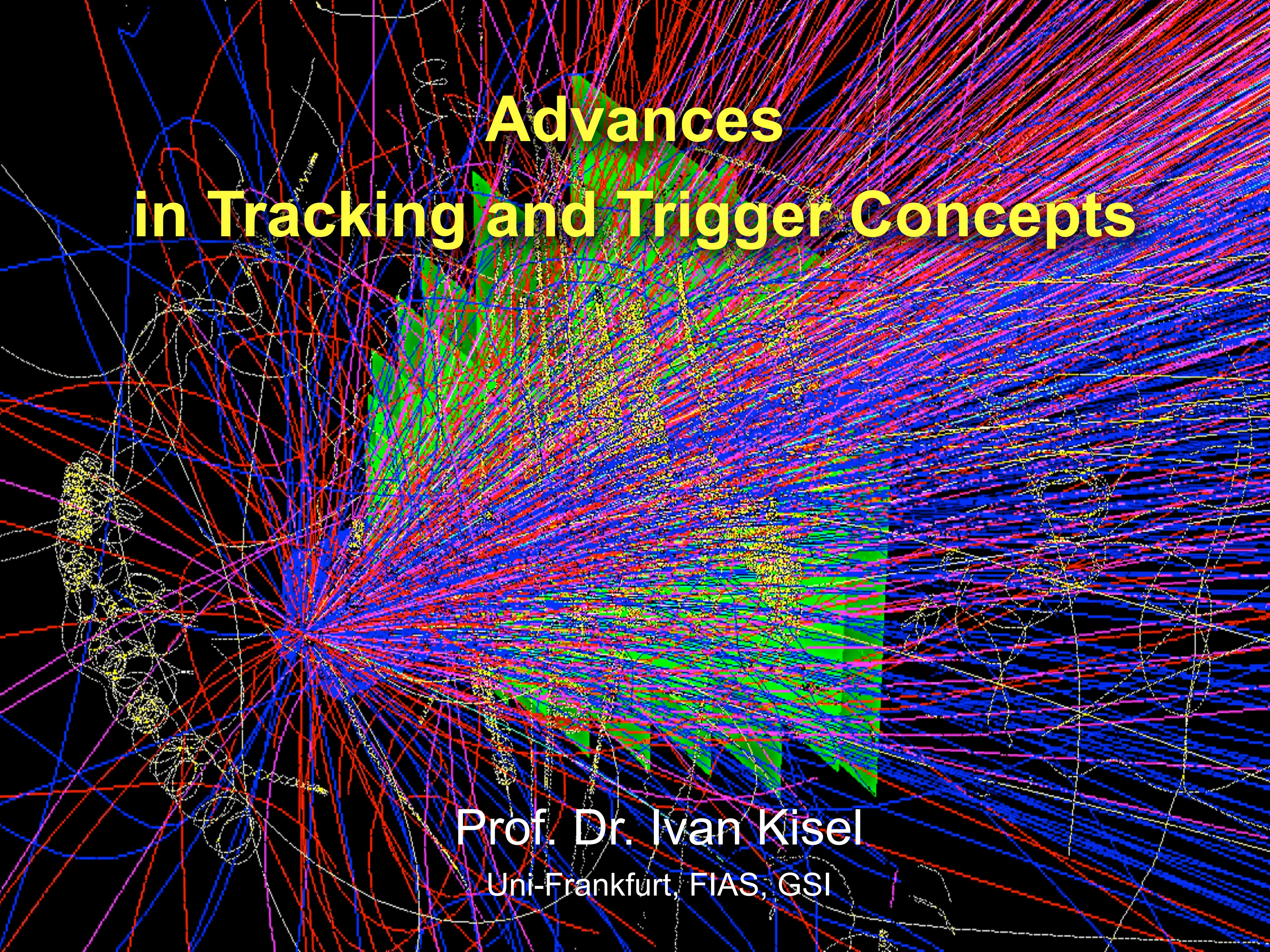


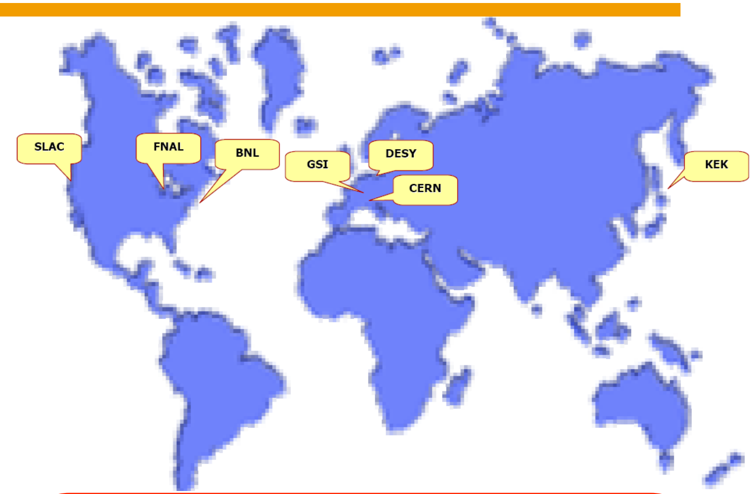
Advances in Tracking and Trigger Concepts

Prof. Dr. Ivan Kisel

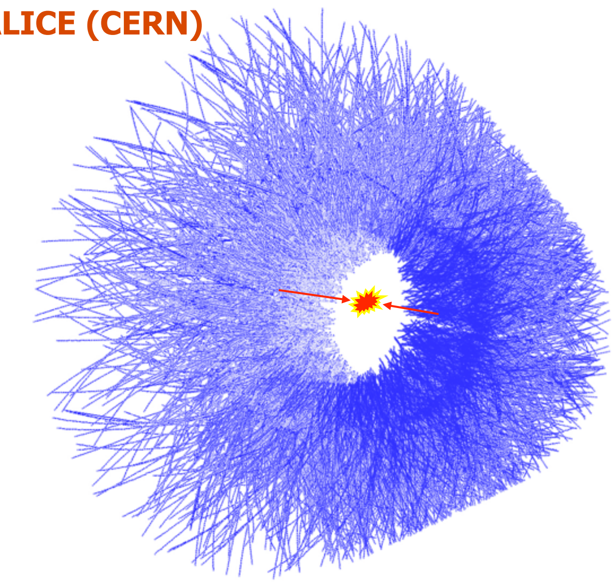
Uni-Frankfurt, FIAS, GSI

HEP Research Centers

Research Center	Accelerator (GeV)	Experiment	Physics
SLAC, USA	PEP-II, e^- x e^+ (9 x 3.1)	BaBar	B-Physics
Fermilab, USA	Tevatron, p x p (1000 x 1000)	D0	Universal
		CDF	Universal
BNL, USA	RHIC, Heavy Ions	PHENIX	Quark-Gluon-Plasma
		STAR	Quark-Gluon-Plasma
KEK, Japan	KEK-B, e^- x e^+ (8 x 3.5)	BELLE	B-Physics
CERN, Switzerland	LHC, p x p (7000 x 7000)	ATLAS	Universal
		CMS	Universal
		ALICE	Quark-Gluon-Plasma
		LHCb	B-Physics
DESY, Germany	HERA, $e^{+/-}$ x p (27.5 x 920)	ZEUS	Proton-Physics
		H1	Proton-Physics
		HERMES	Spin-Physics
		HERA-B	B-Physics
FAIR/GSI, Germany	SIS 100/300, p, Heavy Ions	PANDA	Quark-Physics
		CBM	Quark-Gluon-Plasma

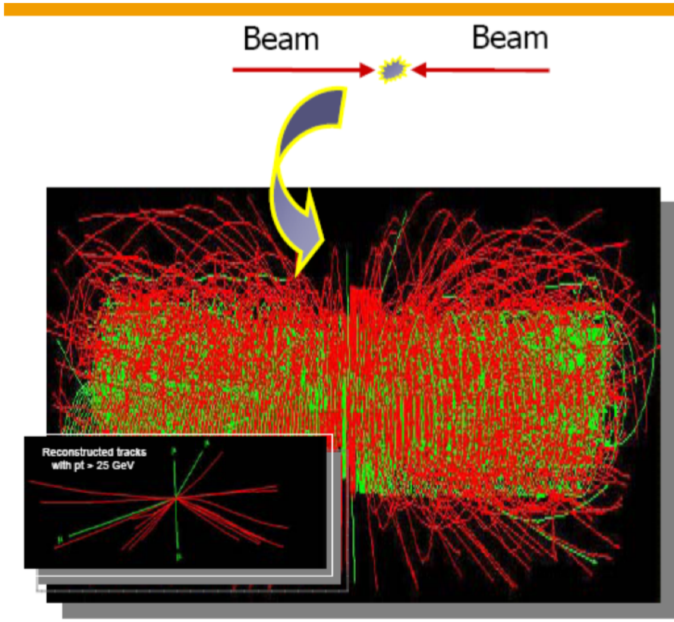


ALICE (CERN)



- 5000 charged particles/collision
- 2000 proton-proton collisions/second
- 300 heavy ion collisions/second
- 15 GB/second data flow (TPC only)

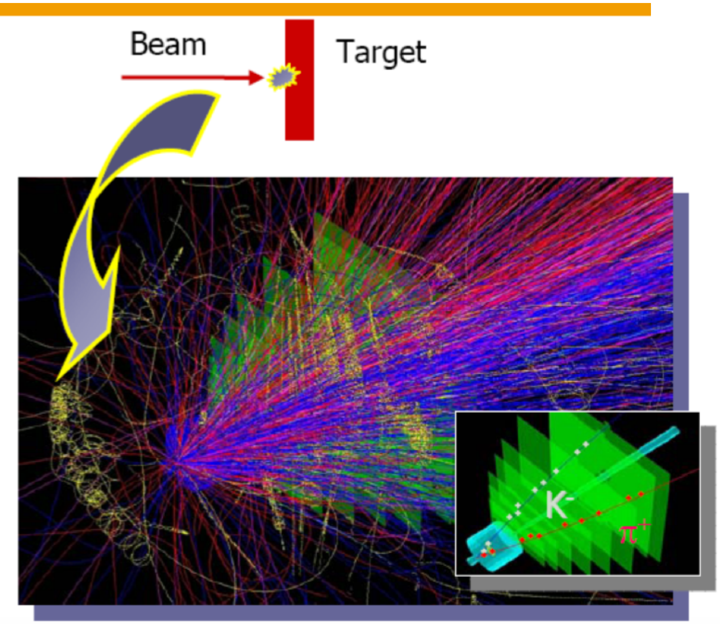
HEP Experiments: Collider and Fixed-Target



Inelastic collisions
$10^7 - 10^9$

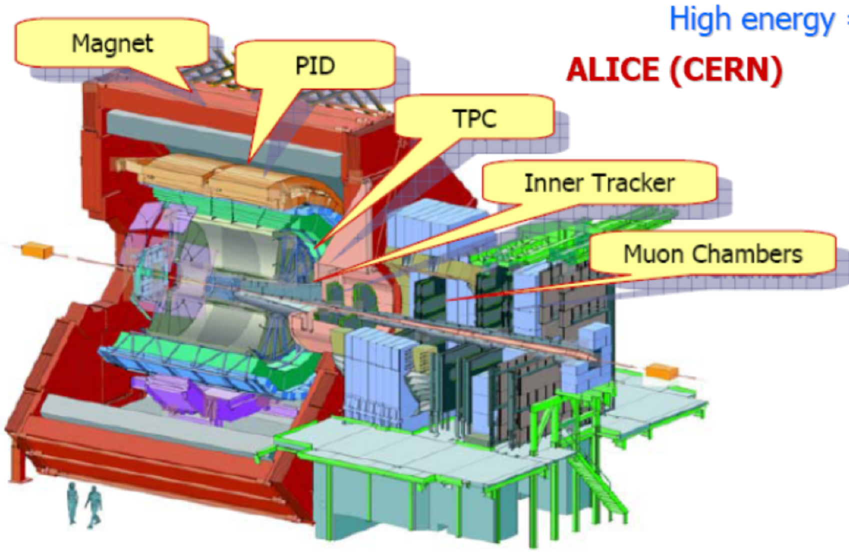
$\mathbf{10^{11}}$

Signal events
$10^2 - 10^{-2}$

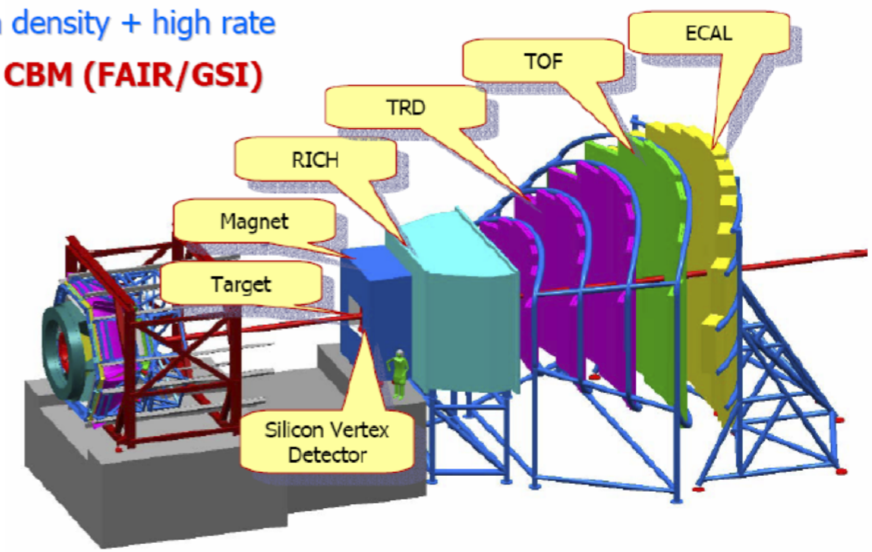


High energy = high density + high rate

ALICE (CERN)

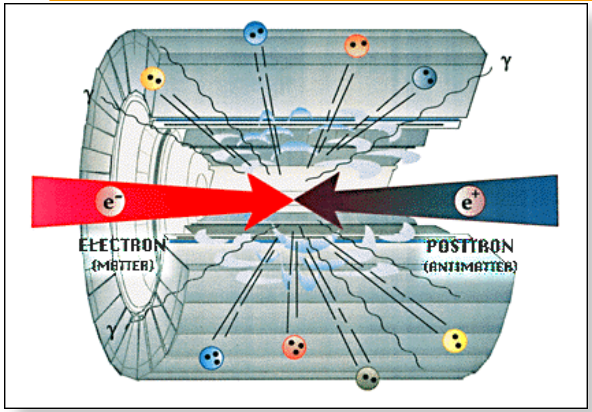


CBM (FAIR/GSI)



HEP Experiments: select interesting physics on-line

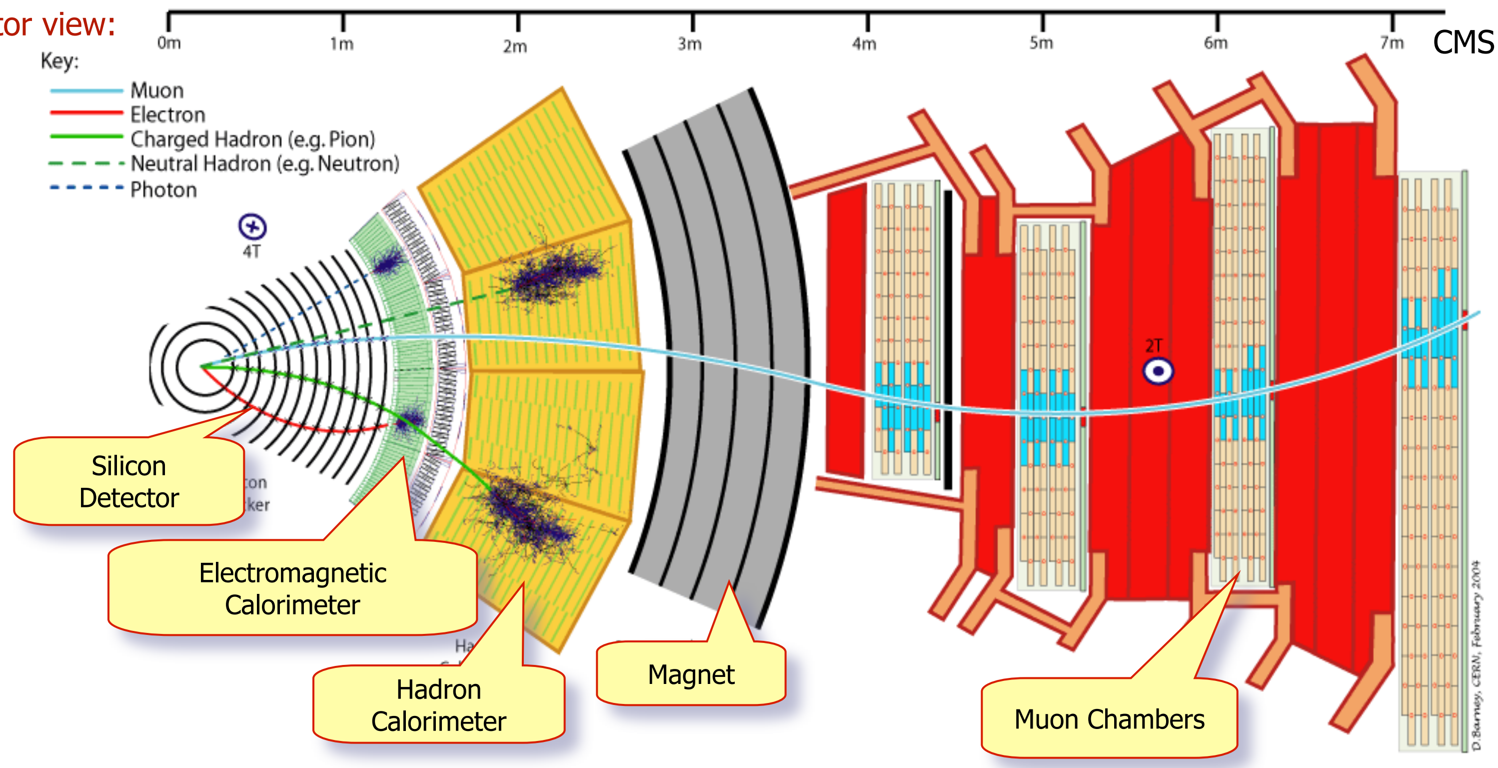
From Raw Data to Physics



1. Particle Accelerator
2. Particle Detectors
3. Data Acquisition (DAQ)
4. Data Reconstruction
5. Physics Analysis

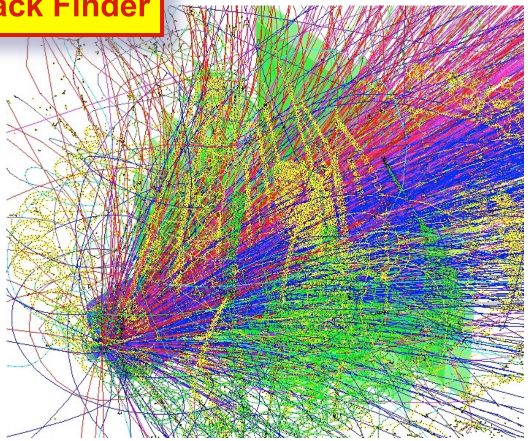
1. Find particle trajectories (tracks)
2. Estimate track parameters
3. Associate particle ID
4. Find interaction point (primary vertex)
5. Find secondary tracks (daughter particles)
6. Find secondary vertices (decay points)
7. Reconstruct short lived decay particles
8. Select physics

Sector view:



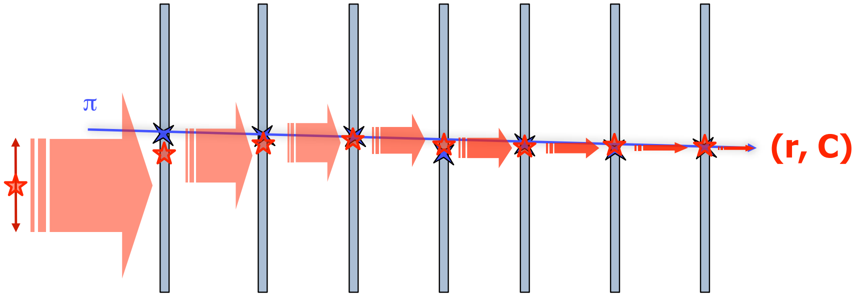
Stages of Event Reconstruction

1 Track Finder



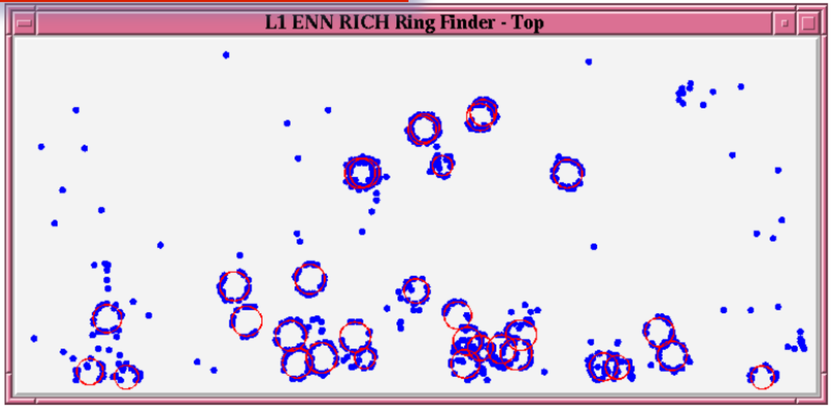
- Conformal Mapping
- Hough Transformation
- Track Following + Kalman Filter
- Cellular Automaton + Kalman Filter

2 Track Fitter



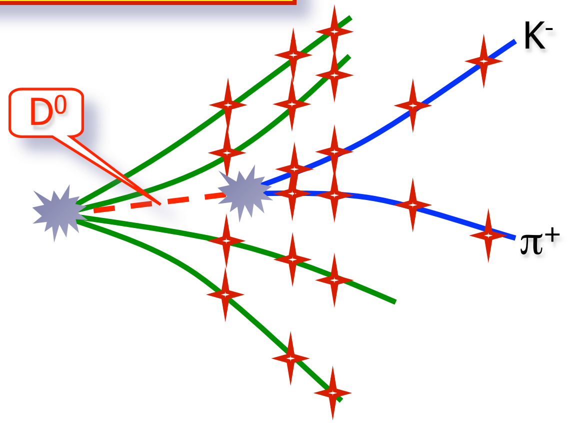
- Kalman Filter

3 Ring Finder (Particle ID)



- Hough Transformation
- Elastic Neural Net

4 Short-Lived Particles Finder



- Kalman Filter

Global Methods: Conformal Mapping + Histogramming

Global methods are especially suitable for fast tracking in projections

Example: Collider experiment with a solenoid, where tracks are circular trajectories

<u>Conformal Mapping:</u>

Transform circles into straight lines

$u = x/(x^2+y^2)$

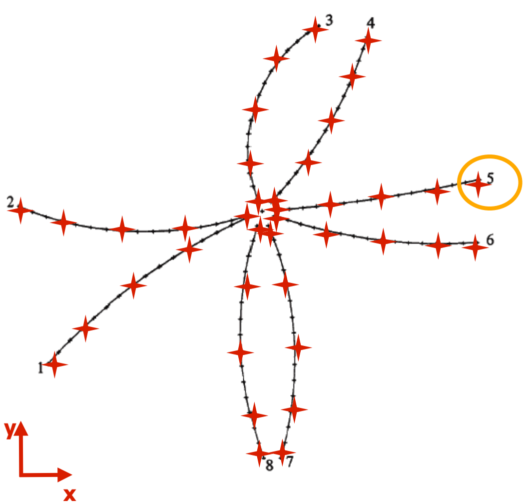
$v = -y/(x^2+y^2)$

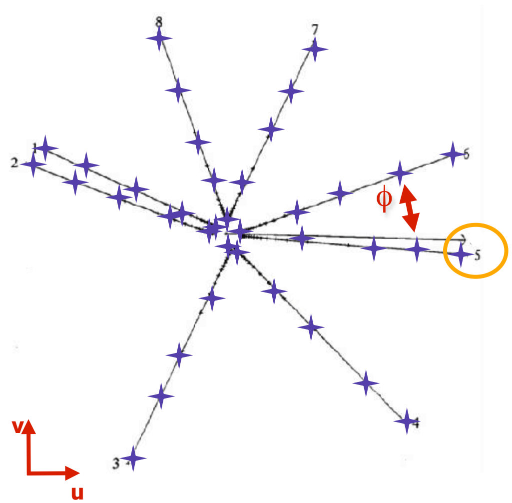
<u>Histogram:</u>

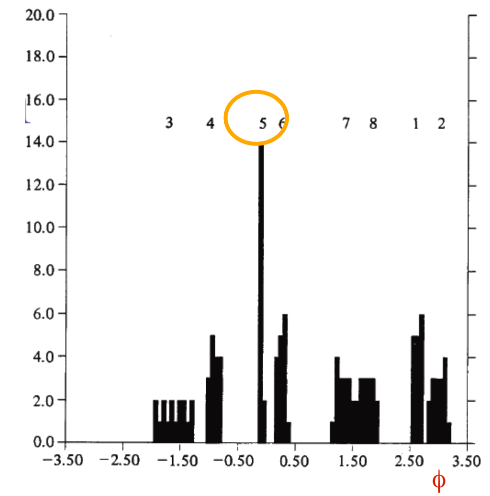
Collect a histogram of azimuth angles ϕ

Find peaks in the histogram

Collect hits into tracks



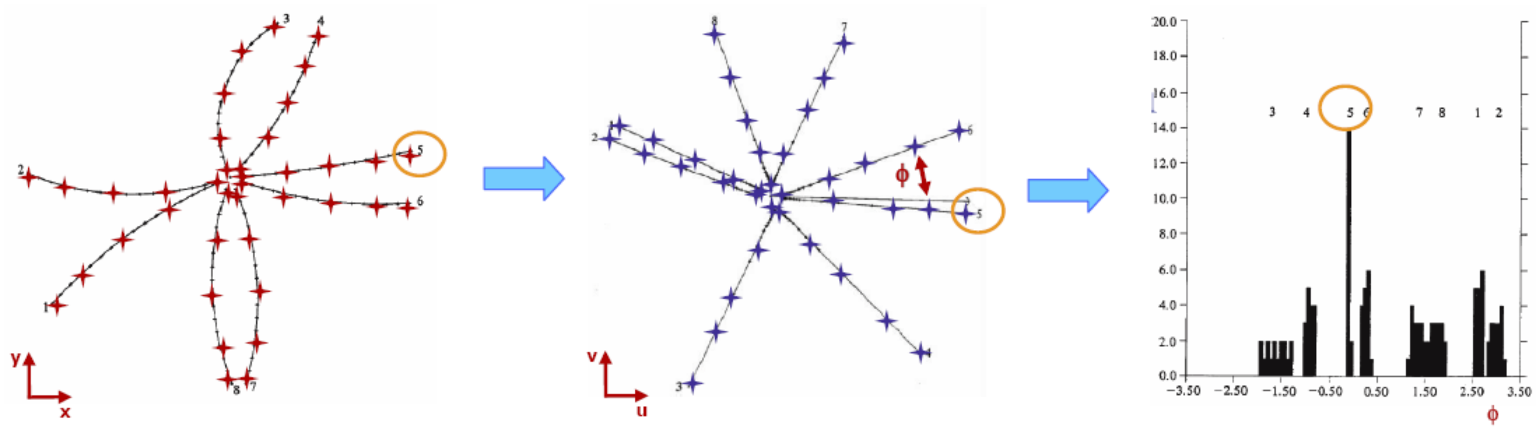




Global Methods: Conformal Mapping + Histogramming

Conformal Mapping:
Transform circles into straight lines
$u = x/(x^2+y^2)$
$v = -y/(x^2+y^2)$

Histogram:
Collect a histogram of azimuth angles ϕ
Find peaks in the histogram
Collect hits into tracks



Features:
- Impressive visual simplification of the problem
- Each step is easy to implement in hardware
- ...

Weak points:
- Needs to know an exact position of the interaction point
- Do not finds tracks, but only approximate track parameters
- No grouping of hits into track candidates
- Finds only primary tracks
- Problems with non-uniform magnetic field
- ...

Useful implemented in hardware and for very simple event topologies only

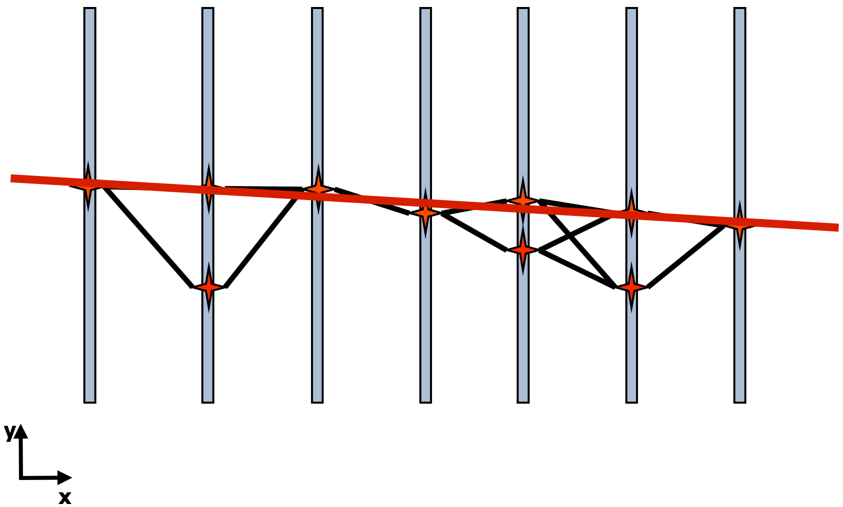
Global Methods: Hough Transformation

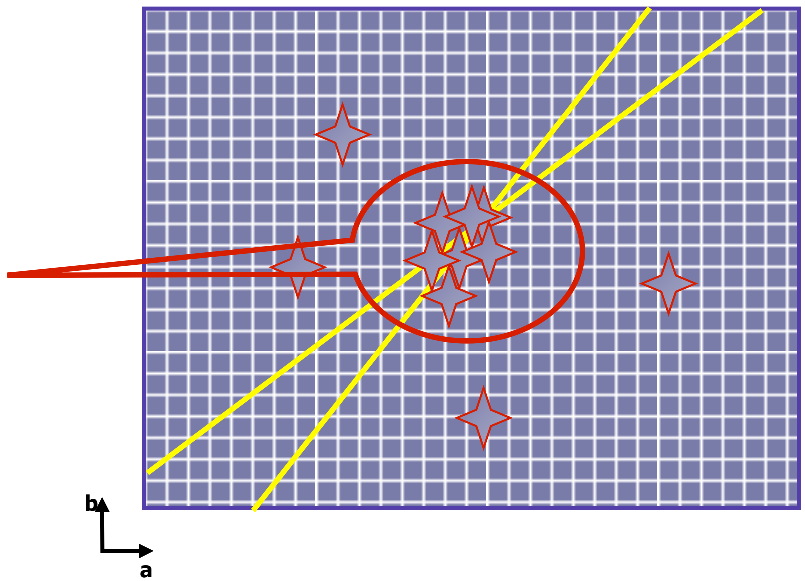
Measurement Space

$$y = a*x + b$$

Parameter Space

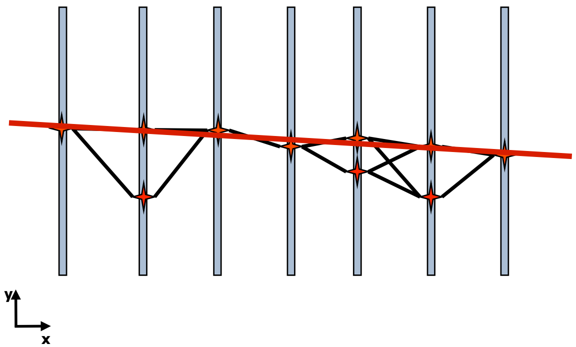
$$b = -x*a + y$$

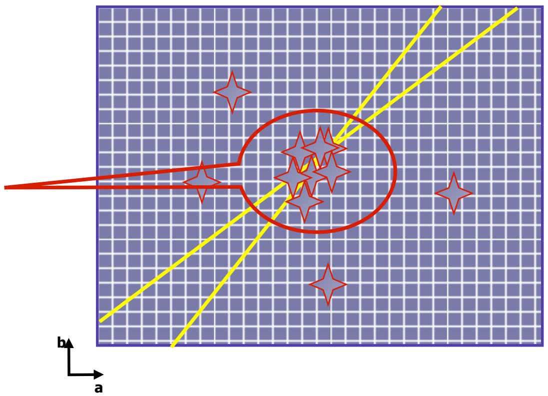




Global Methods: Hough Transformation

Measurement Space $y = a*x + b$ → **Parameter Space** $b = -x*a + y$



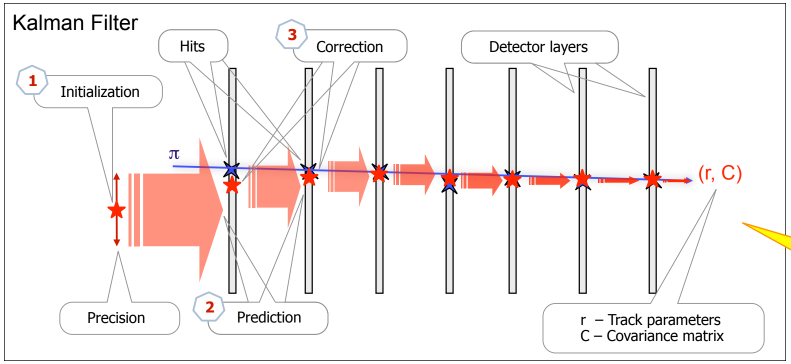


Features:
- Generalization of the histogramming method
- Easy to implement in hardware
- ...

Weak points:
- Needs a global track model
- Appropriate only for uniform magnetic fields
- Does not include multiple scattering
- Provides only track parameters etc.
- No competition between track candidates
- Needs a lot of memory (x, y, tx, ty, q/p -> 5D histogramming)
- ...

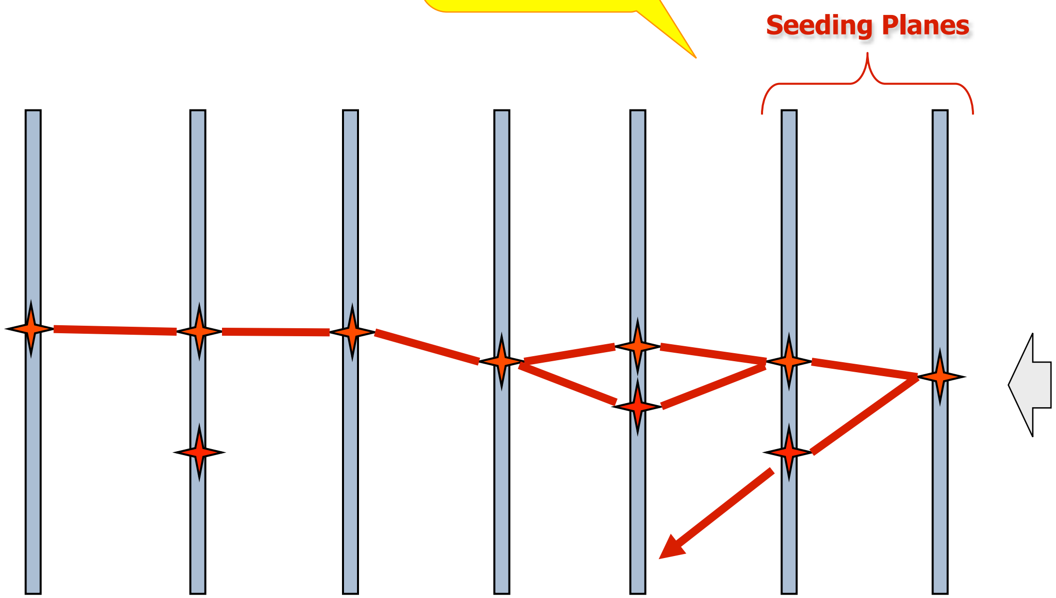
Useful implemented in hardware and for simple event and trigger topologies

Local Methods: Kalman Filter for Track Following

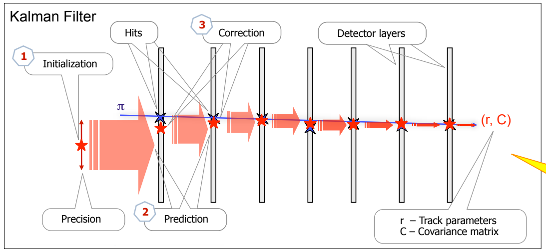


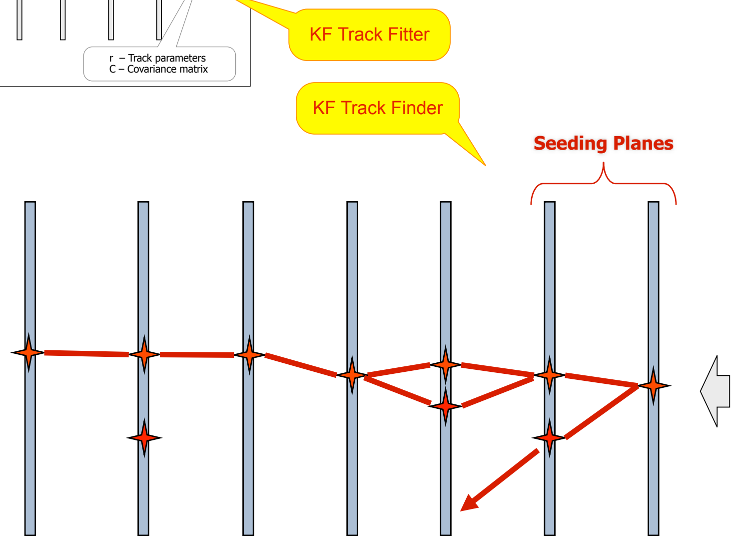
KF Track Fitter

KF Track Finder



Local Methods: Kalman Filter for Track Following





Features:

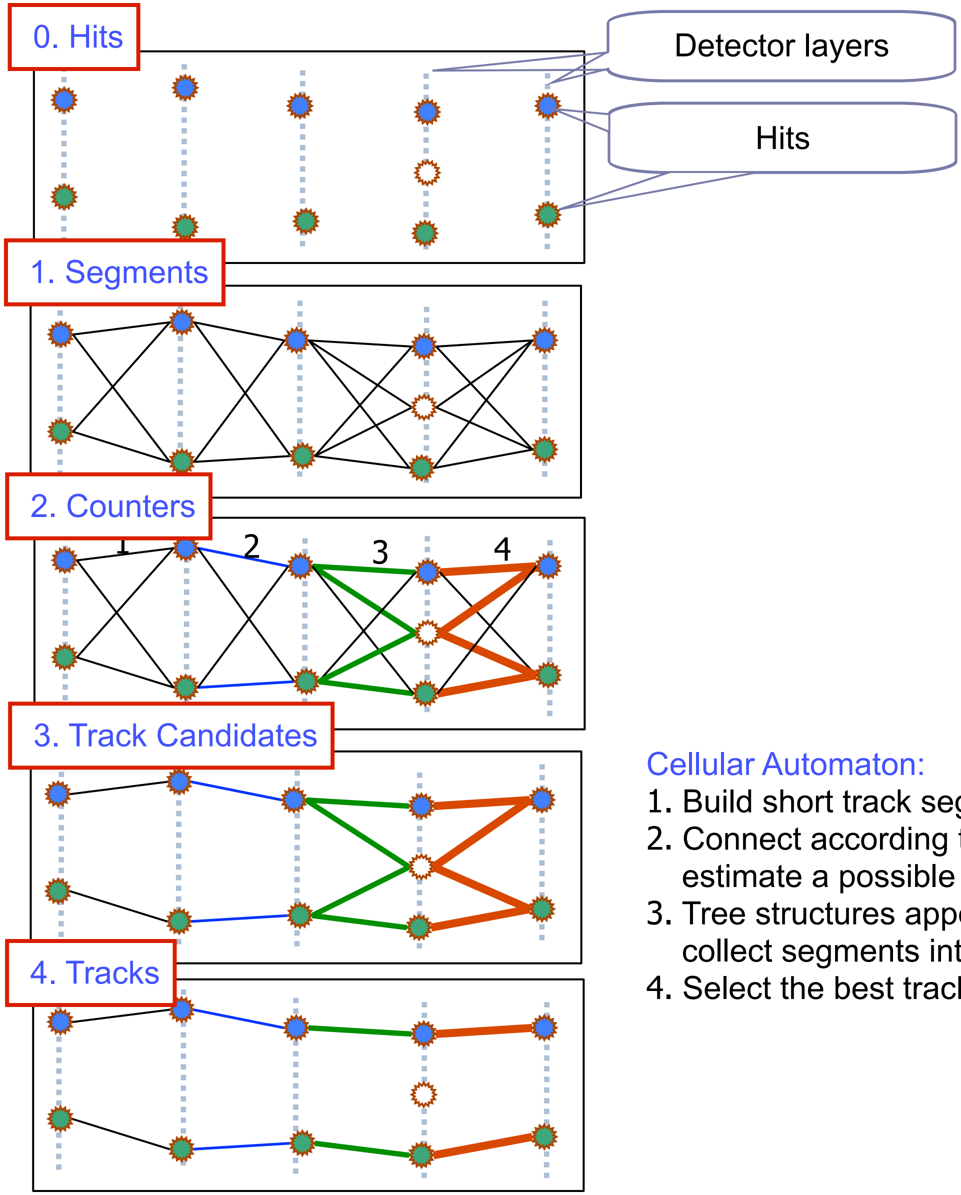
- Psychologically easy to accept hit by hit track finding
- Combined track finder and fitter based on KF
- Development of a new experiment starts with an ideal MC track finder and a realistic KF track fitter, then it is easy to implement the track finder as track following
- ...

Weak points:

- Based on a single track approach
- Needs seeding (starting short track segments)
- Efficiency is limited by the seeding efficiency (and detectors!)
- Works at the hit level, searching for hits within a region
- Repeats the same calculations, after discarding track candidates
- No global competition between track candidates
- ...

Useful for relatively simple event topologies and as a second after the ideal track finder

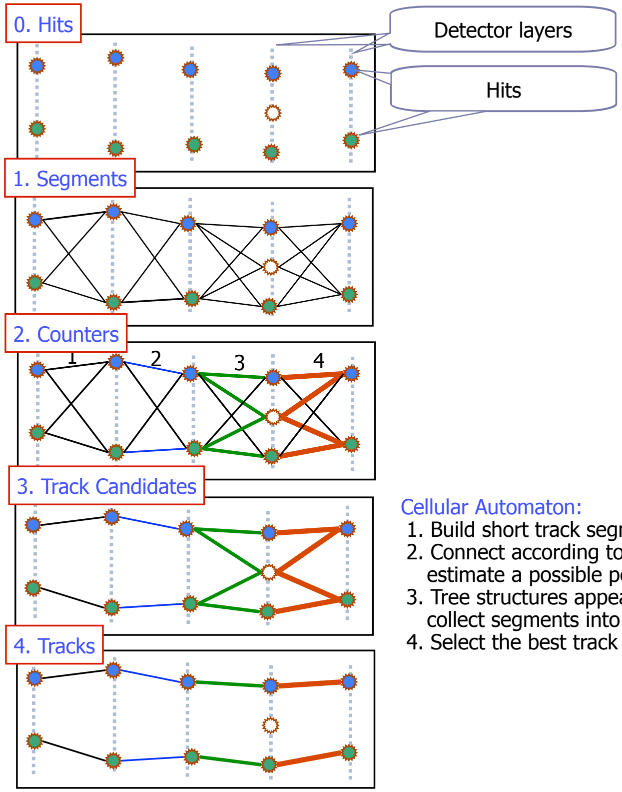
Local Methods: Cellular Automaton as Track Finder



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.

Local Methods: 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.

Features:
- Local relations -> parallel algorithm
- Staged implementation: hits -> segments -> tracks
- Polynomial (2nd order) combinatorics
- Track competition at the global level
- Includes the KF fitter, if necessary, for high track densities
- ...

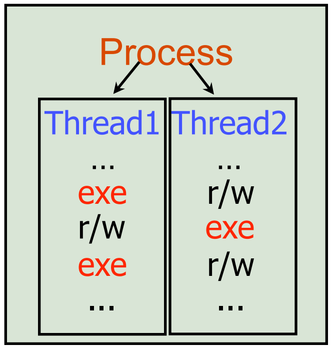
Weak points:
- Not easy to understand a parallel algorithm (Game of Life)
- Parallel hardware is coming now
- ...

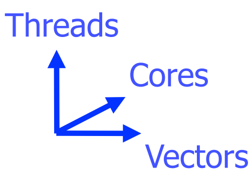
Useful for complicated event topologies with large combinatorics and for parallel hardware

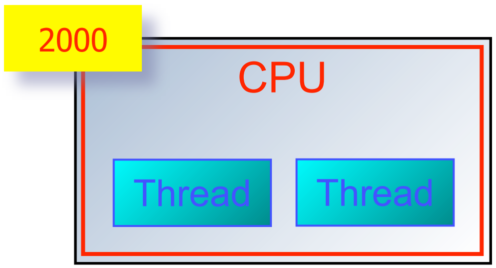
Many-core HPC: Cores, Threads and Vectors

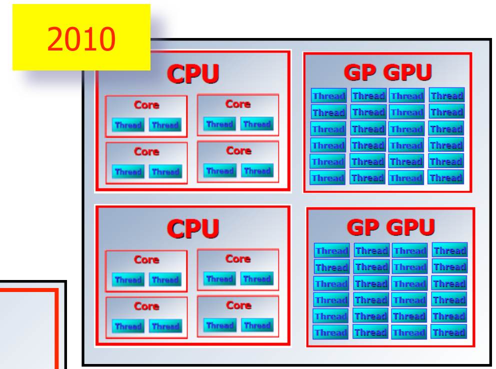
HEP experiments work with high data rates, therefore need High Performance Computing (HPC) !

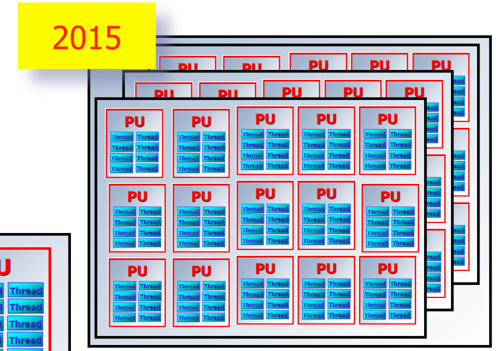
Cores and Threads realize the task level of parallelism

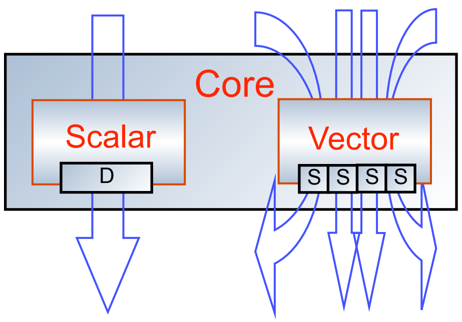












Vectors (SIMD) = data level of parallelism

SIMD = Single Instruction, Multiple Data

Remarks:

+ hardware parallelism for free;
- many-core machine is not batch farm;
+ use core/thread parallelism at the event level;
- scalar code is useless;
+ use vector programming;

Speed-up = N sockets * N cores * (1 + HT) * SIMD width

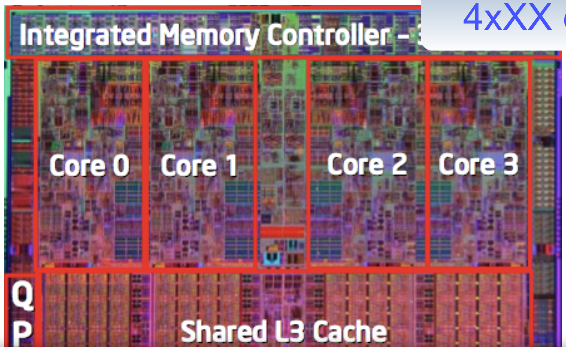
S lxir075@gsi = 4 * 10 * (1 + 0.3) * 4 = 200

Fundamental redesign of traditional approaches to data processing is necessary

Many-Core CPU/GPU Architectures

Intel/AMD CPU

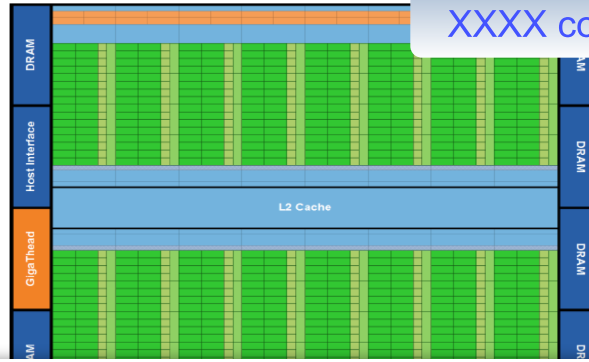
4xXX cores



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

ATI/NVIDIA GPU

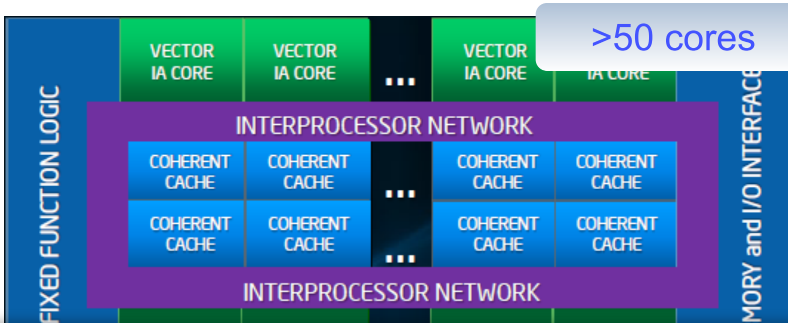
XXXX cores



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

Intel Xeon Phi

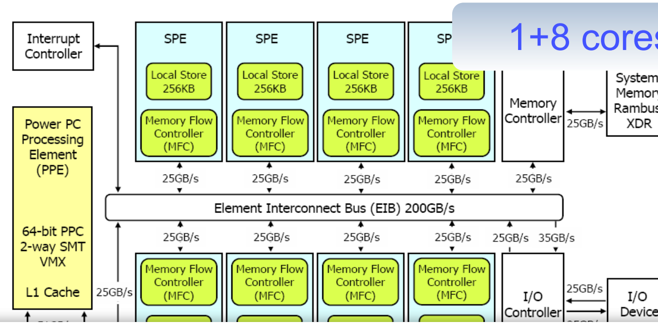
>50 cores



- 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

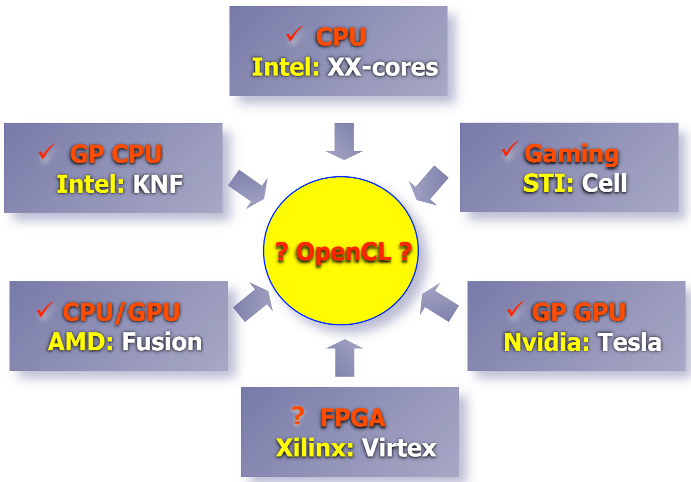
1+8 cores



- 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

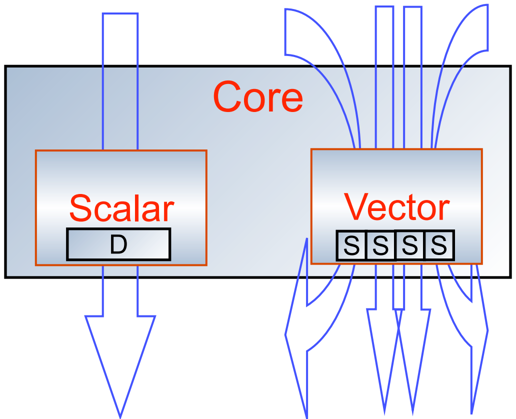


- Intel ArBB (Array Building Blocks)
  - Extension to the C language
  - Intel CPU/GPU specific
  - SIMD exploitation for automatic parallelism
- NVIDIA CUDA (Compute Unified Device Architecture)
  - Defines hardware platform
  - Generic programming
  - Extension to the C language
  - Explicit memory 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

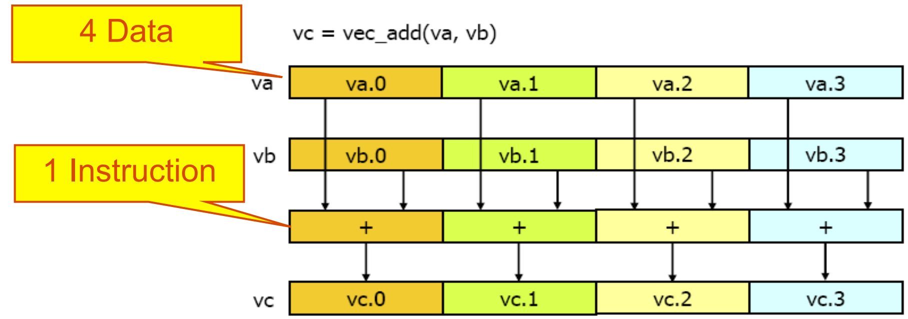
Choice of CPU/GPU/Programming is a practical question

Vectorization: SIMD (Single Instruction Multiple Data)

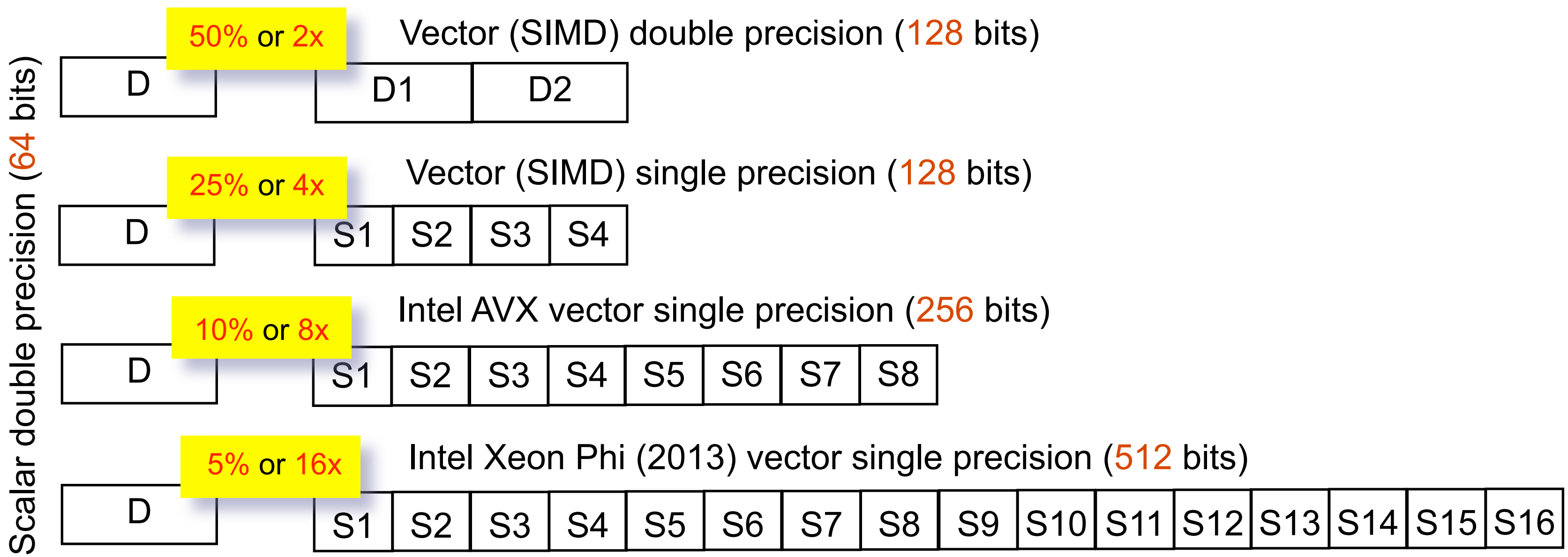
SIMD exploits data stream parallelism



Scalar or Vector = Slower or Faster



Complication:
a+b ---> _mm_add_ps(a, b)



Use headers to overload +, -, *, / operators --> the source code is unchanged !

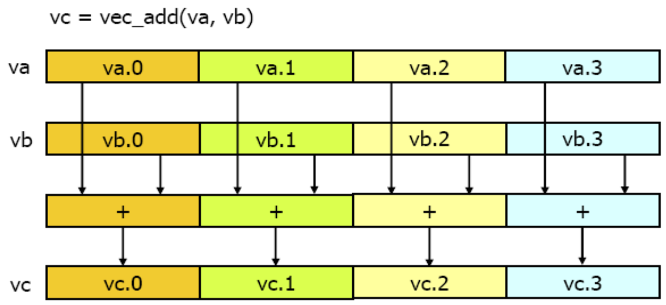
Header (Intel's SSE), later Vector Classes (Vc)

SIMD instructions

```
typedef F32vec4 Fvec_t;
 /* Arithmetic Operators */
 friend F32vec4 operator +( const F32vec4 &a, const F32vec4 &b ) { return _mm_add_ps(a, b); }
 friend F32vec4 operator -( const F32vec4 &a, const F32vec4 &b ) { return _mm_sub_ps(a, b); }
 friend F32vec4 operator *( const F32vec4 &a, const F32vec4 &b ) { return _mm_mul_ps(a, b); }
 friend F32vec4 operator /( const F32vec4 &a, const F32vec4 &b ) { return _mm_div_ps(a, b); }
 /* Functions */
 friend F32vec4 min( const F32vec4 &a, const F32vec4 &b ){ return _mm_min_ps(a, b); }
 friend F32vec4 max( const F32vec4 &a, const F32vec4 &b ){ return _mm_max_ps(a, b); }
 /* Square Root */
 friend F32vec4 sqrt( const F32vec4 &a ){ return _mm_sqrt_ps (a); }
 /* Absolute value */
 friend F32vec4 fabs( const F32vec4 &a ){ return _mm_and_ps(a, _f32vec4_abs_mask); }
 /* Logical */
 friend F32vec4 operator&( const F32vec4 &a, const F32vec4 &b ){ // mask returned
   return _mm_and_ps(a, b);
 }
 friend F32vec4 operator|( const F32vec4 &a, const F32vec4 &b ){ // mask returned
   return _mm_or_ps(a, b);
 }
 friend F32vec4 operator^( const F32vec4 &a, const F32vec4 &b ){ // mask returned
   return _mm_xor_ps(a, b);
 }
 friend F32vec4 operator!( const F32vec4 &a ){ // mask returned
   return _mm_xor_ps(a, _f32vec4_true);
 }
 friend F32vec4 operator||( const F32vec4 &a, const F32vec4 &b ){ // mask returned
   return _mm_or_ps(a, b);
 }
 /* Comparison */
 friend F32vec4 operator<( const F32vec4 &a, const F32vec4 &b ){ // mask returned
   return _mm_cmplt_ps(a, b);
 }
```

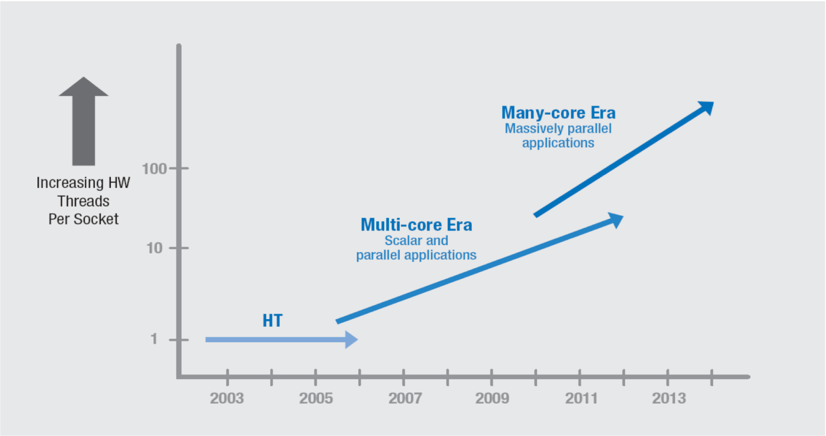
Code (Part of the Kalman Filter)

```
inline void AddMaterial( TrackV &track, Station &st, Fvec_t &qp0 )
{
  cnst mass2 = 0.1396*0.1396;

  Fvec_t tx = track.T[2];
  Fvec_t ty = track.T[3];
  Fvec_t txtx = tx*tx;
  Fvec_t tyty = ty*ty;
  Fvec_t txtx1 = txtx + ONE;
  Fvec_t h = txtx + tyty;
  Fvec_t t = sqrt(txtx1 + tyty);
  Fvec_t h2 = h*h;
  Fvec_t qp0t = qp0*t;

  cnst c1=0.0136, c2=c1*0.038, c3=c2*0.5, c4=-c3/2.0, c5=c3/3.0, c6=-c3/4.0;

  Fvec_t s0 = (c1+c2*st.logRadThick + c3*h + h2*(c4 + c5*h +c6*h2) )*qp0t;

  Fvec_t a = (ONE+mass2*qp0*qp0t)*st.RadThick*s0*s0;

  CovV &C = track.C;

  C.C22 += txtx1*a;
  C.C32 += tx*ty*a; C.C33 += (ONE+tyty)*a;
}
```

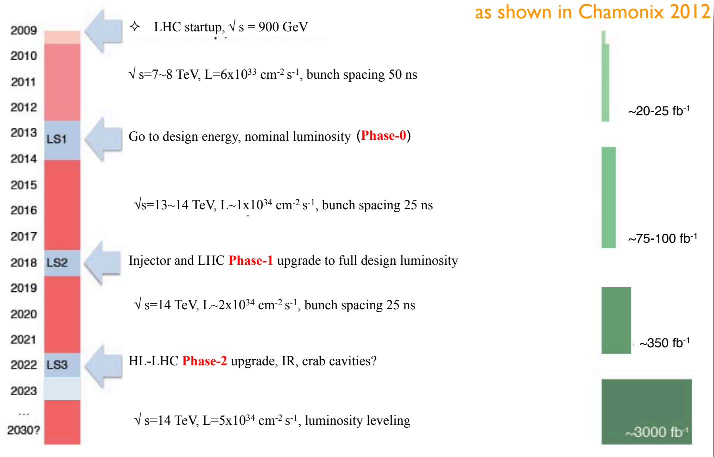


Use of headers to vectorize the code

How to Parallelize Reconstruction in Running (!) Experiments



S. Borkar et al. (Intel), "Platform 2015: Intel Platform Evolution for the Next Decade", 2005.



1	Modify the existing code	ATLAS
2	Implement a new algorithm in the combinatorial part	CMS
3	Merge on-line and off-line codes	ALICE
4	Use an advanced existing algorithm as seed finder	STAR
5	Design and develop a new code from scratch	CBM

Provide stable and reproducible/better results

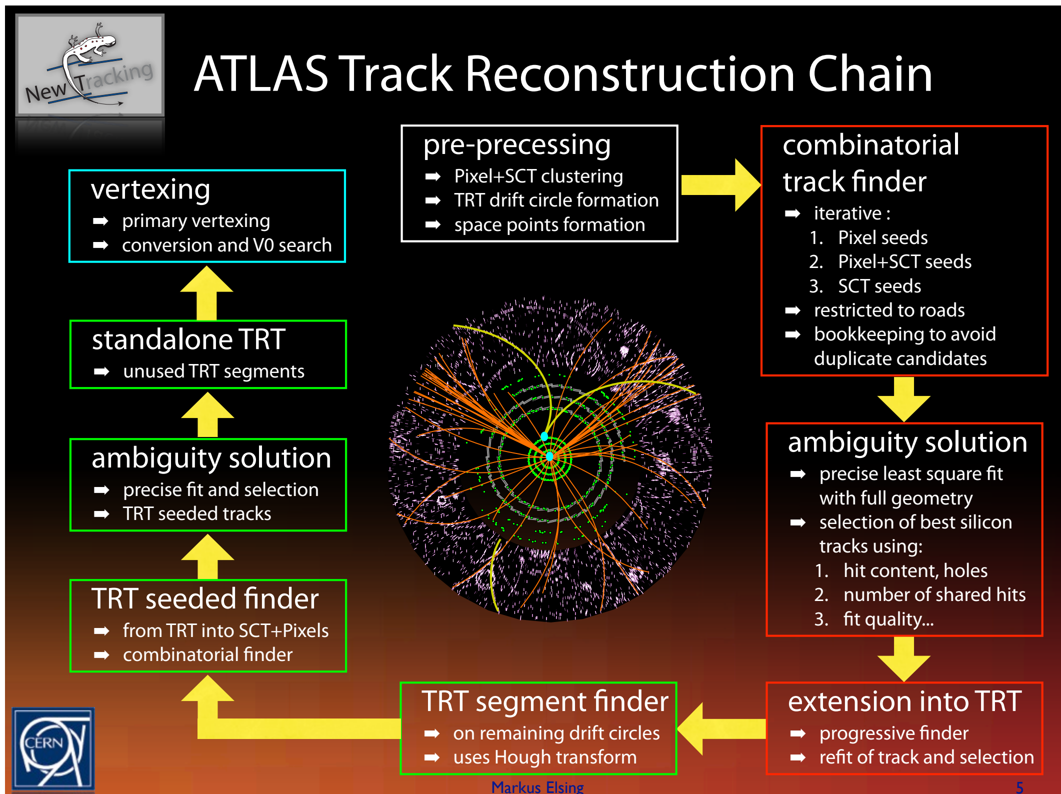
ATLAS

Off-line

Auto-vectorization
Multithreading
Optimization
...

HLT

Integration with off-line



Marcus Elsing, 4th International Workshop for Future Challenges in Tracking and Trigger Concepts, CERN, 28-30.11.2012

LS1: Modify the existing code

CMS

Off-line

Combinatorial Track Following
Gaussian Sum Filter
...

Off-line

Hough Transform ?

Current tracking limitations

Iterative tracking is performing well in early steps; early steps are able to provide fast the bulk of tracking efficiency and, timing wise, they are pretty stable with respect to PU.

Problems are in later steps designed (loosen cuts) to recover efficiency for more difficult (e.g. displaced) tracks.

Late steps result in too many fake seeds with respect to early steps; each seed needs to be propagated in the attempt of building a track. This is time consuming and time per final good track is not favorable for these later steps.

The long story short: combinatorics...

Track building (e.g. full track reconstruction) is eventually able to get rid of these fake proto-tracks; similarly, more information should be used to clean up not useful seeds in advance. But this requires time.

New developments should address this paradox.

24.10.2012 Cerati / Sguazzoni / Stenson 41

Giacomo Sguazzoni, 4th International Workshop for Future Challenges in Tracking and Trigger Concepts, CERN, 28-30.11.2012

LS1: Implement a new algorithm in the combinatorial part

ALICE

Off-line

Track Following

Cellular Automaton (seeds) + Track Following

HLT

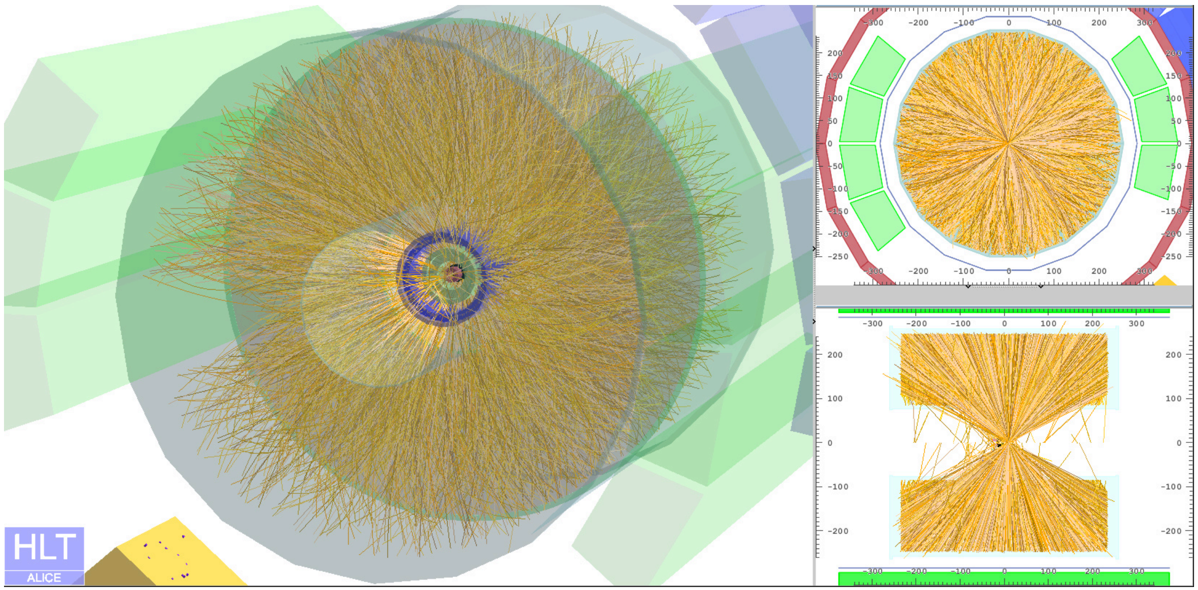
Conformal Mapping + Hough Transform

X

Cellular Automaton (CPU, GPU, CPU+GPU)

CA + TF (CPU, GPU, CPU+GPU) ?

ALICE HLT: Event of the first HI run with the GPU CA tracker



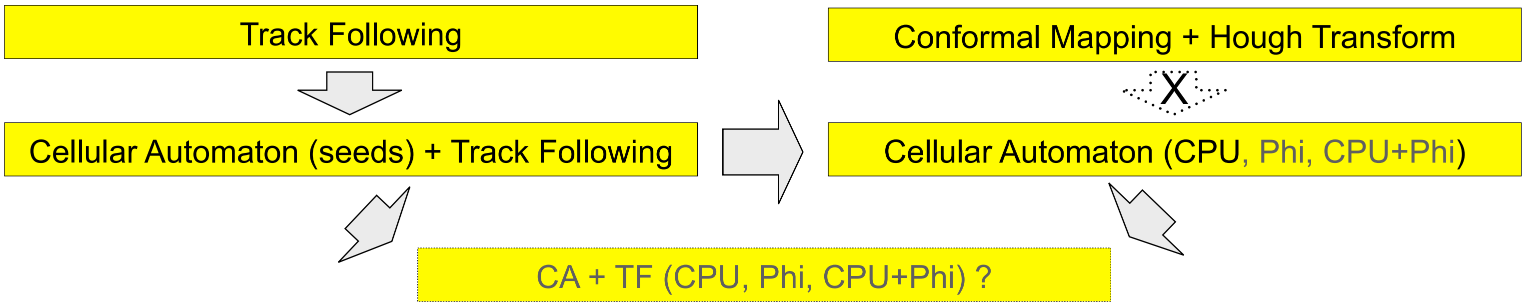
ALICE HLT Group

Merge off-line and on-line codes

STAR

Off-line

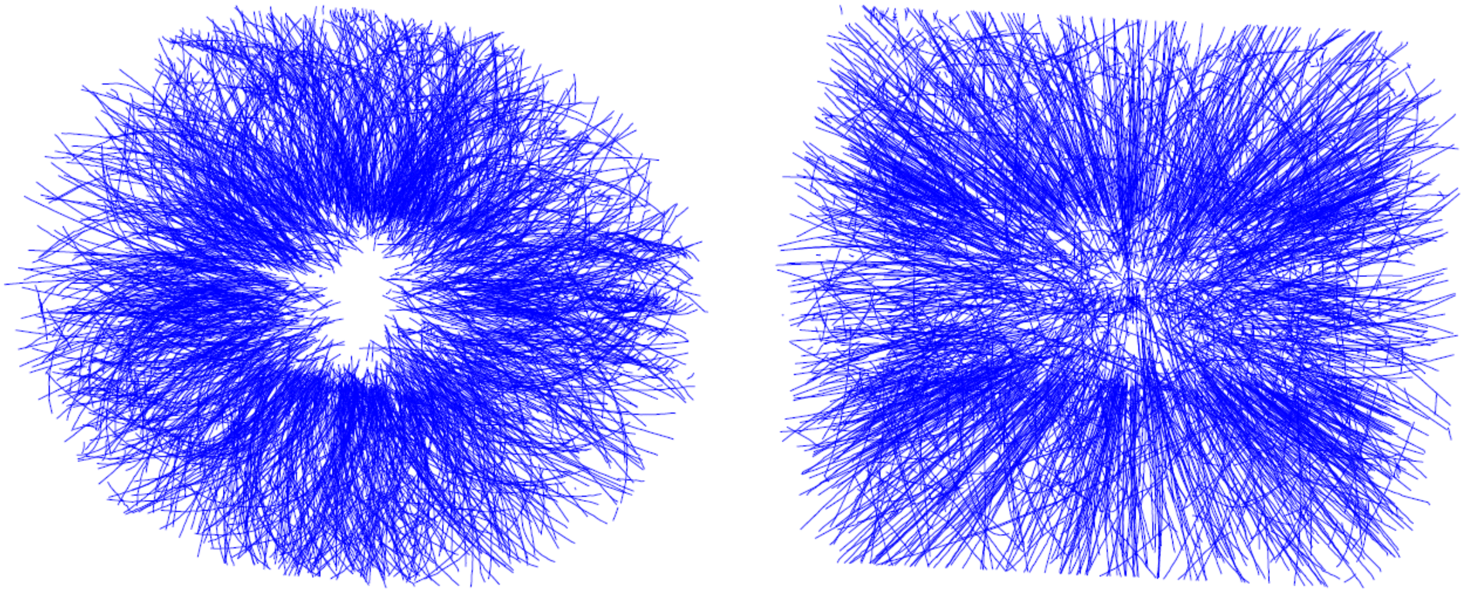
HLT



Au-Au event with 1446 tracks

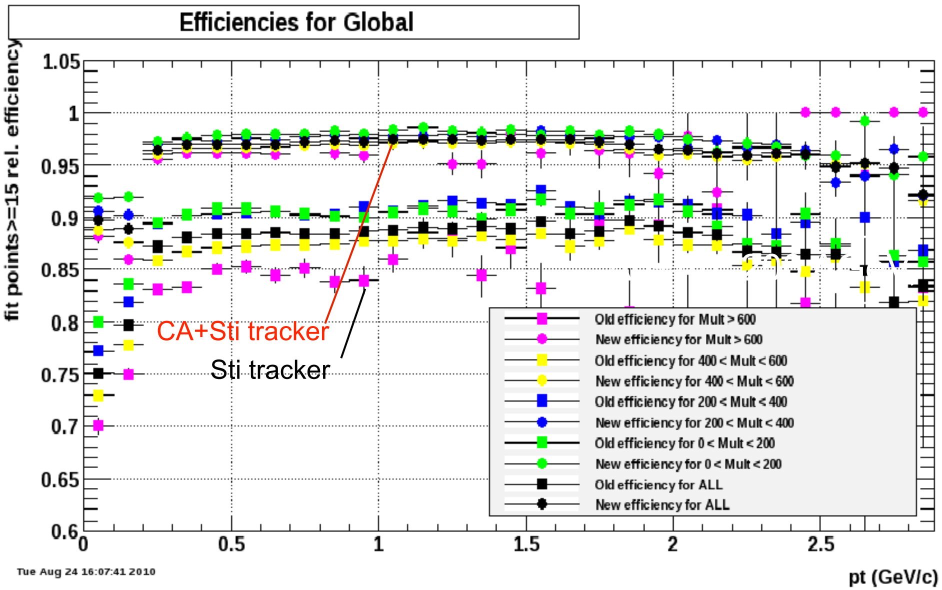
Front view

Side view



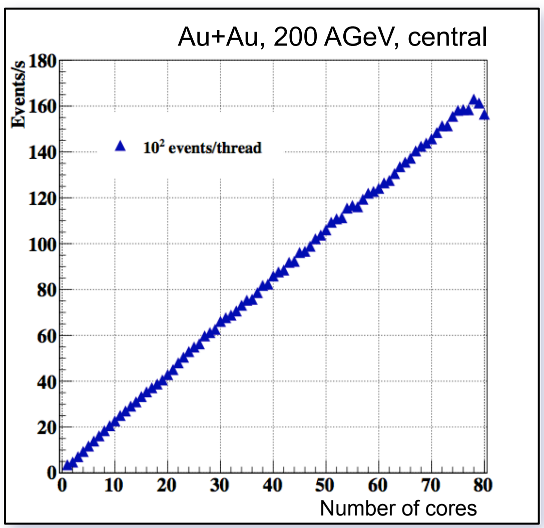
Use an advanced existing algorithm as seed finder

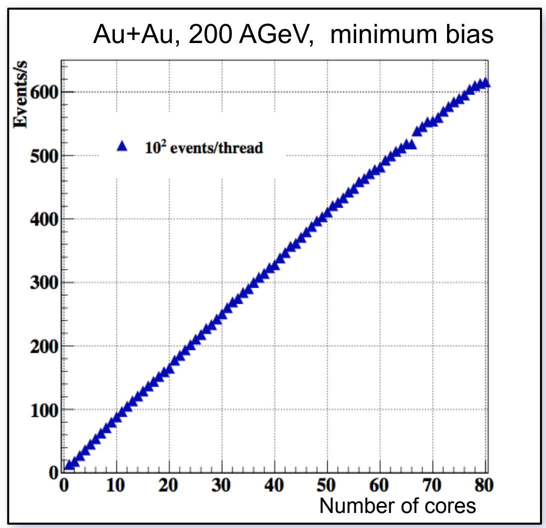
STAR TPC CA Track Finder

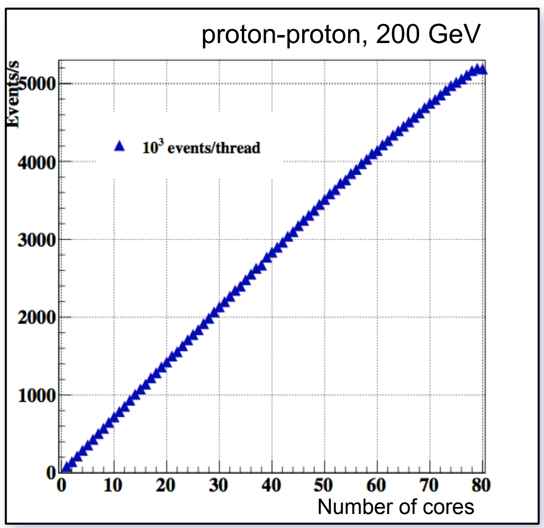


Efficiency and ratio, %	
Ref Set	96.6
All Set	88.6
Clone	10.6
Ghost	12.6
Tracks/ev	659
Time/ev, ms	47

All set: $p \geq 0.05$ GeV/c
Reference set: $p \geq 1$ GeV/c
Ghost: purity < 90%







lxir075.gsi.de: 4 Intel Xeon Westmere CPU E7-4860, 10 cores per CPU, HT, 2.27 GHz, 24 MB L3 cache, 64 GB RAM

CA is stable w.r.t. track multiplicity,10 (40 HLT) times faster than the TF track finder and has strong scalability up to 80 cores.

CBM

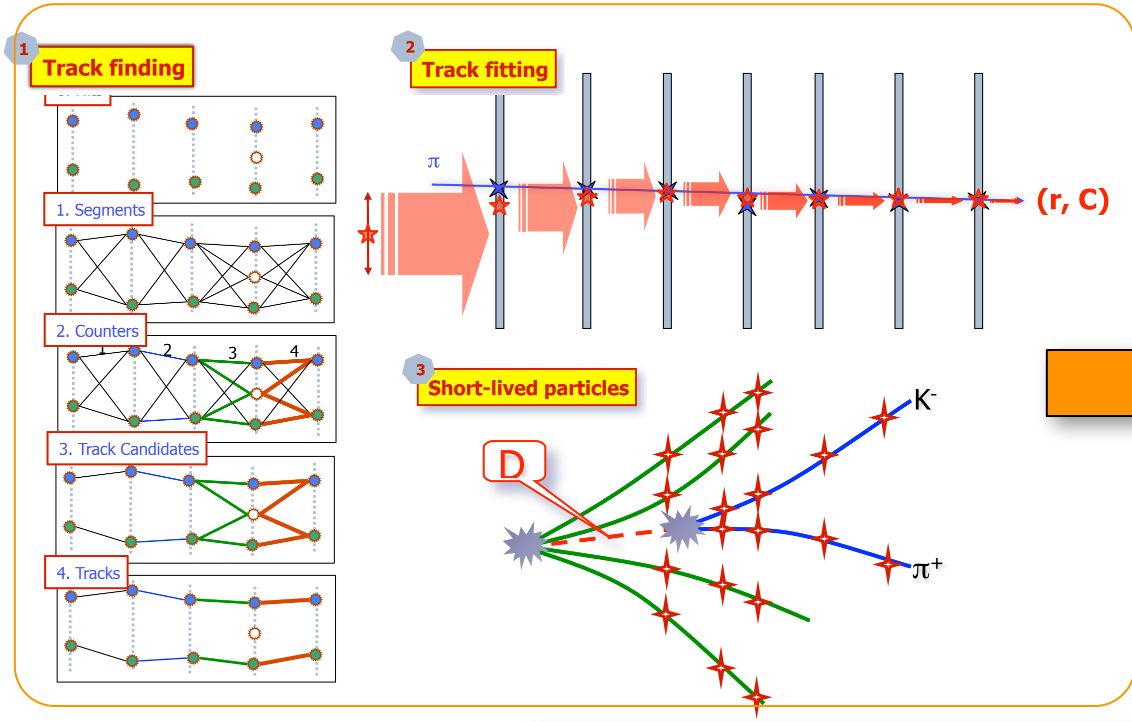
Off-line | Data Oriented Programming | FLES

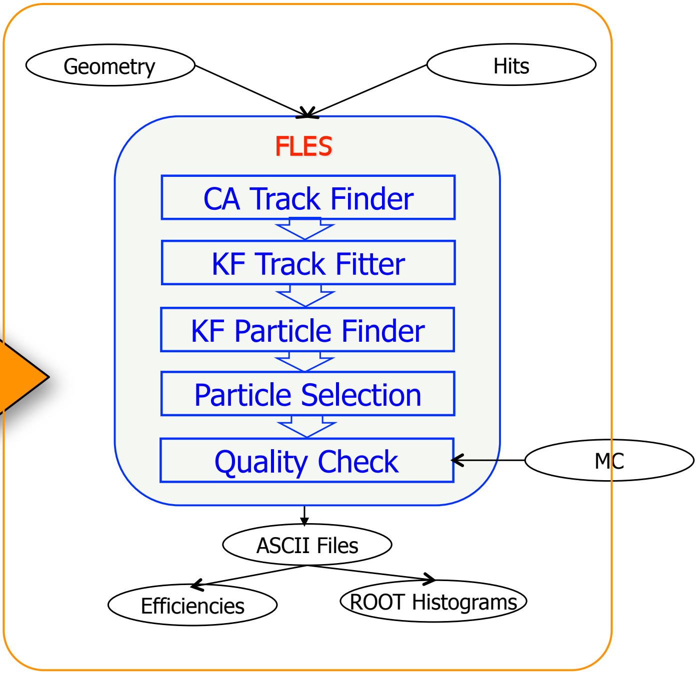
Cellular Automaton
Track Following (PID)

=

Cellular Automaton

Cellular Automaton
(CPU, Phi, GPU, CPU+Phi+GPU) ?





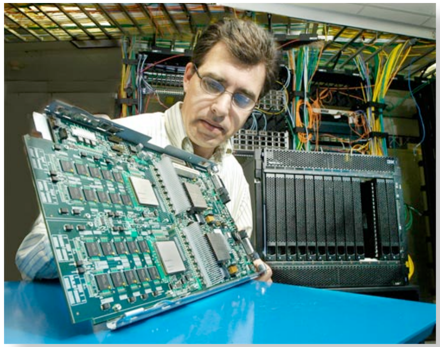
Design and develop a new code from scratch

CBM: Kalman Filter Track Fit on Cell

	Stage	Description	Time/track	Speedup
		Initial scalar version	12 ms	–
Intel	1	Approximation of the magnetic field	240 μs	50
Intel	2	Optimization of the algorithm	7.2 μs	35
Intel	3	Vectorization	1.6 μs	4.5
Cell	4	Porting to SPE	1.1 μs	1.5
Cell	5	Parallelization on 16 SPEs	0.1 μs	10
		Final simdized version	0.1 μs	$\boxed{120000}$

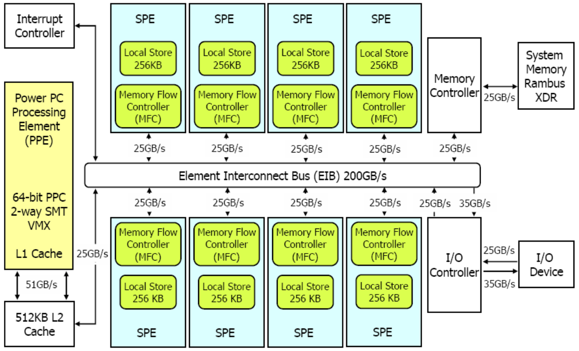
10000x faster on any PC

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



blade11bc4 @IBM, Böblingen:
2 Cell Broadband Engines, 256 kB LS, 2.4 GHz

The KF speed was increased by 5 orders of magnitude



Motivated by, but not restricted to Cell !

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

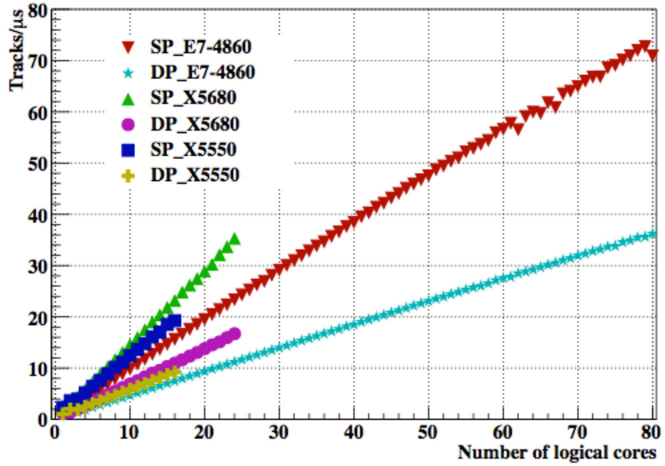
Implementations

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

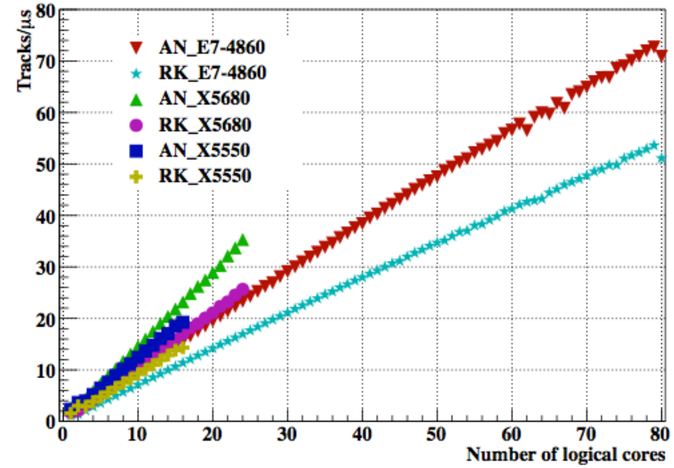
Parallelization (many-cores):
- Open MP
- ITBB
- ArBB
- OpenCL

Precision:
- single
- double

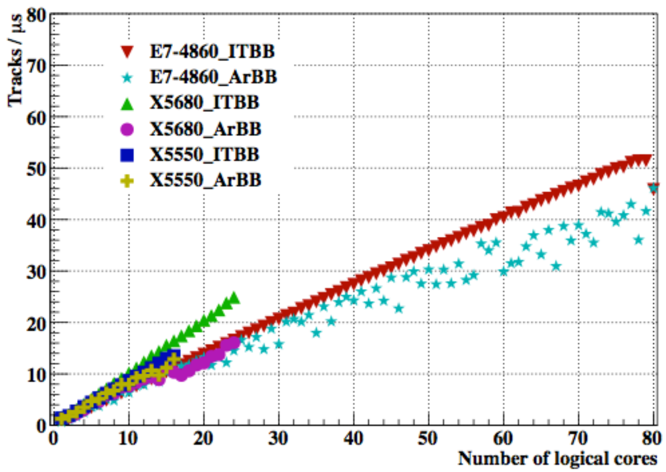
Conventional KF DP vs. SP



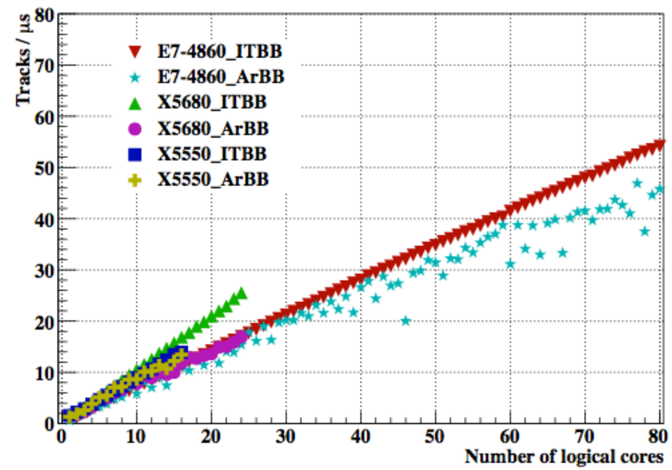
Conventional KF RK4 vs. Analytical



Square-Root KF

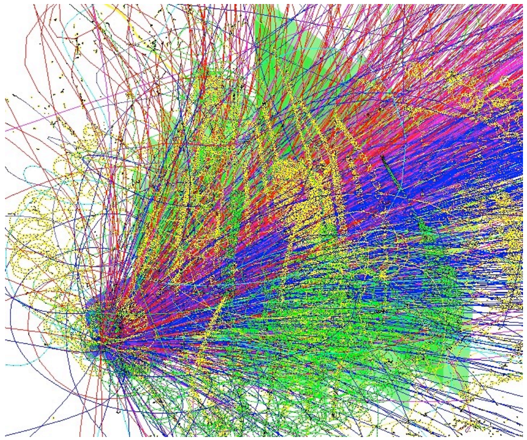


UD KF

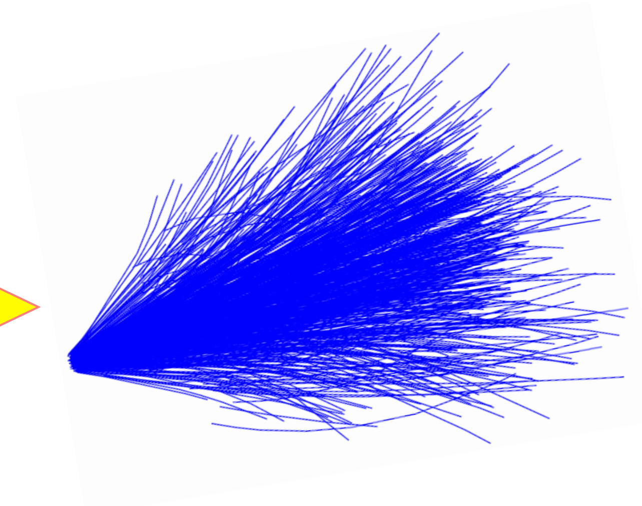


Strong many-core scalability of the Kalman filter library

CBM Track Finding Methods: from Pixels to Strip Detectors



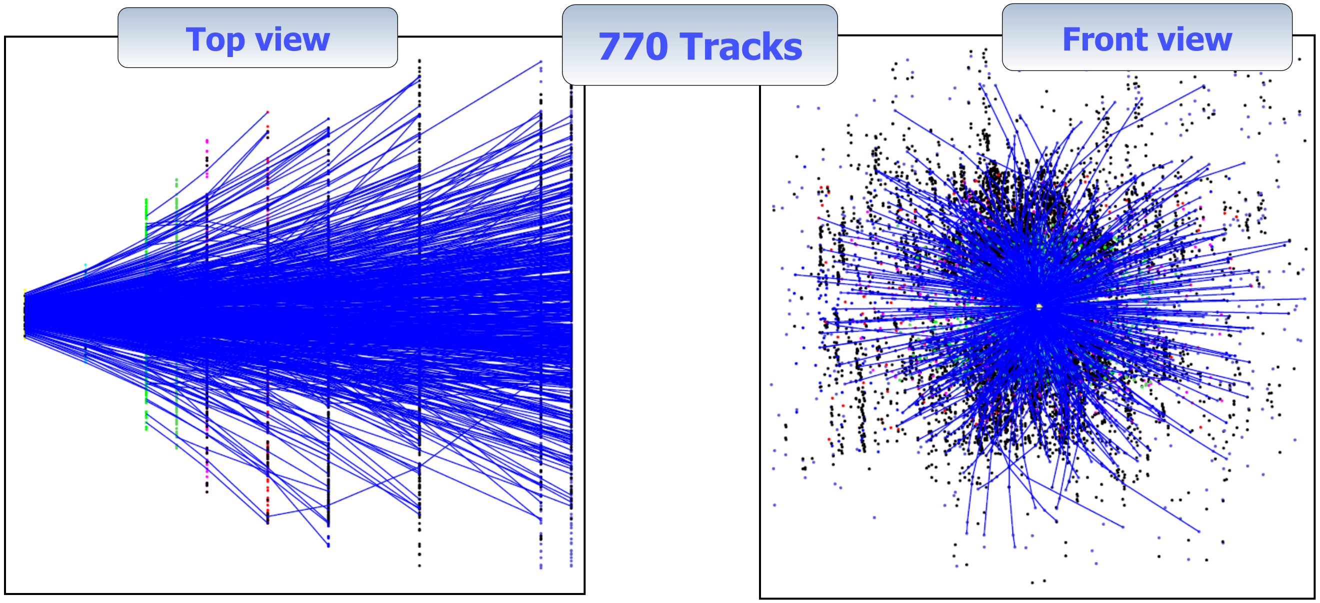
?

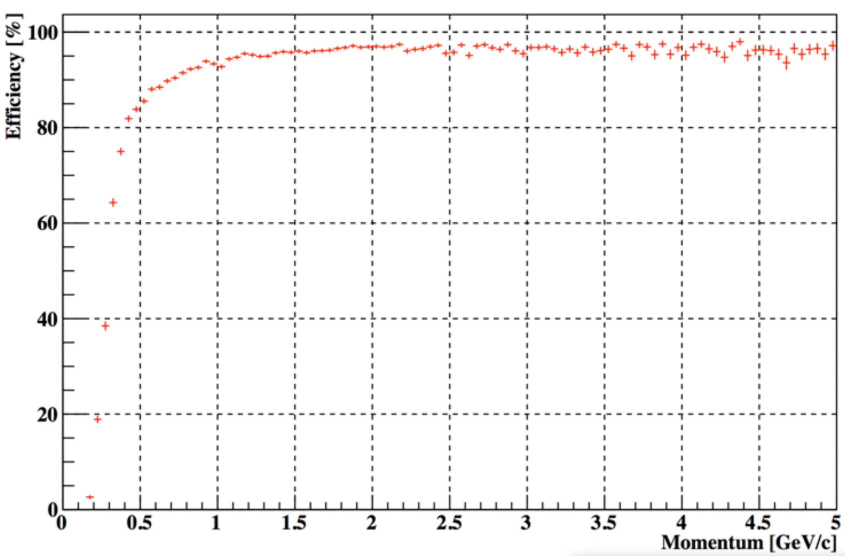


	Developer	Tracking Method	< 2005	> 2005
1	LHEP JINR, Dubna	Conformal Mapping	✓	✗
2	ZITI, Mannheim	Hough Transformation	✓	✗
3	LIT JINR, Dubna	Track Following	✓	✗
4	Uni-Heidelberg, GSI, FIAS	Cellular Automaton	✓	✓

Cellular Automaton is appropriate for complicated event topologies with large combinatorics

CBM CA Track Finder: Efficiency

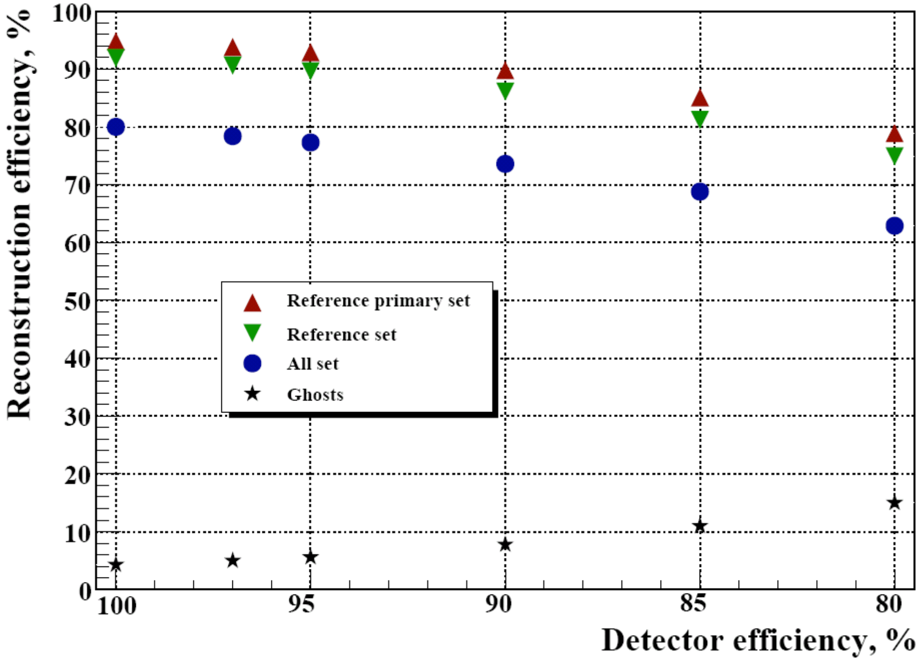




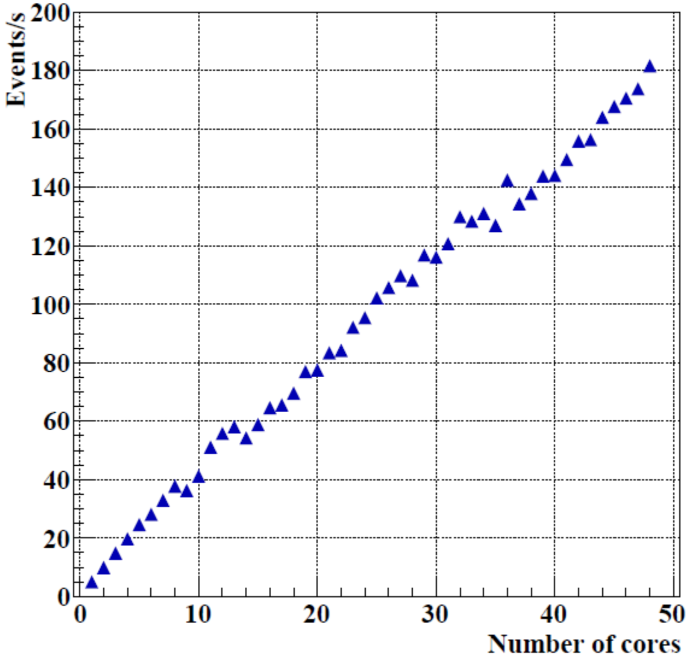
	Efficiency, %	
	mbias	central
Primary high-p tracks	97.1	96.2
Primary low-p tracks	90.4	90.7
Secondary high-p tracks	81.2	81.4
Secondary low-p tracks	51.1	50.6
All tracks	88.5	88.3
Clone level	0.2	0.2
Ghost level	0.7	1.5
Reconstructed tracks/event	120	591
Time/event/core	8.2 ms	57 ms

Efficient and stable event reconstruction

CBM CA Track Finder: Reliability and Scalability



Central Au-Au collisions

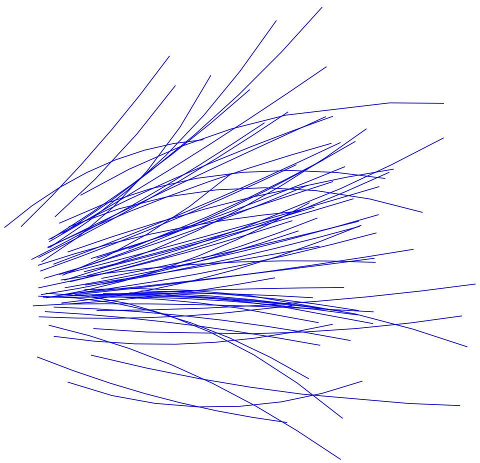


- AMD 6164EH
- 12 cores per CPU, 1.7 GHz
- Openlab CERN

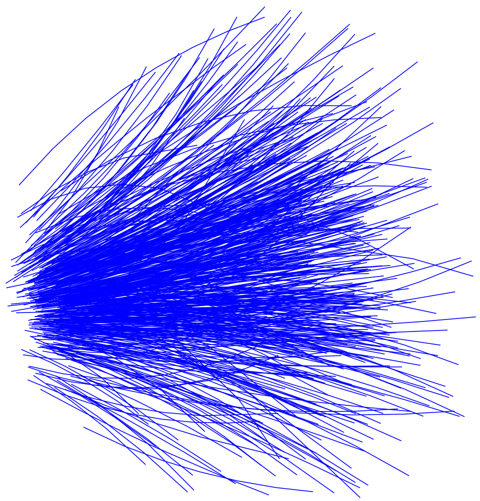
Stable algorithm down to 80% detector efficiency and strong many-core scalability

CBM CA Track Finder at High Track Multiplicity

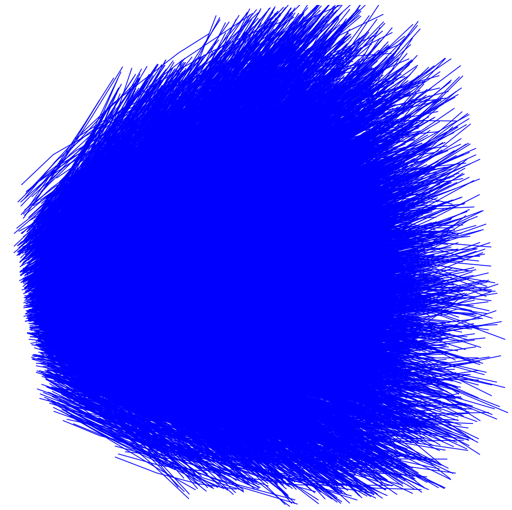
A number of minimum bias events is gathered into a group, which is then treated by the track finder as one event



1 minimum bias event
$\langle N_{reco} \rangle = 109$



5 minimum bias events
$\langle N_{reco} \rangle = 572$

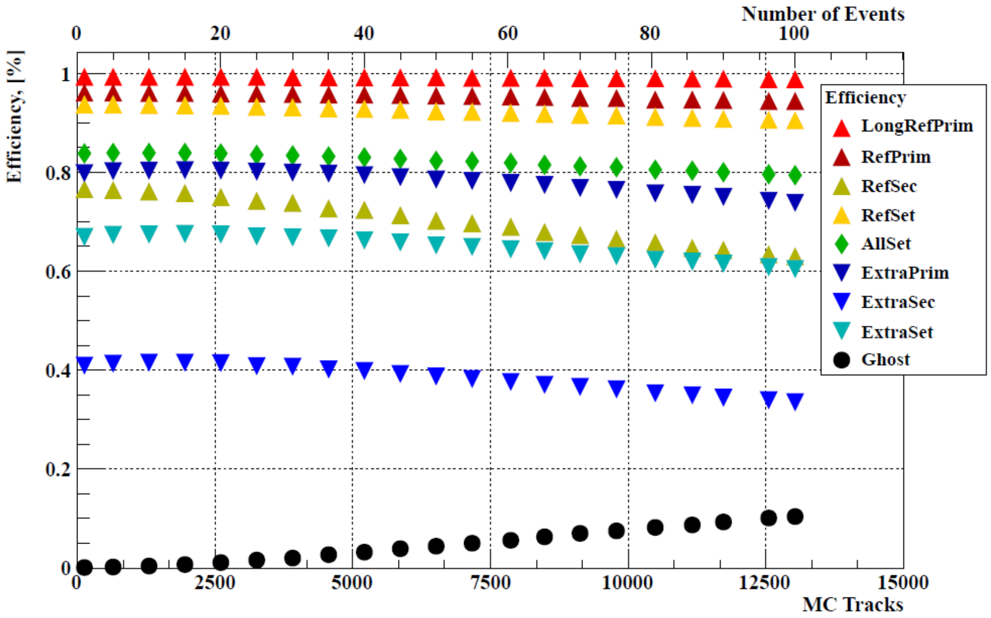


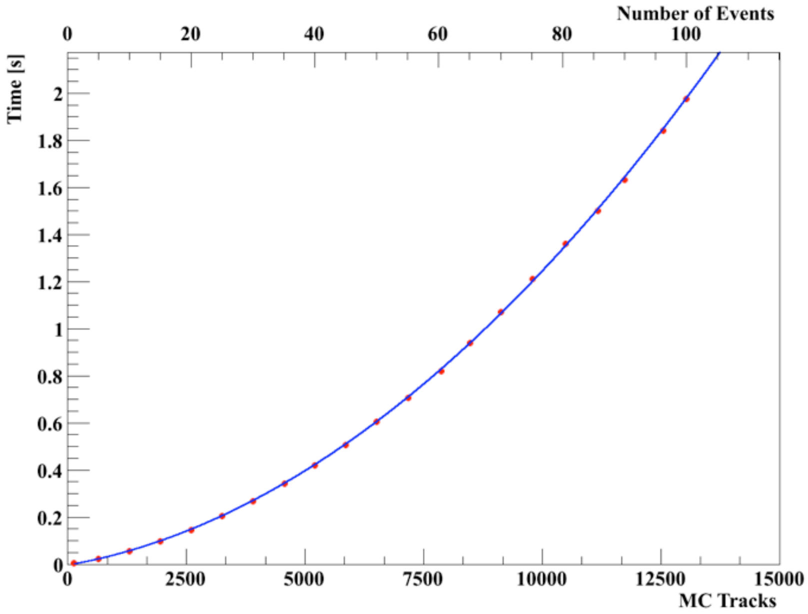
100 minimum bias events
$\langle N_{reco} \rangle = 10340$

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

Towards 4D (space+time) event reconstruction

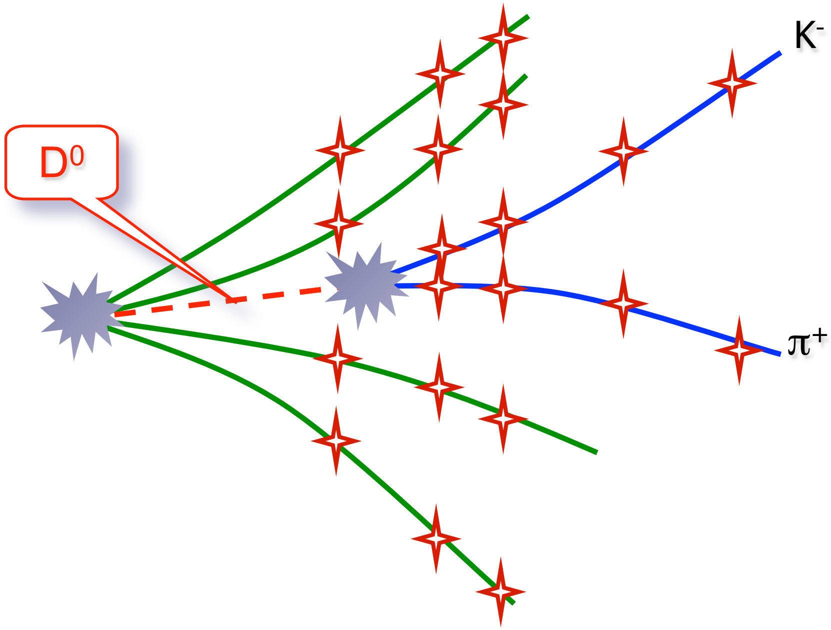
CBM CA Track Finder: Efficiency and Time vs. Track Multiplicity





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

CBM KF Particle: Vertices and Decayed Particles



State vector

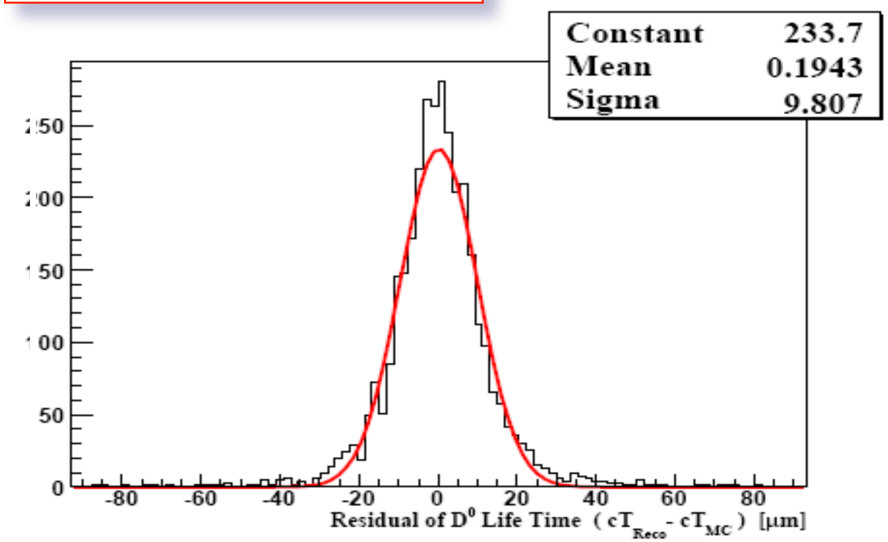
Position, direction, momentum and energy

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

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

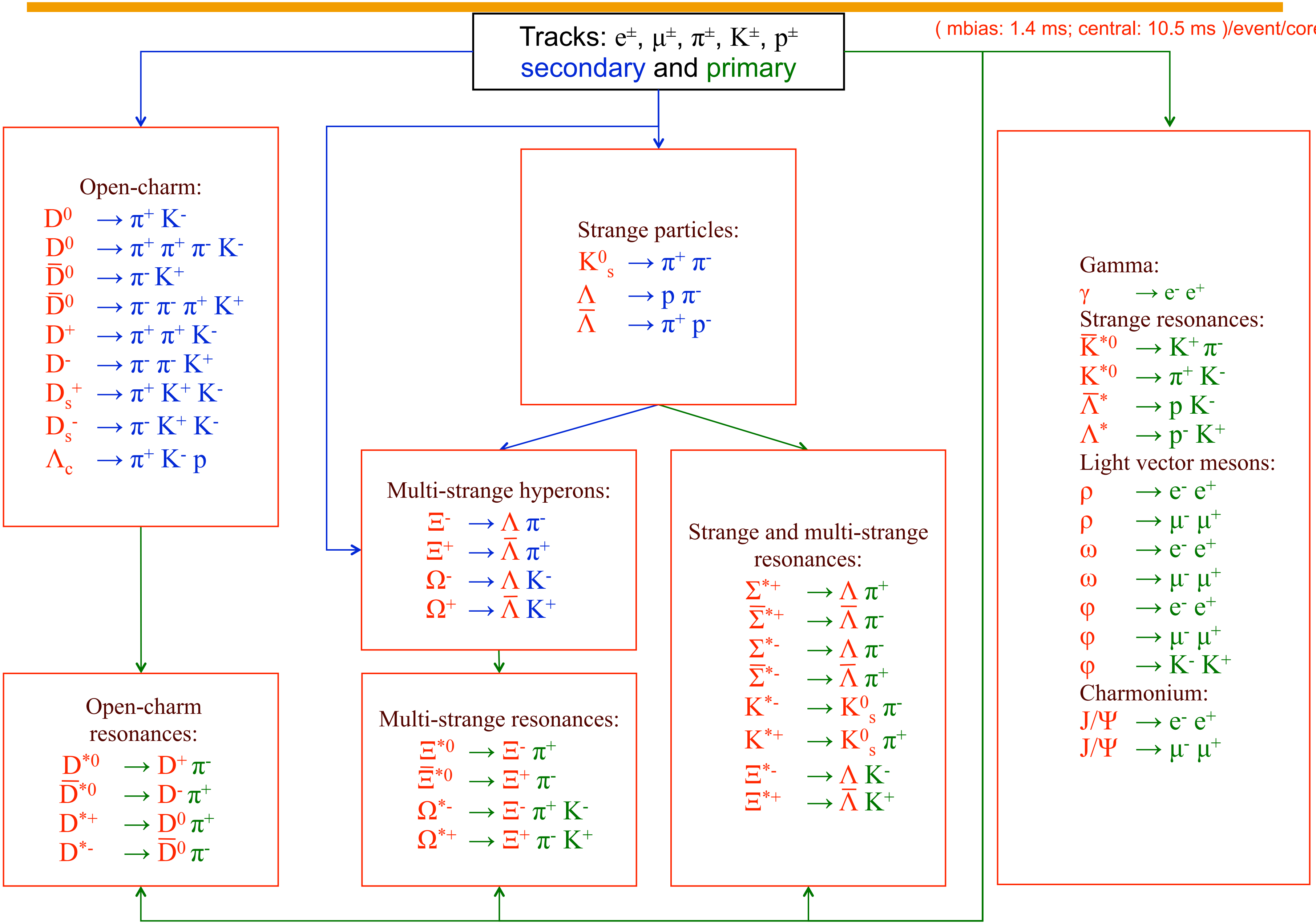
$x, y, z, p_x, p_y, p_z, E, m, L, c\tau$

```
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
```

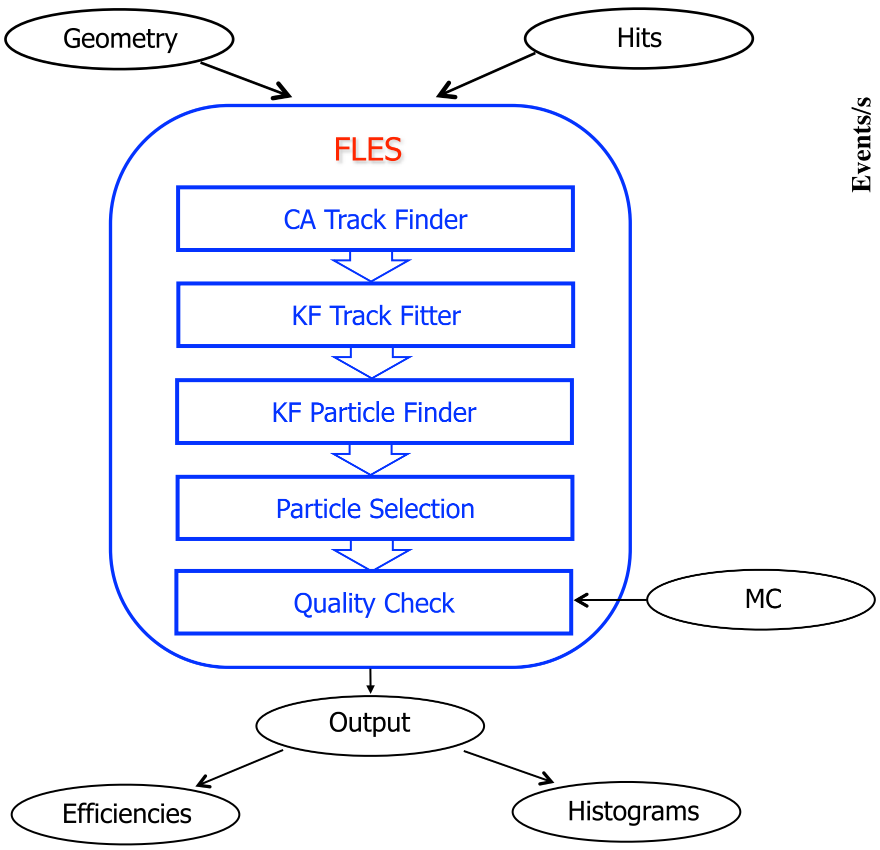


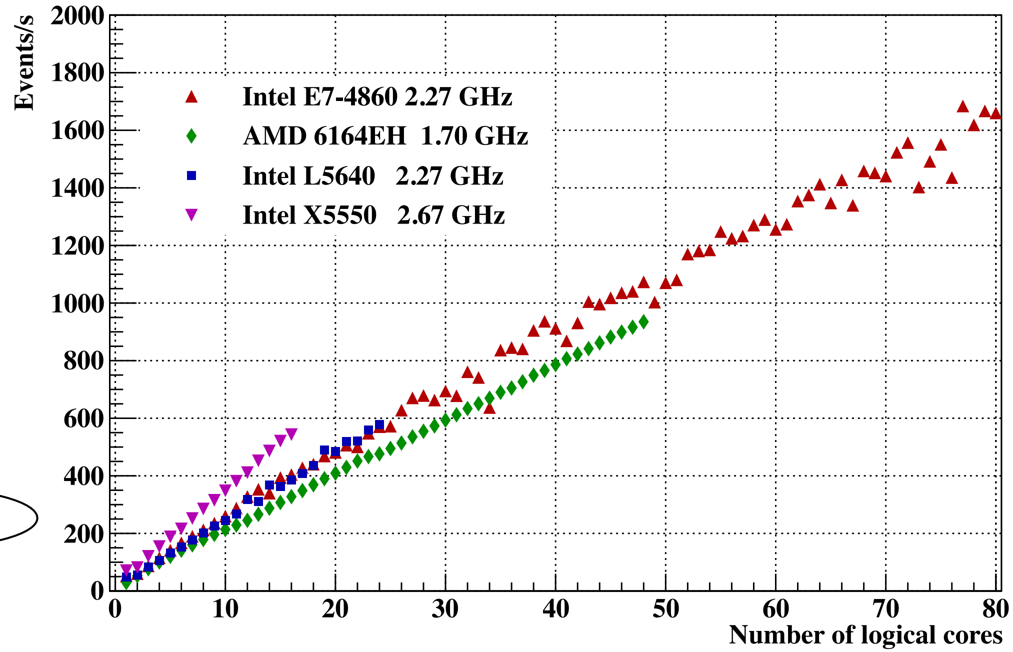
KFParticle provides uncomplicated approach to physics analysis (used in CBM, ALICE and STAR)

CBM KF Particle Finder for Physics Analysis and Selection



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

CBM: Parallelization in the Event Reconstruction

Algorithm	Vector SIMD	MultiThreading	CUDA	OpenCL CPU/GPU
Hit Producers				
STS KF Track Fit	✓	✓	✓	✓/✓
STS CA Track Finder	✓	✓		
MuCh Track Finder	✓	✓	✓	
TRD Track Finder	✓	✓	✓	
RICH Ring Finder	✓	✓		(✓/✓)
Vertexing (KFParticle)	✓	✓		
Off-line Physics Analysis	✓			
FLES Analysis and Selection	✓	✓		

Andrzej Nowak (OpenLab, CERN) by Hans von der Schmitt (ATLAS) at GPU Workshop, DESY, 15-16 April 2013

	SIMD	Instr. Level Parallelism	HW Threads	Cores	Sockets	Factor	Efficiency
MAX	4	4	1.35	8	4	691.2	100.0%
Typical	2.5	1.43	1.25	8	2	71.5	10.3%
HEP	1	0.80	1	6	2	9.6	1.4%
CBM@FAIR	4	3	1.3	8	4	499.2	72.2%

x Algorithm
x Memory

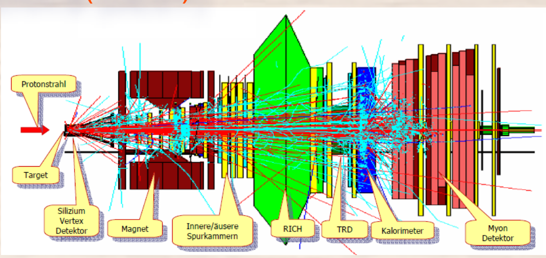
Parallelization becomes a standard in the CBM experiment

Software Evolution: Many-Core Barrier

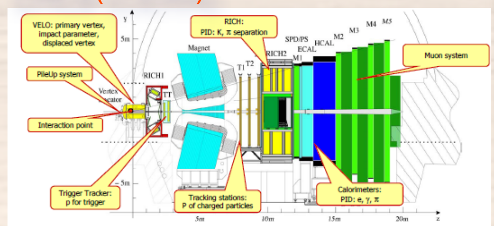
Scalar single-core OOP

Many-core HPC era

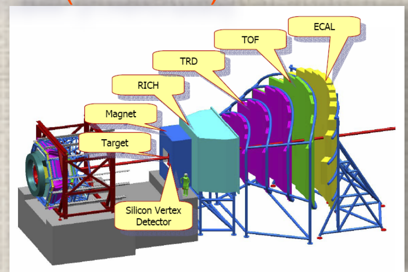
HERA-B (DESY)



LHCb (CERN)

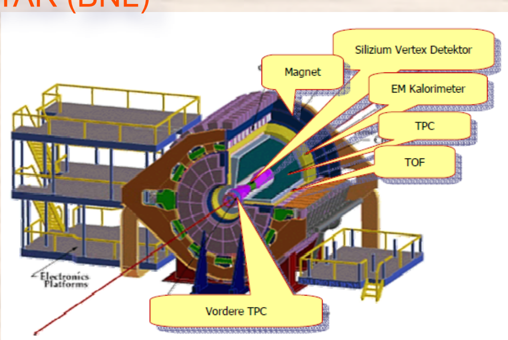


CBM (FAIR/GSI)

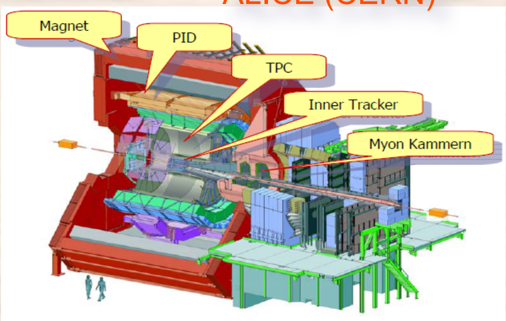


1990 2000 2010 t

STAR (BNL)



ALICE (CERN)



Consolidate efforts of:

- Physicists
- Mathematicians
- Computer scientists
- Developers of parallel languages
- Many-core CPU/GPU producers

1990 2000 2010 t

Software redesign can be synchronized between the experiments

Consolidate Efforts: Common Reconstruction Package

Uni-Frankfurt/FIAS:
Vector classes
CPU/GPU implementation

GSI:
Algorithms development
Many-core optimization

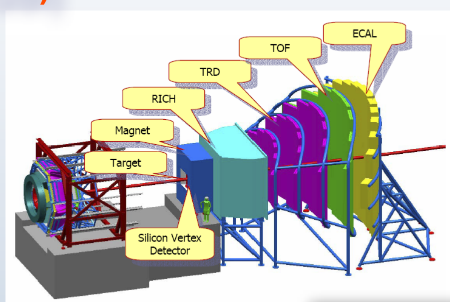
OpenLab (CERN):
Many-core optimization
Benchmarking

HEPHY (Vienna)/Uni-Gjovik:
Kalman Filter track fit
Kalman Filter vertex fit

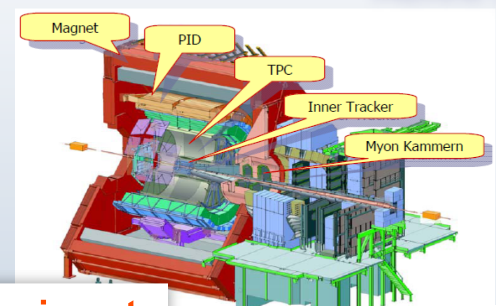
Common Reconstruction Package

Intel:
ArBB/OpenCL implementation
Benchmarking

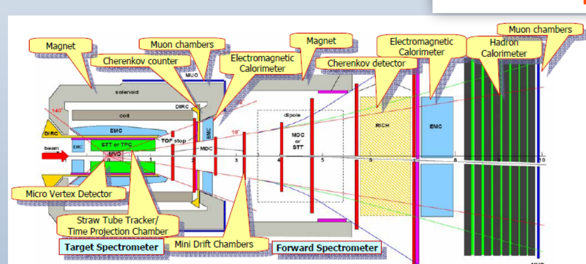
CBM (FAIR/GSI)

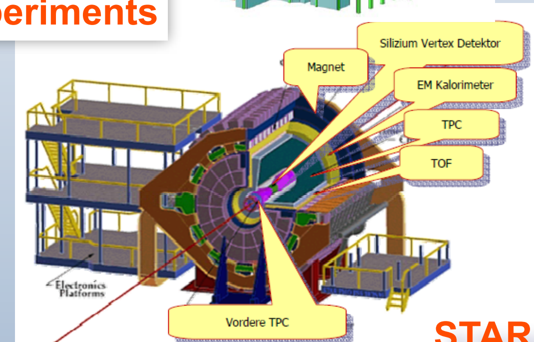


ALICE (CERN)



Host Experiments





PANDA (FAIR/GSI)

STAR (BNL)

Consolidate Efforts: International Workshops

International Workshop for Future Challenges in Tracking and Trigger Concepts

1st GSI, Darmstadt, Germany, 07-11.06.2010;
2nd CERN, Geneva, Switzerland, 07-08.07.2011;
3rd FIAS, Frankfurt, Germany, 27-29.02.2012;
4th CERN, Geneva, Switzerland, 28-30.11.2012;
5th BNL, Brookhaven or LBNL, Berkeley, USA, this year.

Contact: Sverre Jarp, Jerome Lauret, Ivan Kisel