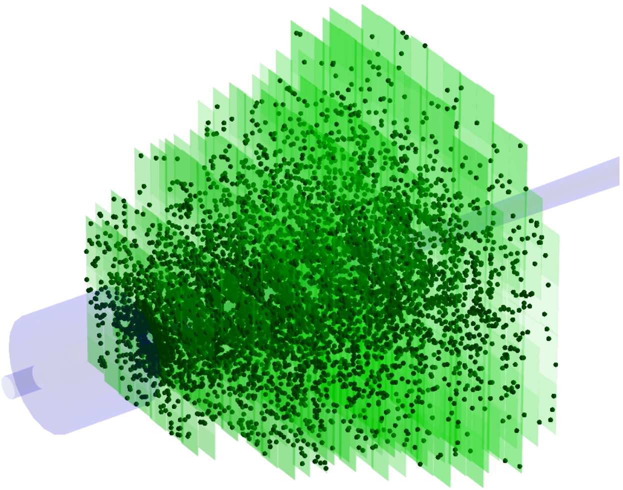
4D Cellular Automaton Track Finder in the CBM Experiment

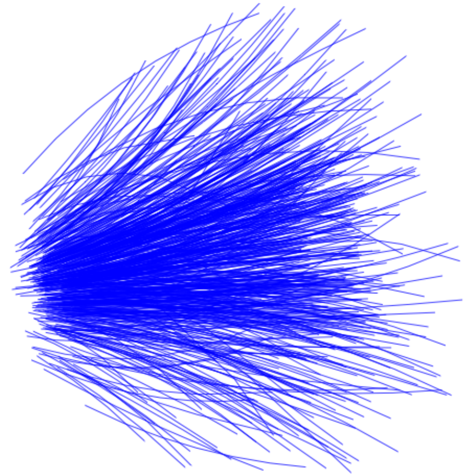
Ivan Kisel
for the CBM Collaboration

Goethe-University Frankfurt am Main
FIAS Frankfurt Institute for Advanced Studies

Reconstruction Challenge in CBM at FAIR/GSI



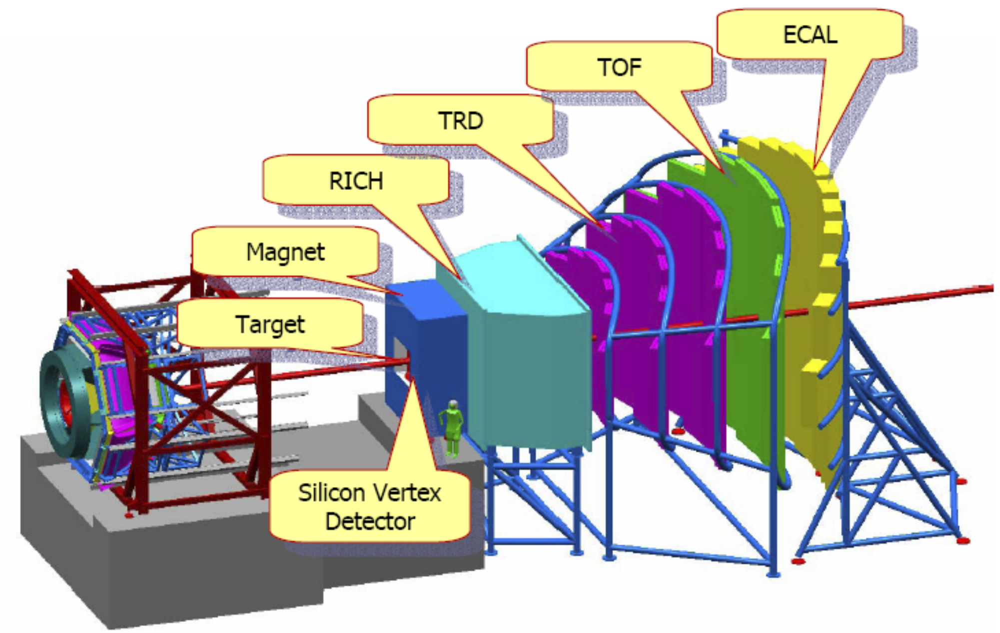




- Future fixed-target heavy-ion experiment
- 10^7 Au+Au collisions/sec
- ~ 1000 charged particles/collision
- Non-homogeneous magnetic field
- Double-sided strip detectors (85% fake space-points)

Full event reconstruction will be done
on-line at the First-Level Event Selection (FLES) and
off-line using the same FLES reconstruction package.

Cellular Automaton (CA) Track Finder
Kalman Filter (KF) Track Fitter
KF short-lived Particle Finder

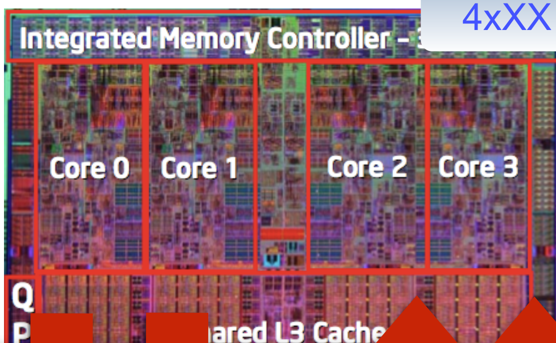


All reconstruction algorithms are vectorized and parallelized.

Many-Core CPU/GPU Architectures

Intel/AMD CPU

4xXX cores



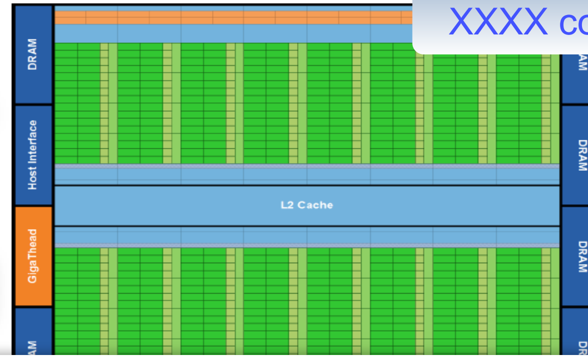
- Optimiz... or lo... tency access ... ache ... ata sets
- Control ... for ... -of-order and ... cula... execution

Math → Memory ←

Parallelism ↓ Math ↓ Memory ↑ #Cores ↑

Nvidia/ATI GPU

XXXX cores

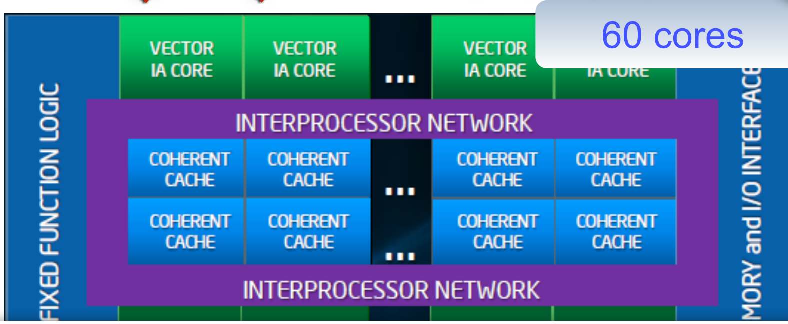


- ...ptimized for data-parallel, throughput computation
- ...lore transistors dedicated to computation

Stability ↗ Memory ↙

Intel Phi

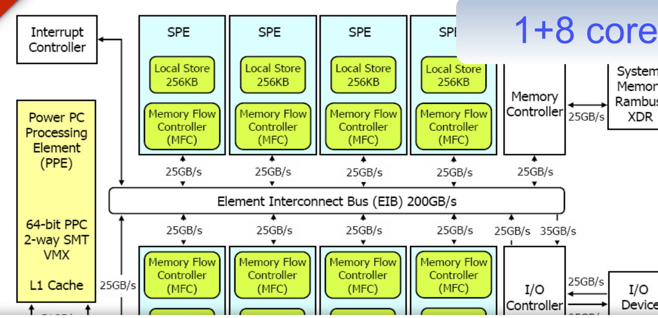
60 cores



- 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, but using the same code

Kalman Filter (KF) 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

Track Propagation:
- Runge-Kutta
- Analytic Formula

Implementations

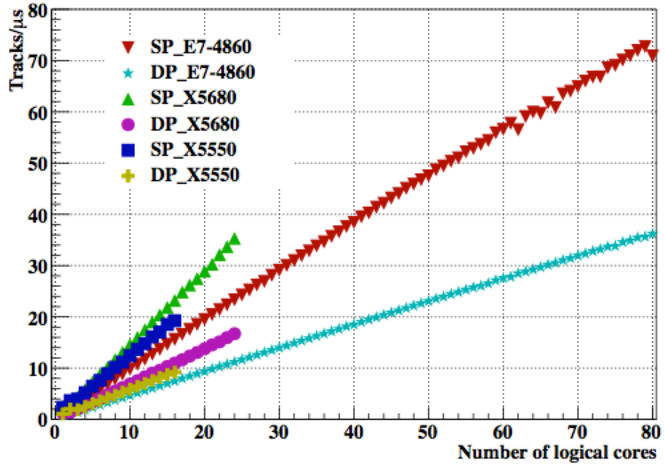
Vectorization (SIMD):
- Header Files
- Vc Vector Classes
- ArBB Array Building Blocks
- OpenCL

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

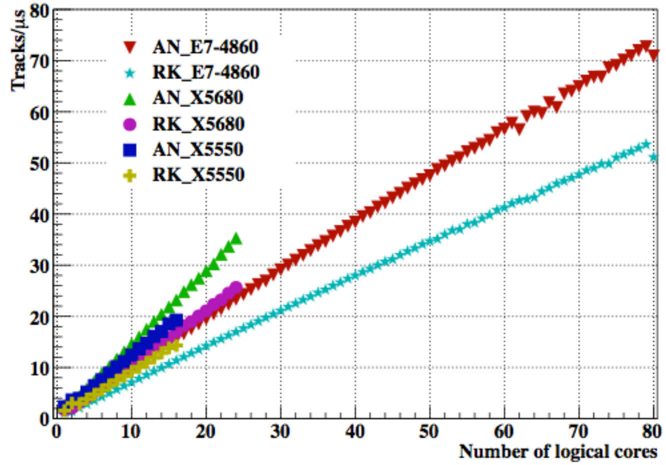
Precision:
- single precision SP
- double precision DP

Comp. Phys. Comm. 178 (2008) 374-383

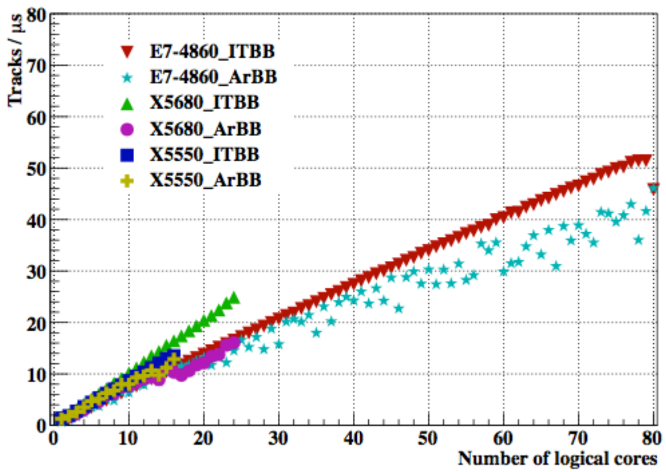
Conventional KF DP vs. SP



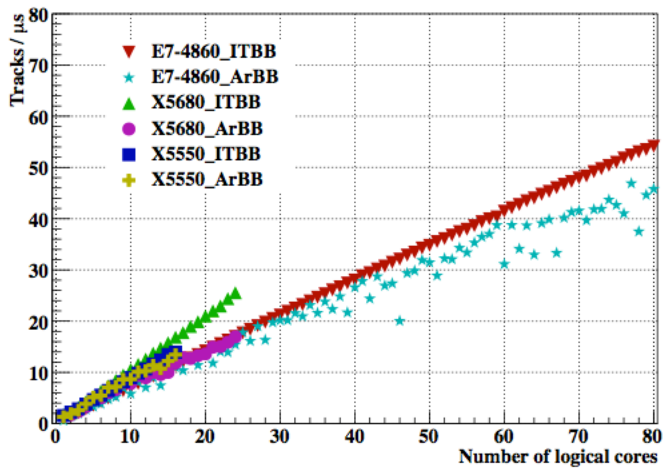
Conventional KF RK4 vs. Analytical



Square-Root KF



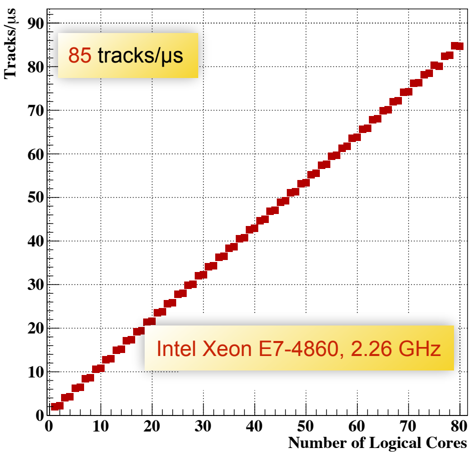
UD KF

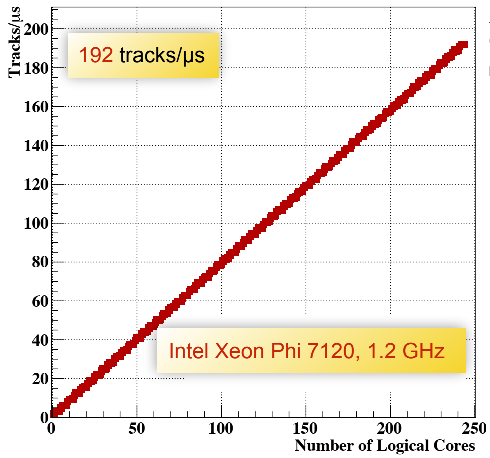


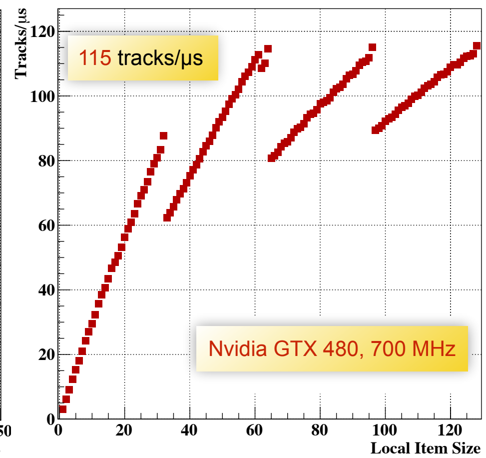
Strong many-core scalability of the Kalman filter library

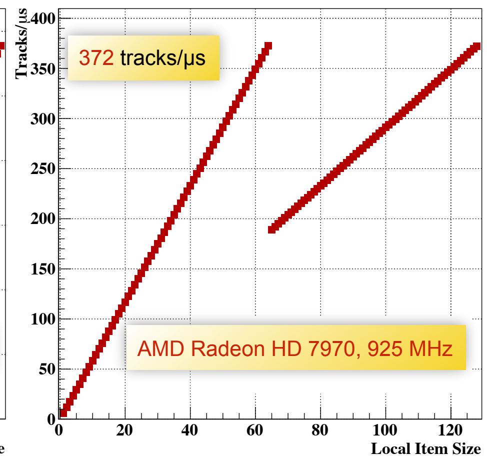
with I. Kulakov, H. Pabst* and M. Zyzak (*Intel)

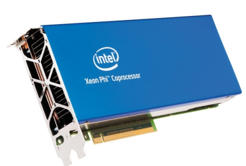
Kalman Filter (KF) Track Fit Library







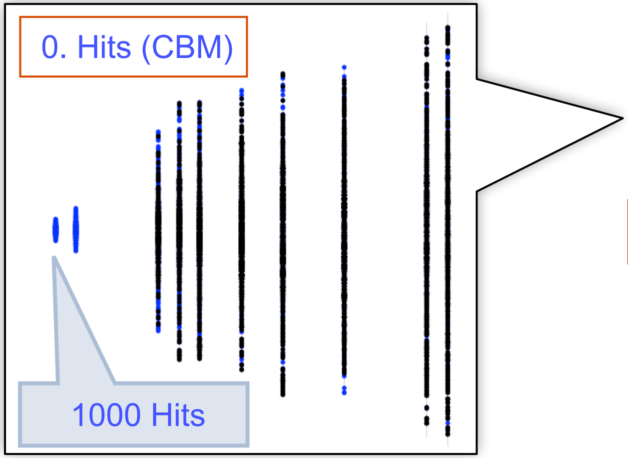


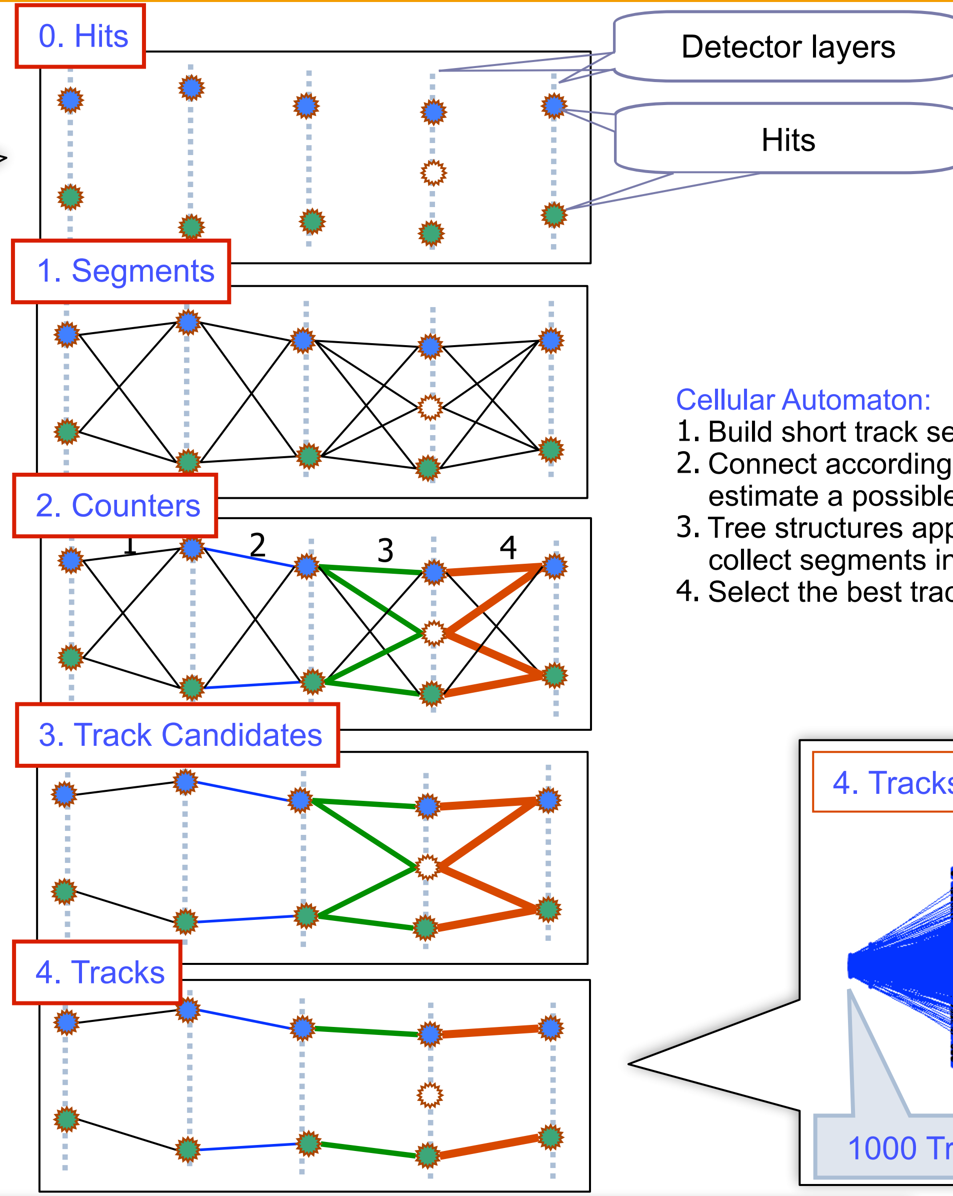


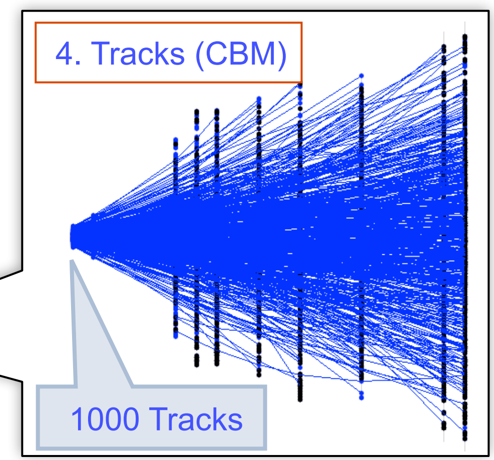
- Scalability with respect to the number of logical cores in a CPU is one of the most important parameters of the algorithm.
- The scalability on the Intel Xeon Phi coprocessor is similar to the CPU, but running four threads per core instead of two.
- In case of the graphic cards the set of tasks is 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.

Full portability of the Kalman filter library

Cellular Automaton (CA) Track Finder







Cellular Automaton:

1. Build short track segments.
2. Connect according to the track model, estimate a possible position on a track.
3. Tree structures appear, collect segments into track candidates.
4. Select the best track candidates.

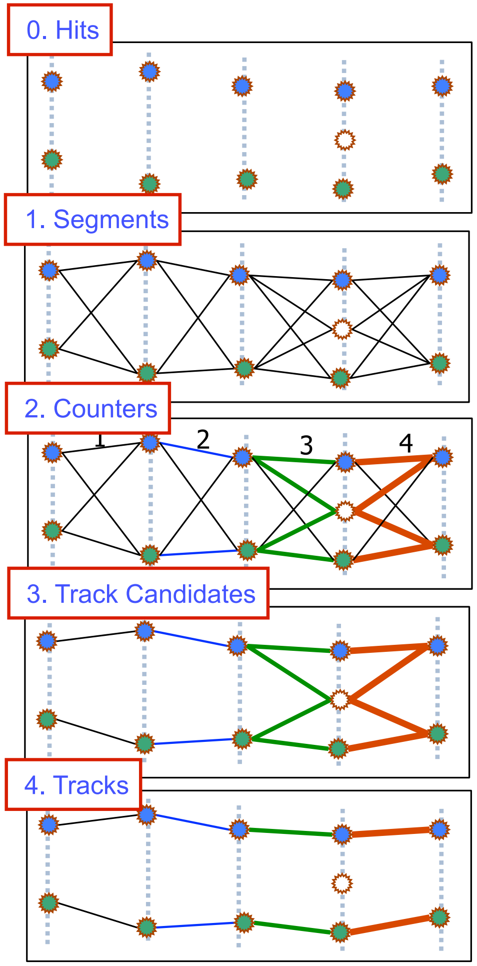
Cellular Automaton:

- local w.r.t. data
- intrinsically parallel
- extremely simple
- very fast

Perfect for many-core CPU/GPU !

Useful for complicated event topologies with large combinatorics and for parallel hardware

CA Track Finder: Pseudocode



Pseudocode for CBM CA Track Finder

```
Sort_Input_Hits_According_to_Grid();

for track_set (high_p_primary, low_p_primary, secondary, broken)

switch (track_set)
  case high_p_primary:
    Build_Triplets (min_momentum_for_fast_tracks,
    primary_track_parameter_initilisation, triplets_wo_gaps);

  case low_p_primary:
    Build_Triplets (min_momentum_for_slow_tracks,
    primary_track_parameter_initilisation, triplets_wo_gaps);

  case secondary:
    Build_Triplets (min_momentum_for_slow_tracks,
    secondary_track_parameter_initilisation, triplets_wo_gaps);

  case broken:
    Build_Triplets (min_momentum_for_slow_tracks,
    secondary_track_parameter_initilisation, triplets_with/
    wo_gaps)

  Find_Neighbours();

  for track_length := NStation to 3 do
    for station := FirstStation to NStation do
      for triplets := First_Triplet_Station to
      Last_Triplet_Station do
         track_candidate = Build_Best_Candidate (triplet);

  Save_Candidates(all_track_candidates);

  Delete_Used_Hits();
```

```
void function Build_Triplets (min_momentum,
prim/sec_track_parameter_initilisation,
triplets_with/wo_gaps)
{
  for station := (NStation-2) to FirstStation do
    for hits_portion := First_Portion_Station to
Last_Portion_Station do

      Find_Singlets(hits_portion);
      Find_Doublets(singlets_in_portion);
      Find_Triplets(doublets_in_portion);
}
```

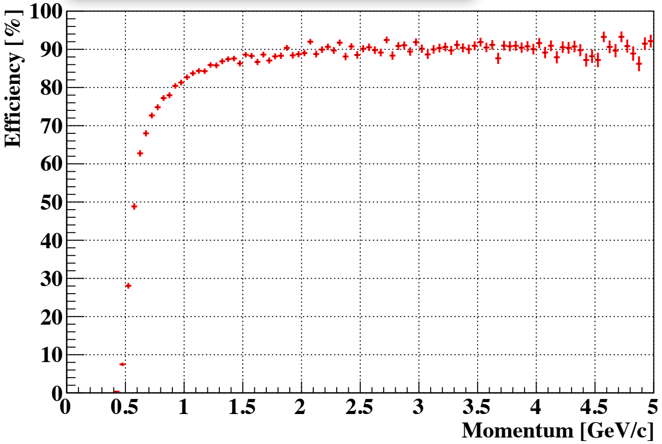
```
void function Find_Neighbours (All_Triplets)

{
  for triplet := First_Triplet to Last_Triplet
do

      Find_Save_Neighbours(triplet);
      Calculate_Level(triplet);

}
```

```
void function
Save_Candidates(All_Track_Candidate)

{
    Sort_Candidates();
      for candidate := First_Candidate to
Last Candidate do
          if (used_hits) discard candidate
          else save candidate;

}
```

Staged track finding

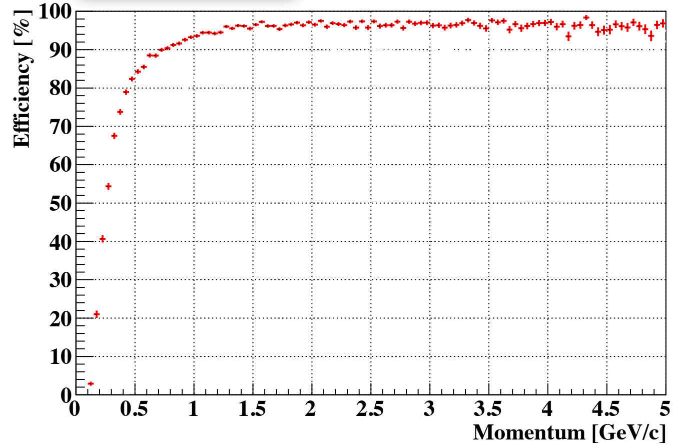
CA Track Finder: Efficiency

(1) high-momentum quasi-primary tracks



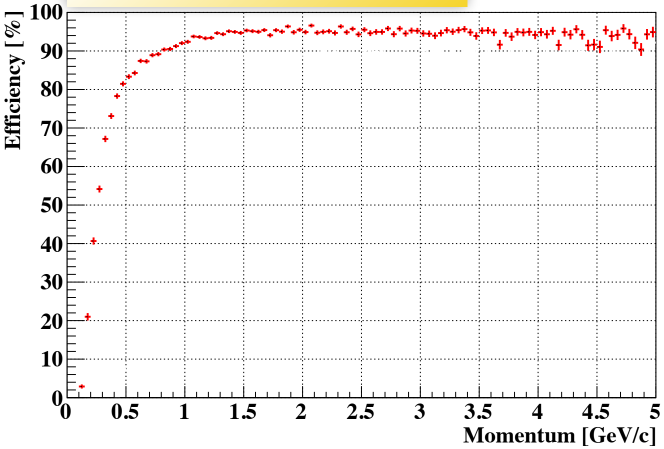
Track category	Eff, %
All tracks	70.4
Primary high-p	94.9
Primary low-p	56.8
Secondary high-p	49.7
Secondary low-p	13.0
Clone level	0.3
Ghost level	0.3
MC tracks found	103
Time, ms/ev	4

(3) secondary tracks



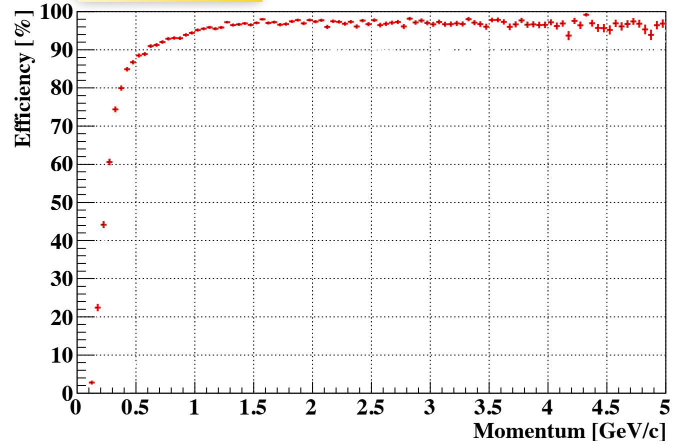
Track category	Eff, %
All tracks	89.2
Primary high-p	97.5
Primary low-p	92.4
Secondary high-p	86.6
Secondary low-p	54.7
Clone level	1.0
Ghost level	5.5
MC tracks found	131
Time, ms/ev	7

(2) low-momentum quasi-primary tracks



Track category	Eff, %
All tracks	87.8
Primary high-p	95.8
Primary low-p	91.4
Secondary high-p	84.5
Secondary low-p	54.2
Clone level	0.9
Ghost level	5.6
MC tracks found	129
Time, ms/ev	6

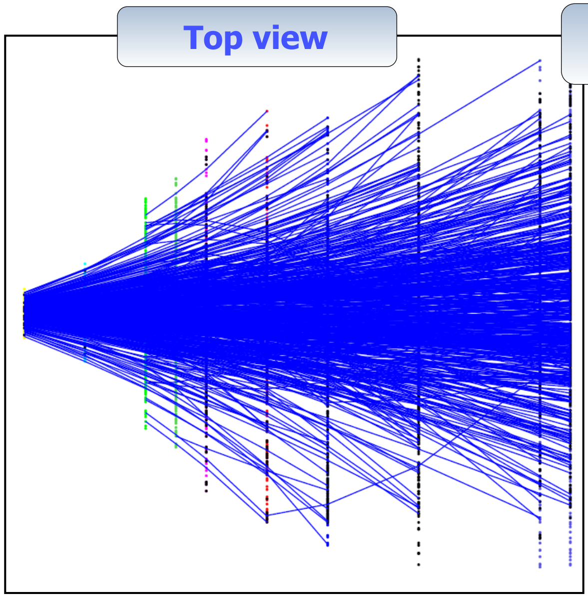
(4) broken tracks

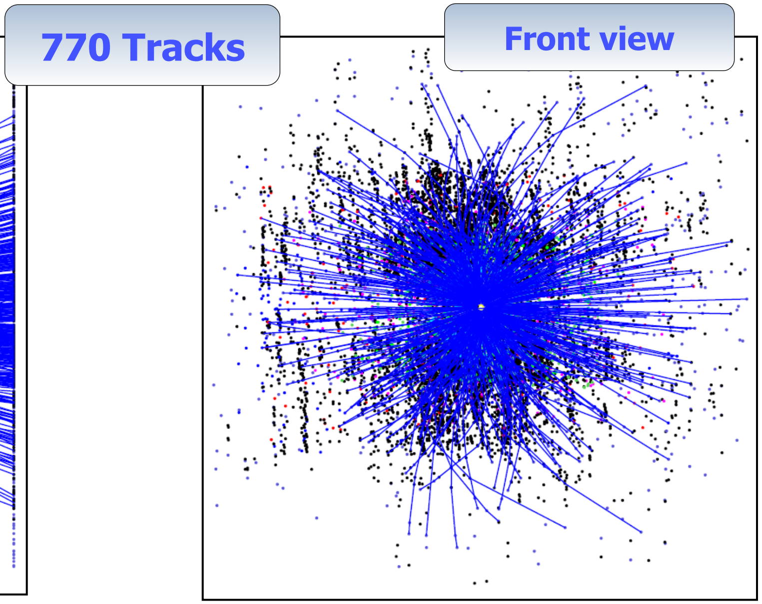


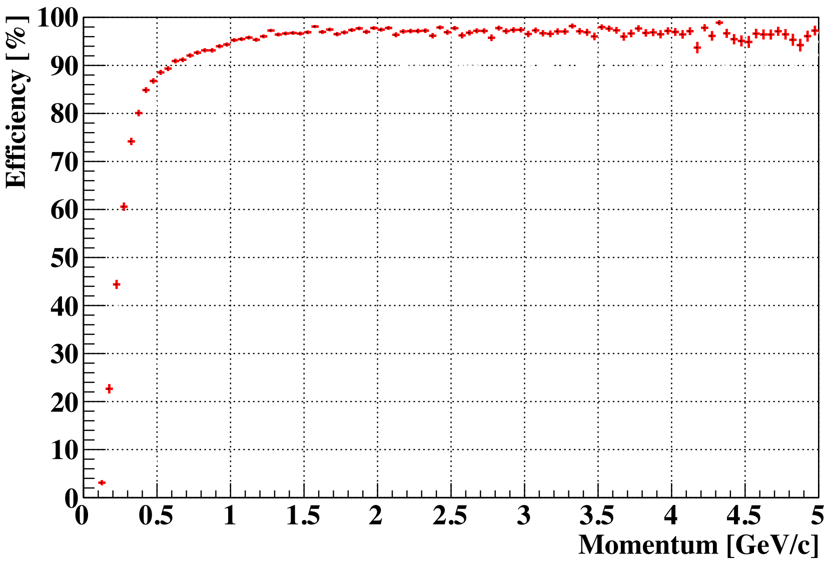
Track category	Eff, %
All tracks	90.9
Primary high-p	97.5
Primary low-p	92.6
Secondary high-p	91.1
Secondary low-p	63.8
Clone level	1.0
Ghost level	5.9
MC tracks found	134
Time, ms/ev	8

Efficient and stable event reconstruction

CA Track Finder: Efficiency





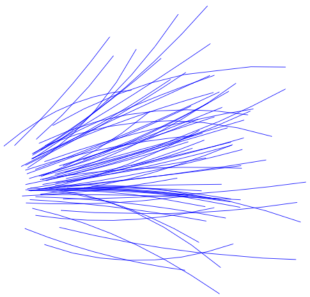


Track category	Eff, %
All tracks	90.9
Primary high-p	97.5
Primary low-p	92.6
Secondary high-p	91.1
Secondary low-p	63.8
Clone level	0.4
Ghost level	5.9
MC tracks found	134
Time, ms/ev	10

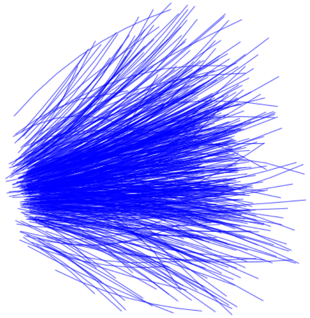
Efficient and stable event reconstruction

CA Track Finder at High Track Multiplicity

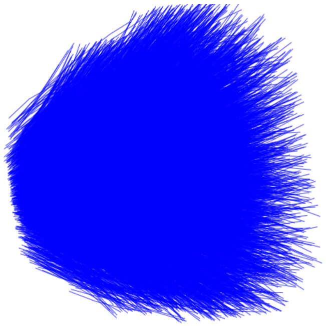
A number of minimum bias events is gathered into a group (super-event), which is then treated by the CA track finder as a single event



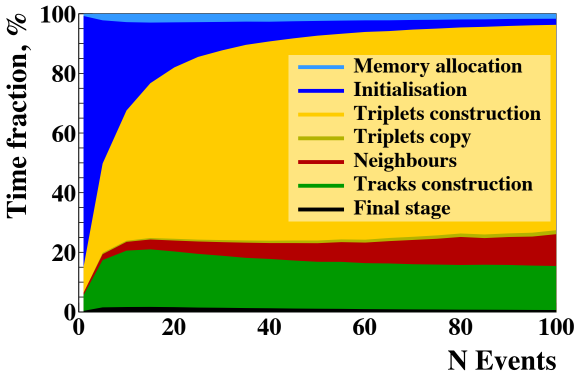
1 mbias event, $\langle N_{reco} \rangle$ = 109

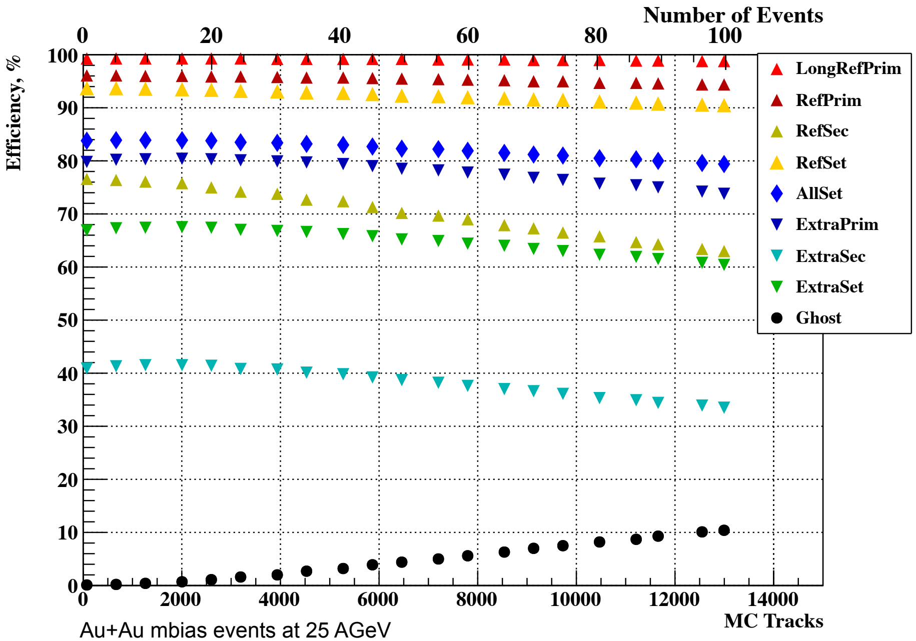


5 mbias events, $\langle N_{reco} \rangle$ = 572

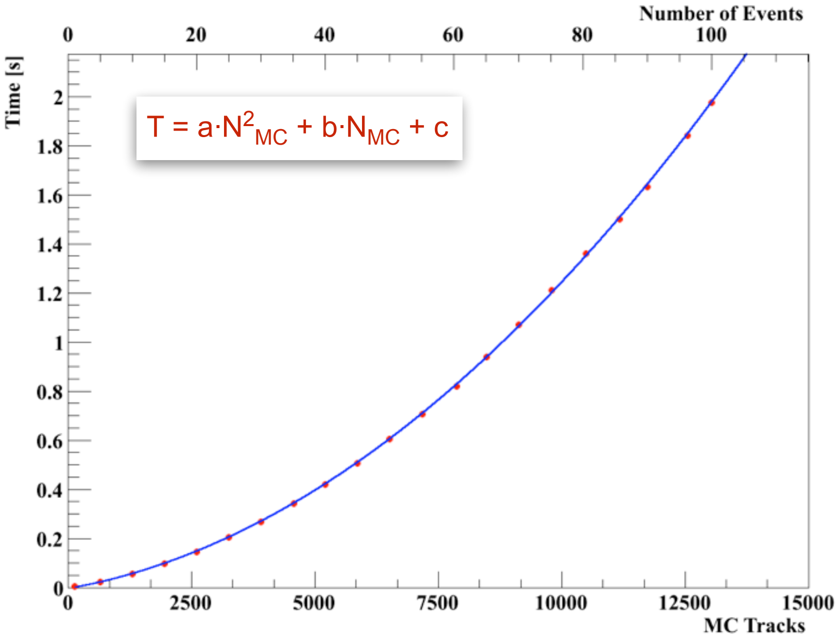


100 mbias events, $\langle N_{reco} \rangle$ = 10340



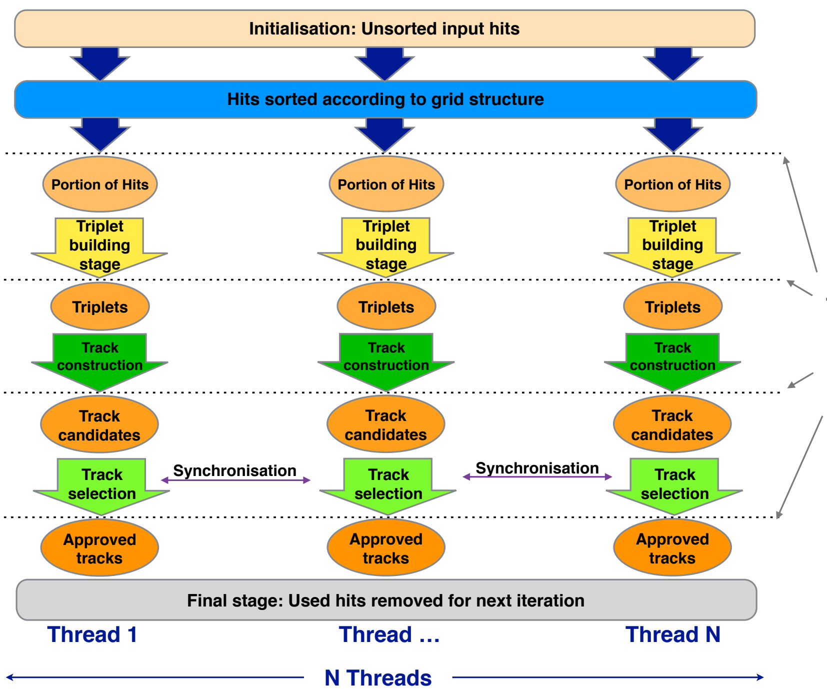


Au+Au mbias events at 25 AGeV



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

Parallelization and 4D Pseudocode



Pseudocode for CBM CA Track Finder

```
Sort_Input_Hits_According_to_Grid();

for track_set (high_p_primary, low_p_primary, secondary, broken)

switch (track_set)
  case high_p_primary:
    Build_Triplets (min_momentum_for_fast_tracks,
    primary_track_parameter_initilisation, triplets_wo_gaps);

  case low_p_primary:
    Build_Triplets (min_momentum_for_slow_tracks,
    primary_track_parameter_initilisation, triplets_wo_gaps);

  case secondary:
    Build_Triplets (min_momentum_for_slow_tracks,
    secondary_track_parameter_initilisation, triplets_wo_gaps);

  case broken:
    Build_Triplets (min_momentum_for_slow_tracks,
    secondary_track_parameter_initilisation, triplets_with/
    wo_gaps)

  Find_Neighbours();

  for track_length := NStation to 3 do
    for station := FirstStation to NStation do
      for triplets := First_Triplet_Station to
      Last_Triplet_Station do
        track_candidate = Build_Best_Candidate (triplet);

  Save_Candidates(all_track_candidates);

  Delete_Used_Hits();
```

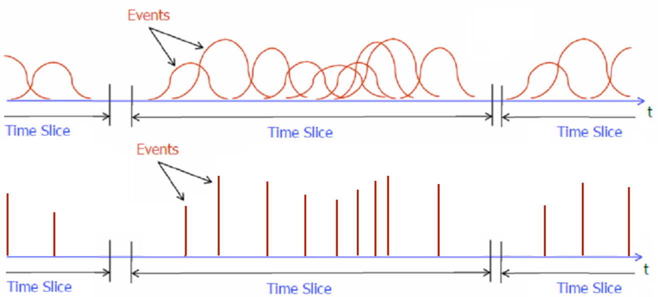
Time-based grid

```
void function Build_Triplets (min_momentum,
prim/sec_track_parameter_initilisation,
triplets_with/wo_gaps)
{
  for station := (NStation-2) to FirstStation do
    for hits_portion := First_Portion_Station to
Last_Portion_Station do

      Find_Singlets(hits_portion);
      Find_Doublets(singlets_in_portion);
      Find_Triplets(doublets_in_portion);
}
```

Cut on hit time

Staged track finding similar to 3D

Time-based (4D) Track Reconstruction with CA Track Finder



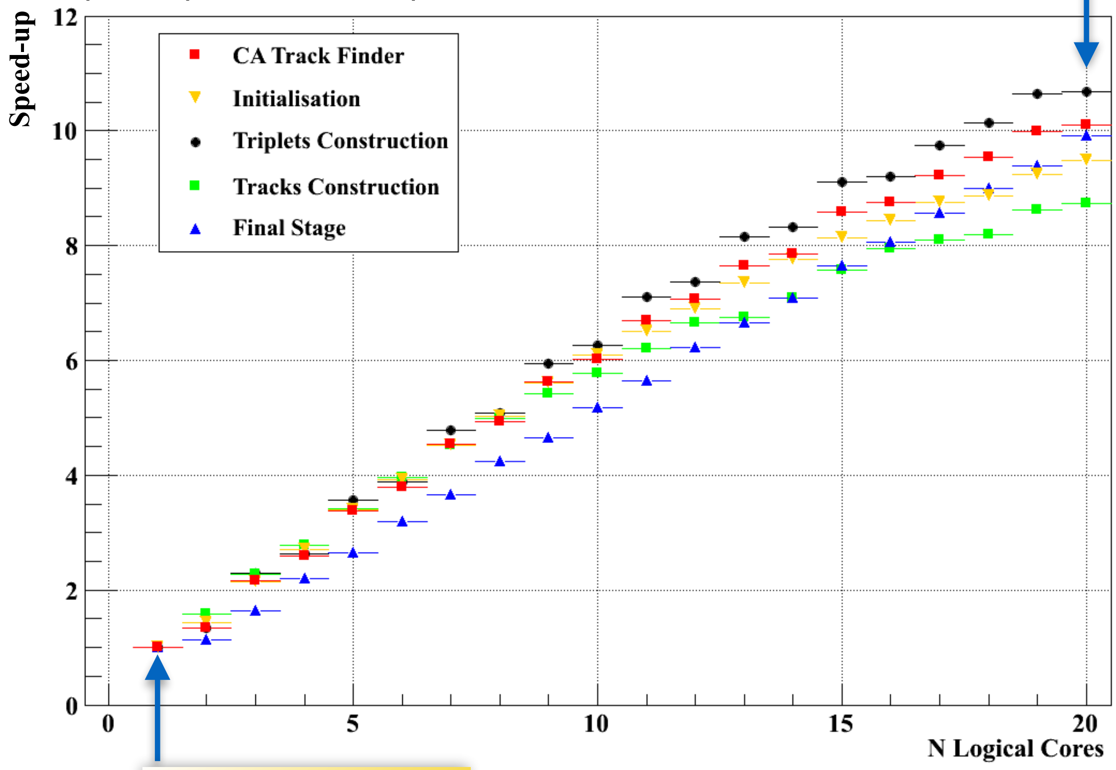
- The beam in the CBM will have no bunch structure, but continuous.
- Measurements in this case will be 4D (x, y, z, t).
- Significant overlapping of events in the detector system.
- Reconstruction of time slices rather than events is needed.

Stage of the algorithm	% of total execution time
Initialisation	8
Triplets construction	64
Tracks construction	15
Final cleaning	13

Efficiency, %	3D	3+1 D	4D
All tracks	83.8	80.4	83.0
Primary high-p	96.1	94.3	92.8
Primary low-p	79.8	76.2	83.1
Secondary high-p	76.6	65.1	73.2
Secondary low-p	40.9	34.9	36.8
Clone level	0.4	2.5	1.7
Ghost level	0.1	8.2	0.3
Time/event/core, ms	8.2	31.5	8.5

Total CA time = 84 ms

Speed-up factor due to parallelization within the time-slice



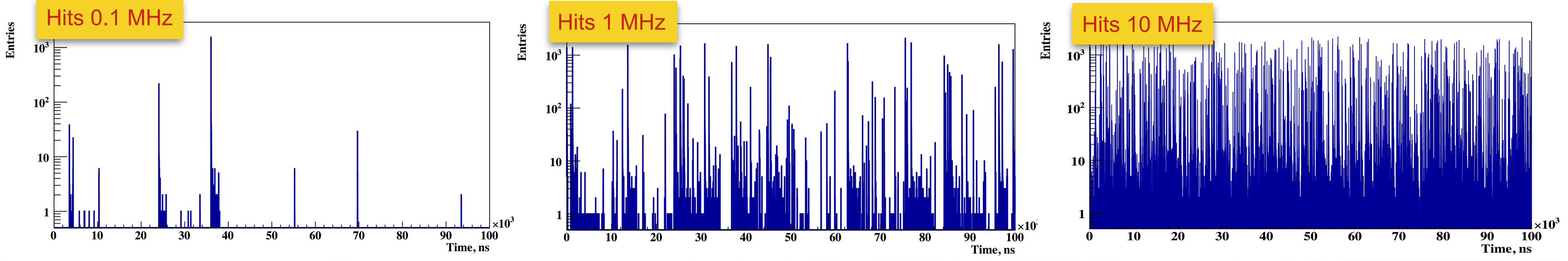
Total CA time = 849 ms

100 mbias events in a time-slice

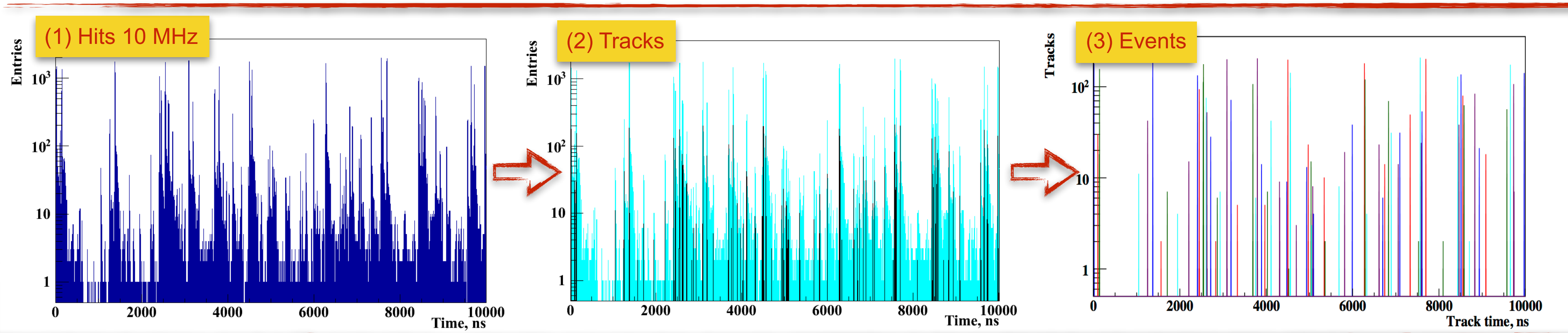
4D event building is scalable with the speed-up factor of 10.1; 3D reconstruction time 8.2 ms/event is recovered in 4D case

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
83% of single events, no splitted events, further analysis with TOF information at the vertexing stage

Summary

- The Kalman Filter track fit library is vectorized, parallelized and portable to CPU/Phi/GPU architectures.
- The Cellular Automaton track finder is vectorized, parallelized and updated for time-based (4D) track finding in time-slices.
- 4D event building is done after all tracks in the time-slice are found.

More details soon:

- V. Akishina, 4D event reconstruction in the CBM experiment, PhD Thesis, Uni-Frankfurt, 2016
- M. Zyzak, Online selection of short-lived particles on many-core computer architectures in the CBM experiment at FAIR, PhD Thesis, Uni-Frankfurt, 2016