

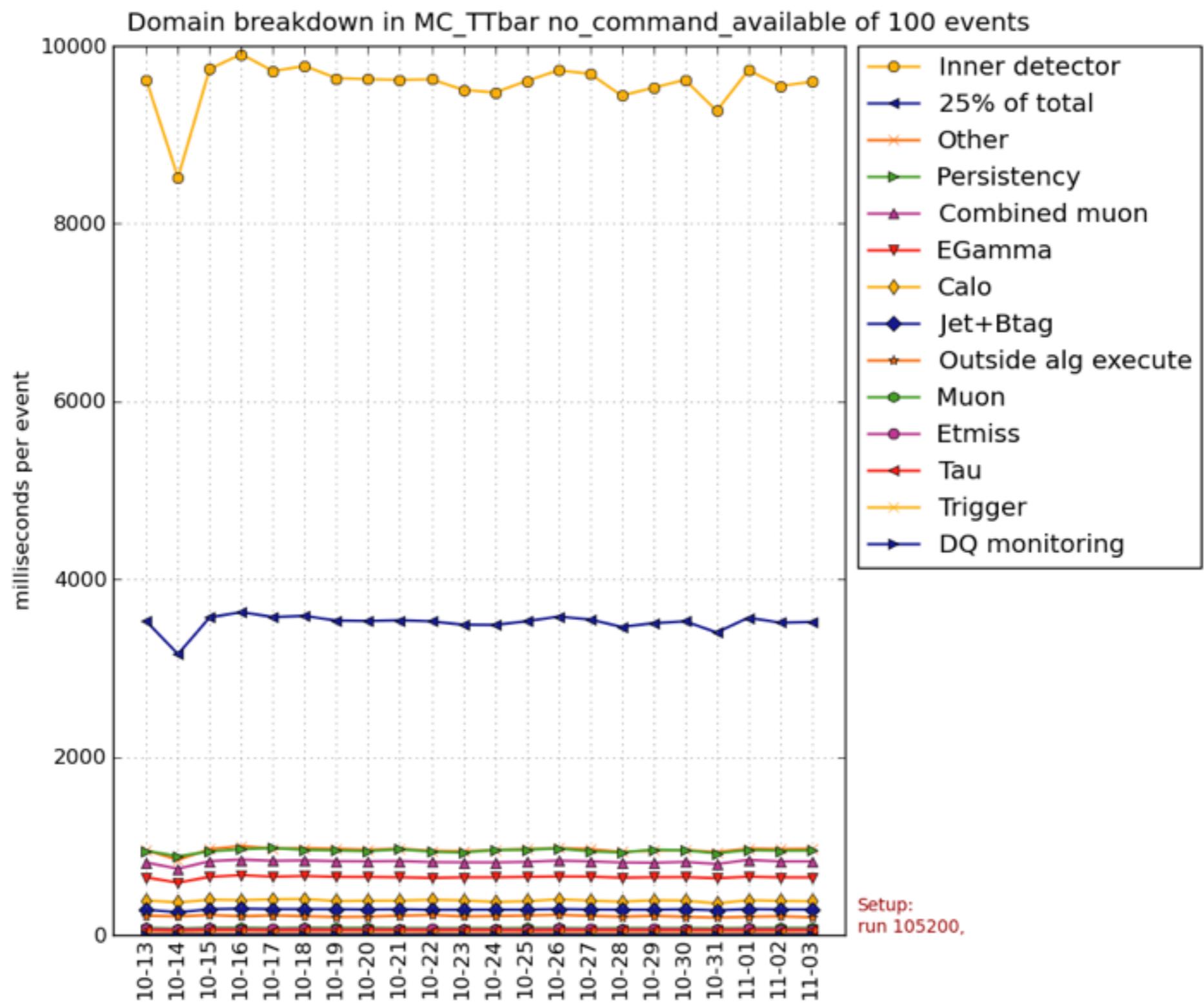


ATLAS ID Track Reconstruction: - a status quo and our potentials

A. Salzburger, CERN



Why do I talk “only” about the Inner Detector ?



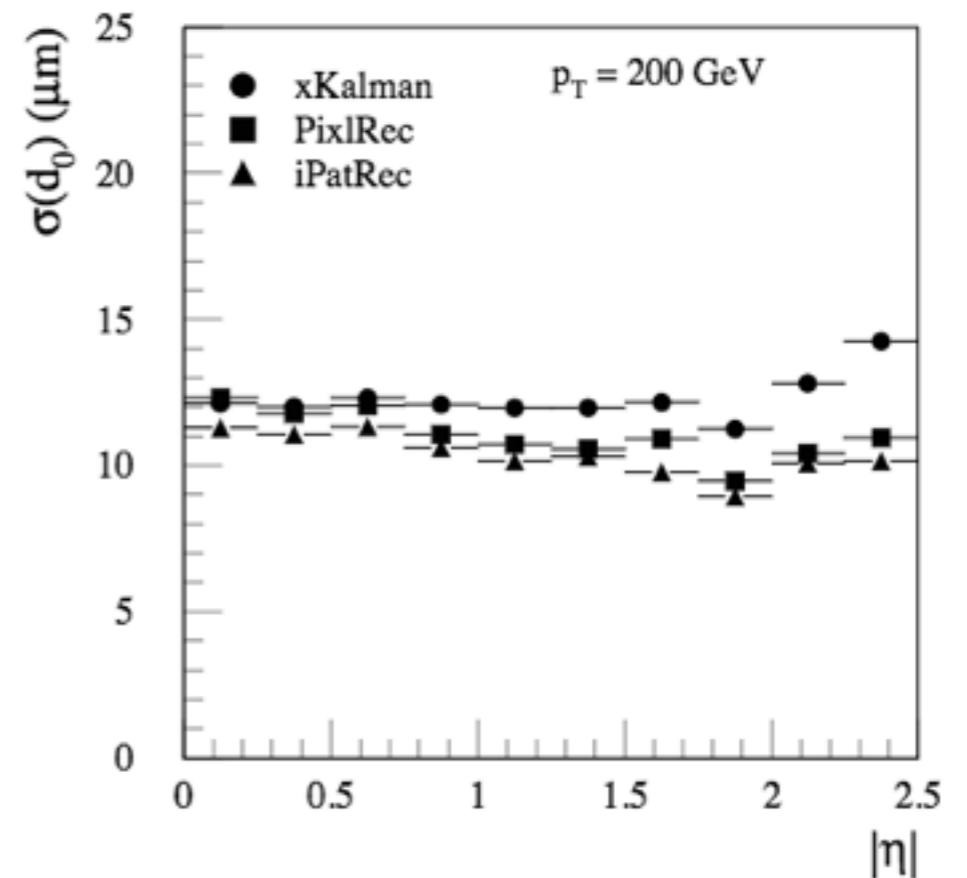
sample : ttbar MC 13 TeV with mu flat from 0 to 40

A historical review - the origin & construction



A historical review (1) - early days

- ▶ ATLAS ID track reconstruction roots date back more than 20 years
 - Inner Detector TDR, Volume I (1997)
reconstruction performed within the FORTRAN-based ATRECON framework, featuring two main “monolithic” reconstruction programs
 - *xKalman & iPatRec*
- ▶ ATLAS shifted to Gaudi-Athena in late 2001 (*I joined ATLAS*)
 - both xKalman and iPatRec were ‘transcribed’ into C++(ish)
translated into the Gaudi-Athena structure of Algorithms and Tools
main shortcomings:
 - *no common Event Data Model*
 - *no common interface to conditions, alignment*
 - *little flexibility for inclusion of new approaches, modules*



A historical review (2) - the RTF

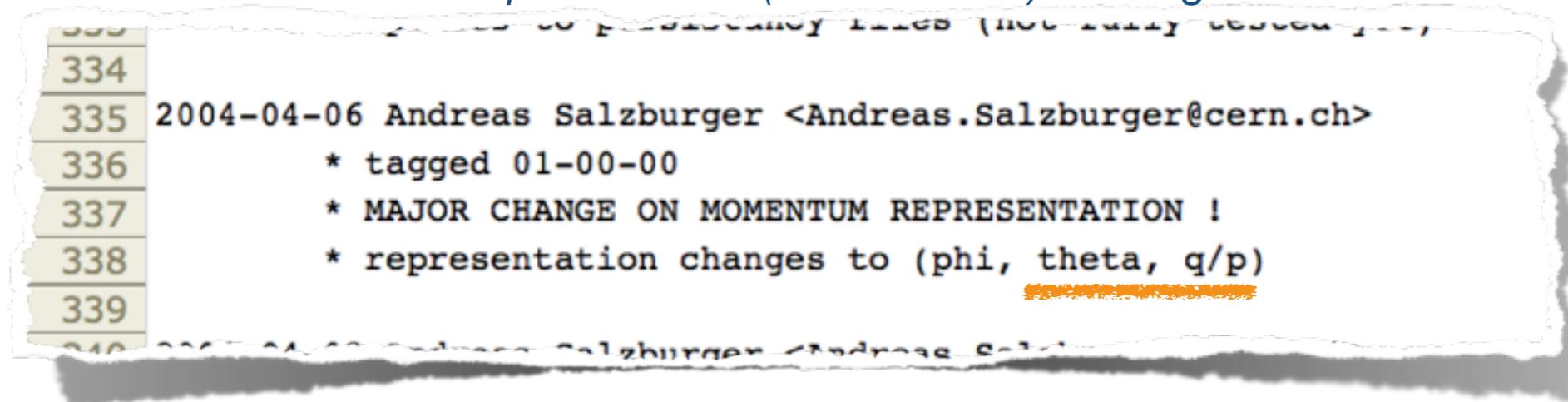
- ▶ ATLAS Reconstruction Task Force (2003)
 - Final RTF Report released in September 2003 (I started working on ATLAS tracking)
[ATL-SOFT-2003-010](#)

start of the New Tracking/Common Tracking project

- Main recommendations:
 - establishment of a common Event Data Model (EDM)
[ATL-SOFT-PUB-2006-004](#), [ATL-SOFT-PUB-2007-003](#)
 - description of ID track reconstruction by:
a sequence of Algorithms using Tools (defined by IAlgTool type interfaces)
[ATL-SOFT-PUB-2007-007](#)
 - encapsulation of common tools (e.g. extrapolation, fitting, calibration)
[ATL-SOFT-2007-005](#)
 - centralise access to common services (e.g. magnetic field, conditions, etc.)
 - split of detector specifics (ID/MS) and common tracking tools
- Additional developments triggered by this:
 - a new tracking geometry description: [ATL-SOFT-PUB-2007-004](#)
 - a fast simulation program (Fatras): [ATL-SOFT-PUB-2008-001](#)

A historical review (3) - from RTF to Run-1

- ▶ Run-1 tracking setup was developed within 4 years
 - Final RTF Report released in September 2003 to NewT note in 2007
[ATL-SOFT-2003-010](#) to [ATL-SOFT-PUB-2007-007](#)
 - Phase 1:
 - review of internal EDM of iPatRec and xKalman
 - establishment of a new EDM based on the two programs, with adaptions
 - *accommodate the MuonSpectrometer (it's a Tracker!) in design*



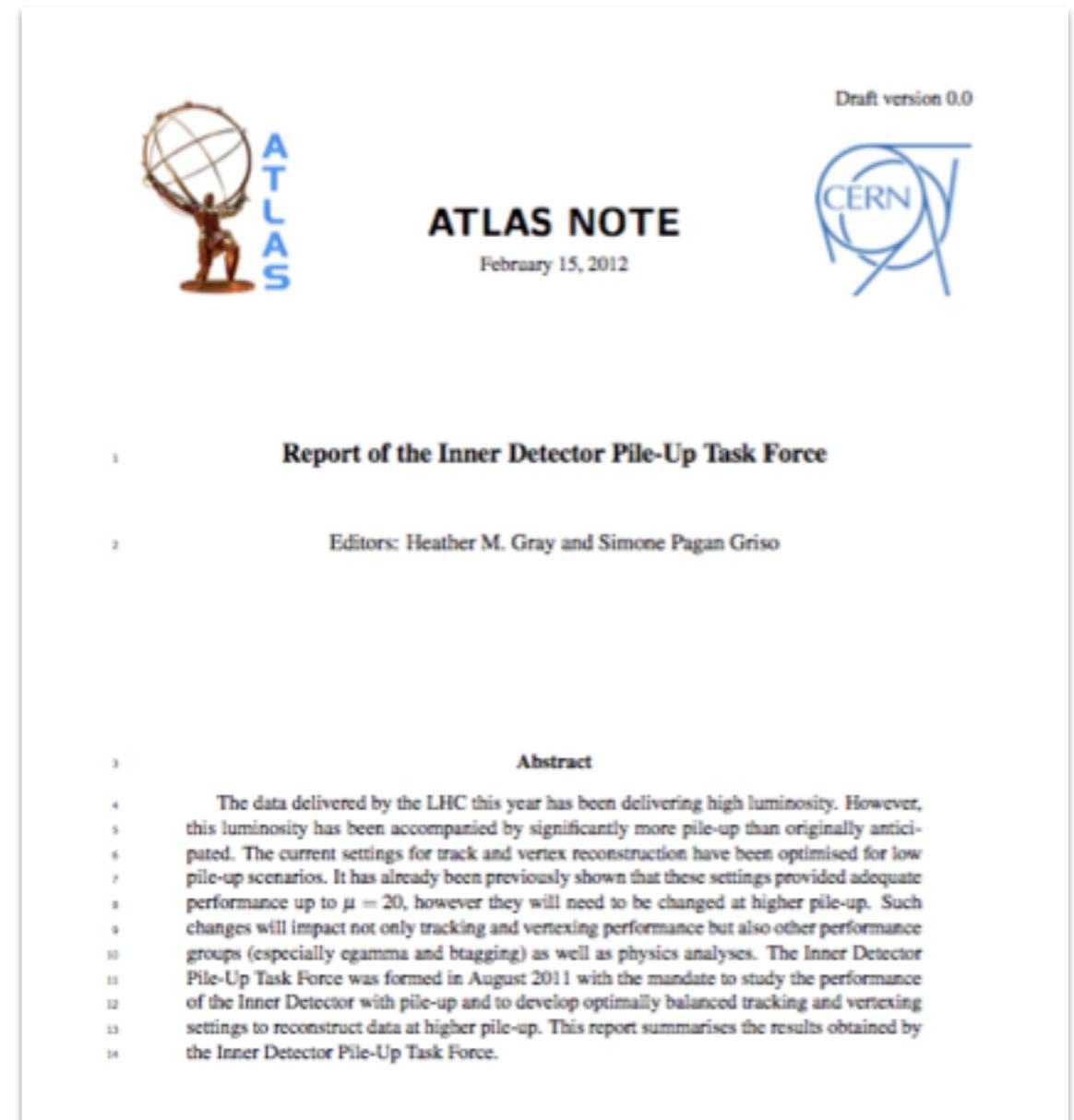
- *introduction of calibrated measurements (from RIO to RIO_onTrack)*
- *establishment of EDM persistency*
(a long history of POOL to T/P conversion)

A historical review (4) - during Run-1

- Run-1 tracking setup was modified substantially during Run-1:

CPU/physics performance driven

- Run-1 LHC performance created some CPU tension
- $\langle\mu\rangle$ was to exceed design average of 23 for 50 ns run of 2011/12
- established a ID Pile-Up task force
 - deal with the CPU problem
 - deal with the increased fake-rate
 - establish a robust vertex reconstruction



Draft version 0.0

ATLAS NOTE
February 15, 2012

CERN

Report of the Inner Detector Pile-Up Task Force

Editors: Heather M. Gray and Simone Pagan Griso

Abstract

The data delivered by the LHC this year has been delivering high luminosity. However, this luminosity has been accompanied by significantly more pile-up than originally anticipated. The current settings for track and vertex reconstruction have been optimised for low pile-up scenarios. It has already been previously shown that these settings provided adequate performance up to $\mu = 20$, however they will need to be changed at higher pile-up. Such changes will impact not only tracking and vertexing performance but also other performance groups (especially egamma and btagging) as well as physics analyses. The Inner Detector Pile-Up Task Force was formed in August 2011 with the mandate to study the performance of the Inner Detector with pile-up and to develop optimally balanced tracking and vertexing settings to reconstruct data at higher pile-up. This report summarises the results obtained by the Inner Detector Pile-Up Task Force.

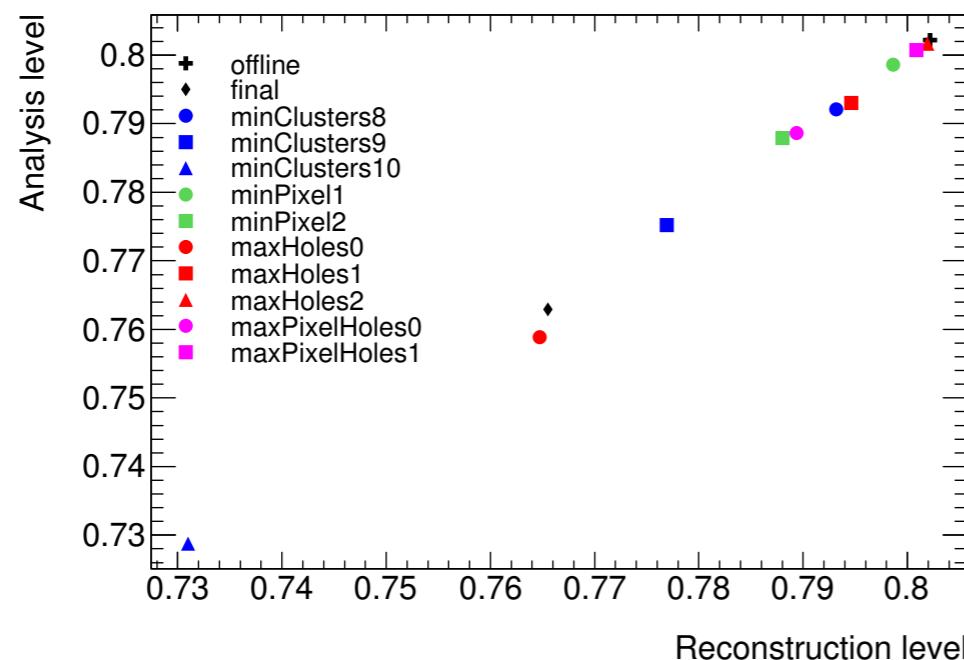
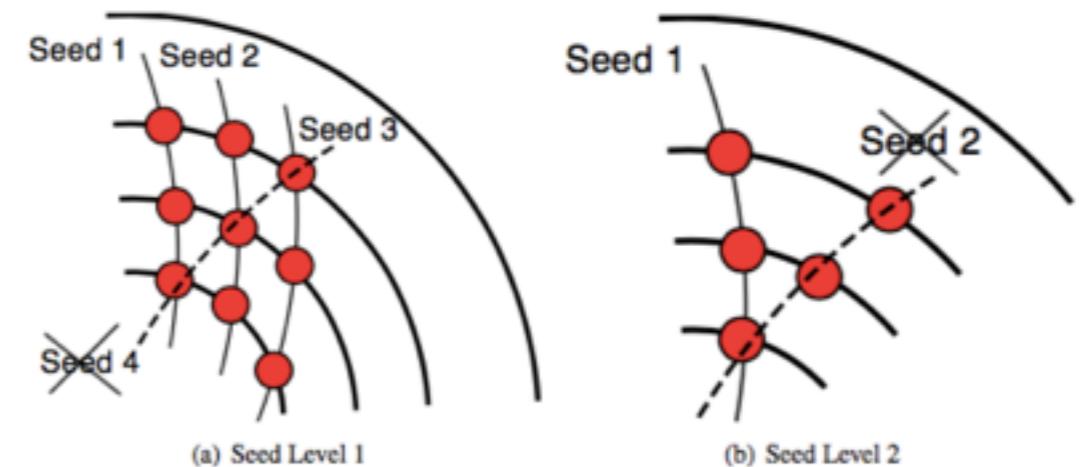
A historical review (5) - during Run-1

- Run-1 tracking setup was modified substantially during Run-1:

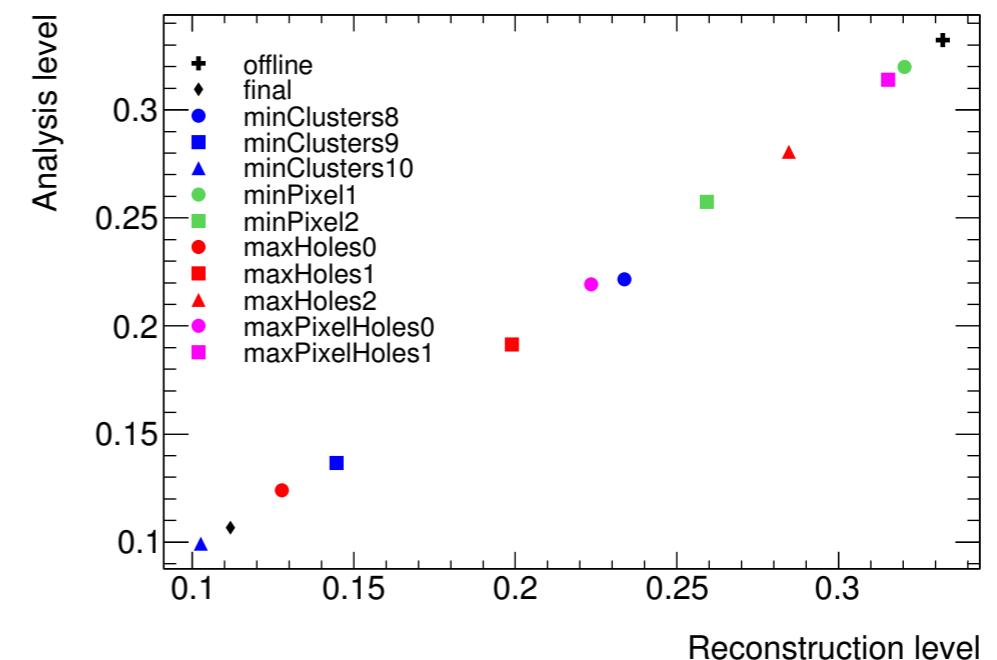
CPU/physics performance driven

- major changes on how the seeding is done (with the aim of having less track candidates at input)
- robust tracking cut study to control fake contribution

slight loss of track reconstruction efficiency accepted in order to control fake contribution



(a) Signal primary efficiency: Analysis vs reco level



(b) Non-primaries: Analysis vs reco level

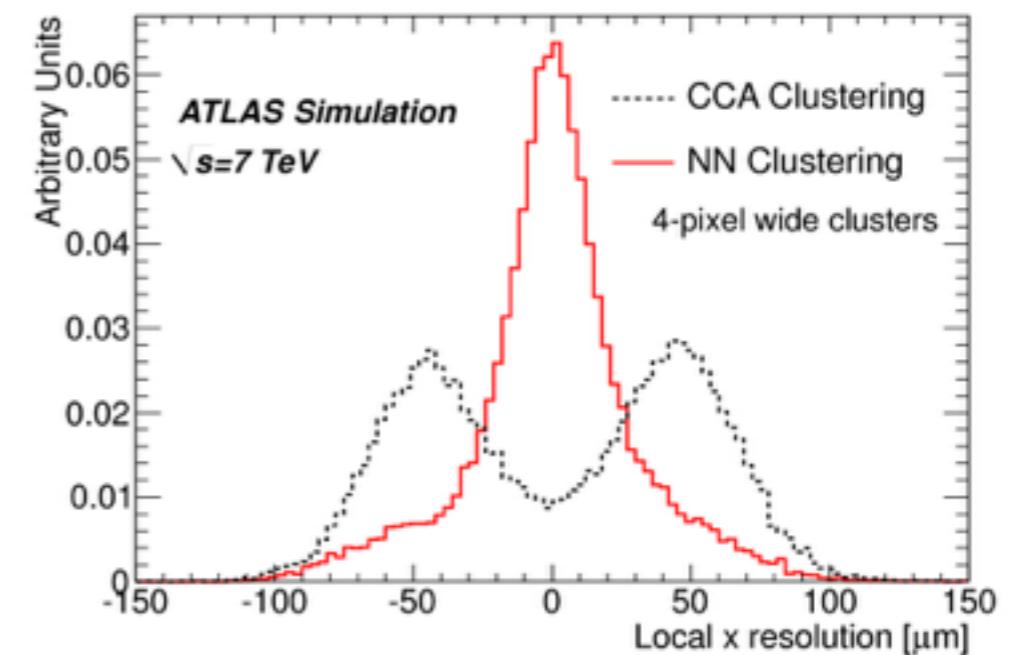
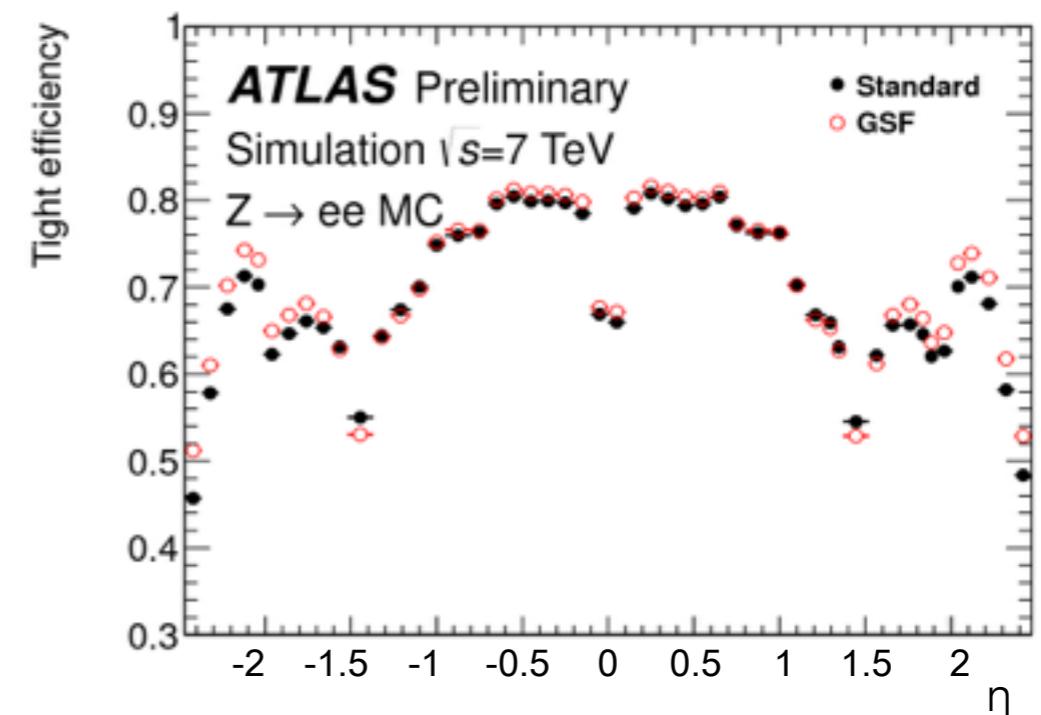
A historical review (6) - during Run-1

- Run-1 tracking setup was modified substantially during Run-1:
physics performance driven

- a dedicated electron pattern recognition
 - switching to a calorimeter-seeded region of interest tracking
 - *boost in electron reconstruction efficiency*

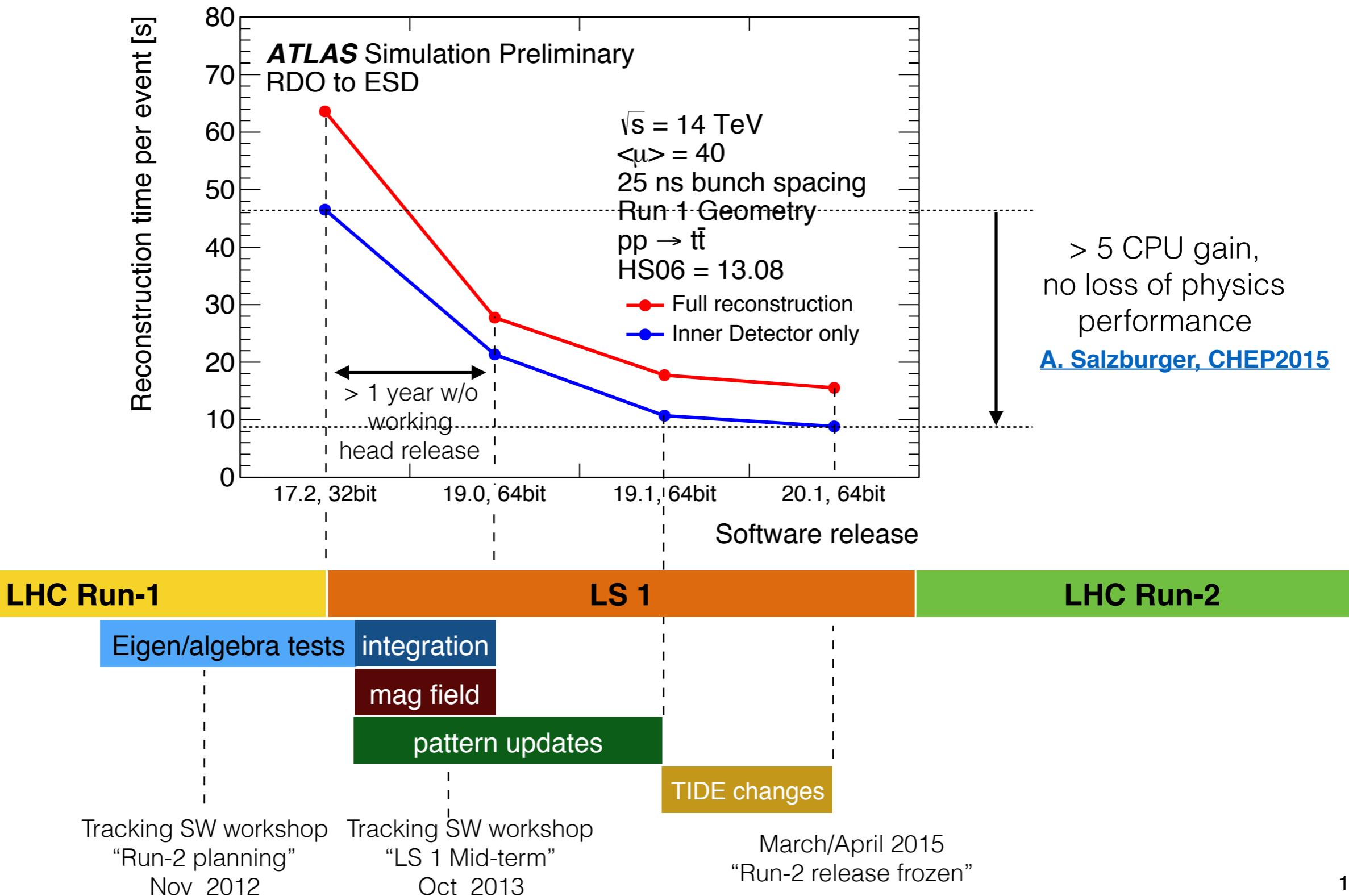
[ATLAS-CONF-2012-047](#)

- the neural network based pixel clustering
 - outcome of the Pixel Cluster Task Force
 - *using a set of neutral networks to estimate the probability of pixel clusters being created by multiple particles*
 - *attempt to split them into sub clusters*
 - [JINST 9 \(2014\) P09009](#)
 - became later the new TIDE group in ATLAS



A historical review (7) - the LS-1 campaign

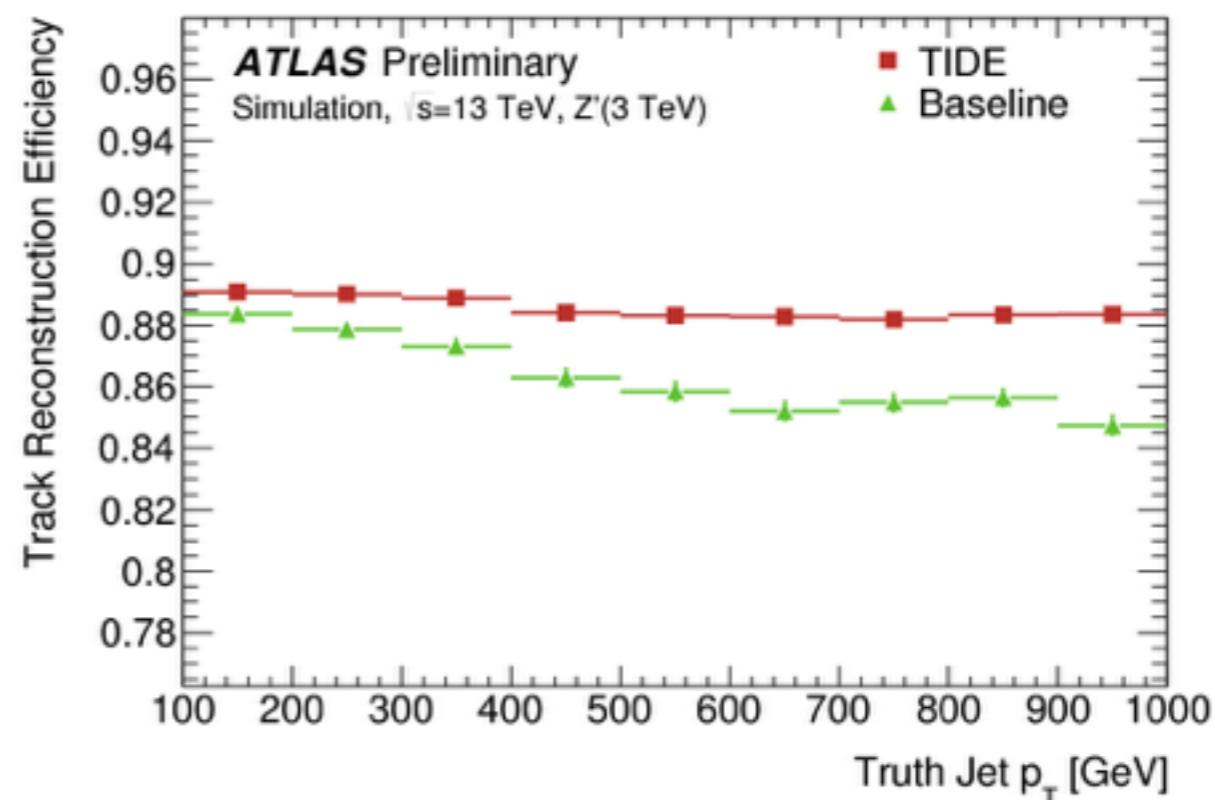
- CPU performance driven SW campaign to optimise the ID tracking



A historical review (8) - additional LS1 changes

personal

- ▶ Newly established TIDE (Tracking in Dense Environments) tracking
 - **former** (Run-1) strategy was to split clusters if they had a high probability of being created by more than one particle before pattern recognition
 - aim to split clusters into siblings in order to have unique hit assignment
 - **changed to** NN estimation of whether a hit is allowed to be changed during ambiguity solving
[ATL-PHYS-PUB-2015-006](#)
 - relax initial hit sharing restrictions and validate/quantify at later stage
 - speed up reconstruction since NN is only called on demand
 - tracks without shared hits do not require any action
 - hit book-keeping is still required (has consequences for concurrency)

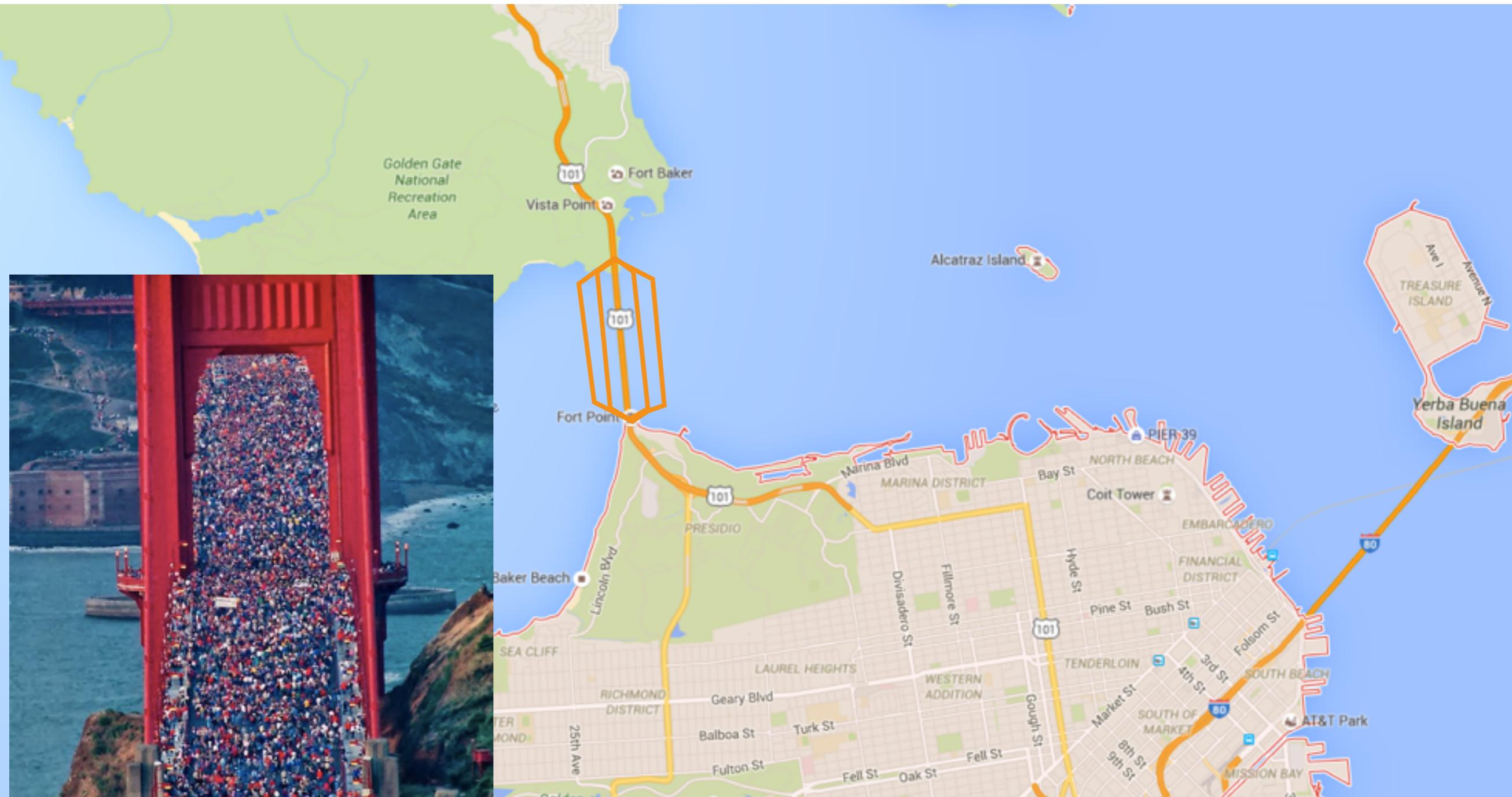


Run-2 status and evolution towards Run-3

- ▶ LS1 SW campaign has secured Run-2 data taking and MC production
 - combinatorial growth of ID track with $\langle\mu\rangle$ remains worry
 - is already of concern for Phase-2 Upgrade (Run-3) simulations expect $\langle\mu\rangle \sim 200$ at HL-LHC (and up to 1000 for FCC-hh studies)
- ▶ Major hotspots/bottlenecks have been removed from Tracking code
 - there are continuous improvements being implemented, maximal O(10%)
 - there are no obvious hotspots to tackle
- ▶ Need to look at different approaches and R&D
 - cut on physics performance (e.g. raising of momentum cut) ?
 - hard scatter (“triggered”) event reconstruction only ?
 - **evolution of code to exploit concurrency**
 - **new approaches in pattern recognition**

main topic of this talk

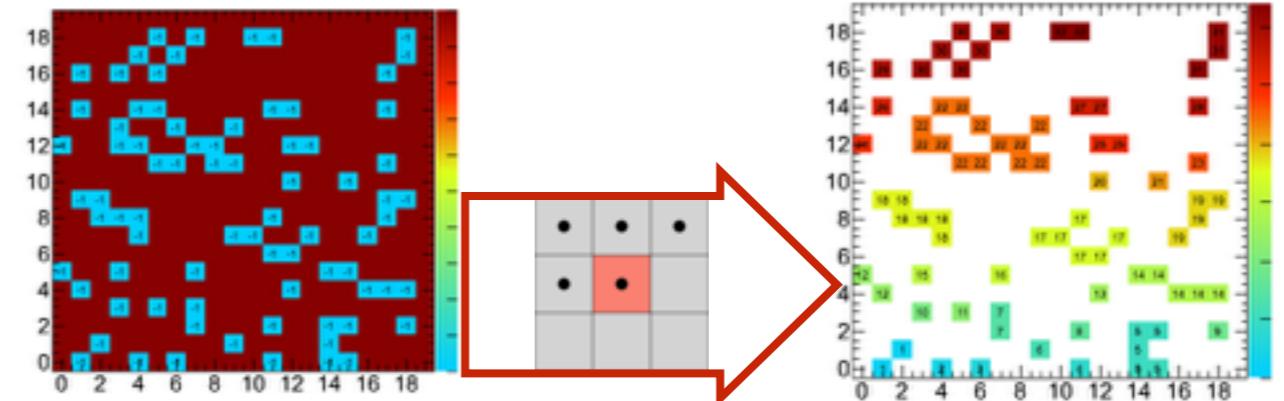
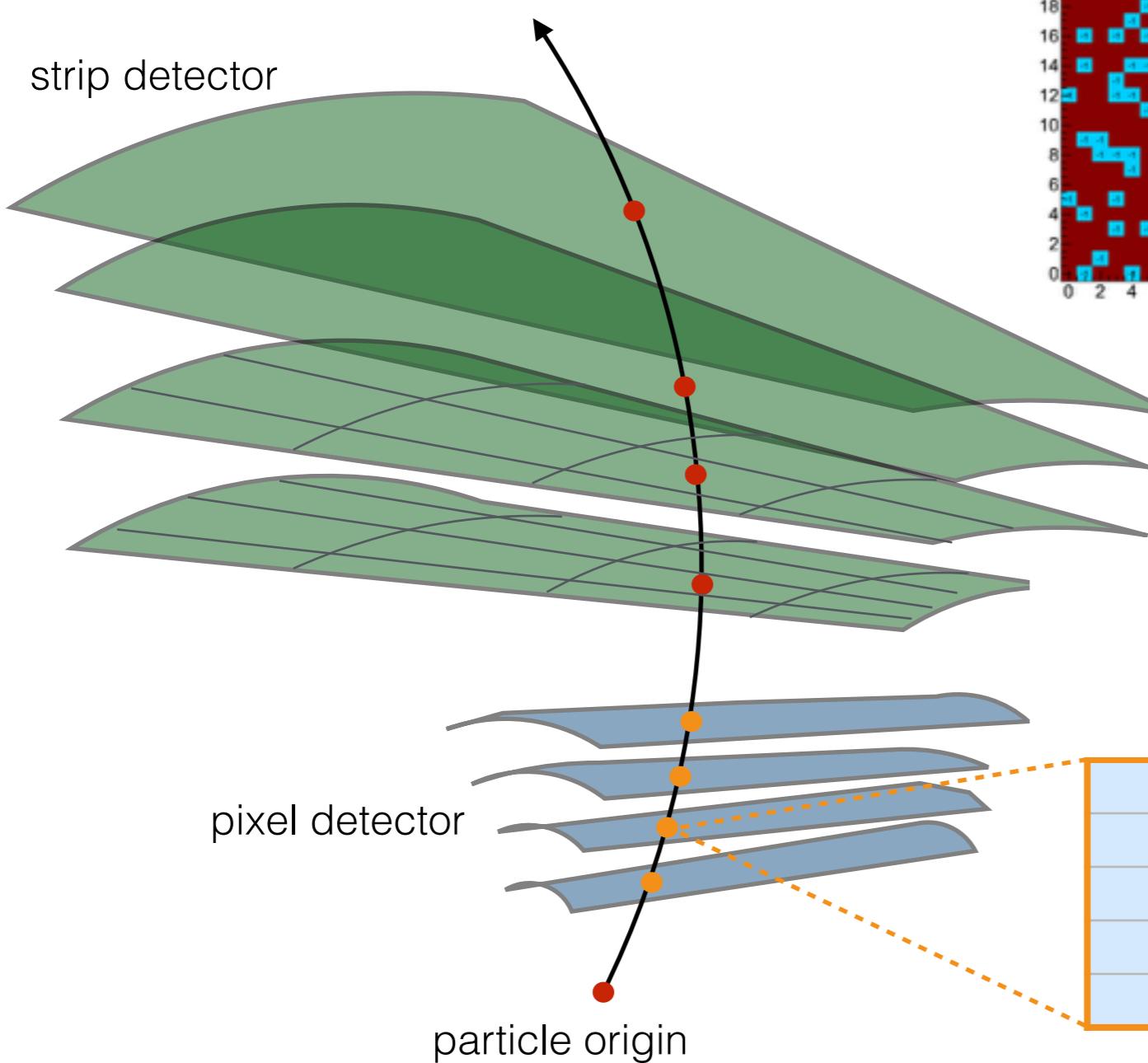
Parallel paths



The nutshell view - data preparation in silicon (1)

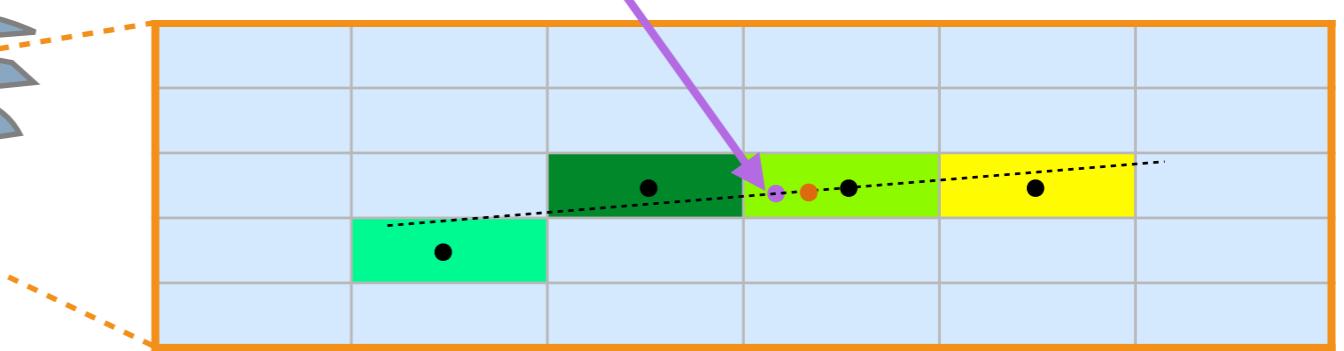
Clustering

finding connected cells (pixels/strips) on module via a connected component analysis



followed by estimation of cluster position
e.g. the charge-weighted approach :

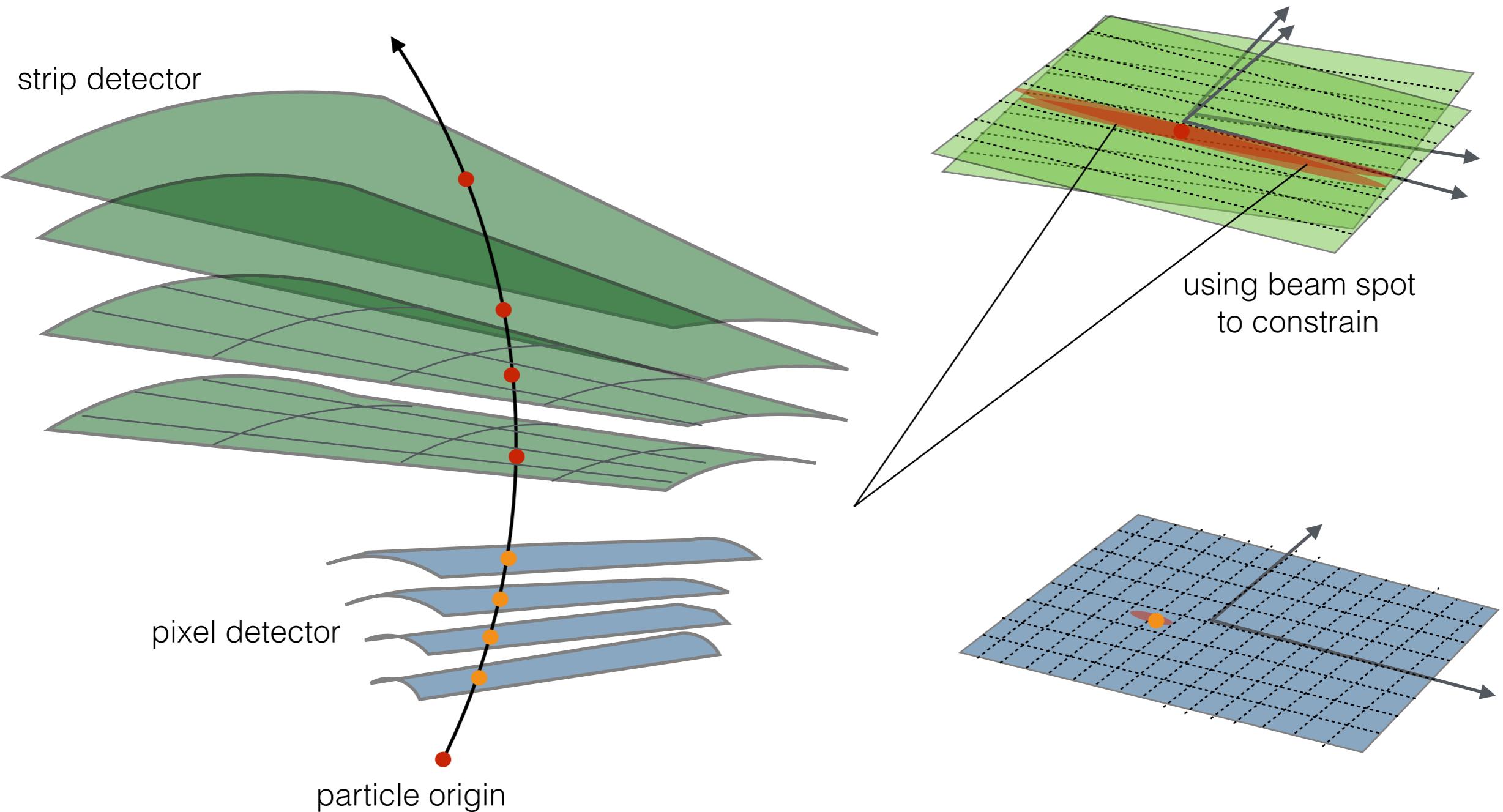
$$m = \frac{1}{\sum_{i=1,N} q_i} \sum_{i=1,N} q_i l_i \quad \begin{matrix} \text{charge collected} \\ \text{in cell } i \end{matrix}$$



The nutshell view - data preparation in silicon (2)

Space Point Formation

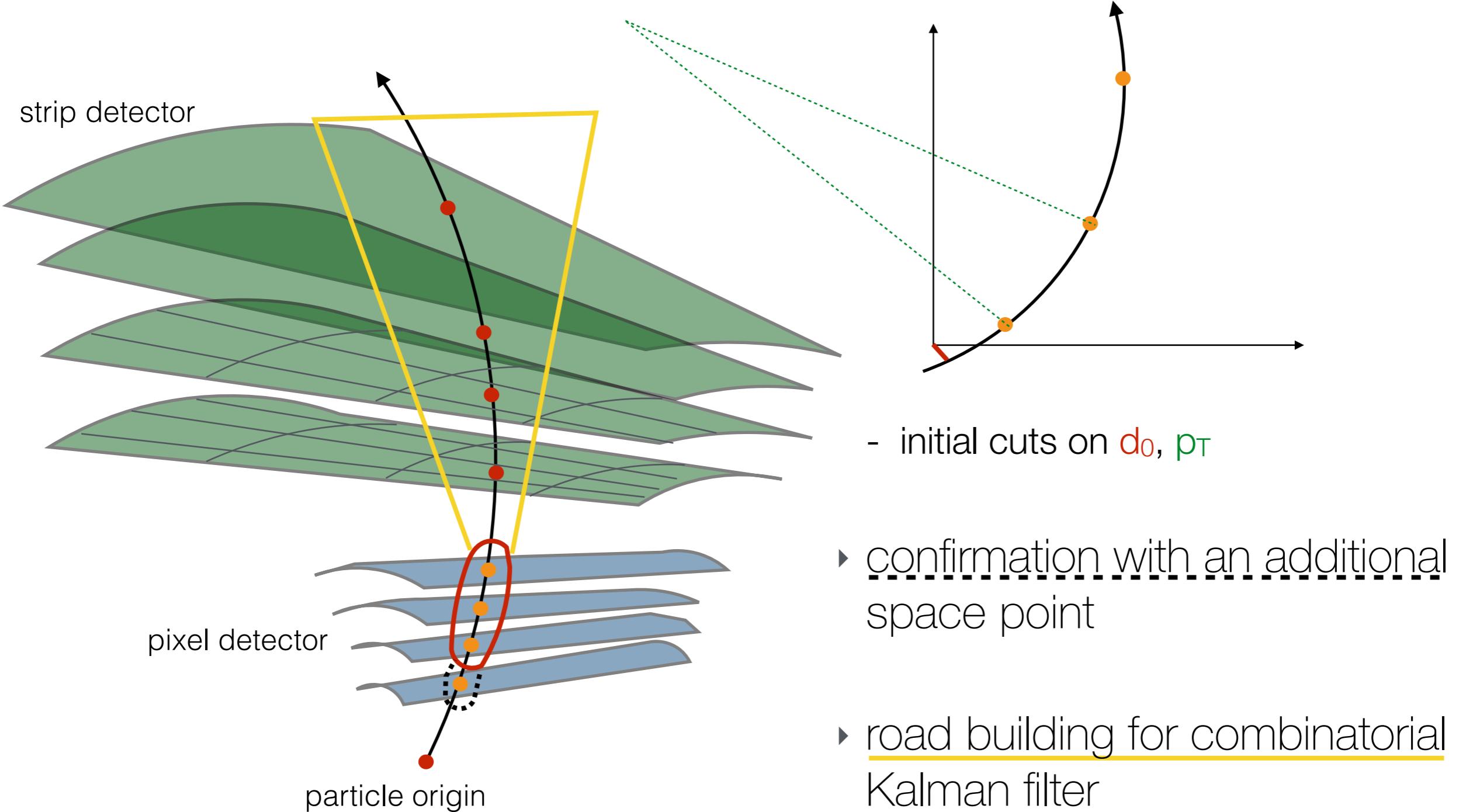
using the local cluster positions & sensor surface to form 3D space points



The nutshell view - track finding (1)

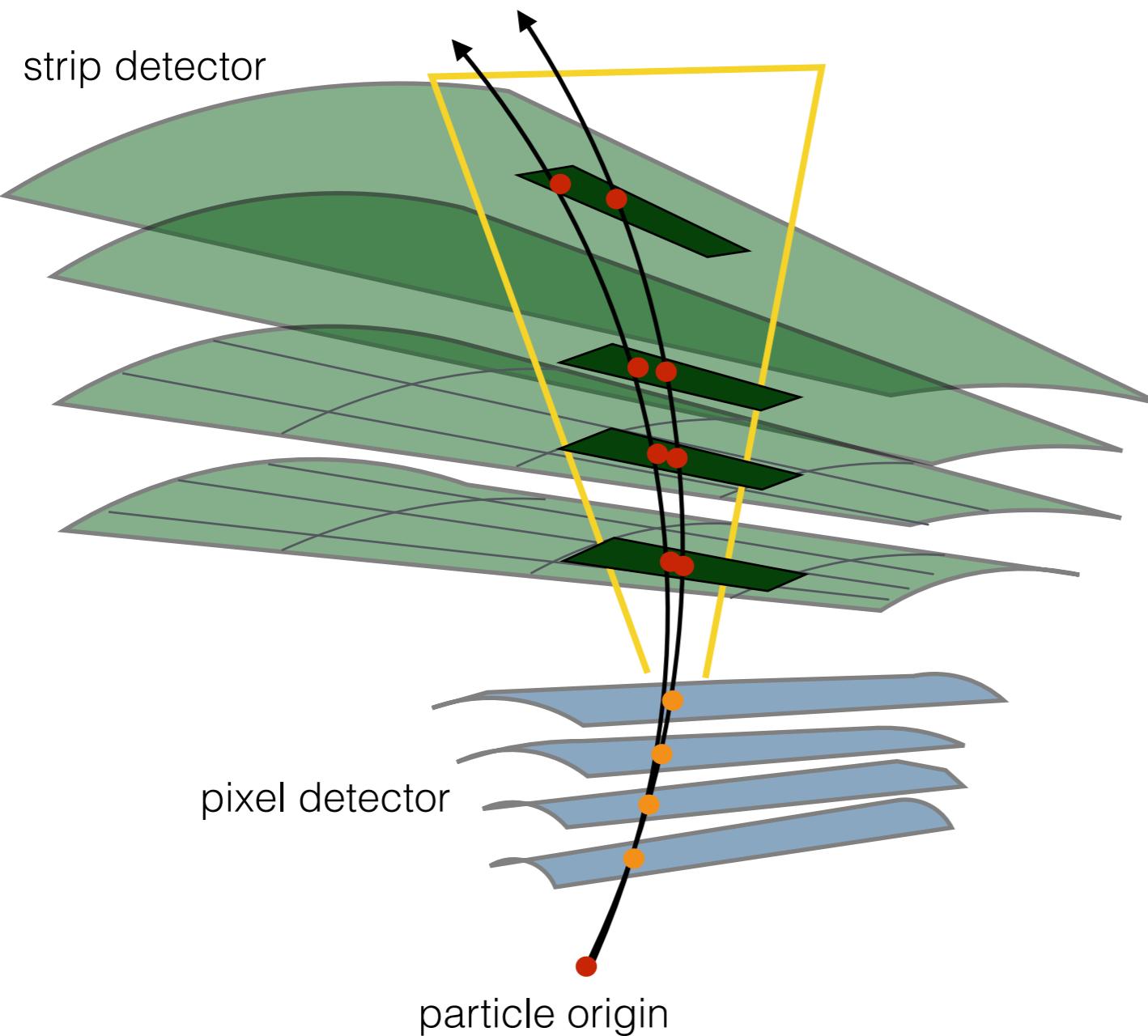
Space Point seeding

- ▶ building triplets of space points



The nutshell view - track finding (2)

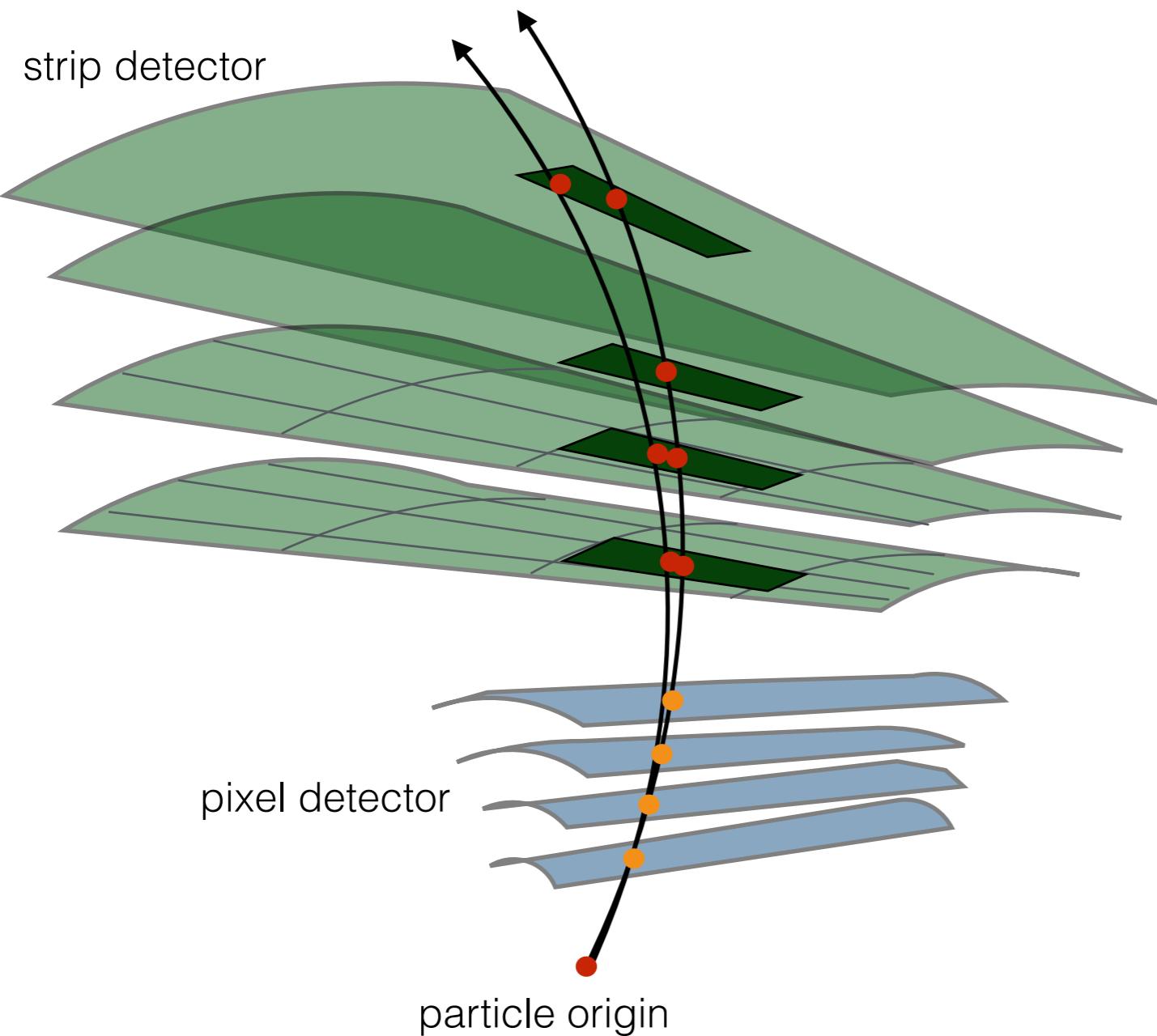
Space point seeded track finding



- ▶ resolve detector elements in a given road and start track candidate search
 - there can be more than one candidate per seed
 - there can be shared hit
 - there can be disregarded seeds
- ▶ dedicated road search for electrons allowing kinks due to energy loss
 - only allowed in seeded regions from the calorimeter

The nutshell view - ambiguity solving

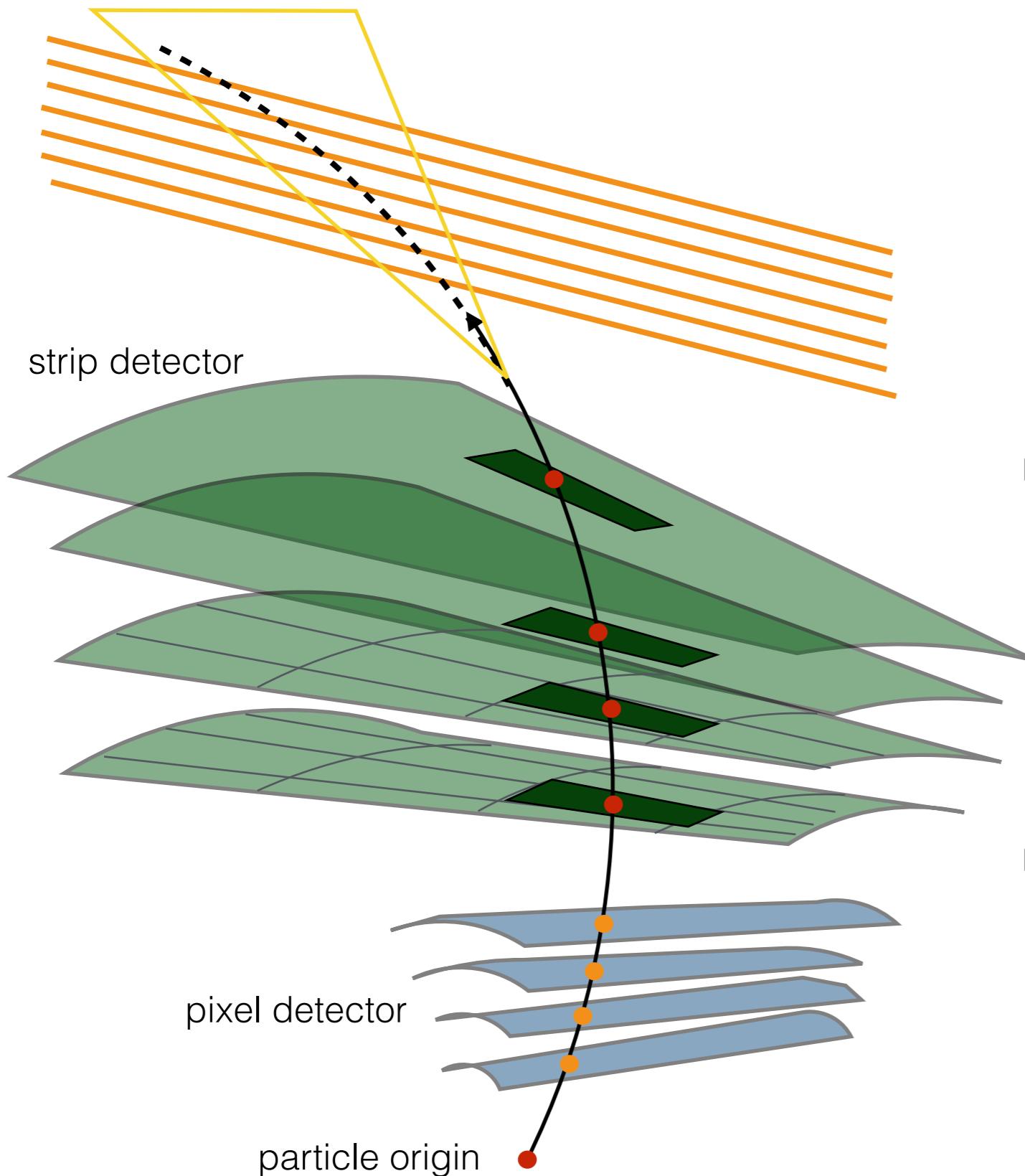
Evaluate the track candidates



- ▶ hit associations
 - using PrdAssociationTool
 - finding shared hits
 - evaluated shared/split probability
- ▶ tracks are ranked (scored)
 - using a scoring tool:
penalties for holes/shared hits
bonus points for hits/good chi²
 - classification problem
(NN based version exists)
- ▶ let tracks with highest score survive

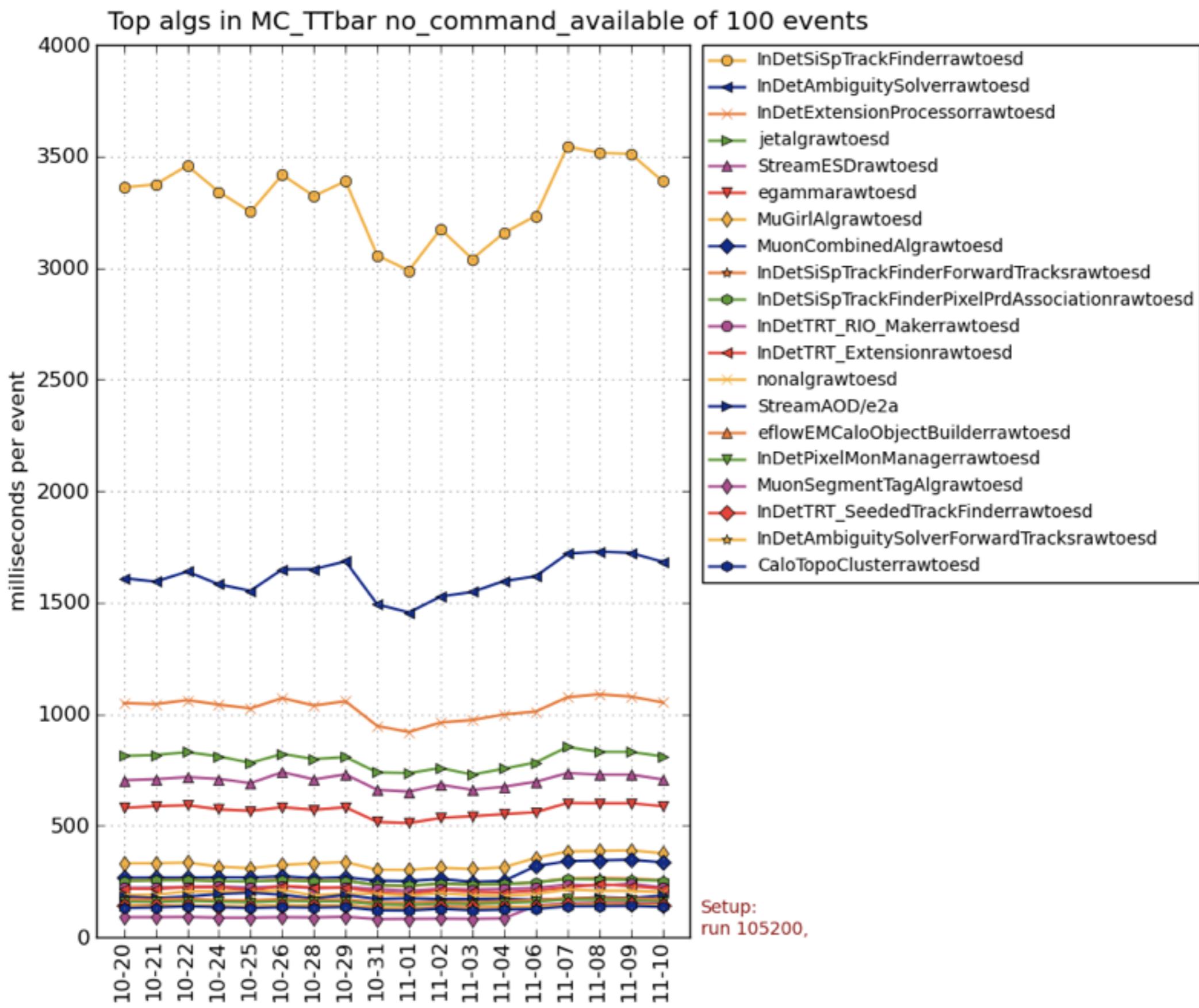
The nutshell view - TRT extension

Evaluate the track candidates



- ▶ Tracks found in Silicon are tested for extension into the TRT
 - road definition by silicon track
 - search for compatible hits and tested for extension using a Kalman filtering
- ▶ a first algorithm finds compatible hits by track following
 - sequential at input
 - first come, first serve
- ▶ a second algorithm evaluates the extension

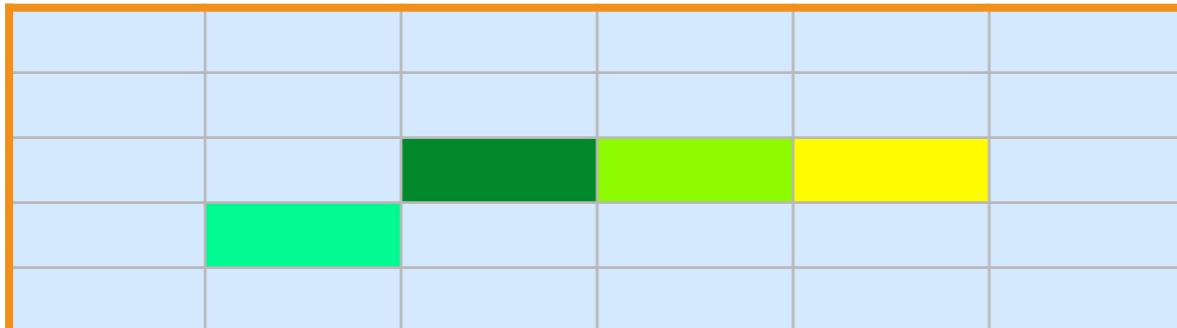
It's all about timing



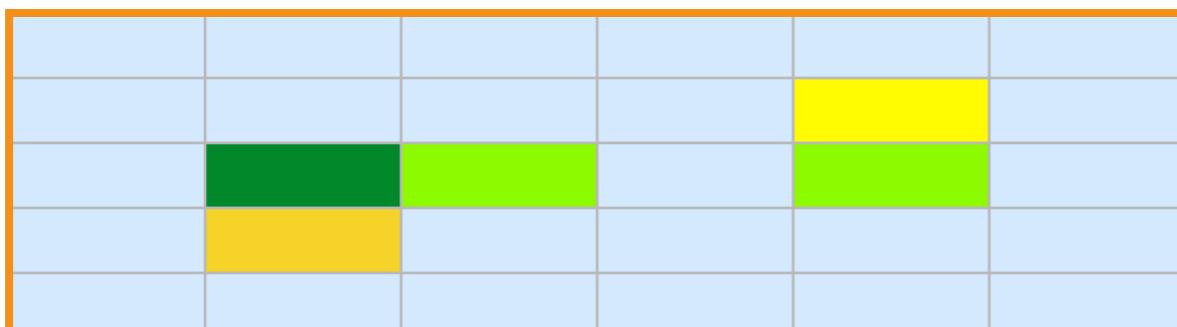
Concurrency usage - Cluster creation

- ▶ Cluster creation is the first stage of local pattern recognition

module i



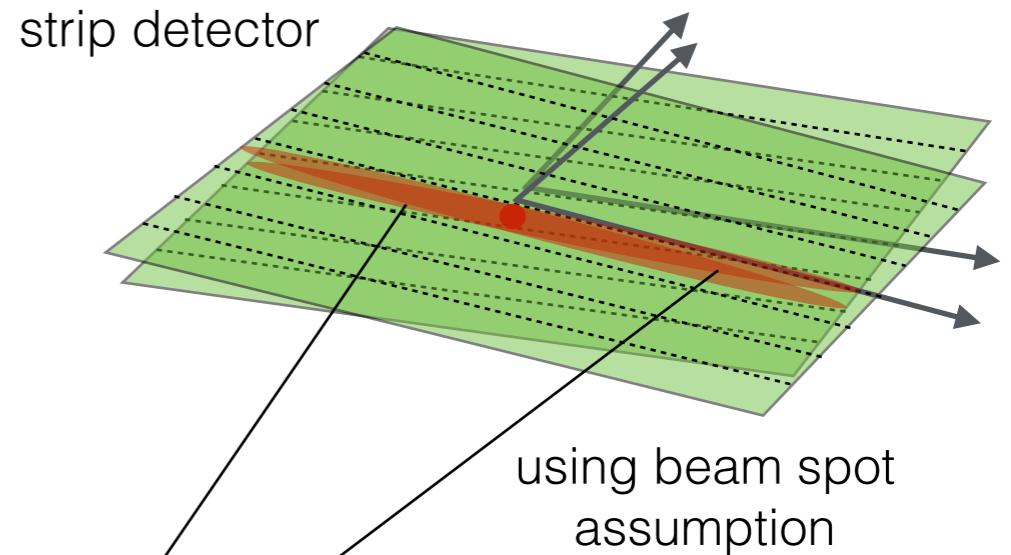
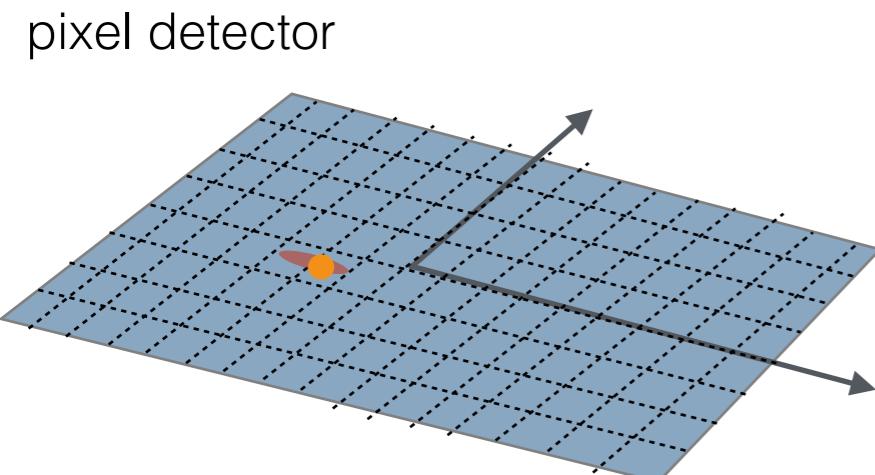
module $i+1$



- ▶ Connected component analysis (CCA) runs per module
- ▶ Could easily be parallelised
 - a perfect use-case for a HiveAlgorithm that opens up many (100s) threads
- ▶ No bookkeeping needed, since every module represents one collection in our Identifiable container module
- ▶ It will not gain us all too much CPU gain, but it is a very good testbed

Concurrency usage - Space point creation

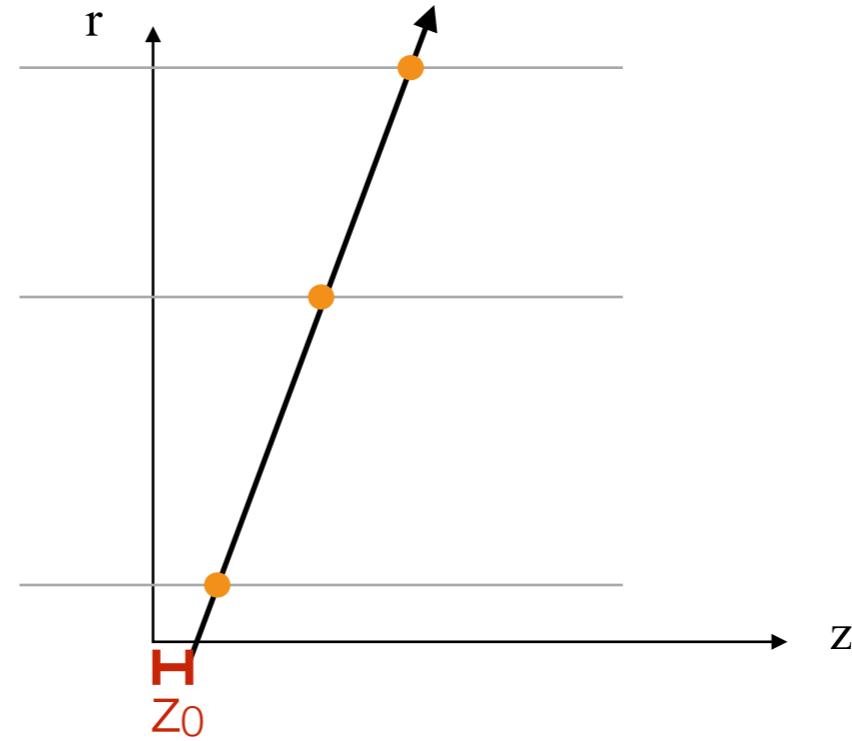
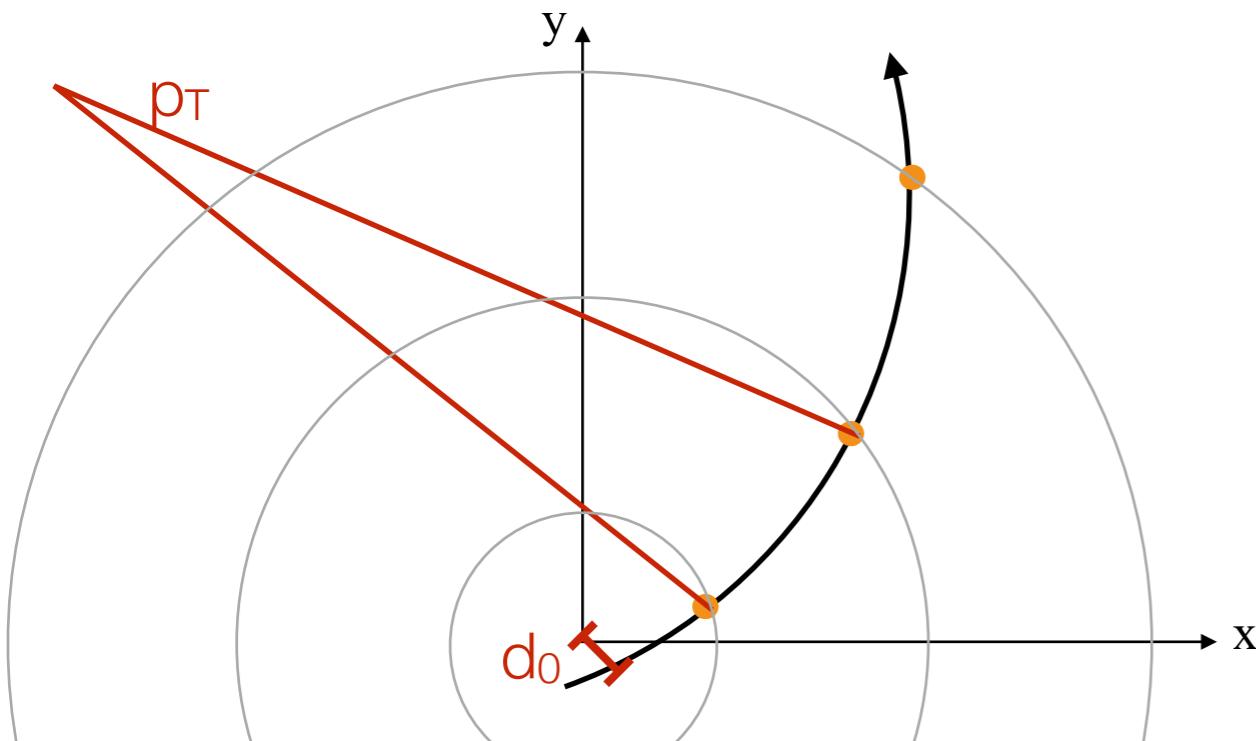
- Space point creation is a very simple activity



- Simple local to global transformations
 - in principle a $\text{vec}(3) \times \text{mat}(3,3) + \text{vec}(3)$ operation, can be done as $\text{vec}(4) \times \text{mat}(4,4)$
 - not much relative CPU time to gain, however a perfect testbed for e.g. aggressive compiler settings & Eigen
 - number of cells is huge (e.t. 10k space points) and memory requirement is minimal

Concurrency usage - seed creation (1)

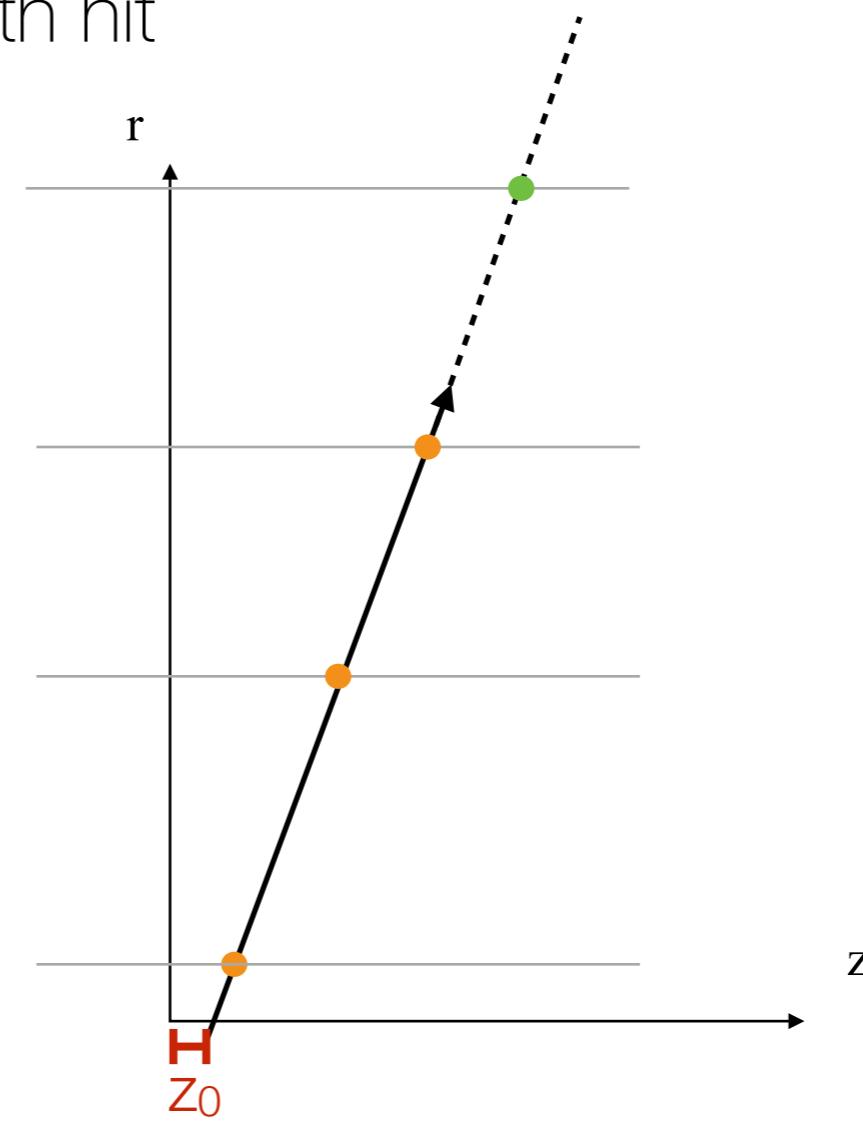
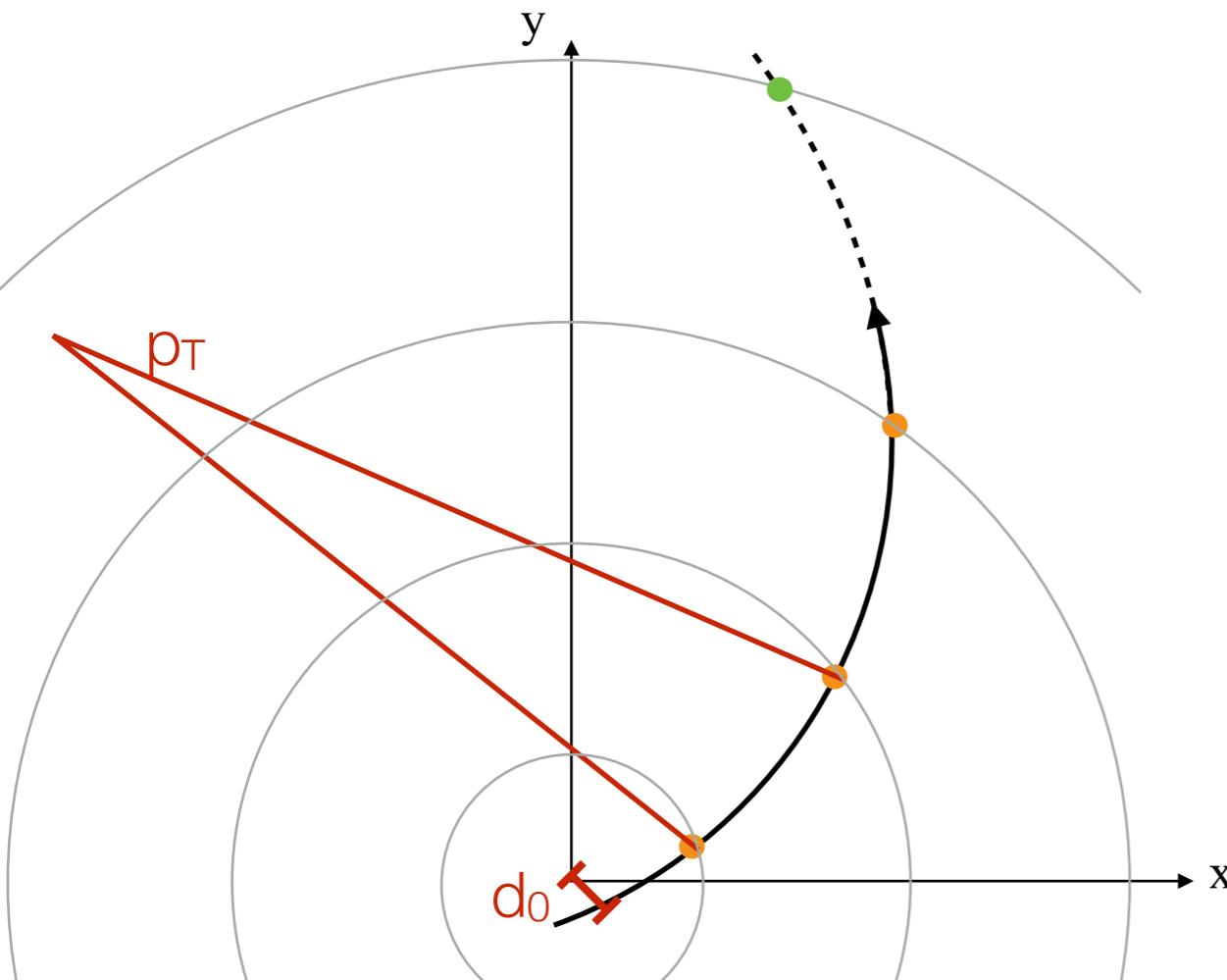
- ▶ Step 1: finding triplets of space points and evaluate their compatibility



- initial cuts on d_0 , z_0 , p_T applied
- looking for seeds in “connected” regions, not just random space point combinations
raises the question of resolving the overlap regions
- currently some book-keeping is done whether space points are already used
makes **concurrency** a bit tricky
having space points multiply used is not a problem per se (-> can be resolved later)

Concurrency usage - seed creation (2)

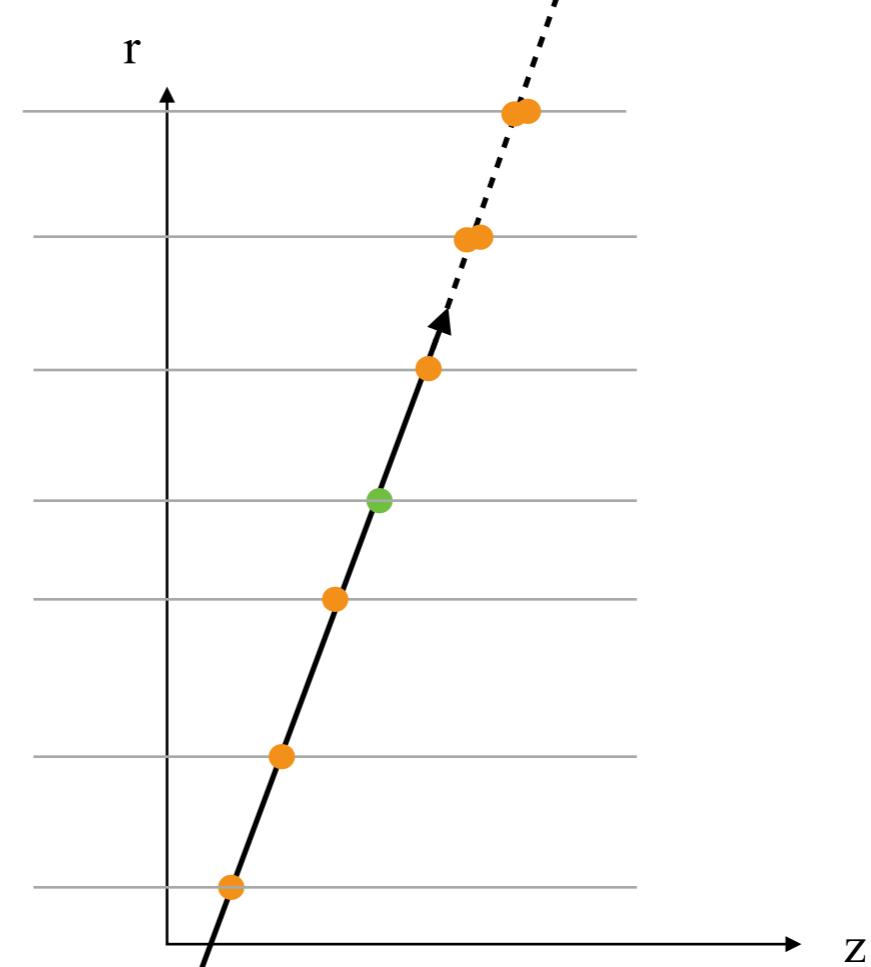
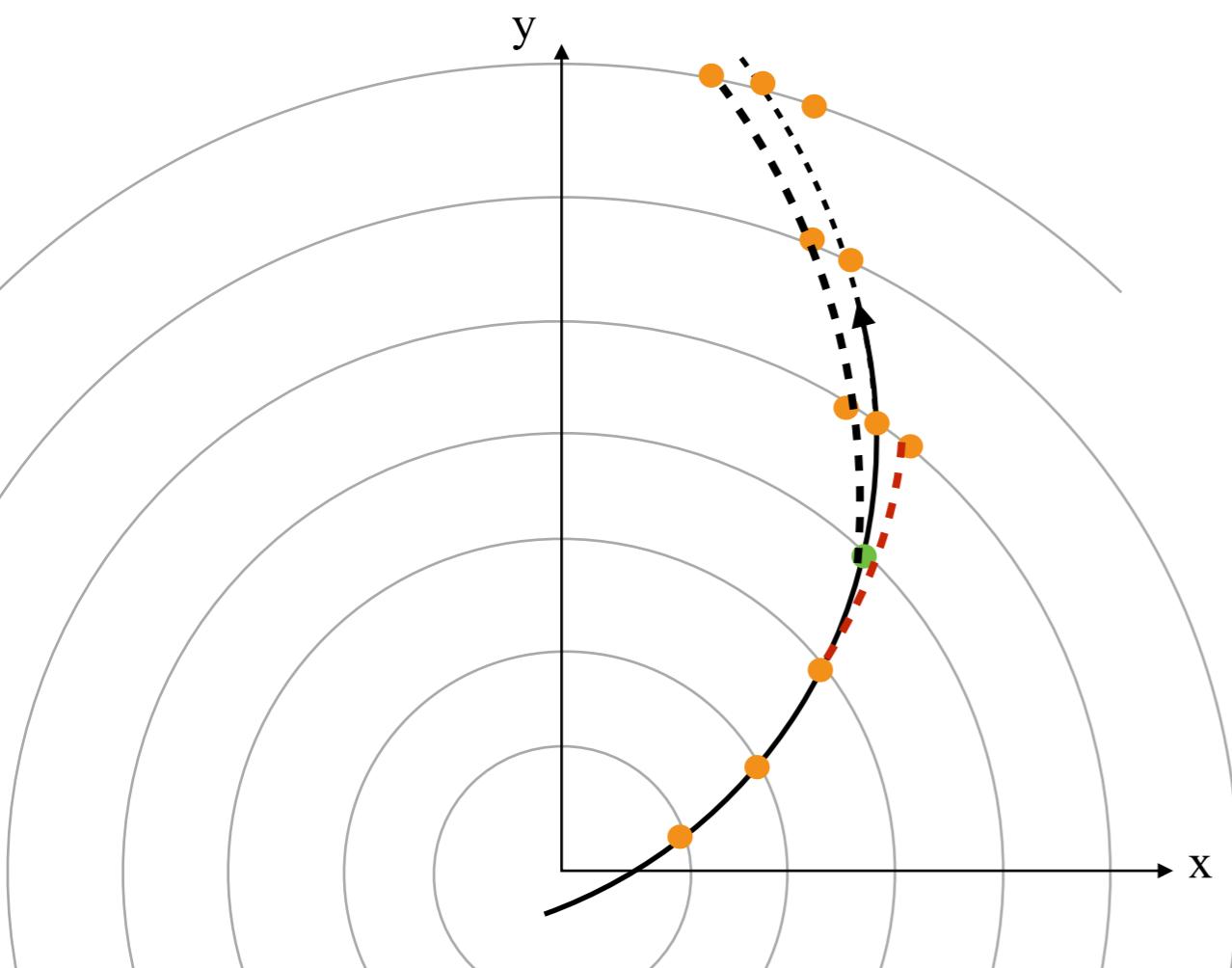
- ▶ Step 2: confirming a seed by a fourth hit



- ▶ Fast confirmation with a fourth space point before entering combinatorial Kalman filter (i.e. track finding)
 - each seed can be treated independently (and thus in parallel)

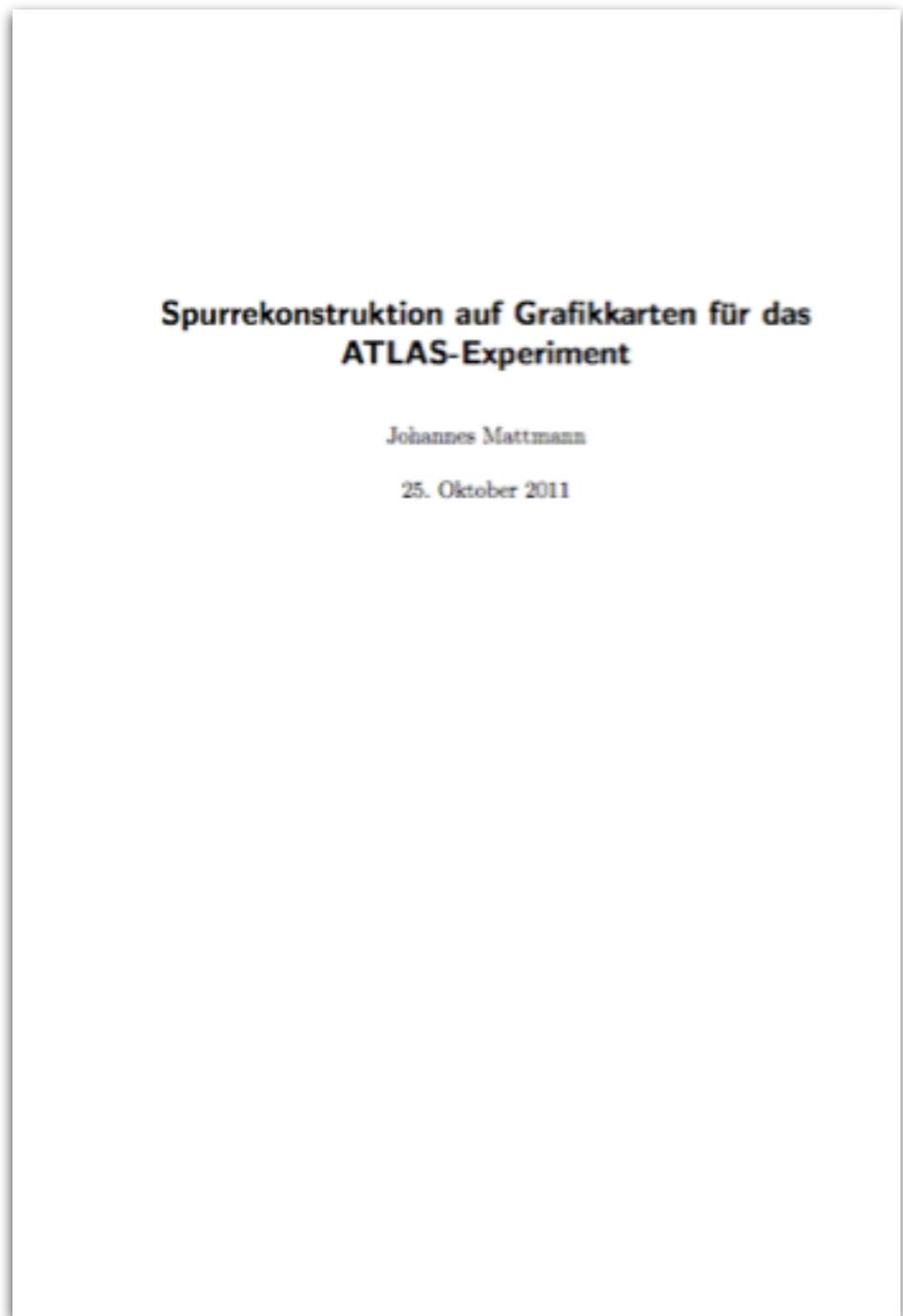
Concurrency usage - track finding (1)

- ▶ Step 3: seed defines a road, track finding within road.

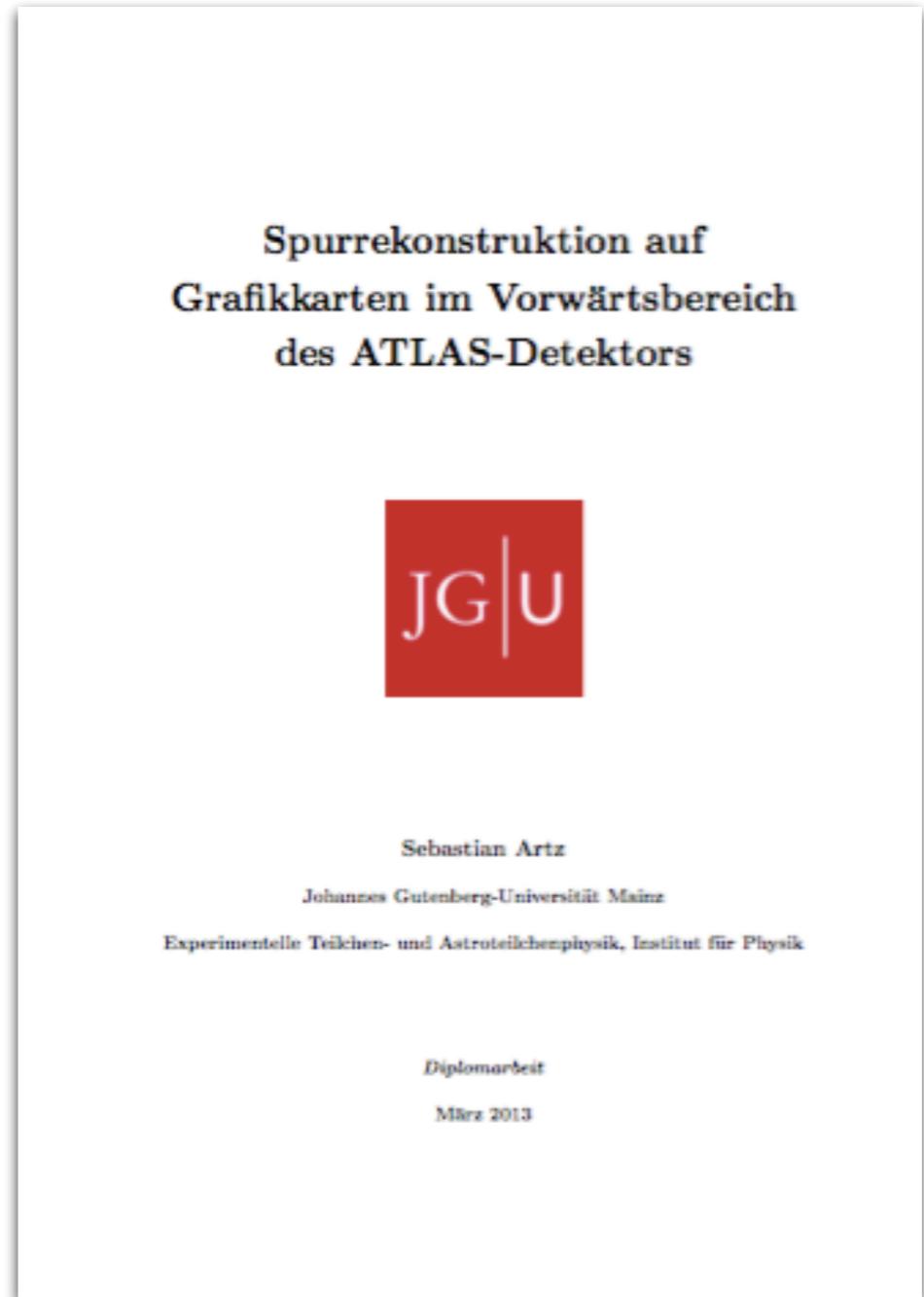


- ▶ Seeds can lead to several track candidates
 - parallel following of track candidates possible
 - open and close threads

GPU based track finding tests (offline)



**Johannes Mattmann, diploma thesis, 2011,
University of Mainz**



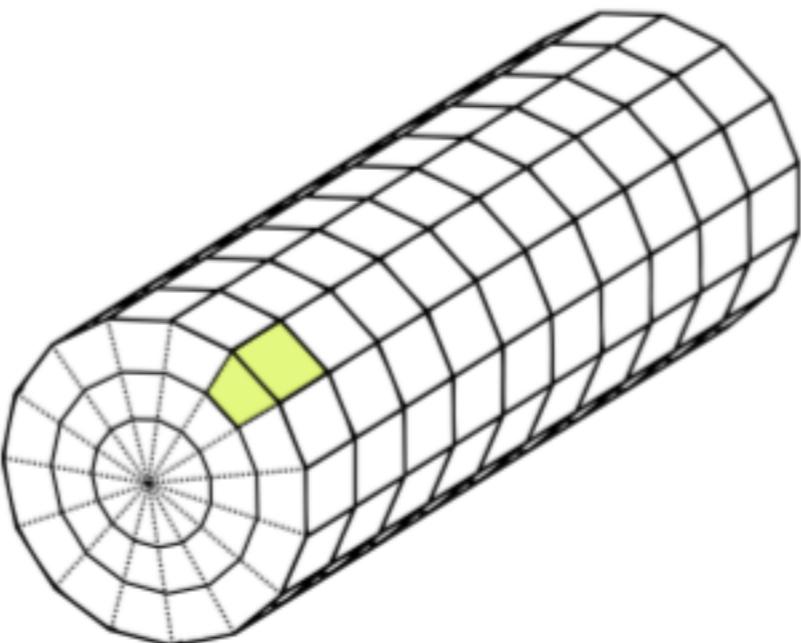
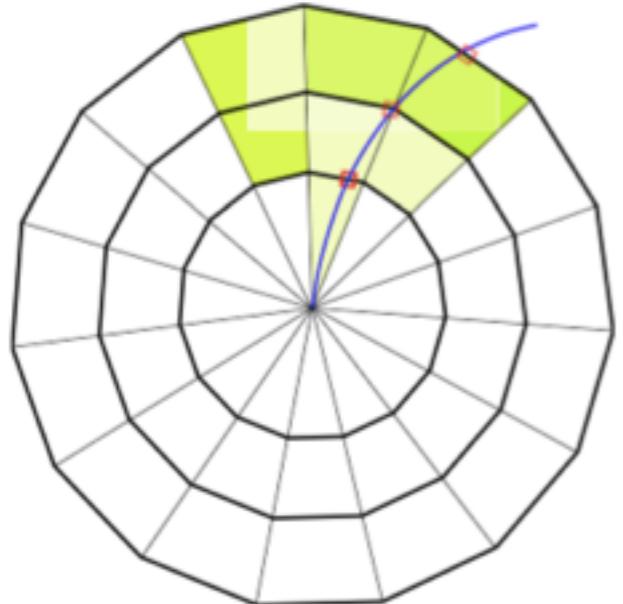
**Sebastian Artz, diploma thesis, 2011,
University of Mainz**

GPU based track finding - JM thesis (1)

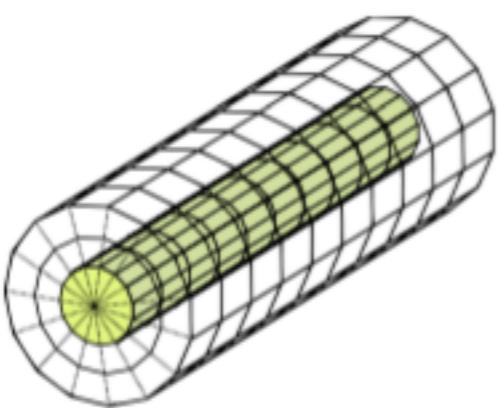
- ▶ diploma thesis of Johannes Mattmann (2011)
- ▶ prototype exploits parallel processing by using detector regions
 - prototype restricted to barrel only (see next slides)
 - seeding restricted to pixels
- ▶ reference CPU implementation & GPU implementation (CUDA)
 - seed finding (fixed to pixel detector)
 - propagation (helical) for road creation
 - track finding within the road
 - no full track fit

GPU based track finding - JM thesis (2)

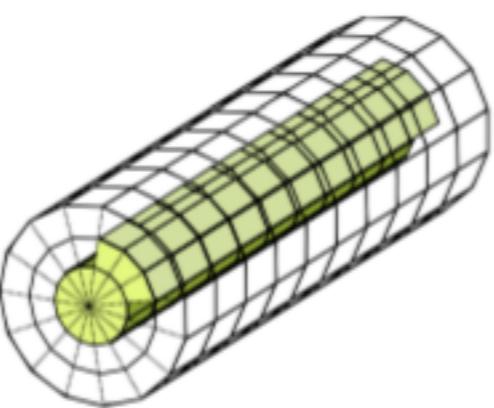
- ▶ Segmentation of the ID into regions



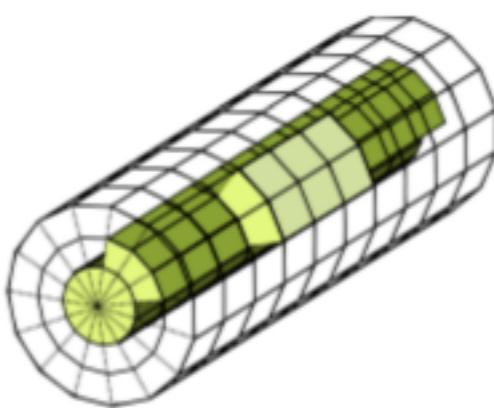
- ▶ Seed finding is done in these regions, moving in layers outwards
 - control the search regions with particle bending



layer0



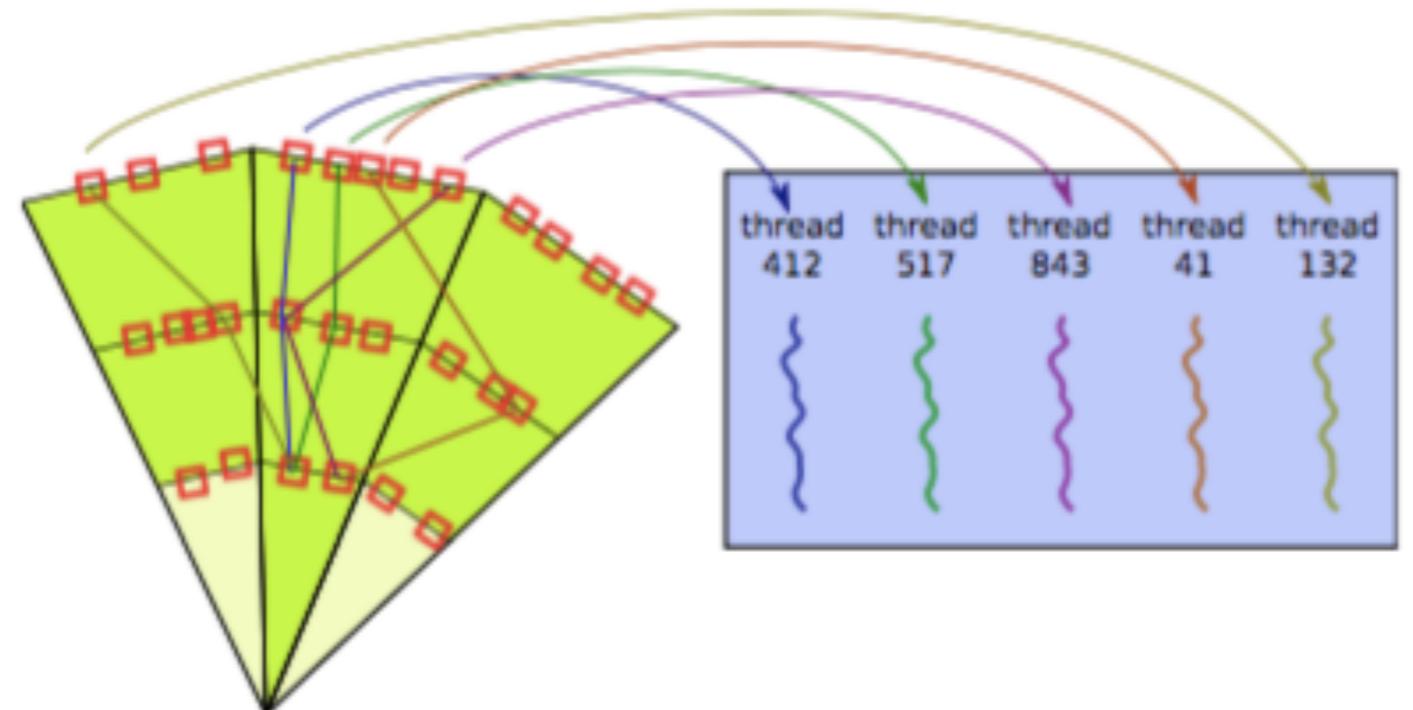
layer1



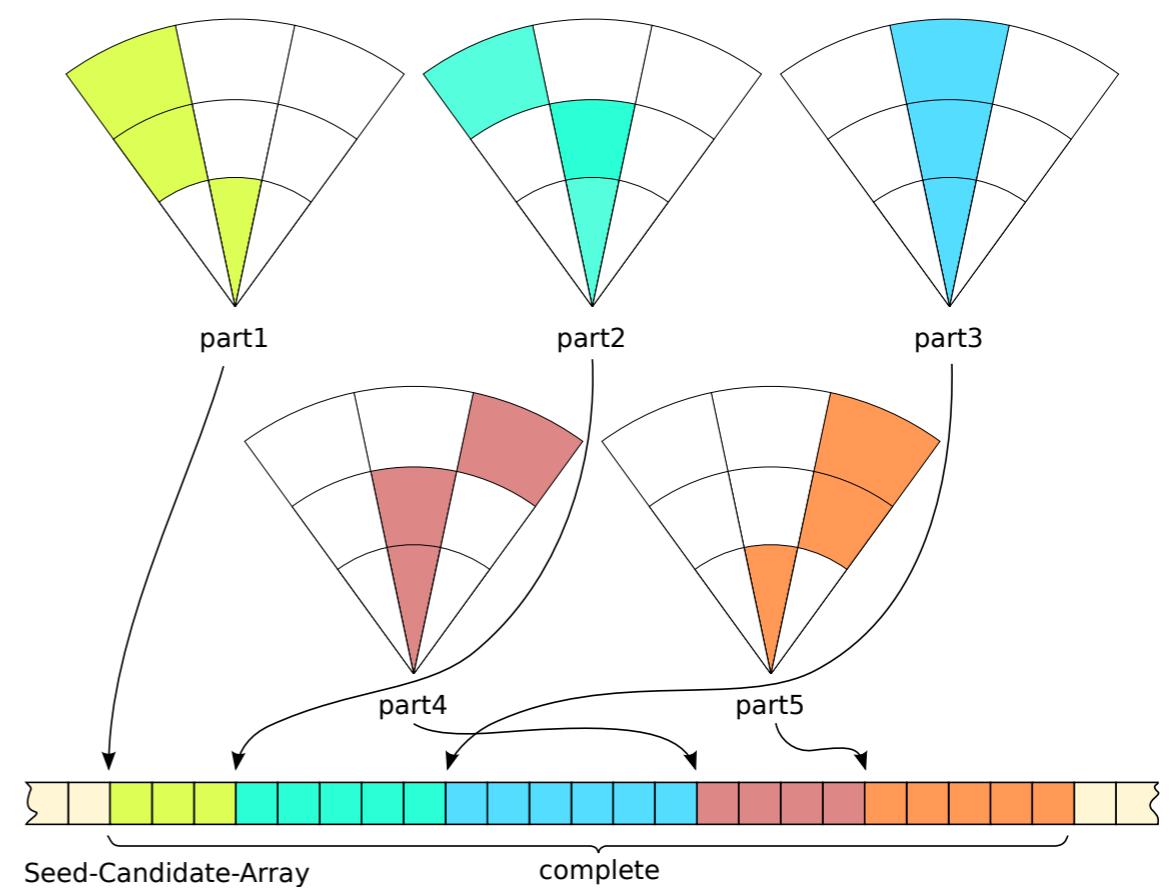
layer2

GPU based track finding - JM thesis (3)

- ▶ Seed finding starts with central cell
 - expands to neighbouring cell respecting the bending
 - cell size is related to minimal momentum cut



- ▶ Fork seeds into different threads
- ▶ Fixed pre-ordering of segment combination in memory

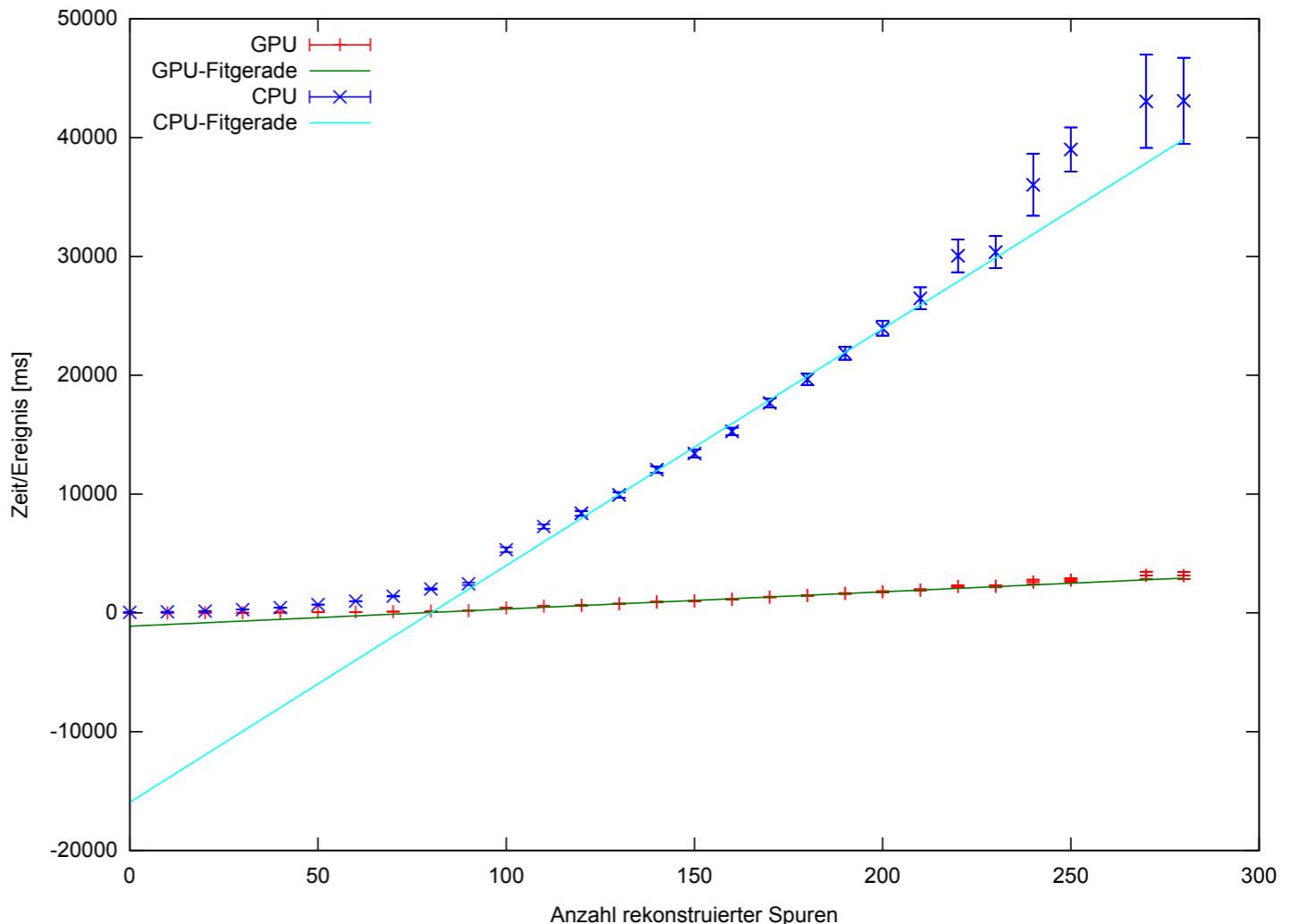


GPU based track finding - JM thesis (4)

- ▶ Some results

- GPU version indeed offers a great scaling behaviour with increasing number of particles
- some initial overhead due to data preparing for GPU architecture

- ▶ Biggest problem is resolving the overlaps between the regions

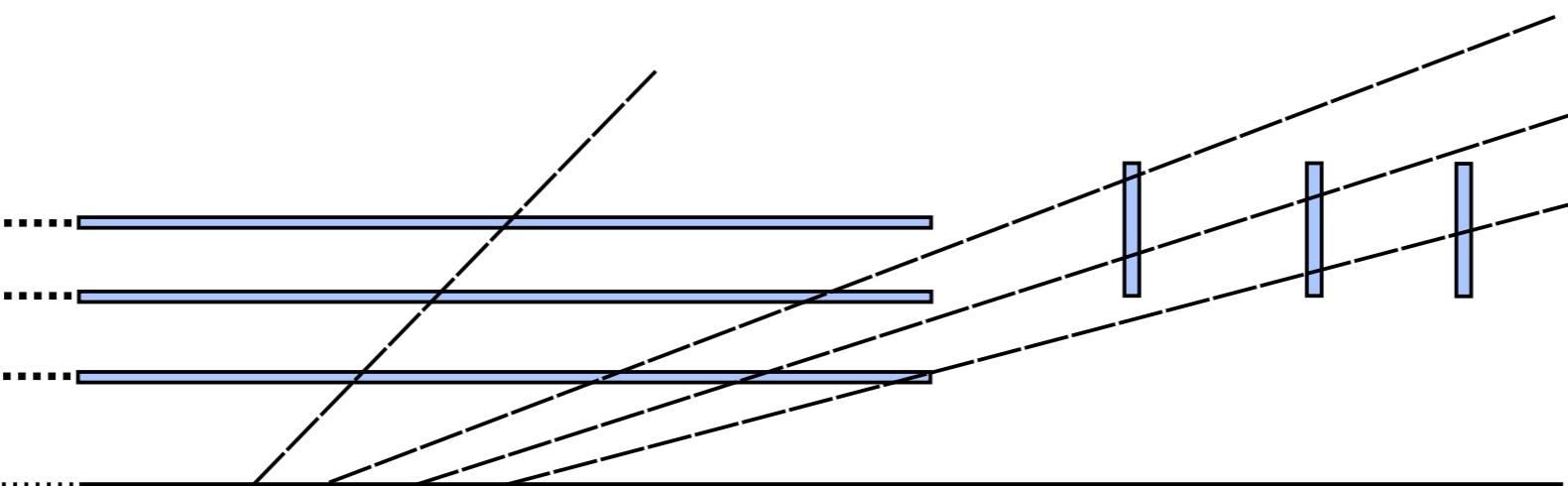


- ▶ Physics performance of sequential approach could not fully be met
 - difficulties to control duplicates/overlaps

GPU based track finding - SA thesis (1)

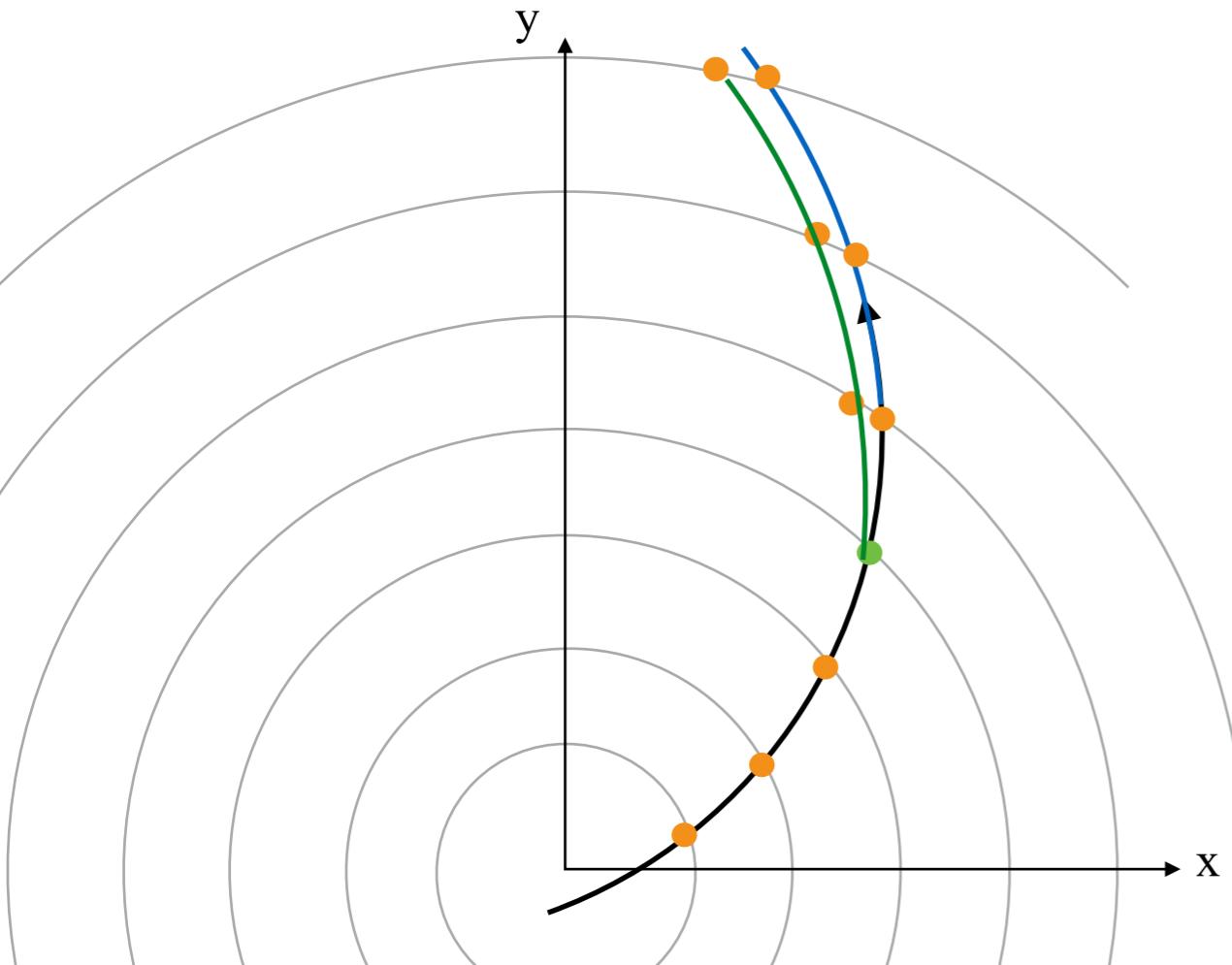
- ▶ Diploma thesis of Sebastian Artz (2013)
- ▶ Extended JM to include endcaps
 - seeding still restricted to Pixels

	Hit 1	Hit 2	Hit 3
Kombination 1	Barrel Layer 0	Barrel Layer 1	Barrel Layer 2
Kombination 2	Barrel Layer 0	Barrel Layer 1	Endkappen Layer 0
Kombination 3	Barrel Layer 0	Endkappen Layer 0	Endkappen Layer 1
Kombination 4	Barrel Layer 0	Endkappen Layer 1	Endkappen Layer 2



Concurrency usage - ambiguity solving (1)

- ▶ Ambiguity solving decides on the best hit association



- ▶ What is the current setup ?
 - is this <**black+green, blue**>
 - or <**black, blue, green**>
 - or <**green, black+blue**>
 - or <**black+ blue**>
 - or <>
- ▶ hypotheses tested & scored
 - they are interconnected via shared hits

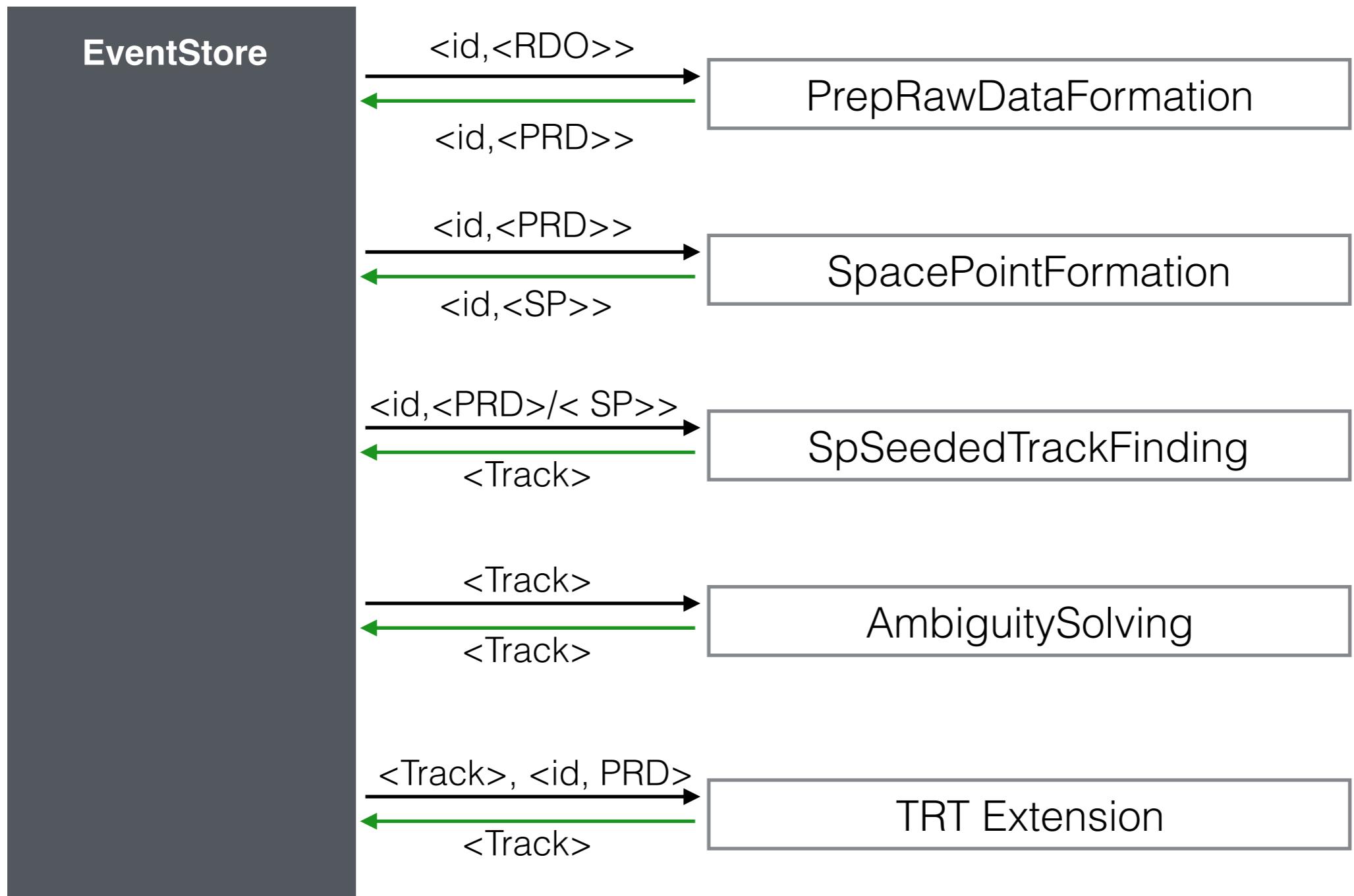
- ▶ Parallelism **could** be around connected sets of tracks
 - final merging and chasing of tracks above certainn score cute

Concurrency Table - Summary

Algorithm	Concurrent Cell	Number of cells	Tests	Rel. CPU of ID	Comments
Cluster creation	per module	$O(1000)$	no tests exist	$O(5\%)$	output merging into identifiable containers
SpacePoint creation	per space point	$O(10000)$	no tests exist	$O(<1\%)$	identifiable container, SIMD ?
SpacePoint seeded track finding	detector region	$O(100)$	GPU based test version from 2011	$O(50\%)$	overlaps are dangerous
Ambiguity Solving	fitting: per track ambiguity: per tracks through shared	$O(1000)$	no tests exist	$O(20\%)$	book keeping of hits is shared
TRT extension	per track	$O(100)$	no tests exist	$O(10\%)$	

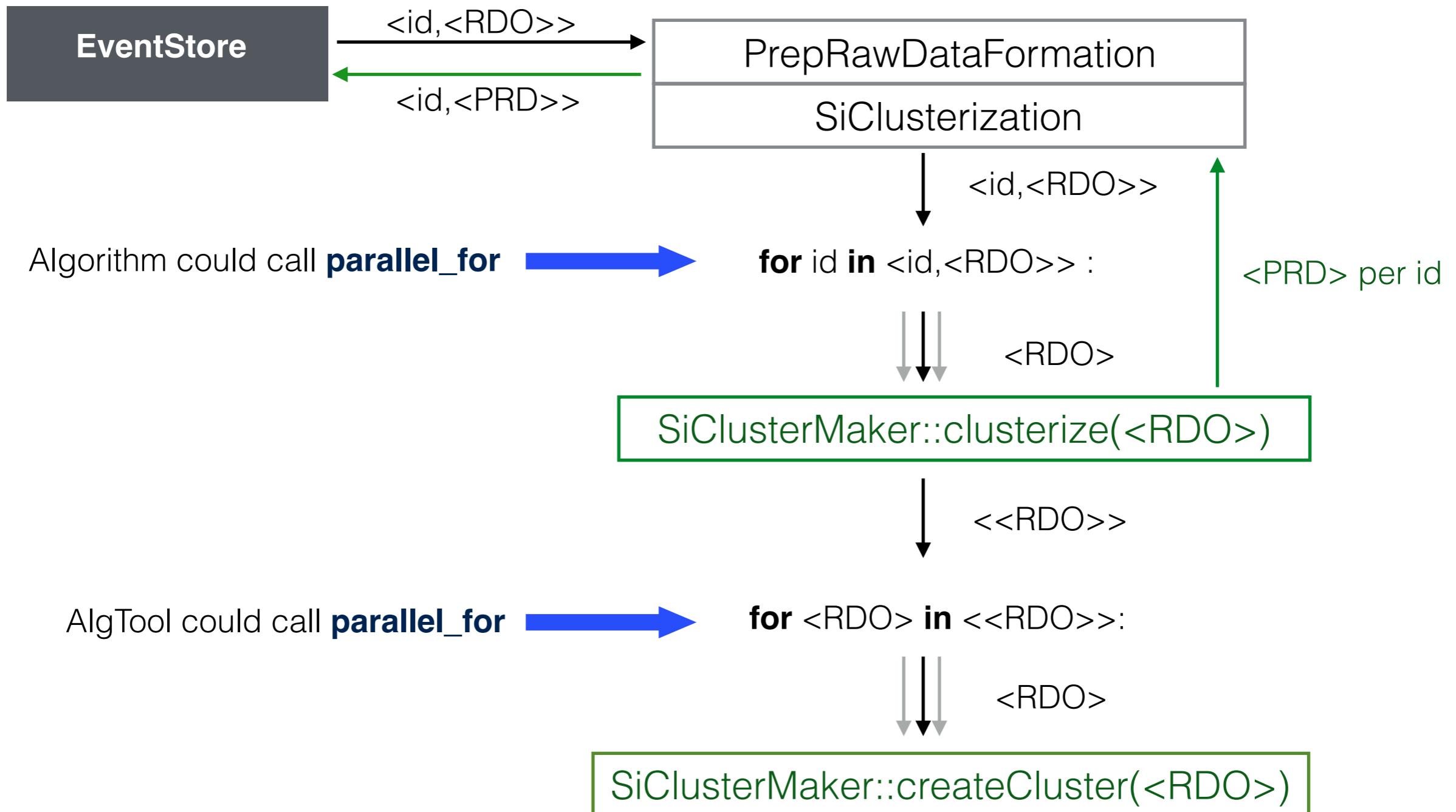
Let's talk about EDM, Algorithms & Tools

- ▶ Current ID track reconstruction is built of a sequence of algorithms
 - following Gaudi-Athena design, they communicate via the blackboard (EventStore)



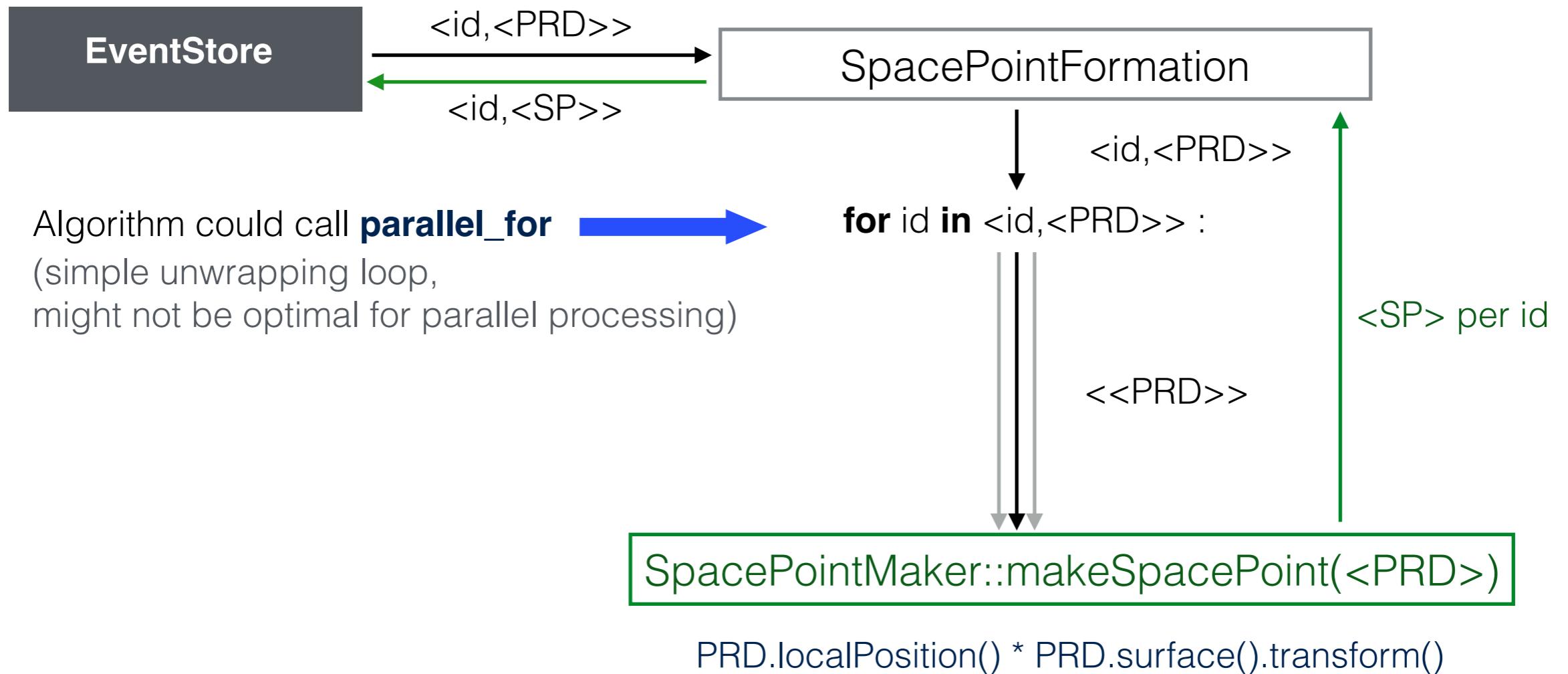
EDM, Algorithms & Tools (1)

- PrepRawData Formation tools/concurrency requirements



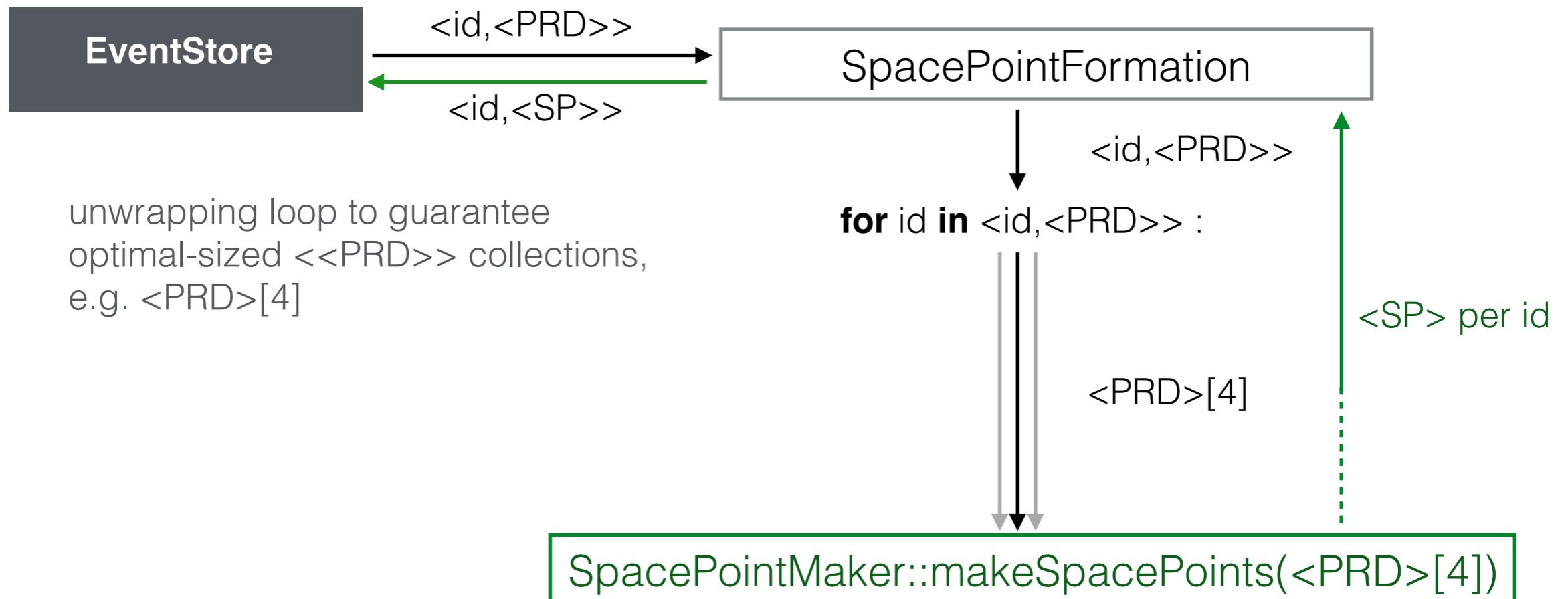
EDM, Algorithms & Tools (2)

- SpacePointFormation concurrency requirement - parallel_for option



EDM, Algorithms & Tools (3)

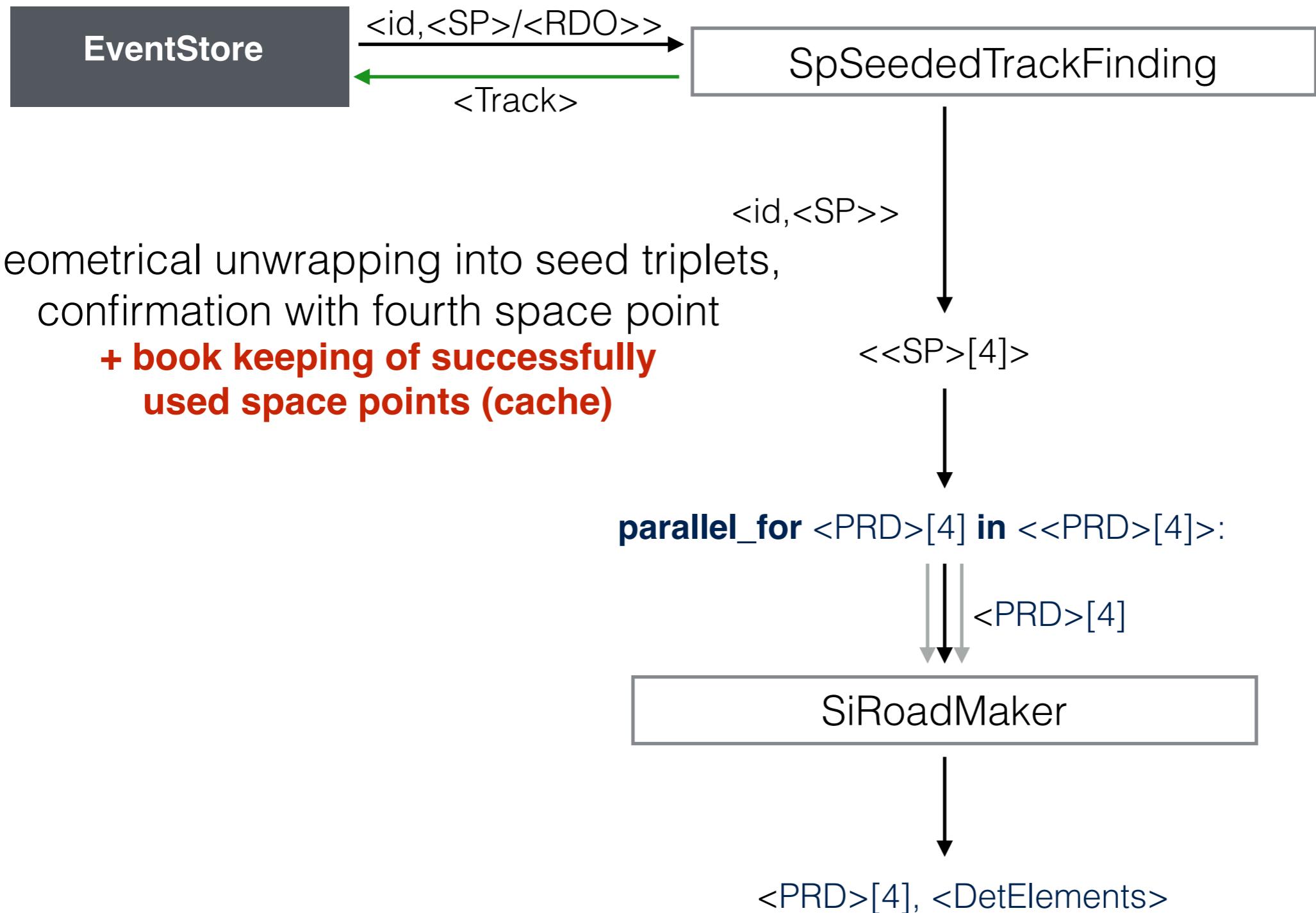
- SpacePointFormation concurrency requirement - SIMD option



output collection has to be per identifier (id), re-ordering/bookkeeping necessary

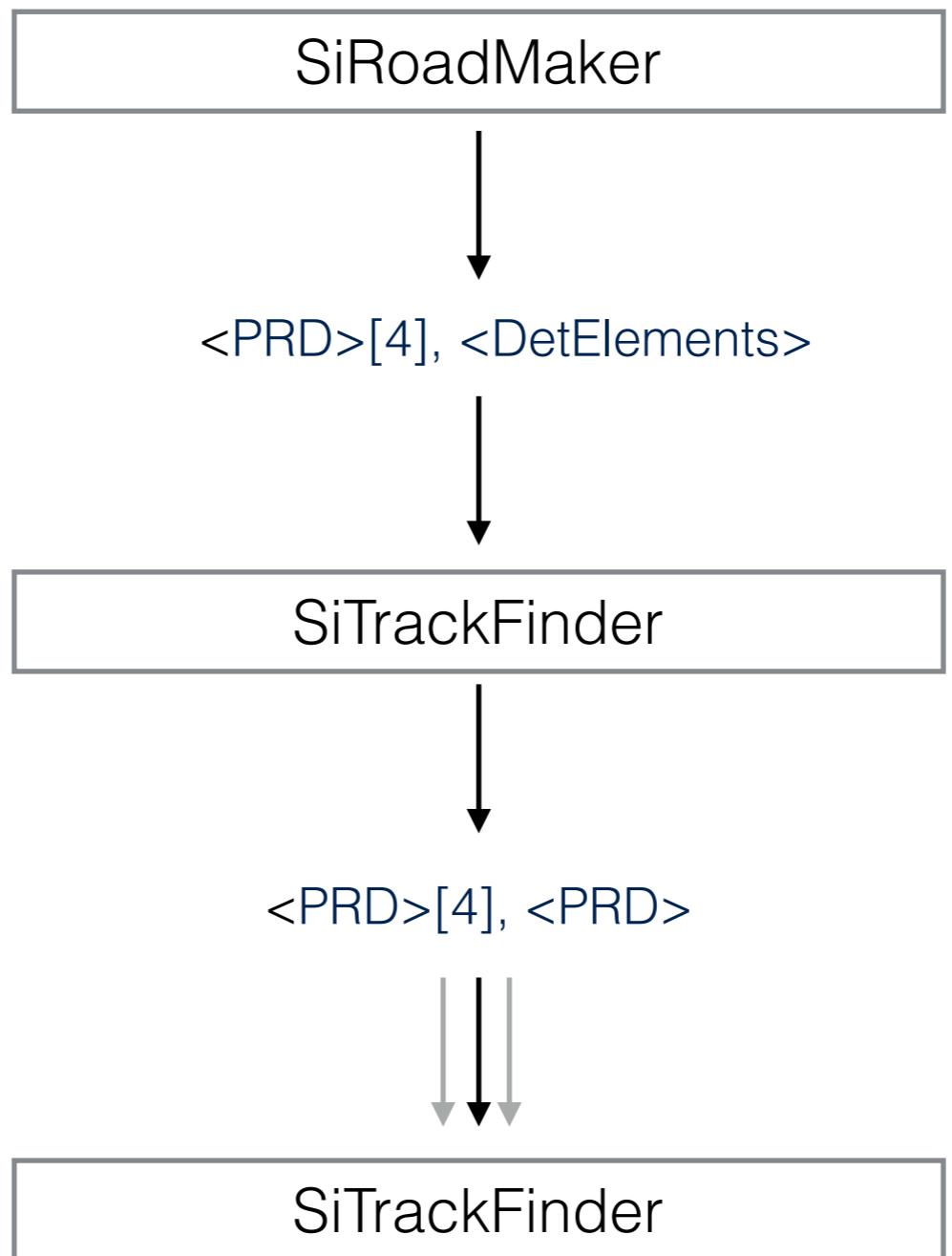
EDM, Algorithms & Tools (4)

- ▶ SiSpSeededTrackFinding tools/concurrency requirements



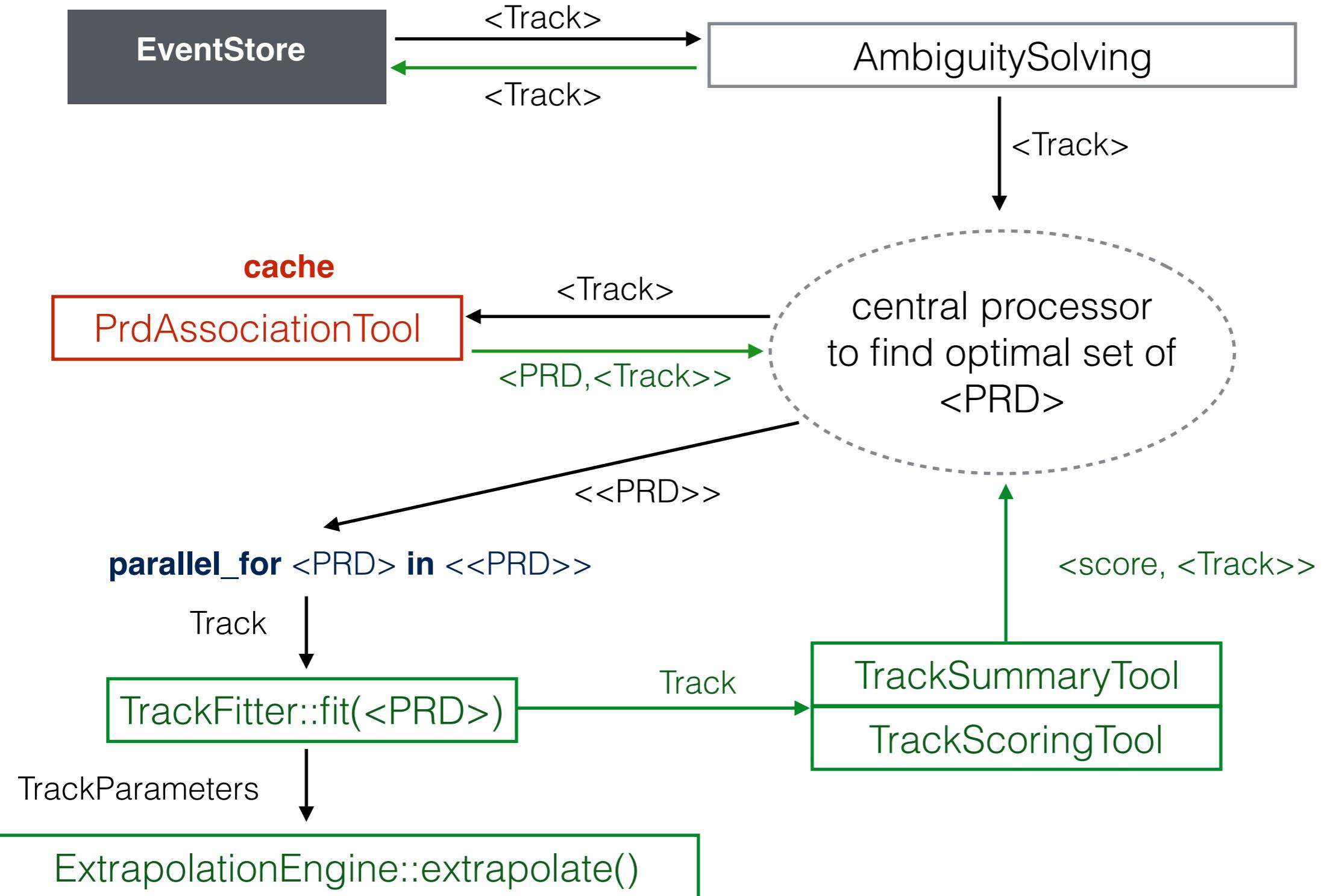
EDM, Algorithms & Tools (5)

combinatorial Kalman filter
can fork to test several
track candidates
new threads could be
opened & closed
depending if the track
candidate survives



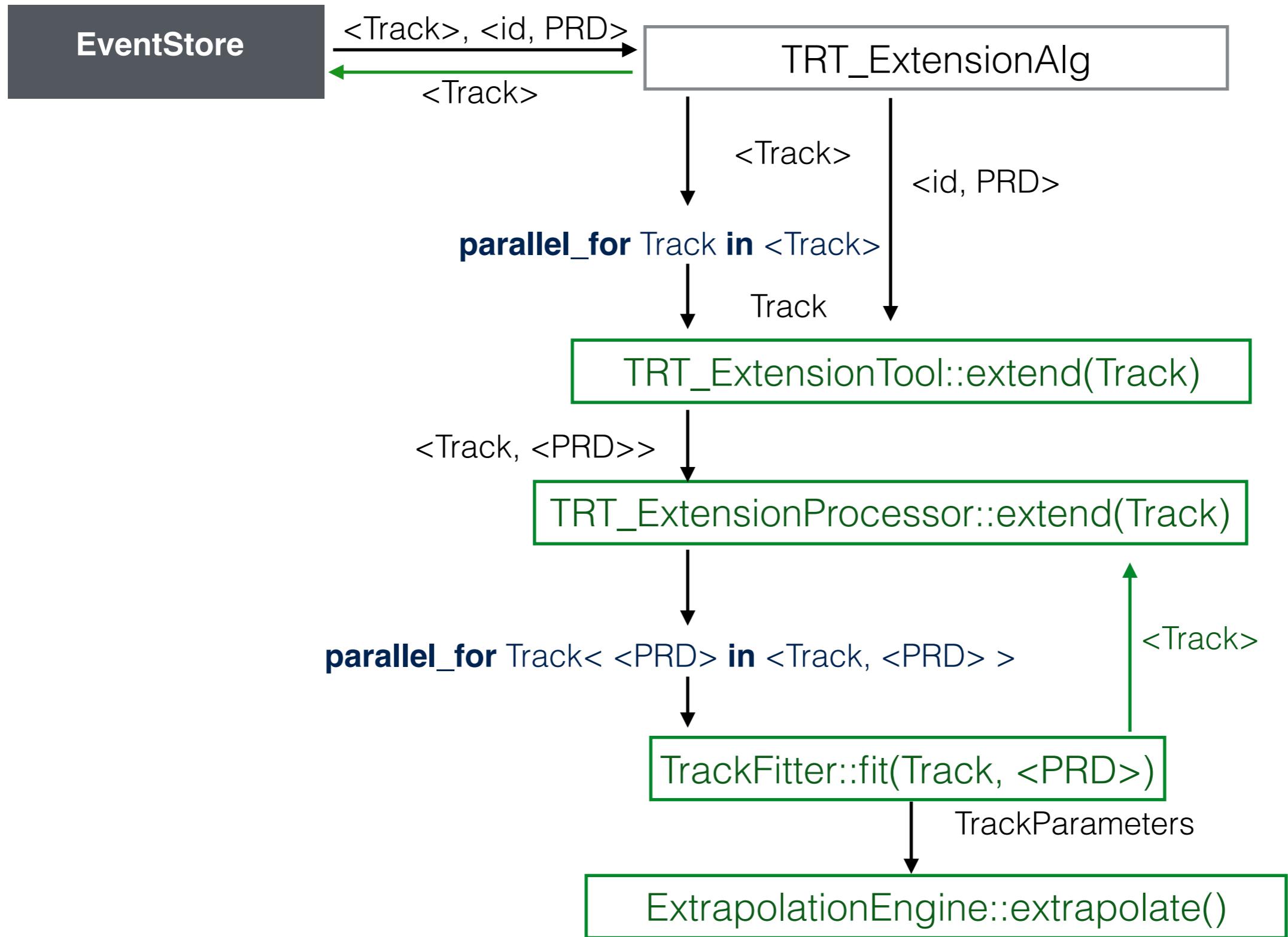
EDM, Algorithms & Tools (6)

- ▶ Ambiguity solving concurrency/tool requirements



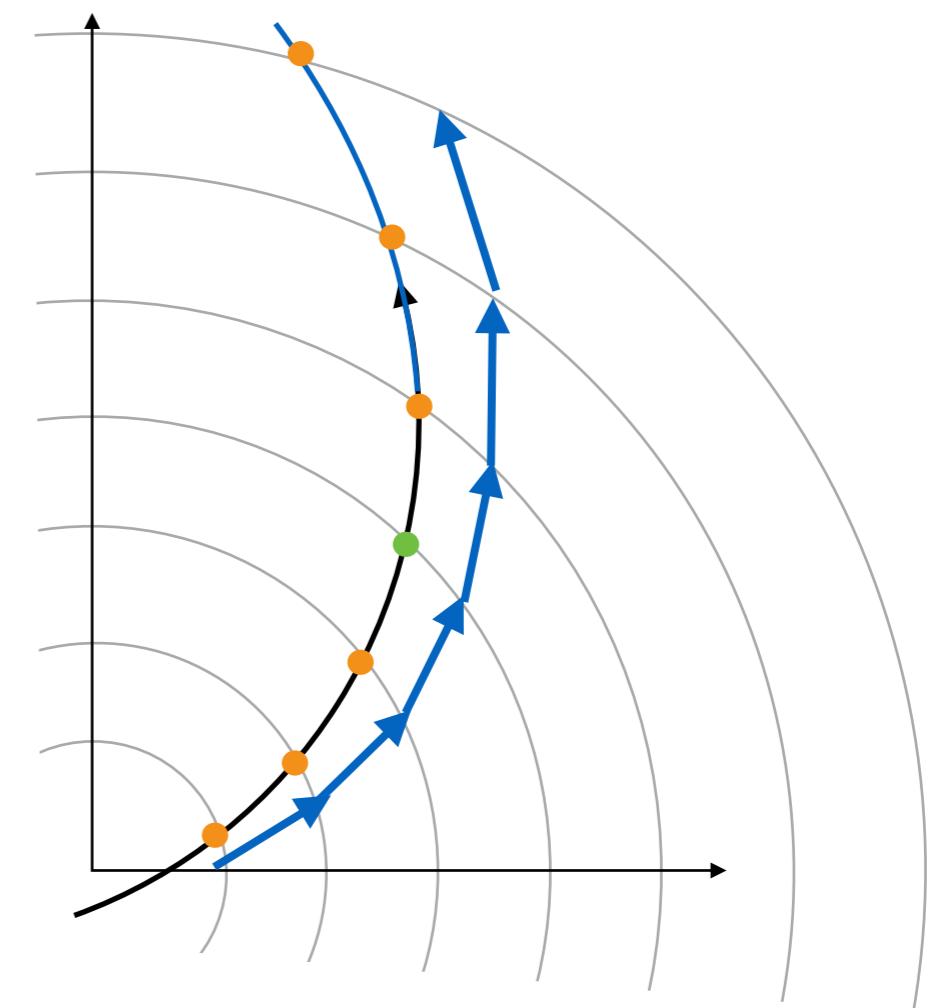
EDM, Algorithms & Tools (7)

- ▶ TRT extension tools/concurrency requirements



Tools example - Extrapolation engine

- ▶ Tools involved in Track reconstruction are pretty evolved
 - one of the most complex tools is the extrapolator used in track finidg*, track fitter, hole search, etc.
 - Run-1 extrapolation tool was about 5k lines not thread safe, as it had an internal cache
 - Re-write as a new `ExtrapolationEngine` prototype now working in current release should/will be changed into a `AthService` (is public tool now)
still **keeps the nice caching functionality**
designed to be thread-safe by a cache visitor (called `ExtrapolationCell`)



e.g. KalmanFilter usually starts at the last destination

```
ExtrapolationCell::restartAtDestination()
```

ExtrapolationEngine (1)

- ▶ new interface with a visitor pattern

```
namespace Trk {

    static const InterfaceID IID_IExtrapolationEngine("IExtrapolationEngine", 1, 0);

    typedef ExtrapolationCell<TrackParameters> ExCellCharged;
    typedef ExtrapolationCell<NeutralParameters> ExCellNeutral;

    class IExtrapolationEngine : virtual public IAlgTool {
        public:

            /** Virtual destructor */
            virtual ~IExtrapolationEngine() {}

            /** AlgTool interface methods */
            static const InterfaceID& interfaceID() { return IID_IExtrapolationEngine; }

            /** charged extrapolation */
            virtual ExtrapolationCode extrapolate(ExCellCharged& ecCharged,
                                                   const Surface* sf = 0,
                                                   PropDirection dir=alongMomentum,
                                                   BoundaryCheck bcheck = true) const = 0;

            /** neutral extrapolation */
            virtual ExtrapolationCode extrapolate(ExCellNeutral& ecNeutral,
                                                   const Surface* sf = 0,
                                                   PropDirection dir=alongMomentum,
                                                   BoundaryCheck bcheck = true) const = 0;

            /** define for which GeometrySignature this extrapolator is valid */
            virtual GeometryType geometryType() const = 0;

        protected:
            //!< SCREEN output formatting (SOP) - unify amongst extrapolation engines
            std::string m_sopPrefix;           //!< prefix for screen output
            std::string m_sopPostfix;          //!< prefix for screen output
    };

} // end of namespace
```

ExtrapolationEngine (2)

- ▶ ExtrapolationCell acts as the cache & steering object

```
/** @class ExtrapolationCell

templated class as an input-output object of the extrapolation,
only public members, since it is a container class

@author Andreas.Salzburger -at- cern.ch
*/

template <class T> class ExtrapolationCell {
public :
    const T&                      startParameters;      //!<< by reference - need to be defined
    const TrackingVolume*           startVolume;          //!<< the start volume - needed for the volumeToVolume loop
    const Layer*                   startLayer;           //!<< the start layer - needed for layerToLayer loop

    const T*                       endParameters;        //!<< by pointer - are newly created and can be optionally 0
    const TrackingVolume*          endVolume;            //!<< the end Volume - can be optionally 0 (needs other trigger to stop)
    const Layer*                  endLayer;             //!<< the end Layer - can be optionally 0 (needs other trigger to stop)

    const T*                       leadParameters;       //!<< the one last truely valid parameter in the stream
    const TrackingVolume*          leadVolume;           //!<< the lead Volume - carrying the navigation stream
    const Layer*                  leadLayer;            //!<< the lead Layer - carrying the navigation stream

    int                           navigationStep;        //!<< a counter of the navigation Step
    double                        pathLength;           //!<< the path length accumulated
    double                        pathLimit;            //!<< the maximal limit of the extrapolation

    ParticleHypothesis           pHypothesis;          //!<< what particle hypothesis to be used, default : pion
    MagneticFieldProperties       mFieldMode;           //!<< what magnetic field mode to be used, default : fullField
    bool                          navigationCurvilinear; //!<< stay in curvilinear parameters where possible, default : true
    bool                          destinationCurvilinear; //!<< return curvilinear parameters even on destination

    std::vector<const T*>        sensitiveParameters;  //!<< parameters on sensitive detector elements
    std::vector<const T*>        passiveParameters;   //!<< parameters on the passive layers
    std::vector<const T*>        boundaryParameters; //!<< parameters on boundary surfaces

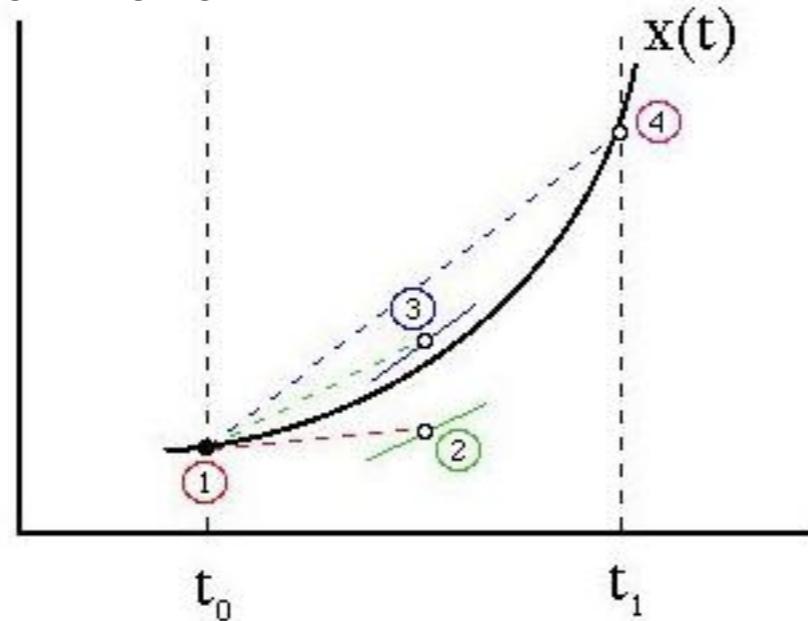
    unsigned int                  extrapolationConfiguration; //!<< integer carrying (by setting bits) the information how to extrapolate -
}
```

serves as Input/Output and Cache of this extrapolation job
- automatically thread safe !

ExtrapolationEngine (3)

- Major part of time in extrapolation is spent on Runge-Kutta integration

- numerical integration to solve equation of motion



$$\begin{aligned} dx_1 &= \Delta t v_{x,n} \\ dv_x 1 &= \Delta t a_x(x_n, y_n, t) \\ dx_2 &= \Delta t (v_{x,n} + \frac{dv_x 1}{2}) \\ dv_x 2 &= \Delta t a_x(x_n + \frac{dx_1}{2}, y_n + \frac{dy_1}{2}, t + \frac{\Delta t}{2}) \\ dx_3 &= \Delta t (v_{x,n} + \frac{dv_x 2}{2}) \\ dv_x 3 &= \Delta t a_x(x_n + \frac{dx_2}{2}, y_n + \frac{dy_2}{2}, t + \frac{\Delta t}{2}) \\ dx_4 &= \Delta t (v_{x,n} + dv_x 3) \\ dv_x 4 &= \Delta t a_x(x_n + dx_3, y_n + dy_3, t + \Delta t) \\ x_{n+1} &= y_n + \frac{dx_1}{6} + \frac{dx_2}{3} + \frac{dx_3}{3} + \frac{dx_4}{6} \\ v_{x,n+1} &= v_{x,n} + \frac{dv_x 1}{6} + \frac{dv_x 2}{3} + \frac{dv_x 3}{3} + \frac{dv_x 4}{4} \end{aligned}$$

- ATLAS has two test versions - none run in production
 - vectorised version (hand-written)
 - 4-dimensional Eigen based version (was about the same speed)

ExtrapolationEngine (4)

- ▶ vectorised Runge-Kutta kernel

```

for(int i=0; i<4L; i++) {
    double* dR     = &P[i];
    double* dA     = &P[i+3];

    double dA0     = H0[ 2]*dA[1]-H0[ 1]*dA[2];
    double dB0     = H0[ 0]*dA[2]-H0[ 2]*dA[0];
    double dC0     = H0[ 1]*dA[0]-H0[ 0]*dA[1];

    if(i==35) {dA0+=A0; dB0+=B0; dC0+=C0;}

    double dA2     = dA0+dA[0];
    double dB2     = dB0+dA[1];
    double dC2     = dC0+dA[2];

    double dA3     = dA[0]+dB2*H1[2]-dC2*H1[1];
    double dB3     = dA[1]+dC2*H1[0]-dA2*H1[2];
    double dC3     = dA[2]+dA2*H1[1]-dB2*H1[0];

    if(i==35) {dA3+=A3-A00; dB3+=B3-A11; dC3+=C3-A22;}

    double dA4     = dA[0]+dB3*H1[2]-dC3*H1[1];
    double dB4     = dA[1]+dC3*H1[0]-dA3*H1[2];
    double dC4     = dA[2]+dA3*H1[1]-dB3*H1[0];

    if(i==35) {dA4+=A4-A00; dB4+=B4-A11; dC4+=C4-A22;}

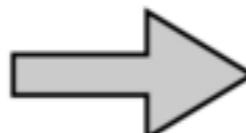
    double dA5     = dA4+dA4-dA[0];
    double dB5     = dB4+dB4-dA[1];
    double dC5     = dC4+dC4-dA[2];

    double dA6     = dB5*H2[2]-dC5*H2[1];
    double dB6     = dC5*H2[0]-dA5*H2[2];
    double dC6     = dA5*H2[1]-dB5*H2[0];

    if(i==35) {dA6+=A6; dB6+=B6; dC6+=C6;}
}

dR[0]+=(dA2+dA3+dA4)*S3; dA[0]=(dA0+dA3+dA3+dA5+dA6)*.333333333333;
dR[1]+=(dB2+dB3+dB4)*S3; dA[1]=(dB0+dB3+dB3+dB5+dB6)*.333333333333;
dR[2]+=(dC2+dC3+dC4)*S3; dA[2]=(dC0+dC3+dC3+dC5+dC6)*.333333333333;

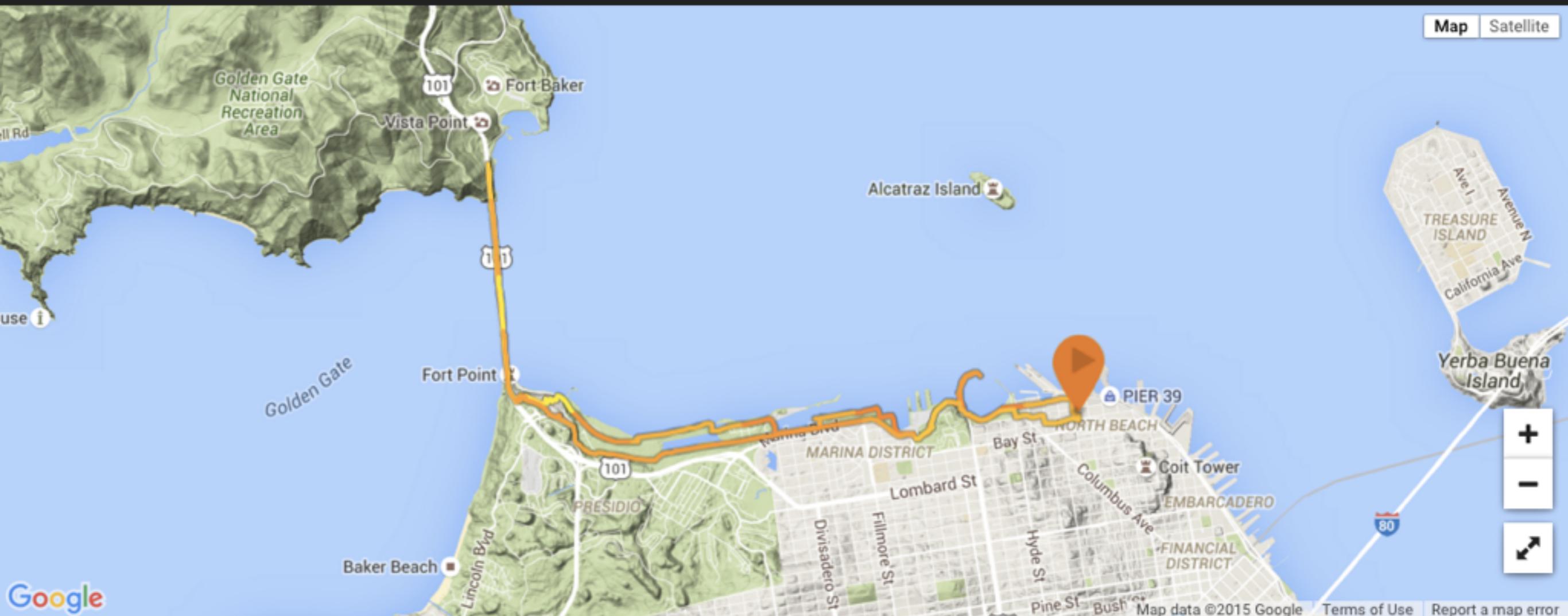
```



- Runge-Kutta integration but up to 2.4x faster using SIMD instructions
 - Future CPUs are expected to further extend SIMD capacities

New paths

- MAP



Remember this ?



the HiggsML challenge

May to September 2014

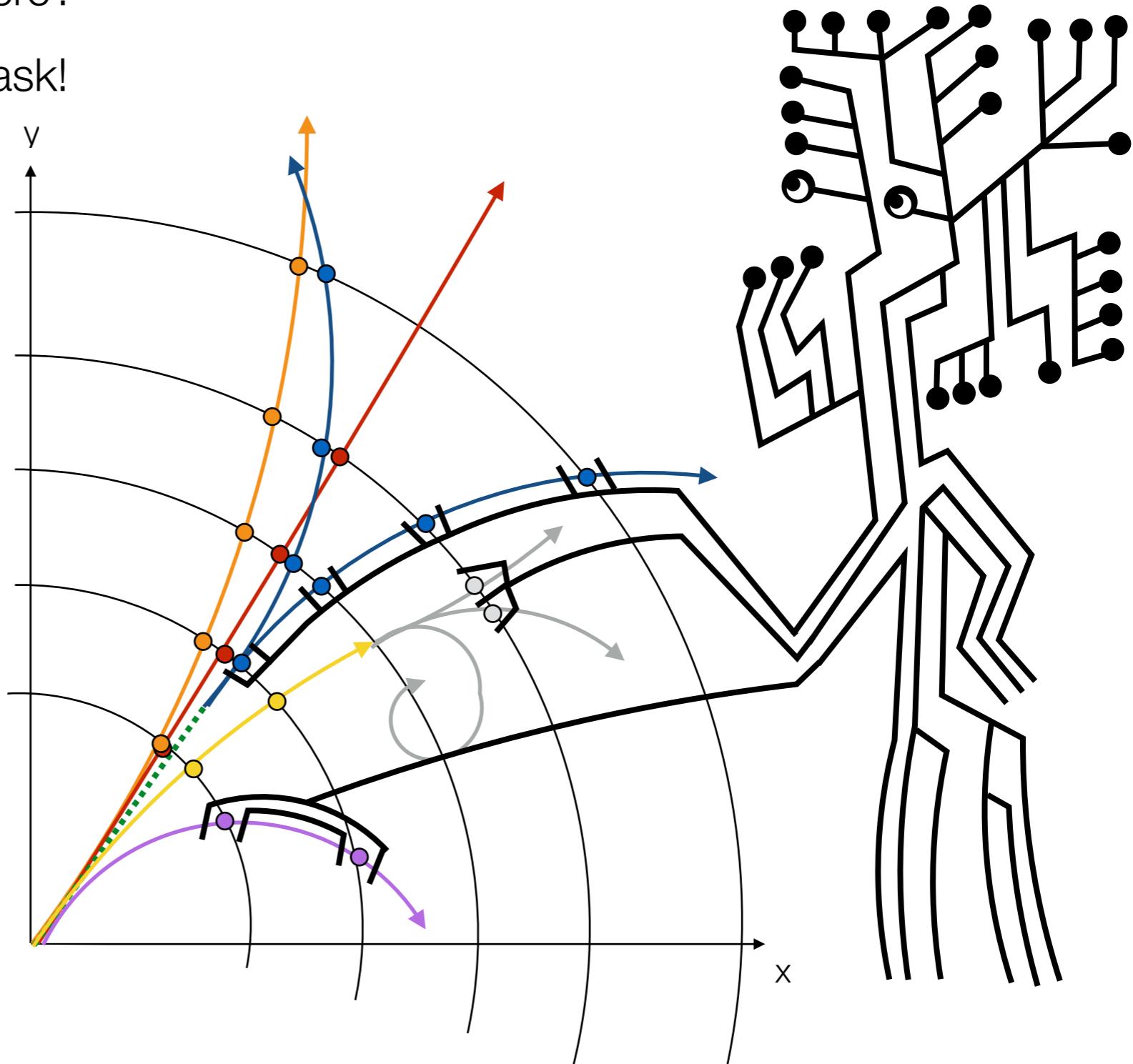
When **High Energy Physics** meets **Machine Learning**



info to participate and compete : <https://www.kaggle.com/c/higgs-boson>

A new challenge lies ahead

- ▶ Are we using the most up-to-date approaches ?
 - are there smarter kids out there?
 - we never know if we do not ask!
- ▶ HEP pattern recognition techniques are more than 25 years old
 - machine learning techniques are flooding the market
 - are they suitable for our complex problems ?



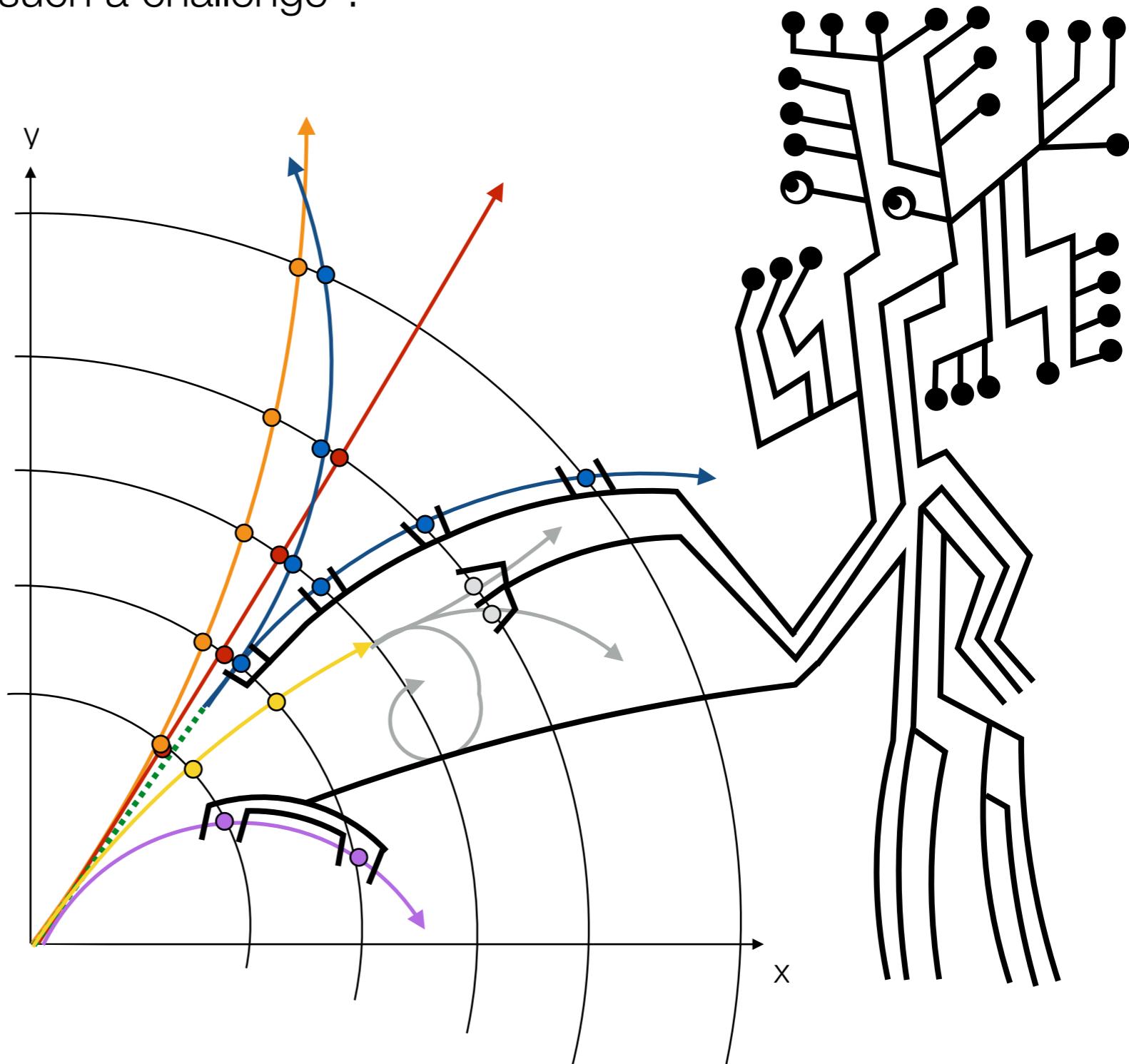
A HEP Tracking Machine Learning Challenge (1)

- ▶ Can we teach the computer to find tracks for us ?

- what is the minimal input for such a challenge ?

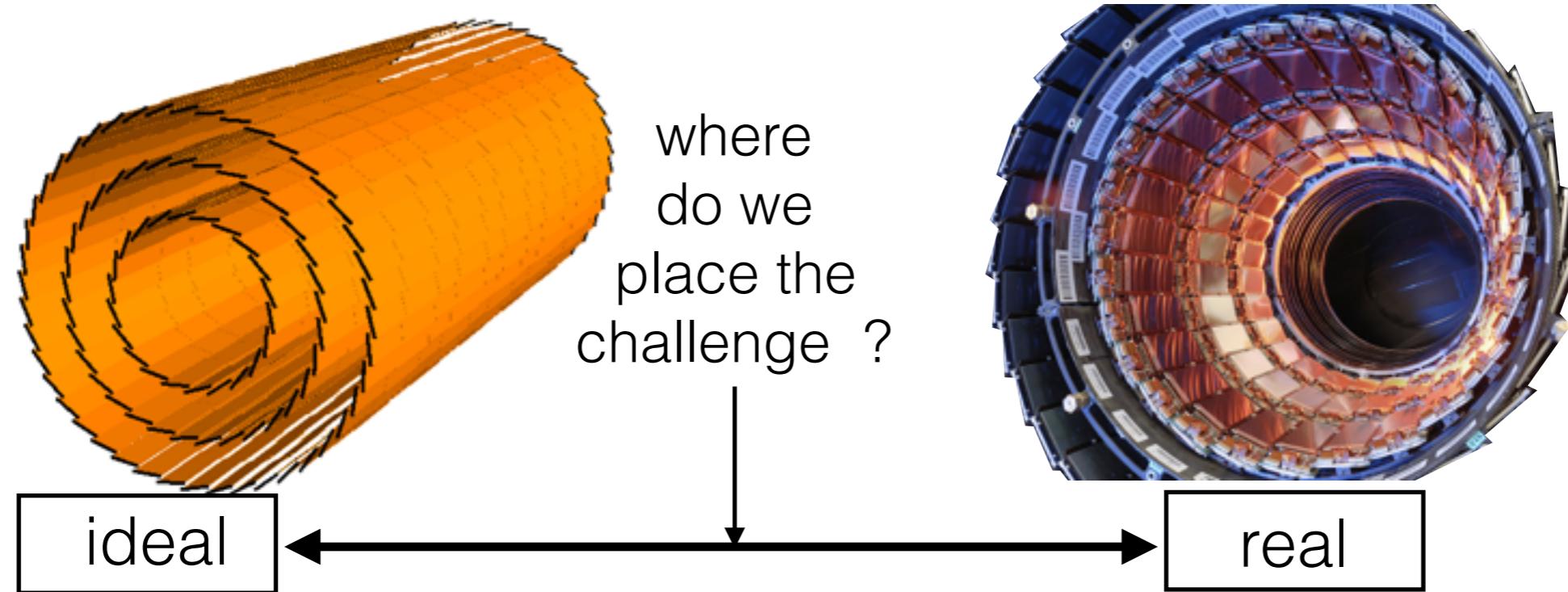
- ▶ Formulation of the challenge is a challenge itself

- balance of simplification versus complexity must be met to make such a challenge useful
 - formulation of the goal is similarly difficult
 - how can we take timing aspects into account ?



A HEP Tracking Machine Learning Challenge (2)

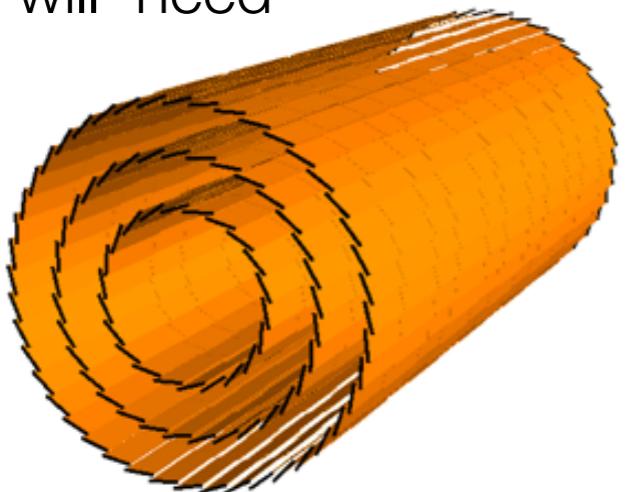
- ▶ Our current understanding - preconditions
 - difficulty of pattern recognition is not only due to combinatorics
 - ideal world (perfect magnetic) field and no process noise, there is an ideal solution:
conformal mapping techniques have been successfully used in the past,
e.g. Hough transformation, Riemann sphare
 - yet, only TPC-like detectors currently allow such semi-algorithmic approaches
FCC detector concepts actually bring more complexity rather than simplicity
 - process noise is a problem, they make the problem messy
multiple coulomb scattering, energy loss effects, hadronic interactions



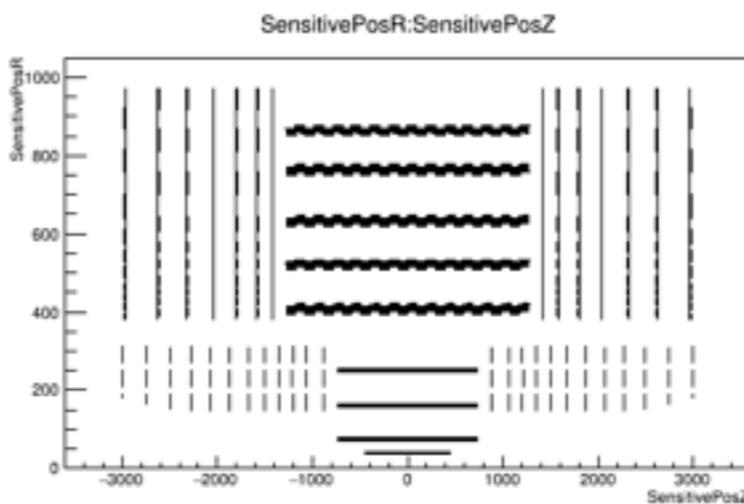
A HEP Tracking Machine Learning Challenge (3)

- Challenge will have to take several stages

- we will need



detector geometry
planar barrel/EC type detector
pixel/strip system

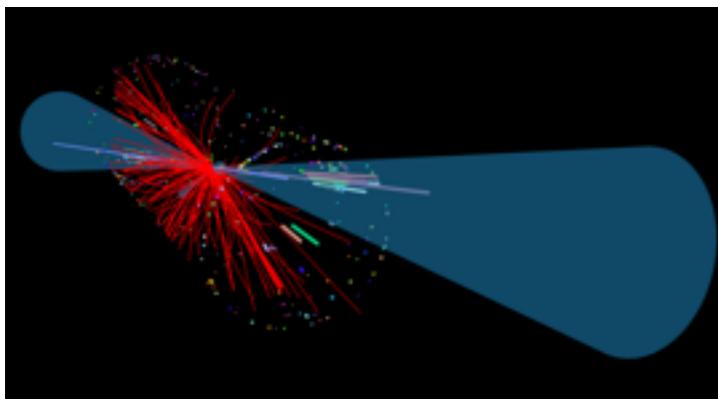


simulation
with the possibility to
simplify where possible

```
1 {  
2   "hits": [  
3     23.04,  
4     -123.2,  
5     83.22  
6   ]  
}
```

Valid JSON

event data
easily readable,
platform independent



visualisation
of geometry,
hits & found tracks



well defined goal
what is success
and how we measure them



different categories
for different
solutions