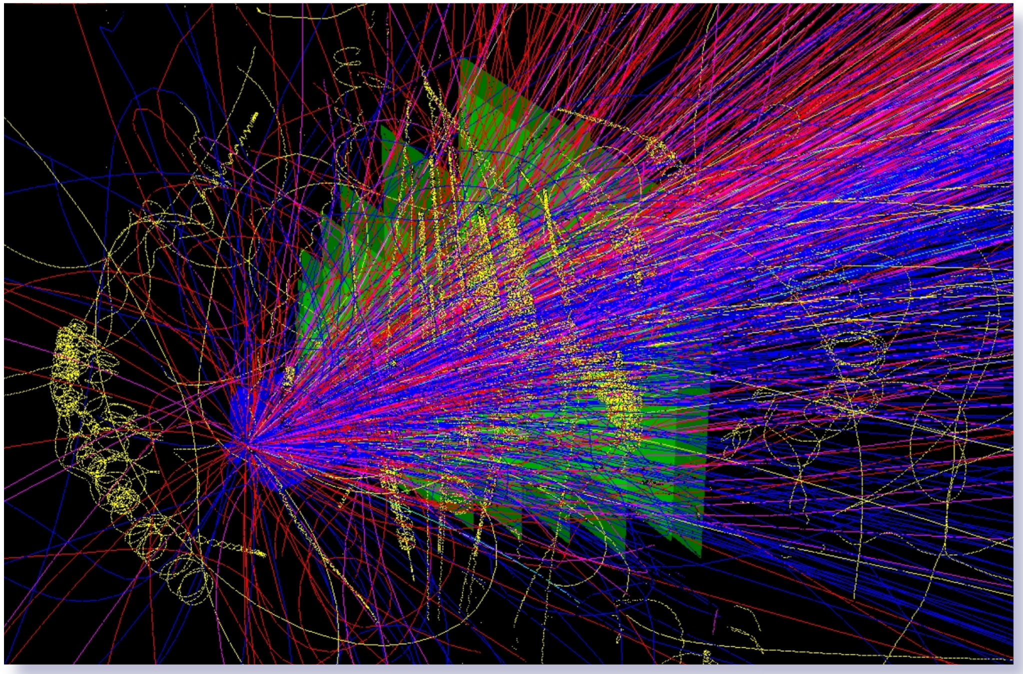
FLES: First Level Event Selection Package for the CBM Experiment

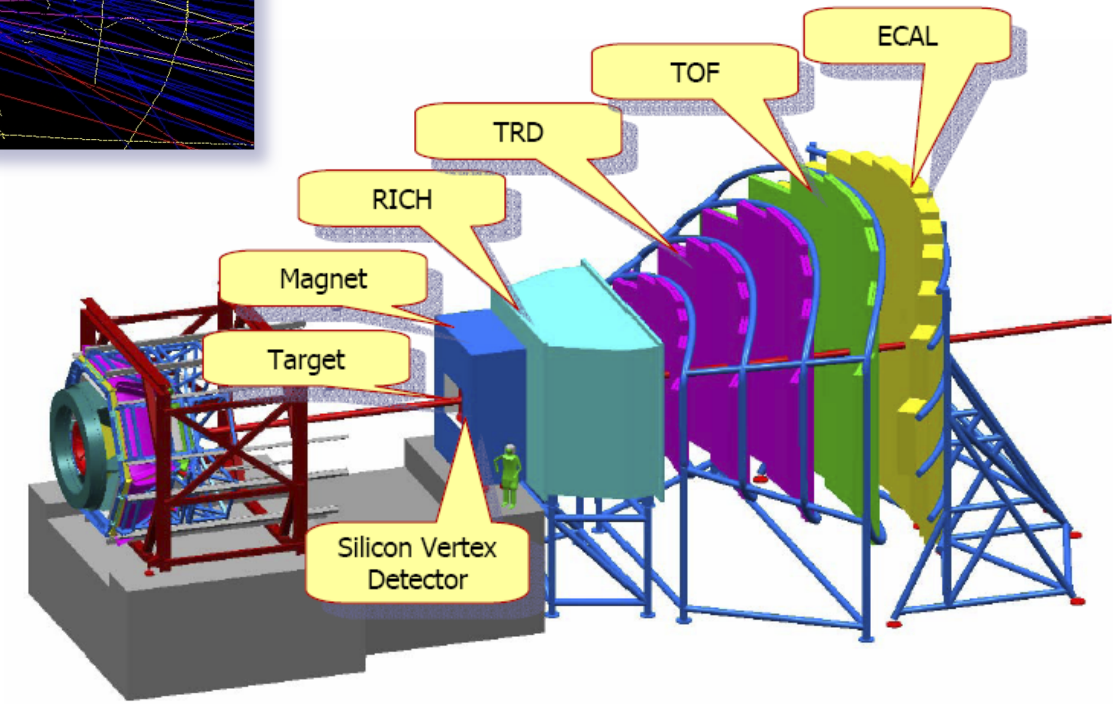
V. Akishina, I. Kisel, I. Kulakov and M. Zyzak

Uni-Frankfurt, FIAS, GSI

Reconstruction Challenge in CBM



* 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)

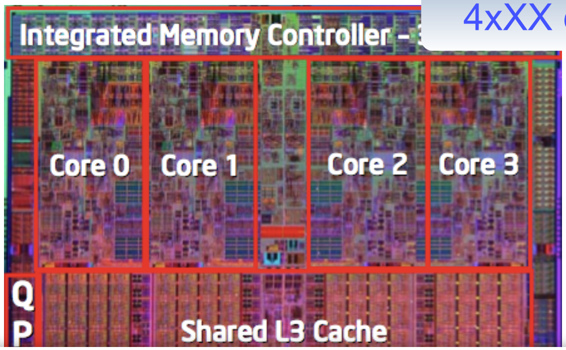


Track reconstruction in STS/MVD
and displaced vertex search
required in the first trigger level

Many-Core CPU/GPU Architectures

Intel/AMD CPU

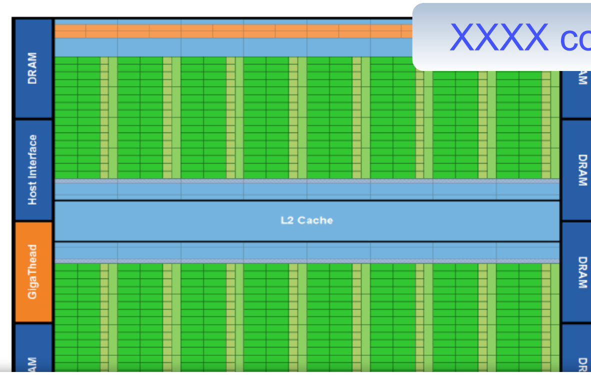
4xXX cores



- Optimized for low-latency access to cached data sets
- Control logic for out-of-order and speculative execution

ATI/NVIDIA GPU

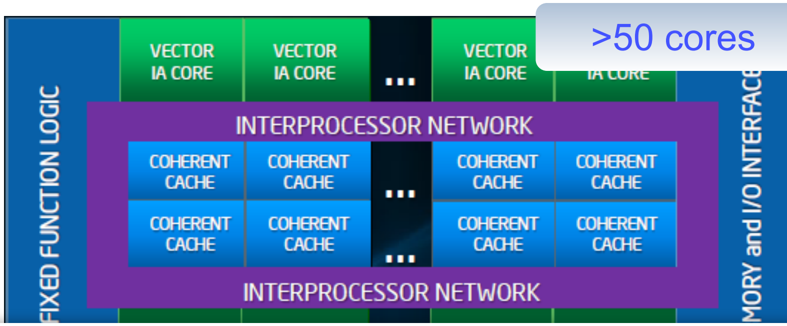
XXXX cores



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

Intel Xeon Phi

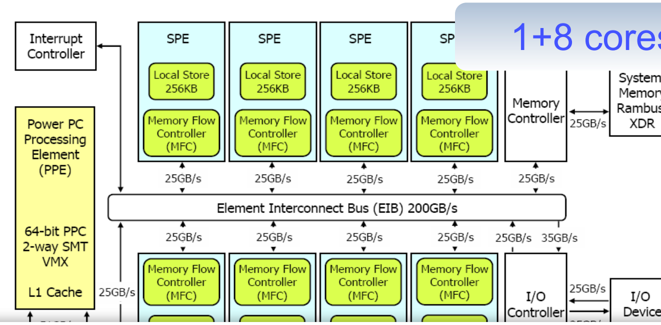
>50 cores



- 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

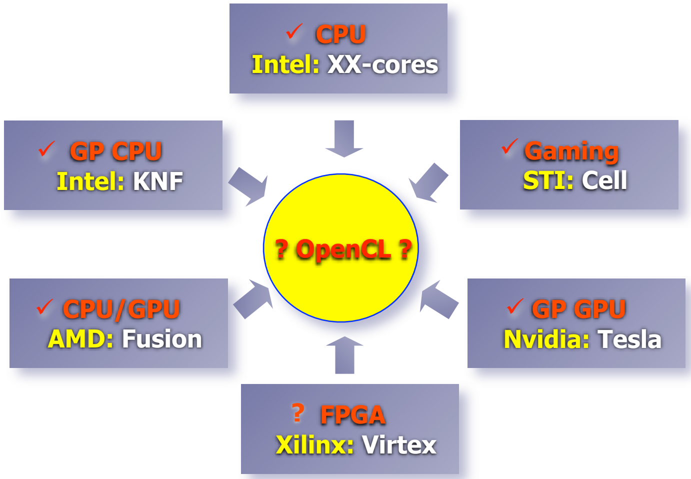
1+8 cores



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

Future systems are heterogeneous

CPU/GPU Programming Frameworks

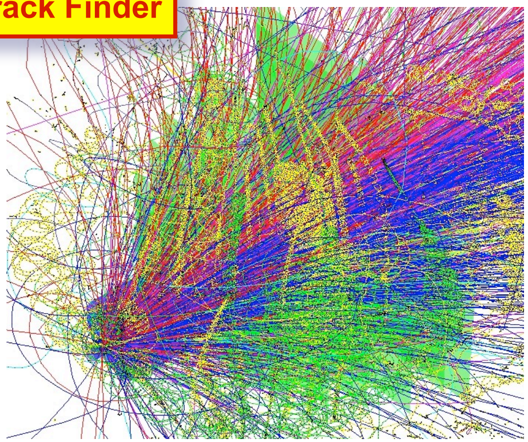


- Intel ArBB (Array Building Blocks)
  - Extension to the C language
  - Intel CPU/GPU specific
  - SIMD exploitation for automatic parallelism
- NVIDIA CUDA (Compute Unified Device Architecture)
  - Defines hardware platform
  - Generic programming
  - Extension to the C language
  - Explicit memory management
  - Programming on thread level
- OpenCL (Open Computing Language)
  - Open standard for generic programming
  - Extension to the C language
  - Supposed to work on any hardware
  - Usage of specific hardware capabilities by extensions
- Vector classes (Vc)
  - Overload of C operators with SIMD/SIMT instructions
  - Uniform approach to all CPU/GPU families
  - Uni-Frankfurt/FIAS/GSI

Choice of CPU/GPU/Programming is a practical question

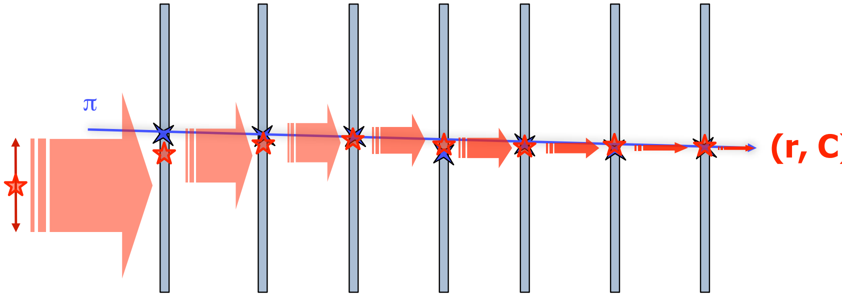
Stages of Event Reconstruction

1 // Track Finder



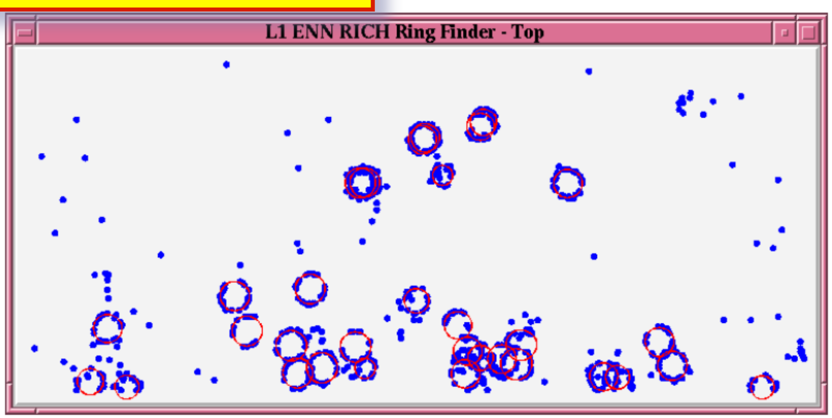
- Cellular Automaton
- Track Following

2 // Track Fitter



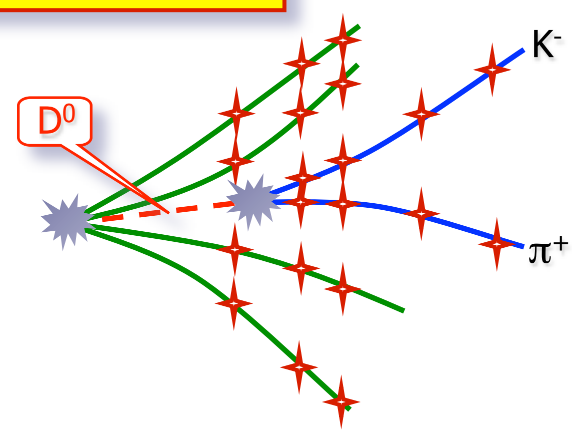
- Kalman Filter

3 // Ring Finder (Particle ID)



- Hough Transformation
- Elastic Neural Net

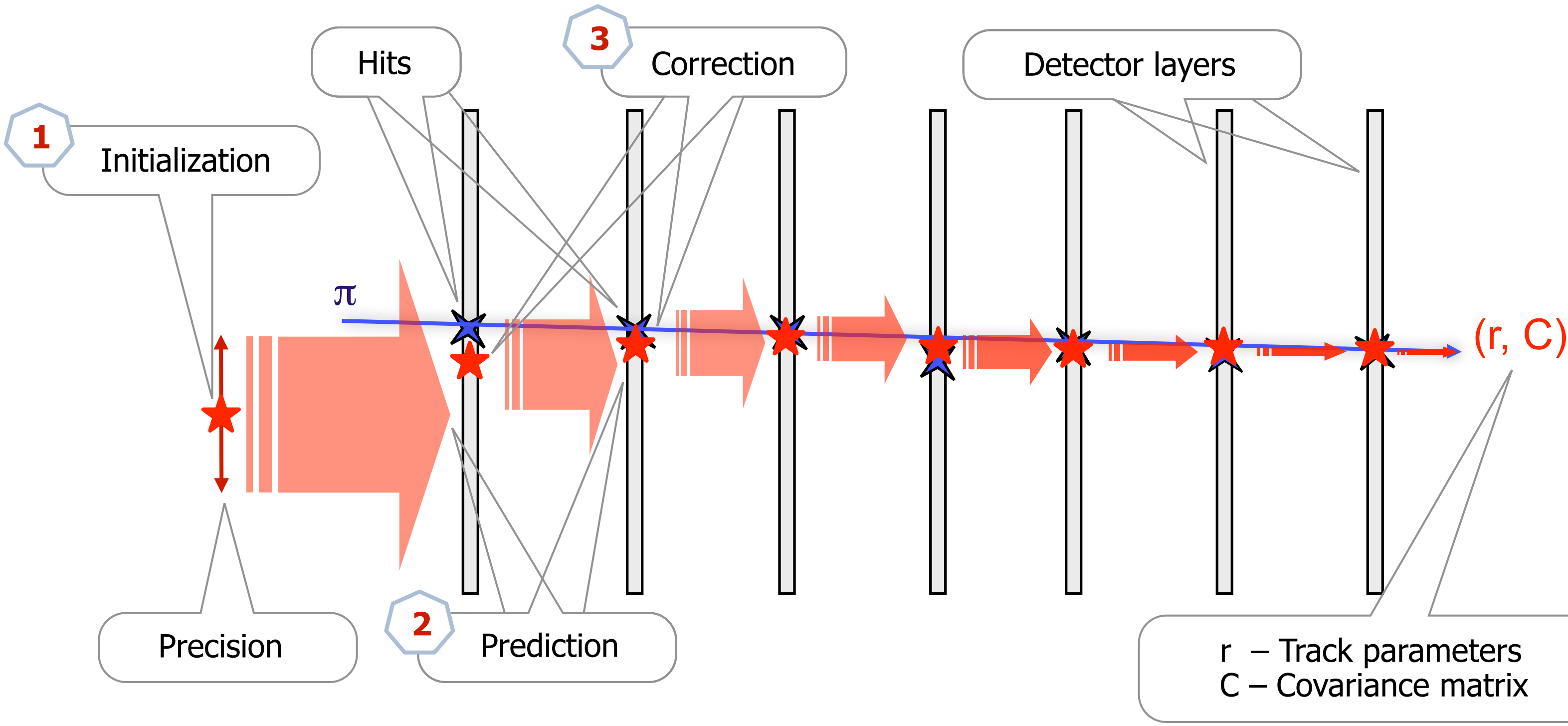
4 // Short-Lived Particles Finder



- Kalman Filter

Kalman Filter based Track Fit

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

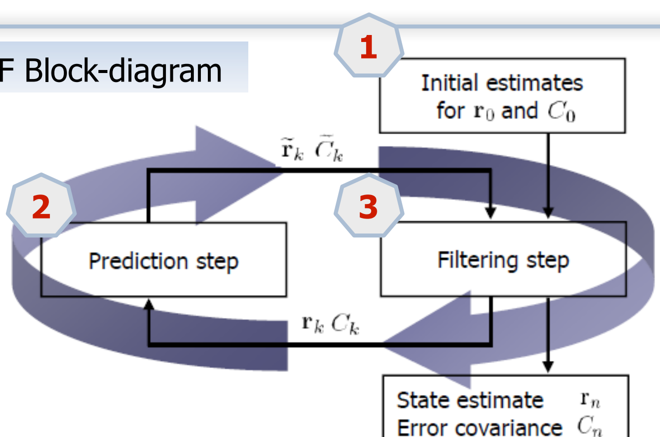


State vector

Position, direction and momentum

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

KF Block-diagram



KF as a recursive least squares method

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

CBM 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

Track Propagation:
- Runge-Kutta
- Analytic Formula

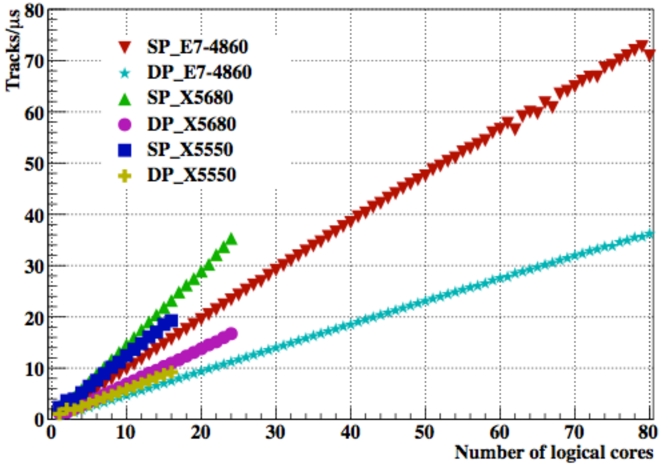
Implementations

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

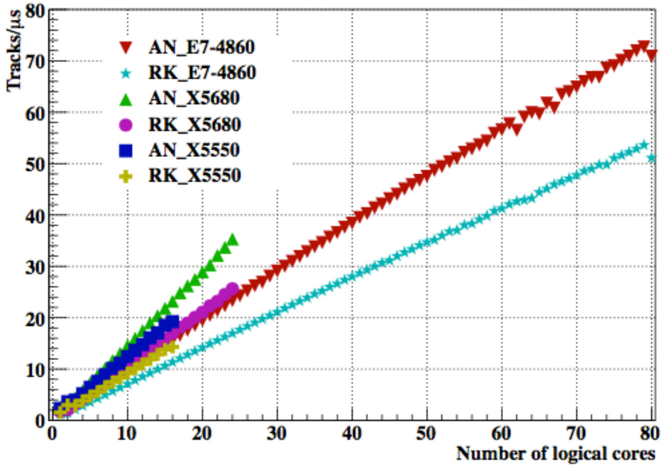
Parallelization (many-cores):
- Open MP
- ITBB
- ArBB
- OpenCL

Precision:
- single
- double

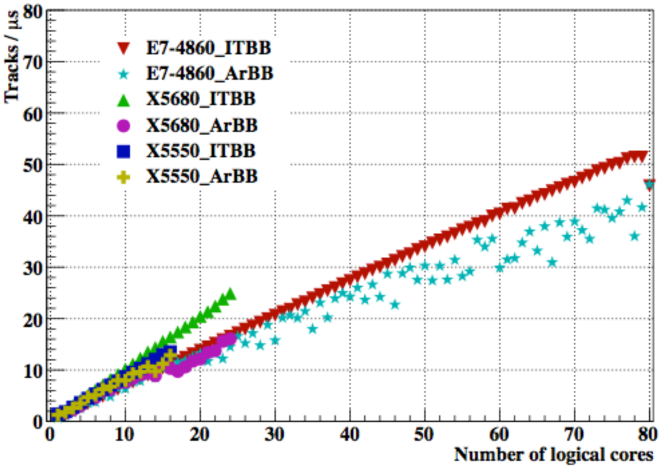
Conventional KF DP vs. SP



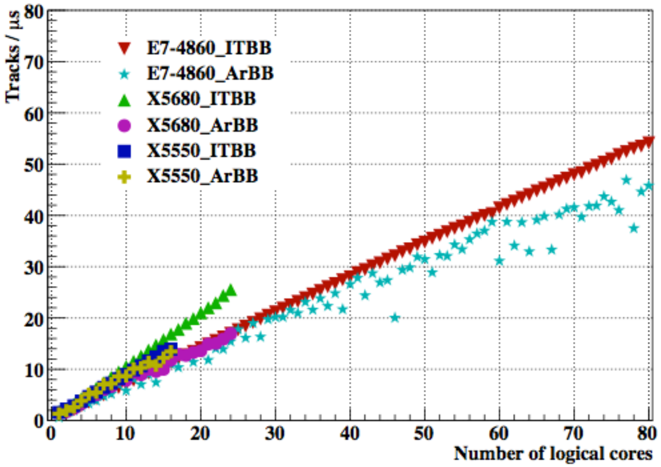
Conventional KF RK4 vs. Analytical



Square-Root KF

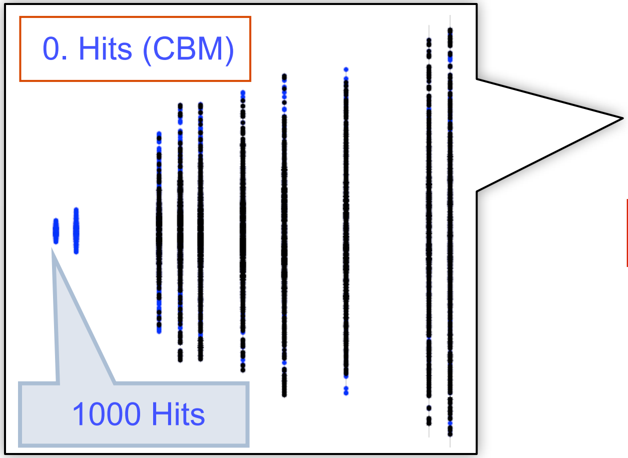


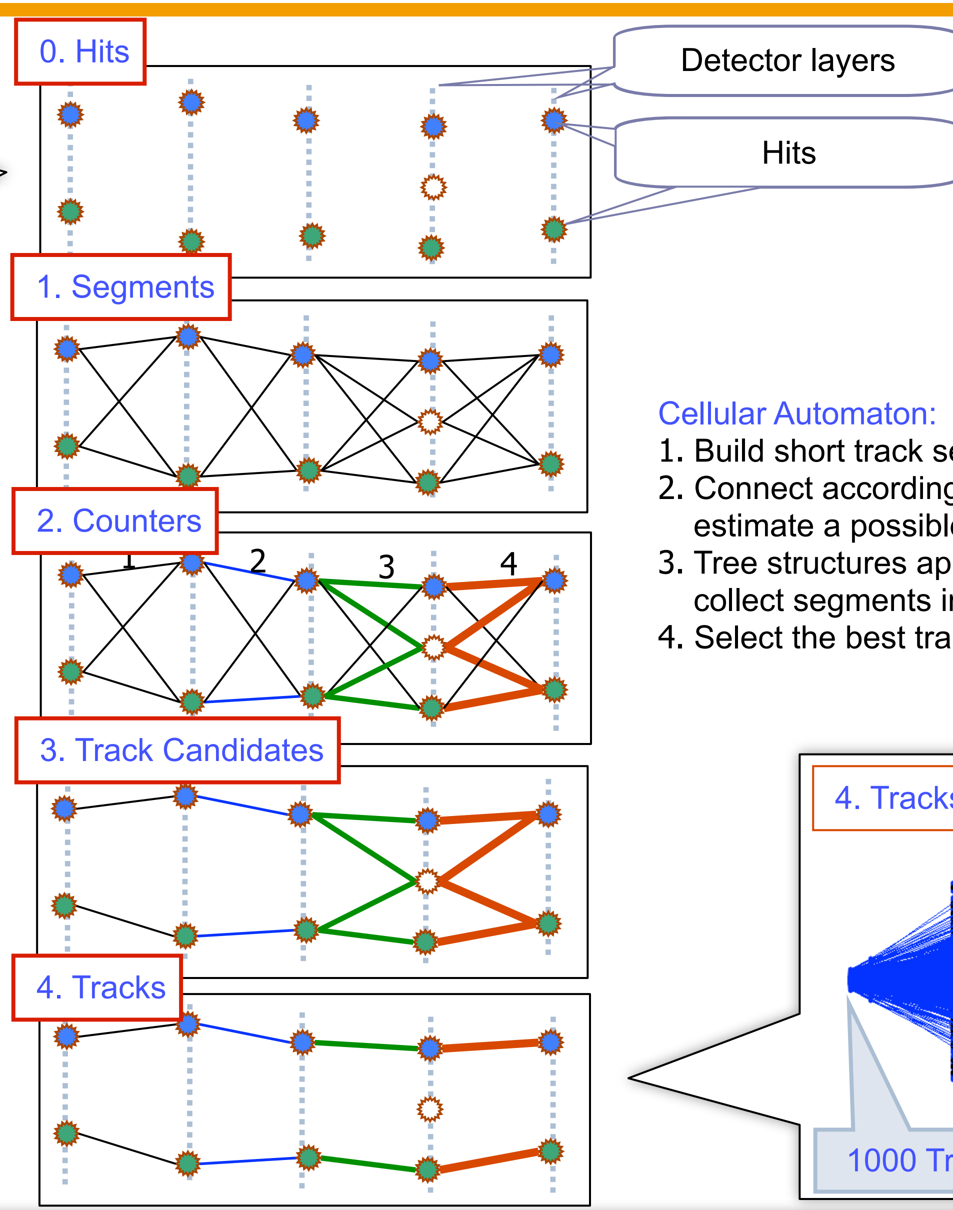
UD KF



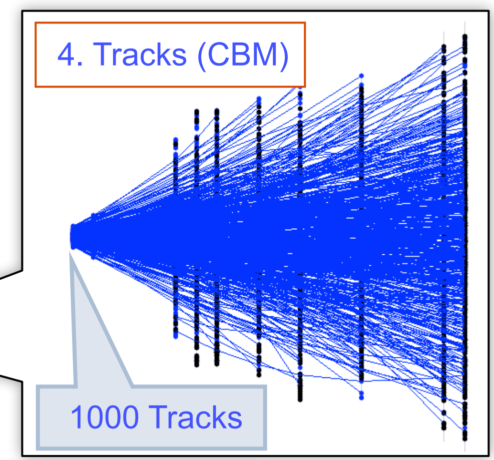
Strong many-core scalability of the Kalman filter library

Cellular Automaton as 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.



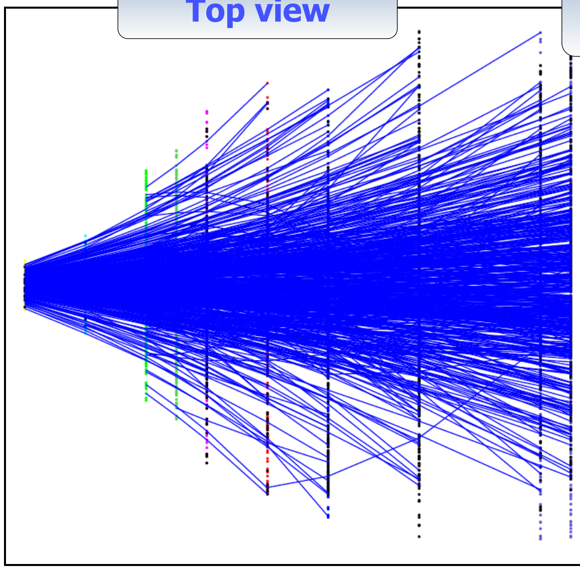
Cellular Automaton:
- local w.r.t. data
- intrinsically parallel
- extremely simple
- very fast

Perfect for many-core CPU/GPU !

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

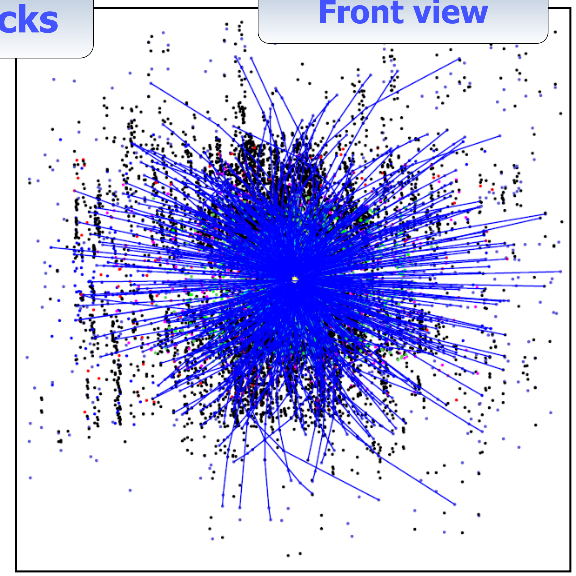
CBM CA Track Finder: Efficiency

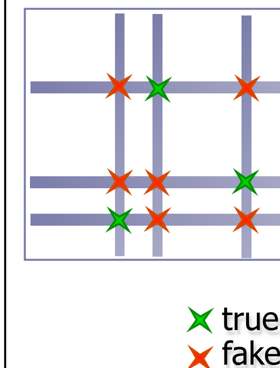
Top view

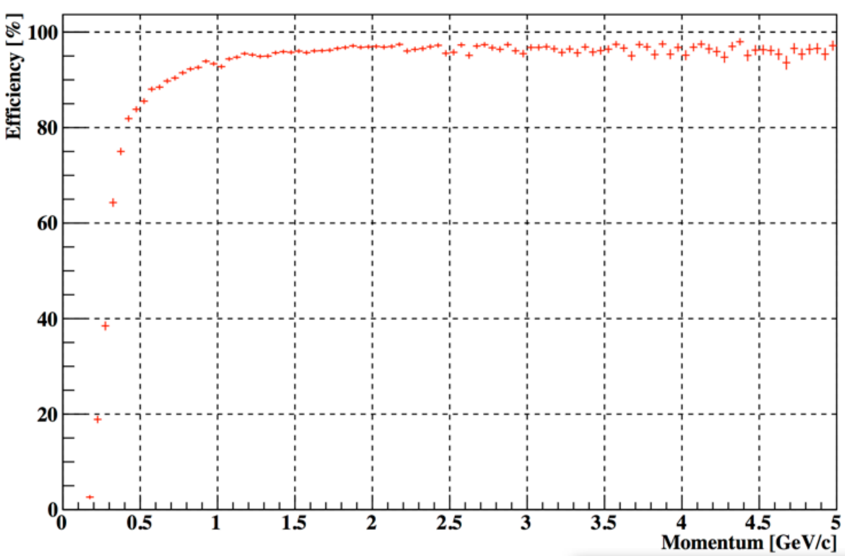


770 Tracks

Front view





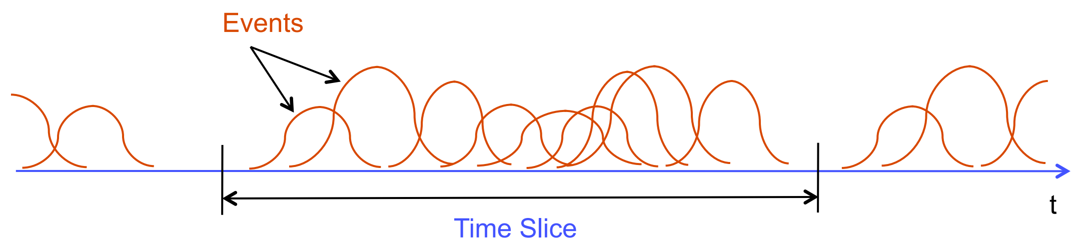


	Efficiency, %	
	mbias	central
Primary high-p tracks	97.1	96.2
Primary low-p tracks	90.4	90.7
Secondary high-p tracks	81.2	81.4
Secondary low-p tracks	51.1	50.6
All tracks	88.5	88.3
Clone level	0.2	0.2
Ghost level	0.7	1.5
Reconstructed tracks/event	120	591
Time/event/core	8.2 ms	57 ms

Efficient and stable event reconstruction

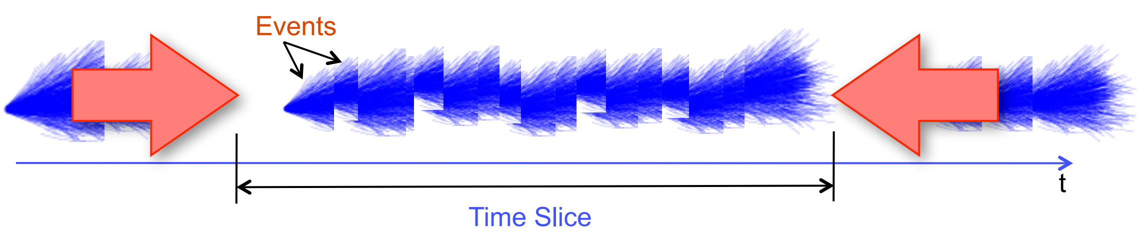
CA Track Finder: Towards 4D Reconstruction

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



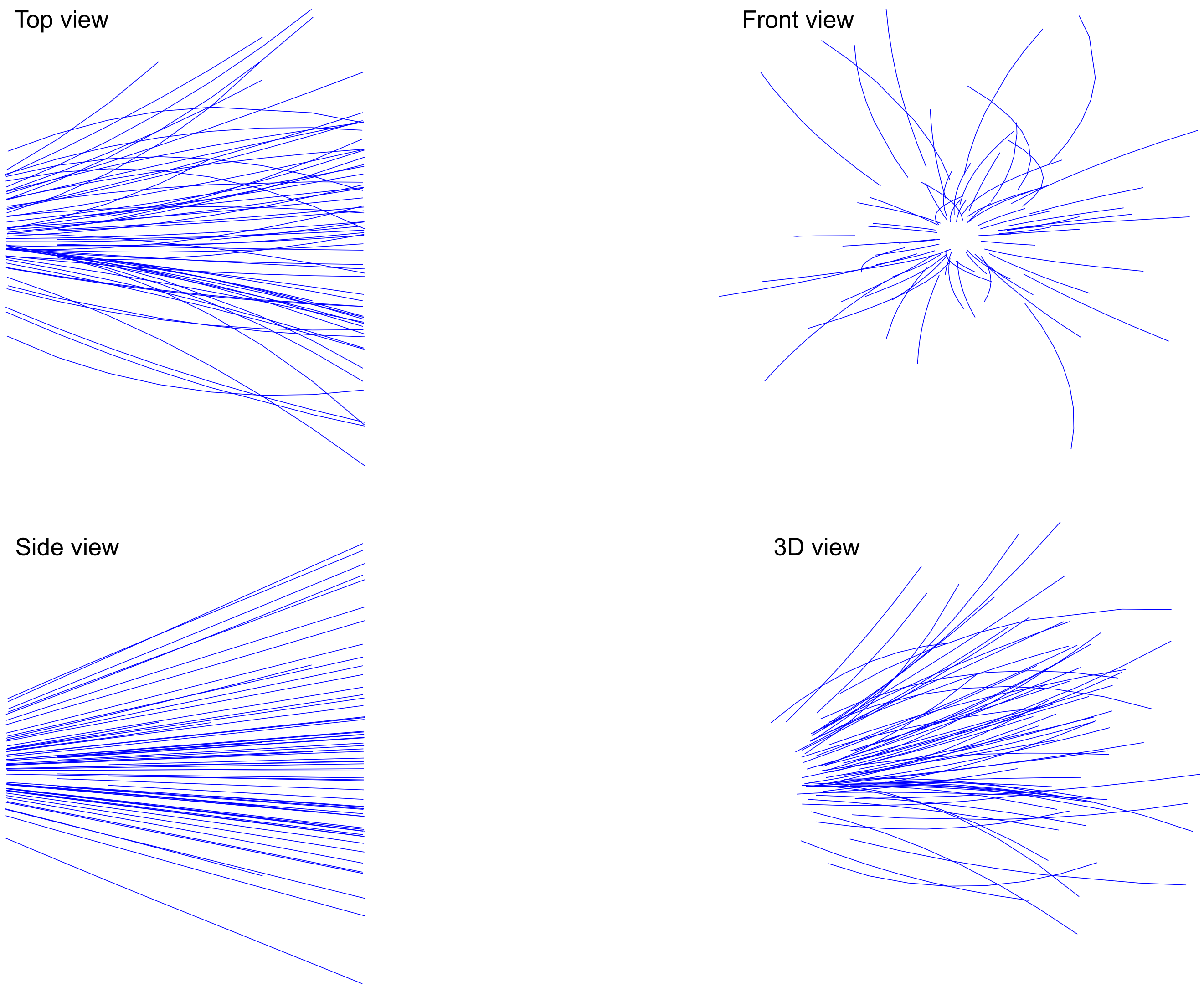
Packed groups of events:

- a number of minimum bias events is gathered into a group, which is then treated by the track finder as one event
- no time measurement is taken into account



A group of minimum bias events is treated as a single event (no time information is used)

Track Finding at Low Track Multiplicity

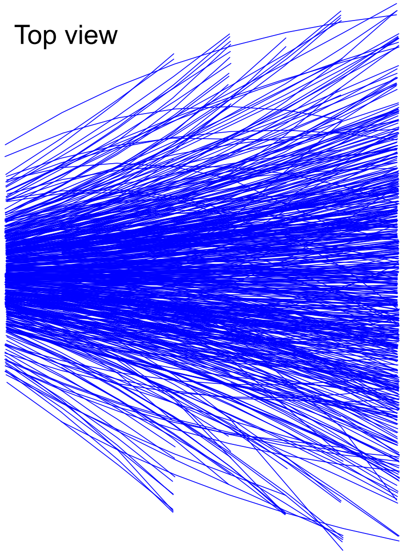


Au+Au mbias events at 25 AGeV, 8 STS, 0 x 7,5 strip angles

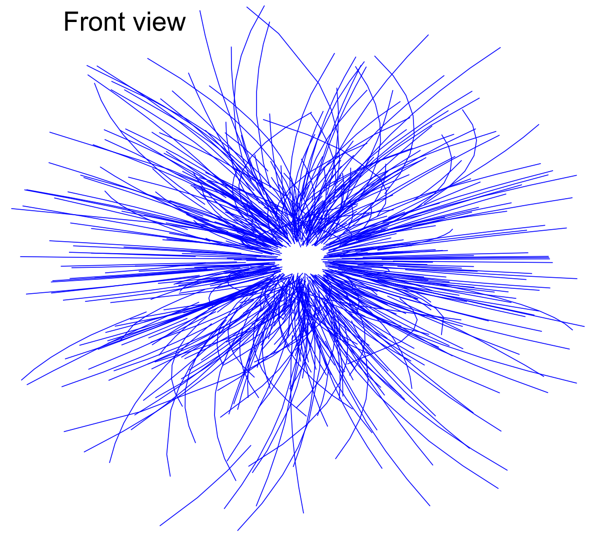
A minimum bias event: average reconstructed track multiplicity 109

Track Finding at Medium Track Multiplicity

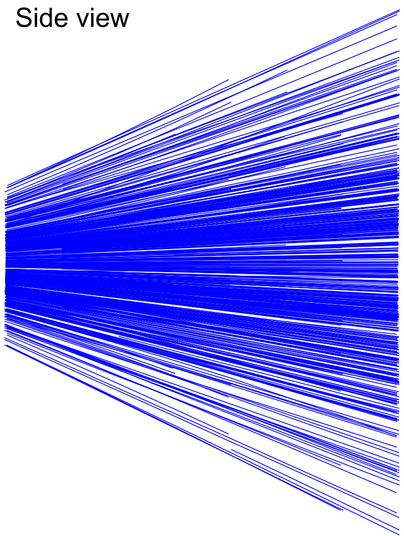
Top view



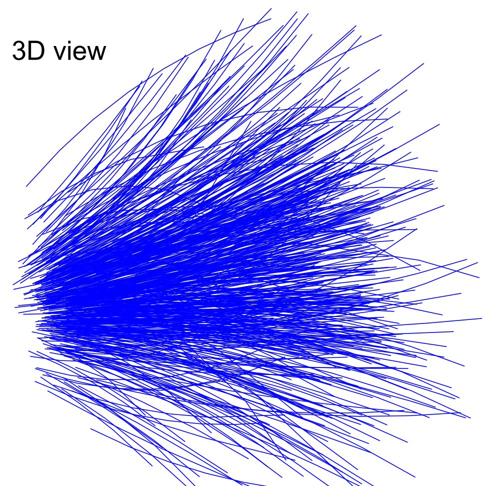
Front view



Side view



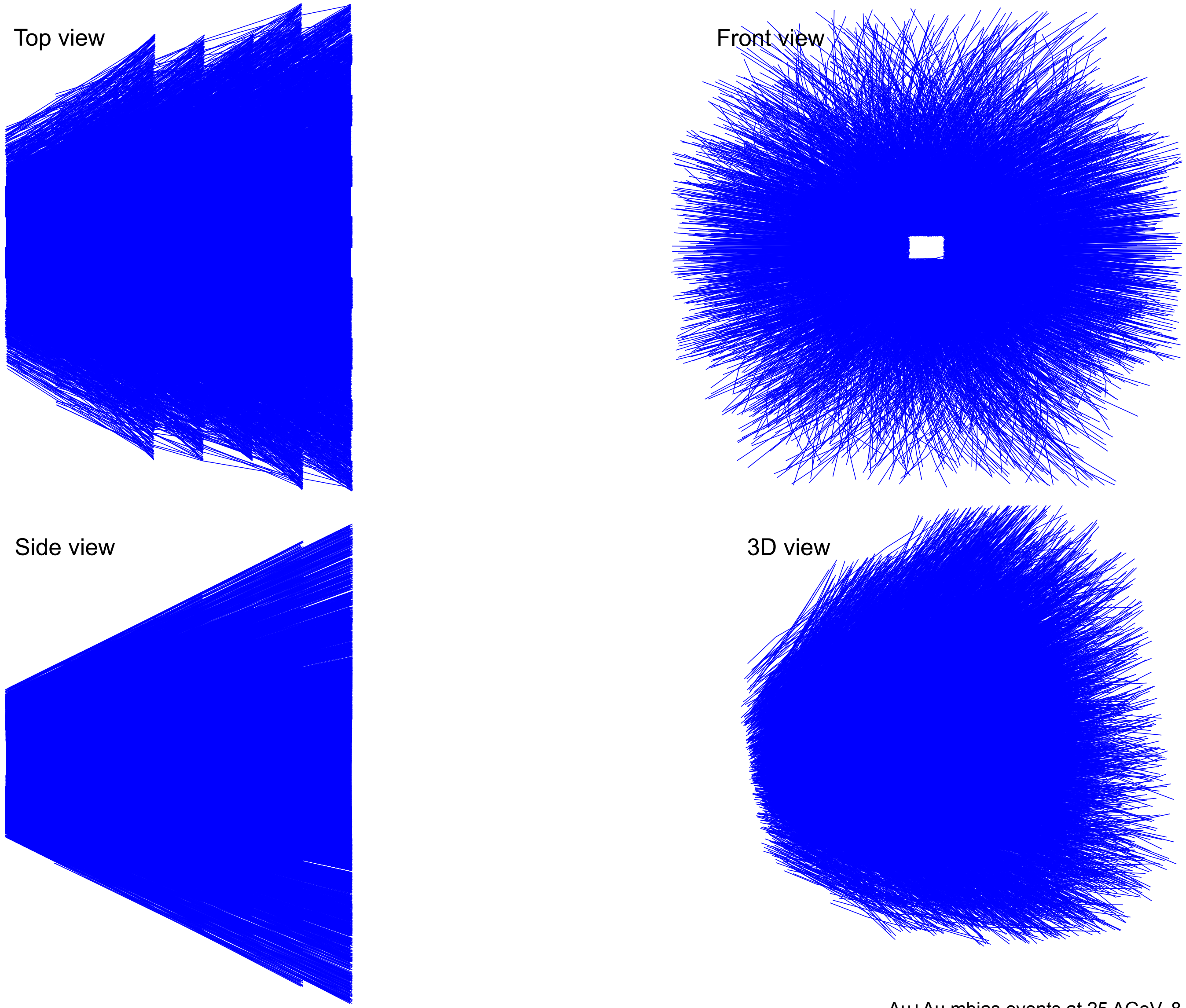
3D view



Au+Au mbias events at 25 AGeV, 8 STS, 0 x 7,5 strip angles

A central event: average reconstructed track multiplicity 572

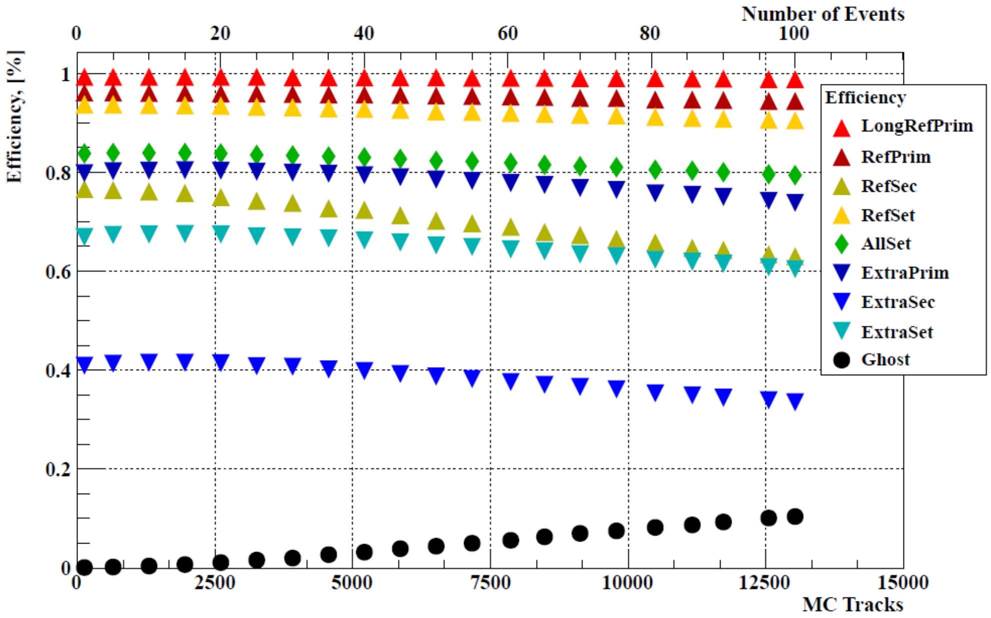
Track Finding at High Track Multiplicity

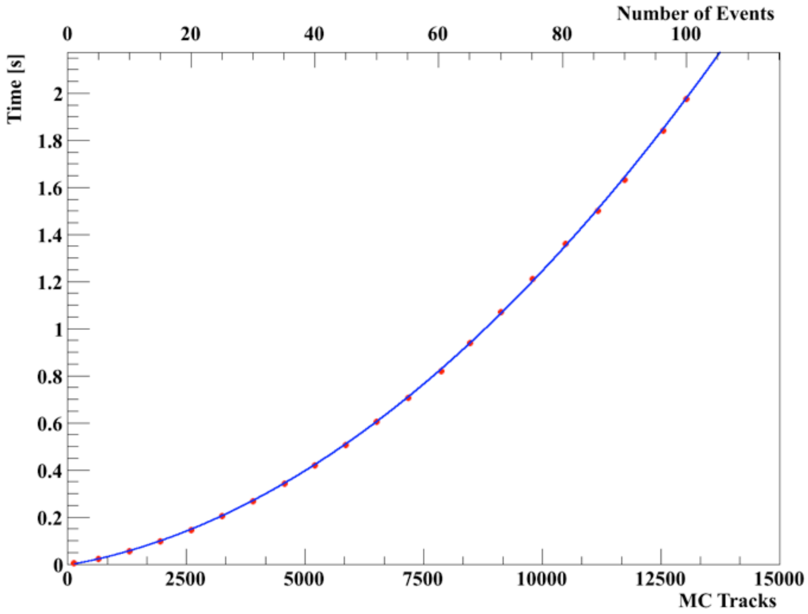


Au+Au mbias events at 25 AGeV, 8 STS, 0 x 7,5 strip angles

A group with 100 minimum bias events: average reconstructed track multiplicity 10340

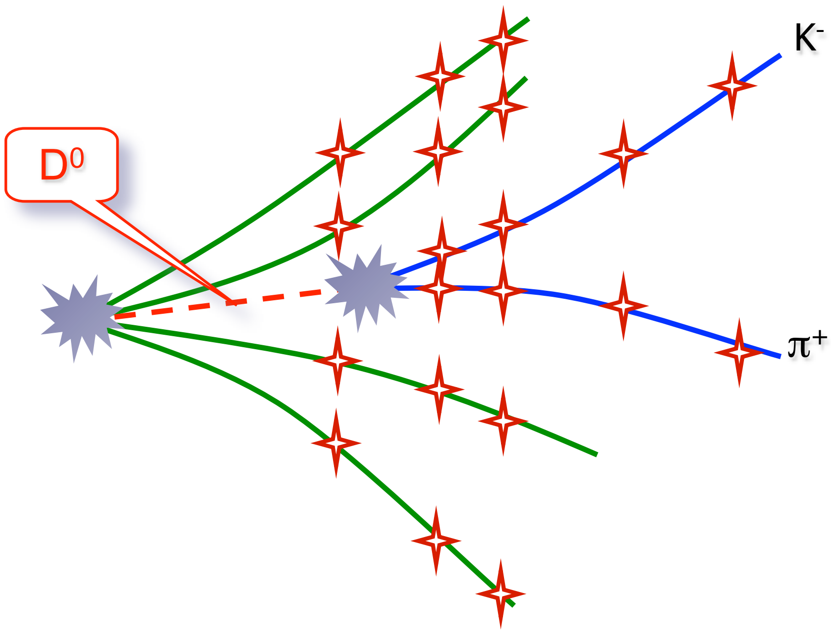
CA Track Finder: Efficiency and Time vs. Track Multiplicity





Stable reconstruction efficiency and time as a second order polynomial up to 100 minimum bias events in a group

KFParticle: Reconstruction of Vertices and Decayed Particles



State vector

Position, direction, momentum and energy

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

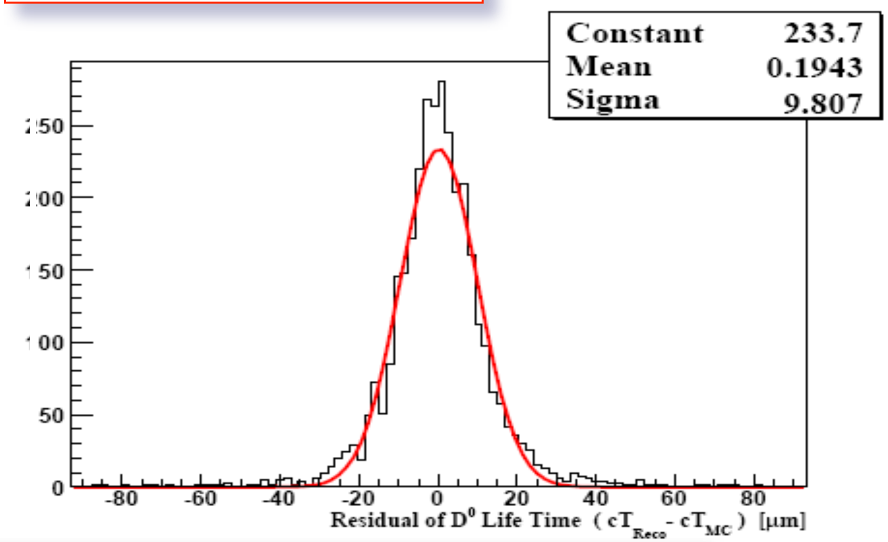
- Mother and daughter particles have the same state vector and are treated in the same way
- Geometry independent
- Kalman filter based

$x, y, z, p_x, p_y, p_z, E, m, L, c\tau$

```
AliKFVertex PrimVtx( ESDPrimVtx );  // Set primary vertex
                                    // Set daughters
AliKFParticle K( ESDp1, -321 ), pi( ESDp2, 211 );

AliKFParticle  D0( K, pi );              // Construct mother
PrimVtx += D0;                           // Improve the primary vertex

D0.SetProductionVertex( PrimVtx ); // D0 is fully fitted
K.SetProductionVertex( D0 );         // K is fully fitted
pi.SetProductionVertex( D0 );        // pi is fully fitted
```



KFParticle provides uncomplicated approach to physics analysis (used in CBM, ALICE and STAR)

KF Particle Finder for Physics Analysis and Selection

(mbias: 1.4 ms; central: 10.5 ms)/event/core

Tracks: $e^\pm$, $\mu^\pm$, $\pi^\pm$, $K^\pm$, $p^\pm$
secondary and primary

Open-charm:
$D^0 \rightarrow \pi^+ K^-$
$D^0 \rightarrow \pi^+ \pi^+ \pi^- K^-$
$\bar{D}^0 \rightarrow \pi^- K^+$
$\bar{D}^0 \rightarrow \pi^- \pi^- \pi^+ K^+$
$D^+ \rightarrow \pi^+ \pi^+ K^-$
$D^- \rightarrow \pi^- \pi^- K^+$
$D_s^+ \rightarrow \pi^+ K^+ K^-$
$D_s^- \rightarrow \pi^- K^+ K^-$
$\Lambda_c \rightarrow \pi^+ K^- p$

Strange particles:
$K^0_s \rightarrow \pi^+ \pi^-$
$\Lambda \rightarrow p\, \pi^-$
$\bar{\Lambda} \rightarrow \pi^+ p^-$

Gamma:
$\gamma \rightarrow e^- e^+$

Strange resonances:
$\bar{K}^{*0} \rightarrow K^+ \pi^-$
$K^{*0} \rightarrow \pi^+ K^-$
$\bar{\Lambda}^* \rightarrow p\, K^-$
$\Lambda^* \rightarrow p^- K^+$

Light vector mesons:
$\rho \rightarrow e^- e^+$
$\rho \rightarrow \mu^- \mu^+$
$\omega \rightarrow e^- e^+$
$\omega \rightarrow \mu^- \mu^+$
$\varphi \rightarrow e^- e^+$
$\varphi \rightarrow \mu^- \mu^+$
$\varphi \rightarrow K^- K^+$

Charmonium:
$J/\Psi \rightarrow e^- e^+$
$J/\Psi \rightarrow \mu^- \mu^+$

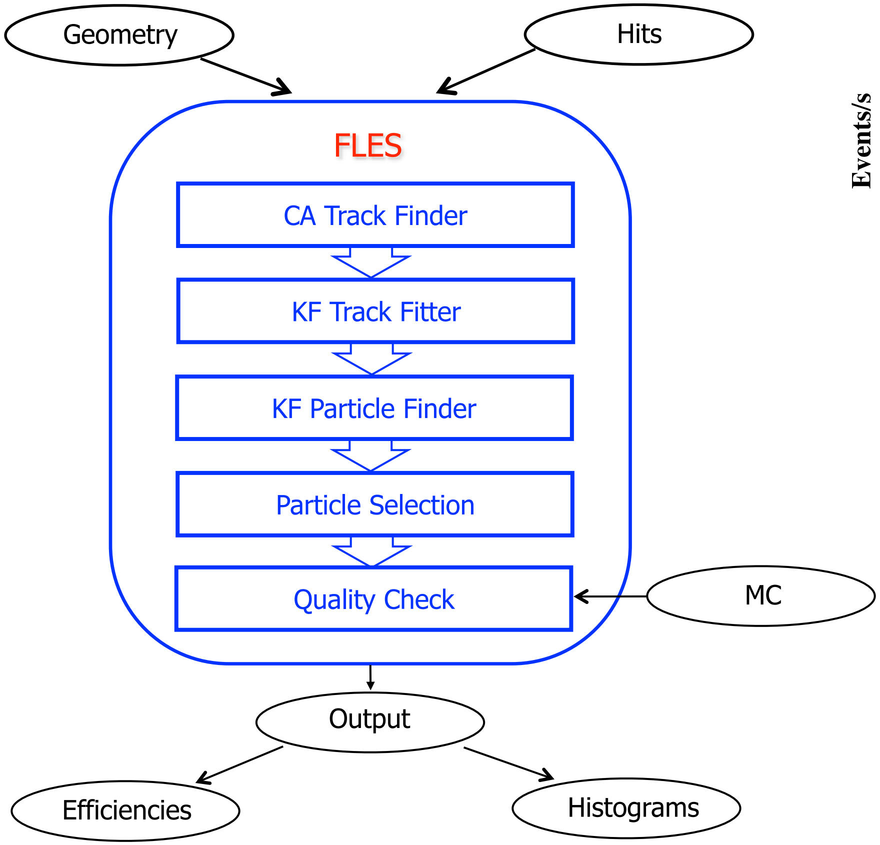
Multi-strange hyperons:
$\Xi^- \rightarrow \Lambda\, \pi^-$
$\Xi^+ \rightarrow \bar{\Lambda}\, \pi^+$
$\Omega^- \rightarrow \Lambda\, K^-$
$\Omega^+ \rightarrow \bar{\Lambda}\, K^+$

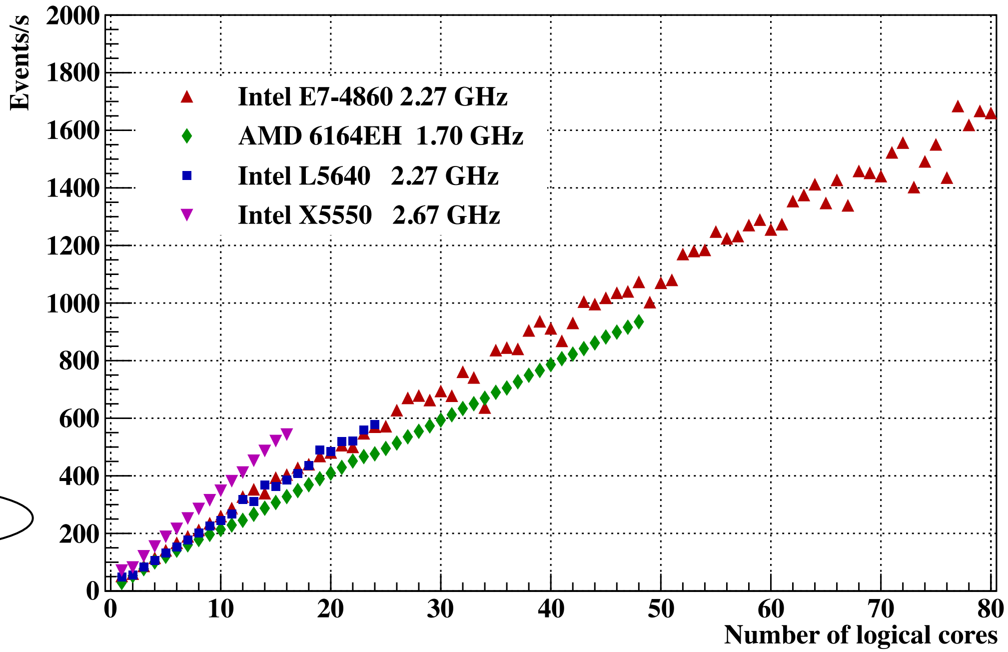
Strange and multi-strange resonances:
$\Sigma^{*+} \rightarrow \Lambda\, \pi^+$
$\bar{\Sigma}^{*+} \rightarrow \bar{\Lambda}\, \pi^-$
$\Sigma^{*-} \rightarrow \Lambda\, \pi^-$
$\bar{\Sigma}^{*-} \rightarrow \bar{\Lambda}\, \pi^+$
$K^{*-} \rightarrow K^0_s\, \pi^-$
$K^{*+} \rightarrow K^0_s\, \pi^+$
$\Xi^{*-} \rightarrow \Lambda\, K^-$
$\Xi^{*+} \rightarrow \bar{\Lambda}\, K^+$

Open-charm resonances:
$D^{*0} \rightarrow D^+ \pi^-$
$\bar{D}^{*0} \rightarrow D^- \pi^+$
$D^{*+} \rightarrow D^0 \pi^+$
$D^{*-} \rightarrow \bar{D}^0 \pi^-$

Multi-strange resonances:
$\Xi^{*0} \rightarrow \Xi^- \pi^+$
$\bar{\Xi}^{*0} \rightarrow \Xi^+ \pi^-$
$\Omega^{*-} \rightarrow \Xi^- \pi^+ K^-$
$\Omega^{*+} \rightarrow \Xi^+ \pi^- K^+$

CBM Standalone First Level Event Selection (FLES) Package





Given n threads each filled with 1000 events, run them on specified n cores, thread/core.

The first version of the FLES package is vectorized, parallelized, portable and scalable

Summary and Plans

• A first version of the FLES package for CBM has been developed

• Add other sub-detectors (with less combinatorics)
• Include time information (4D time-slices)
• Investigate scalability on a farm