Alternative KF Track Finding

23 August 2024

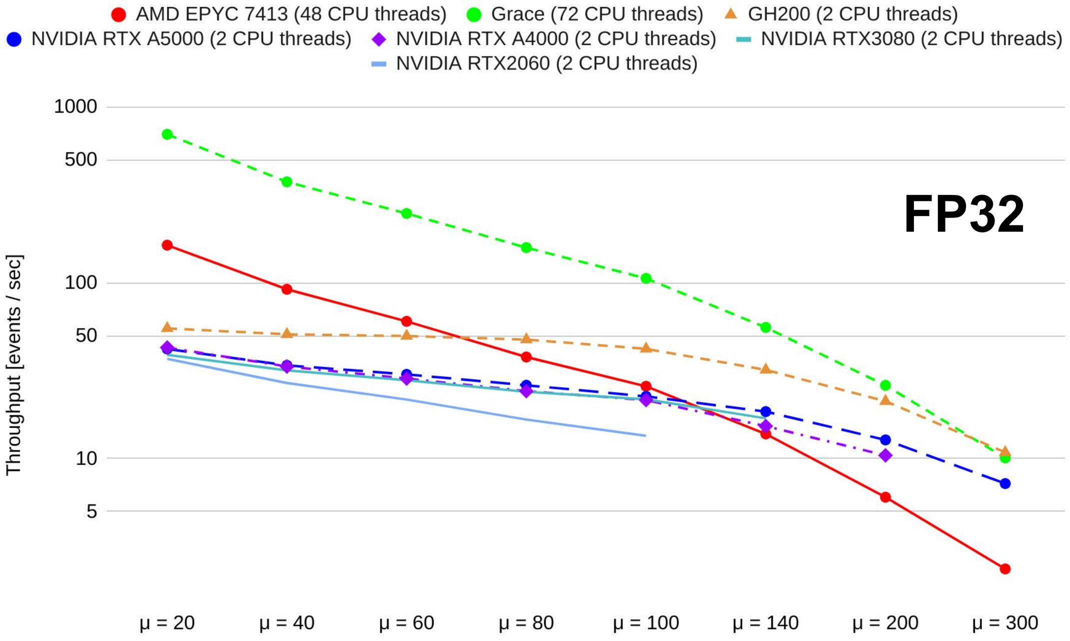
Markus Elsing

Recent TRACCC Results...

Performance studies on TRACCC on full track finding on ODD

- Still early stage, of course
- First results on tracking performance and throughput

Few things to notice:



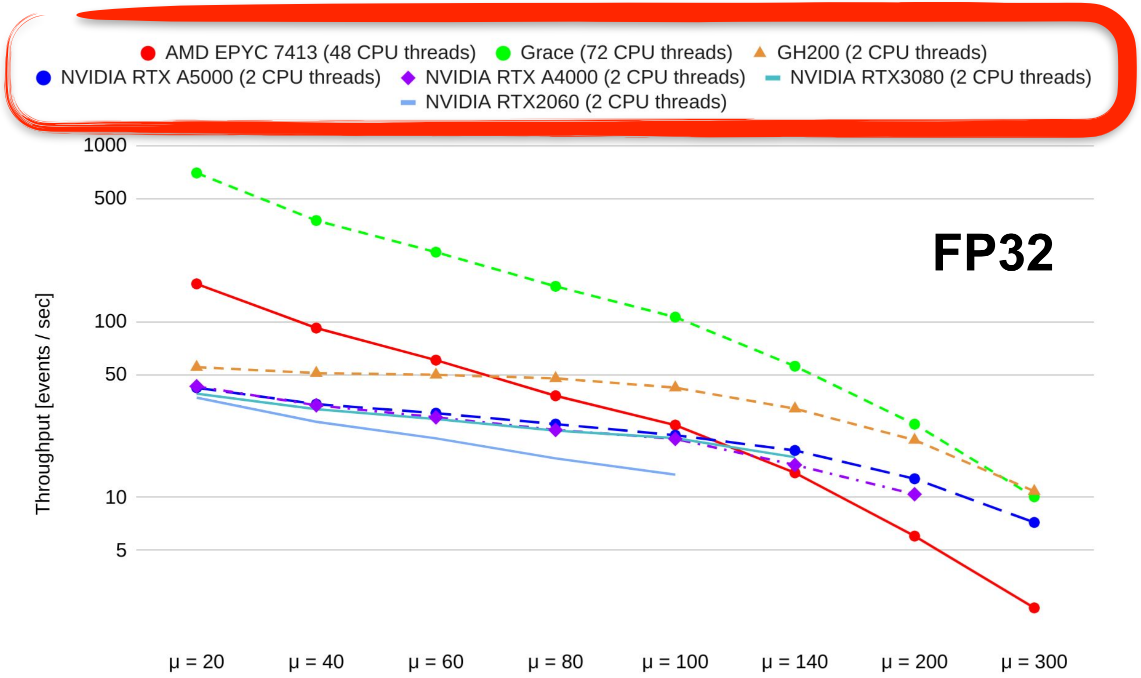
Recent TRACCC Results…

Performance studies on TRACCC on full track finding on ODD

- Still early stage, of course
- First results on tracking performance and throughput

Few things to notice:

- Results are TRACCC, also on CPU



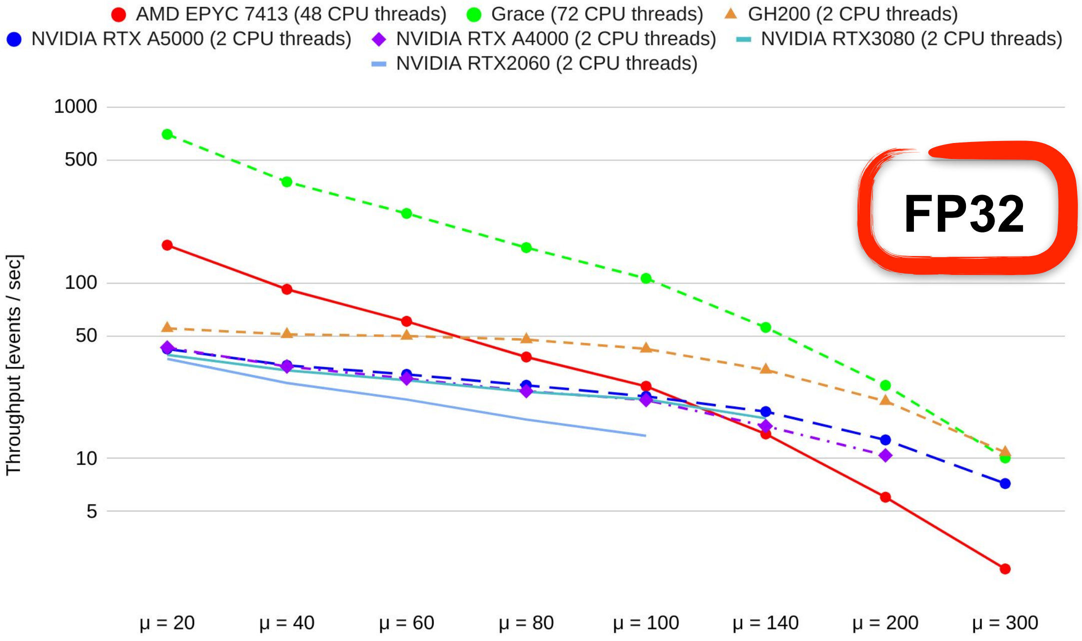
Recent TRACCC Results…

Performance studies on TRACCC on full track finding on ODD

- Still early stage, of course
- First results on tracking performance and throughput

Few things to notice:

- Results are TRACCC, also on CPU
- FP32 with offline tracking code gives performance issues



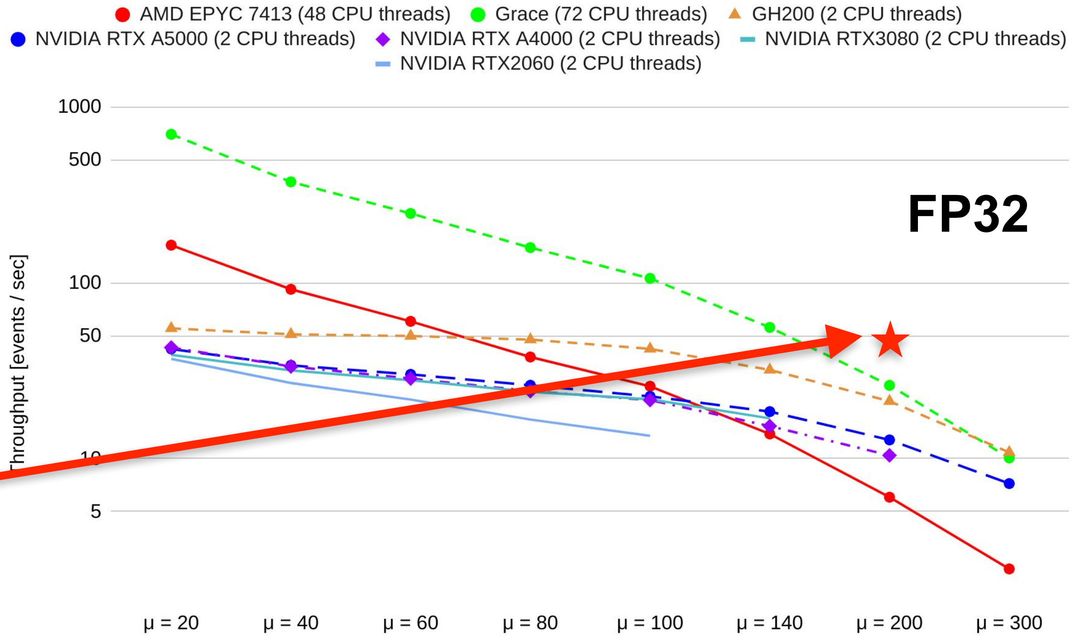
Recent TRACCC Results...

Performance studies on TRACCC on full track finding on ODD

- Still early stage, of course
- First results on tracking performance and throughput

Few things to notice:

- Results are TRACCC, also on CPU
- FP32 with offline tracking code gives performance issues
- ACTS on ODD on EPYC



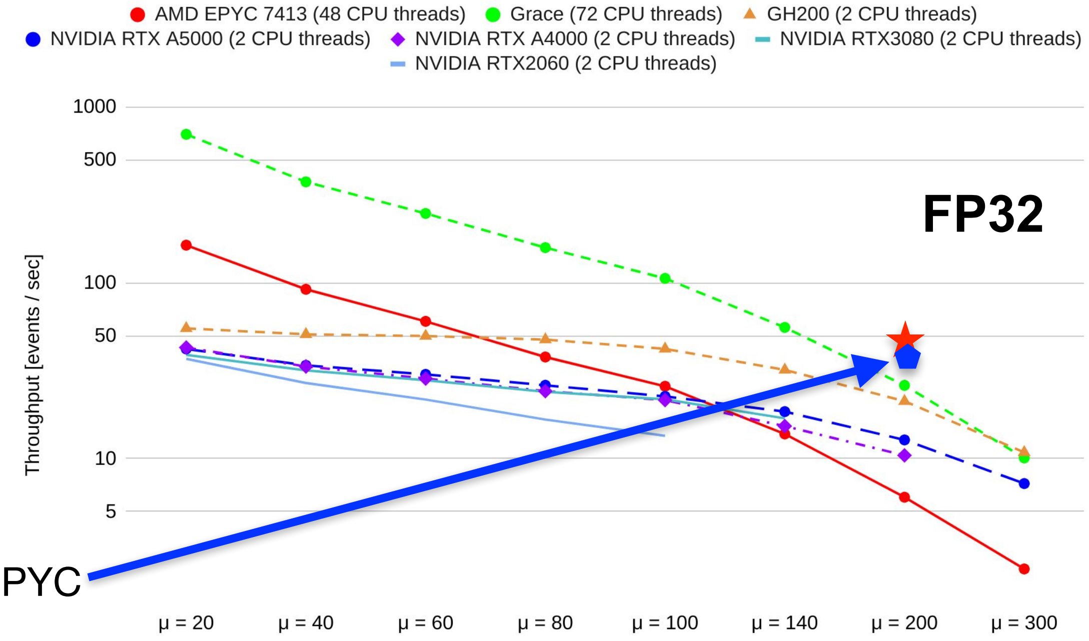
Recent TRACCC Results...

Performance studies on TRACCC on full track finding on ODD

- Still early stage, of course
- First results on tracking performance and throughput

Few things to notice:

- Results are TRACCC, also on CPU
- FP32 with offline tracking code gives performance issues
- ACTS on ODD on EPYC
- ACTS fast tracking on FULL ITk on EPYC



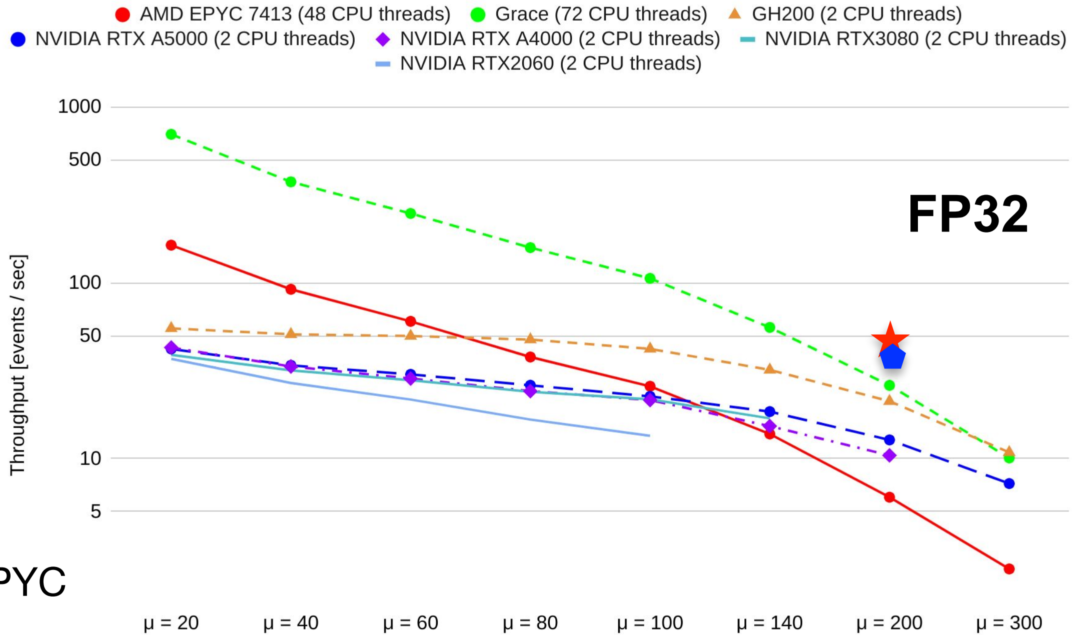
Recent TRACCC Results...

Performance studies on TRACCC on full track finding on ODD

- Still early stage, of course
- First results on tracking performance and throughput

Few things to notice:

- Results are TRACCC, also on CPU
- FP32 with offline tracking code gives performance issues
- ACTS on ODD on EPYC
- ACTS fast tracking on FULL ITk on EPYC
- The benchmark is throughput / KCHF at full physics performance



Recent TRACCC Results...

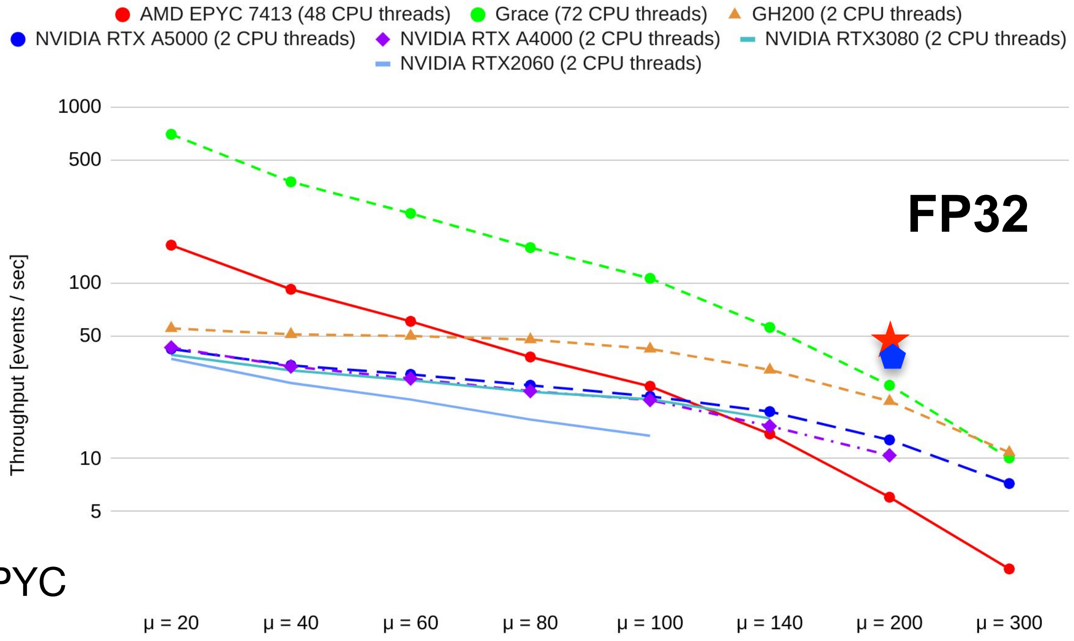
Performance studies on TRACCC on full track finding on ODD

- Still early stage, of course
- First results on tracking performance and throughput

Few things to notice:

- Results are TRACCC, also on CPU
- FP32 with offline tracking code gives performance issues
- ACTS on ODD on EPYC
- ACTS fast tracking on FULL ITk on EPYC
- The benchmark is throughput / KCHF at full physics performance

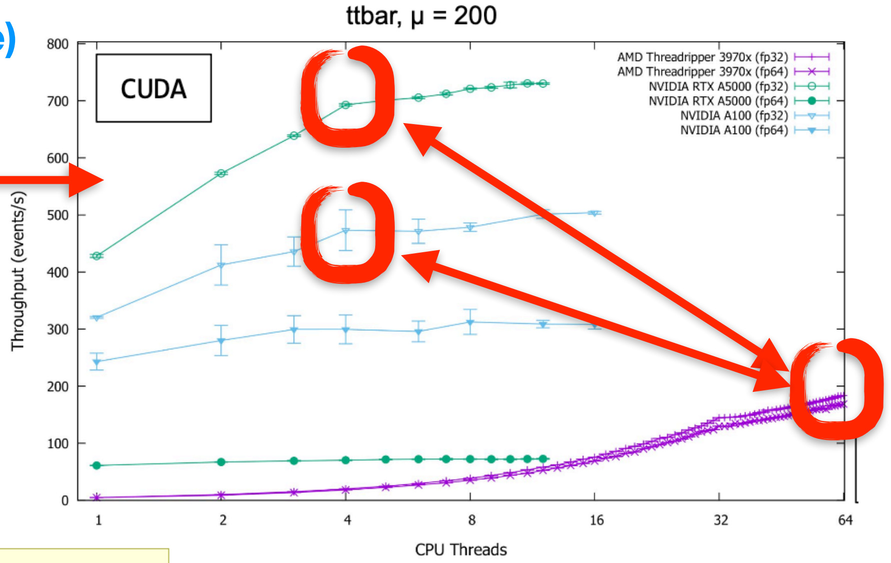
➡ **This is a good start, but the gap is sizeable !**

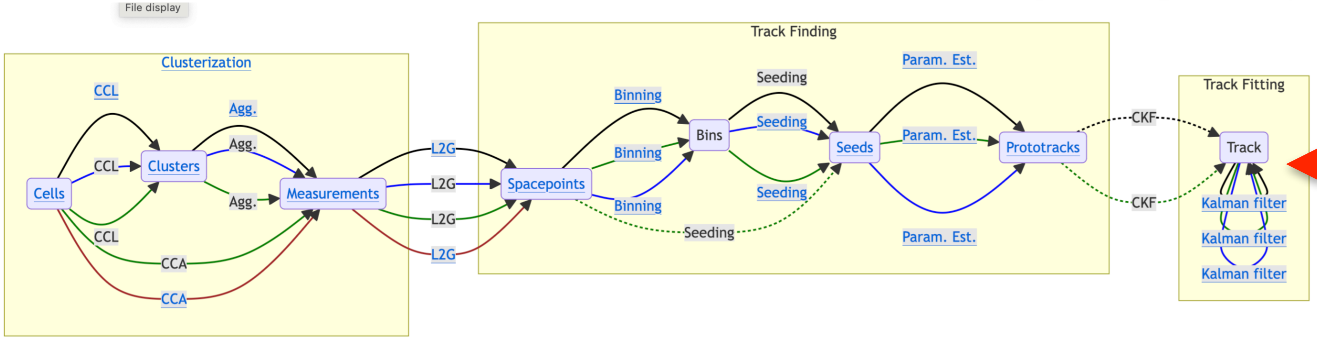


Current TRACCC Reconstruction Chain

TRACCC strives to reproduce ACTS (ATLAS offline) reconstruction chain

- Early stages from hits to track seeds are all relatively well suited for GPUs in terms of algorithmic approaches and decomposition
- Results show a sizeable speedup, but same comments on 32bit, ODD vs (full ITk) ACTS, and throughput / KCHF apply
- Limit is (C)KF track finding which is almost serial



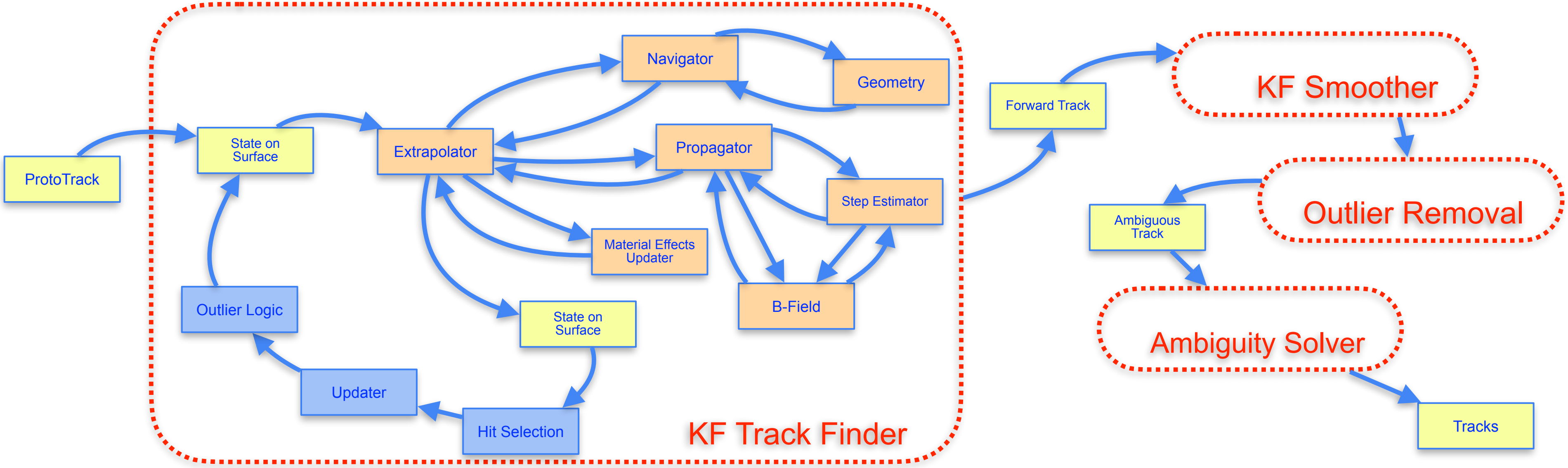


- Already the diagram shows that this is not as nicely composed into suitable algorithmic kernels

What could be done to improve Track Finding ?

The Combinatorial Klaman Filter involves several (nested) loops of different length, branching and sequences of decisions, not suited for GPU processing

- In reality the GPU code (like the offline) does not run a full combinatorial filter, but a progressive scan taking only the best hit on each sensor surface
- While this is ok, even a KF track finder is quite involved



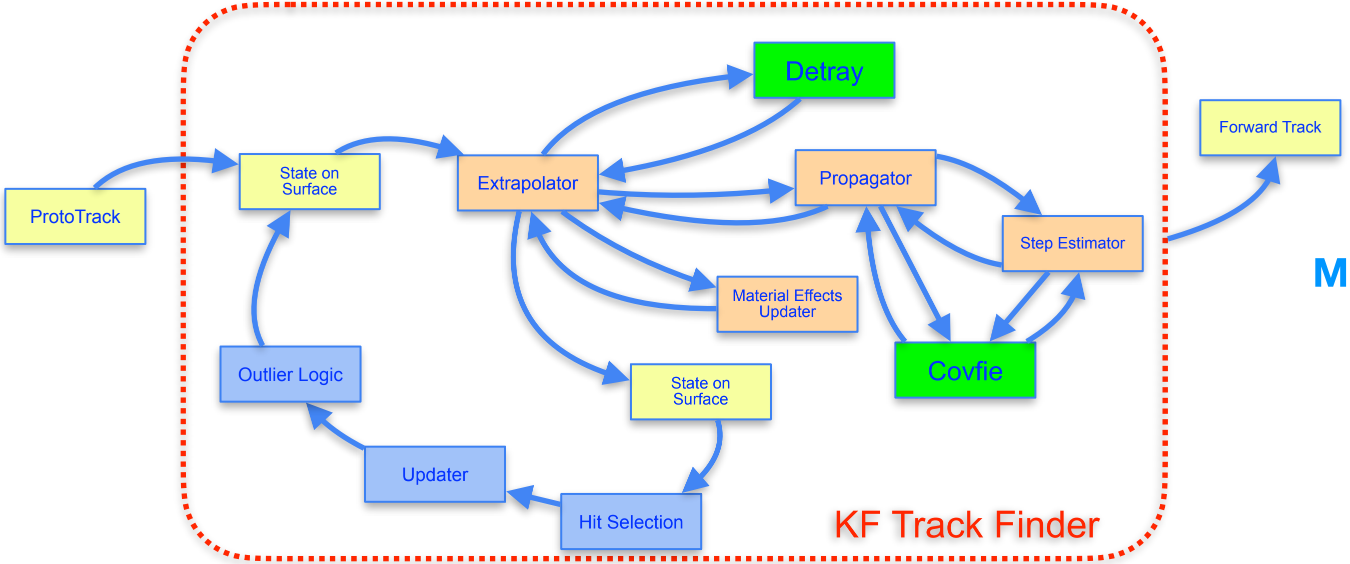
What could be done to improve Track Finding ?

Let's focus on the KF Track Finder first

- Discuss KF Smoother, Outlier Removal and Ambiguity Solver later

Important developments to port functionality onto GPUs

- Detray for navigating the geometry, Covfie for B-Field lookups



Main issue stays !

- Several entangled loops and branching

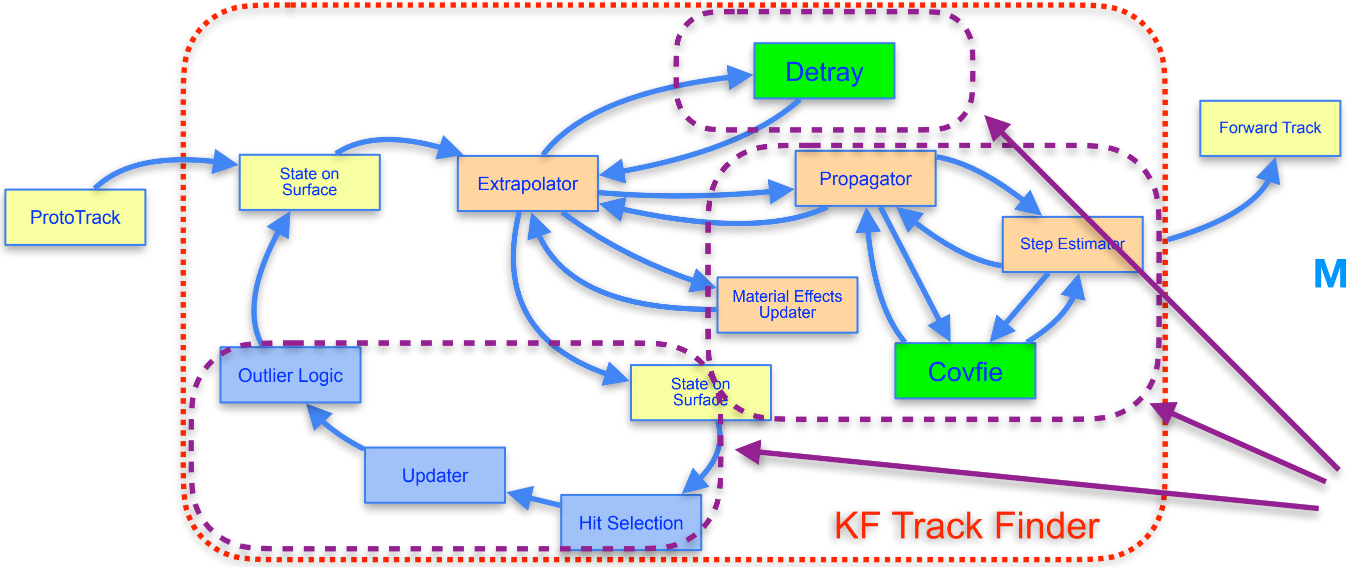
What could be done to improve Track Finding ?

Let's focus on the KF Track Finder first

- Discuss KF Smoother, Outlier Removal and Ambiguity Solver later

Important developments to port functionality onto GPUs

- Detray for navigating the geometry, Covfie for B-Field lookups



Main issue stays !

- Several entangled loops and branching
- Could we disentangle the loops ?

KF Track Finding Approach

To disentangle the loops we have to rethink the algorithmic approach

- ACTS implements the NewTracking CKF based on the original Kalman Filter (Fruehwirth et al.)
- Mathematical approach works well on CPUs, but requires the entangled loops

Let's look a bit closer...

- The propagator implements the track model, it provides the transport Jacobian

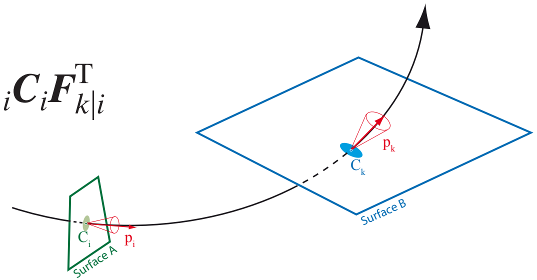
track following in mathematical terms:

$$\boldsymbol{q}_k = \boldsymbol{f}_{k|i}(\boldsymbol{q}_i)$$

convariance: $\boldsymbol{C}_k = \boldsymbol{F}_{k|i}\boldsymbol{C}_i\boldsymbol{F}^{\mathrm{T}}_{k|i}$

with: $\boldsymbol{f}_{k|i}$ ~ track model

$\boldsymbol{F}_{k|i} = \dfrac{\partial \boldsymbol{q}_k}{\partial \boldsymbol{q}_i}$ ~ Jacobi matrix



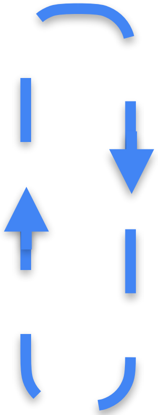
KF Track Finding Approach

To disentangle the loops we have to rethink the algorithmic approach

- ACTS implements the NewTracking CKF based on the original Kalman Filter (Fruehwirth et al.)
- Mathematical approach works well on CPUs, but requires the entangled loops

Let's look a bit closer...

- The KF iteratively propagates and then updates the prediction with a measurement, propagates...



1. propagate p_{k-1} and its covariance C_{k-1} :

$$\boldsymbol{q}_{k|k-1} = \boldsymbol{f}_{k|k-1}(\boldsymbol{q}_{k-1|k-1})$$

$$\boldsymbol{C}_{k|k-1} = \boldsymbol{F}_{k|k-1}\boldsymbol{C}_{k-1|k-1}\boldsymbol{F}^{\mathrm{T}}_{k|k-1} + \boldsymbol{Q}_k$$

with Q_k ~ noise term (M.S.)

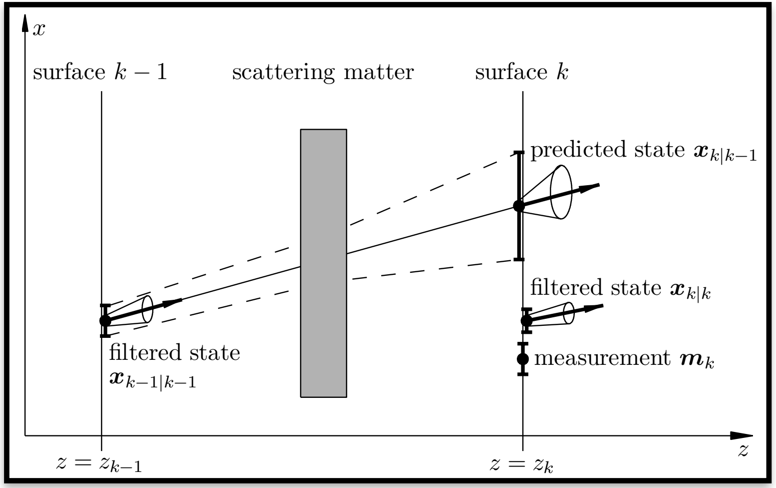
2. update prediction to get $q_{k|k}$ and $C_{k|k}$:

$$\boldsymbol{q}_{k|k} = \boldsymbol{q}_{k|k-1} + \boldsymbol{K}_k[\boldsymbol{m}_k - \boldsymbol{h}_k(\boldsymbol{q}_{k|k-1})]$$

$$\boldsymbol{C}_{k|k} = (\boldsymbol{I} - \boldsymbol{K}_k\boldsymbol{H}_k)\boldsymbol{C}_{k|k-1}$$

with K_k ~ gain matrix :

$$\boldsymbol{K}_k = \boldsymbol{C}_{k|k-1}\boldsymbol{H}_k^{\mathrm{T}}(\boldsymbol{G}_k + \boldsymbol{H}_k\boldsymbol{C}_{k|k-1}\boldsymbol{H}_k^{\mathrm{T}})^{-1}$$


x
surface $k-1$
scattering matter
surface k
predicted state $\boldsymbol{x}_{k|k-1}$
filtered state $\boldsymbol{x}_{k|k}$
filtered state $\boldsymbol{x}_{k-1|k-1}$
measurement $\boldsymbol{m}_k$
$z = z_{k-1}$
$z = z_k$
z

Alternative Formulation of Kalman Filter

An alternative Formulation of the Kalman Filter uses a Reference Trajectory

- Rudi, Pierre and I used this for DELPHI at the time...

Mathematically this is a different way of linearising the fitting problem

- Taylor expansion of the track parameters $\mathbf{q \sim q_0 + \delta q}$ **+ higher-terms**
- Stick this into the track model gives

$$\boxed{\mathbf{f(q) \sim f(q_0 + \delta q + \text{higher-terms}) \sim\ f(q_0) + F{\cdot}\delta q + \text{higher-terms}}}$$

Reference Trajectory

Transport Jacobian

Formulate the Kalman Filter as a fit for **δq**

- Mathematically one needs to replace $\mathbf{q_{k|k-1}}$ with $\mathbf{\delta q_{k|k-1}}$ making it:

$$\boxed{\mathbf{q_{k|k-1} = f_{k|k-1}(q_{k-1|k-1}) \sim f_{k|k-1}(q_{0,k-1}) + F_{k|k-1}{\cdot}\delta q_{k-1|k-1}}}$$

- Hence, the call to the propagator (track model) is replaced by the Reference Trajectory $\mathbf{f(q_0)}$!

Alternative Formulation of Kalman Filter

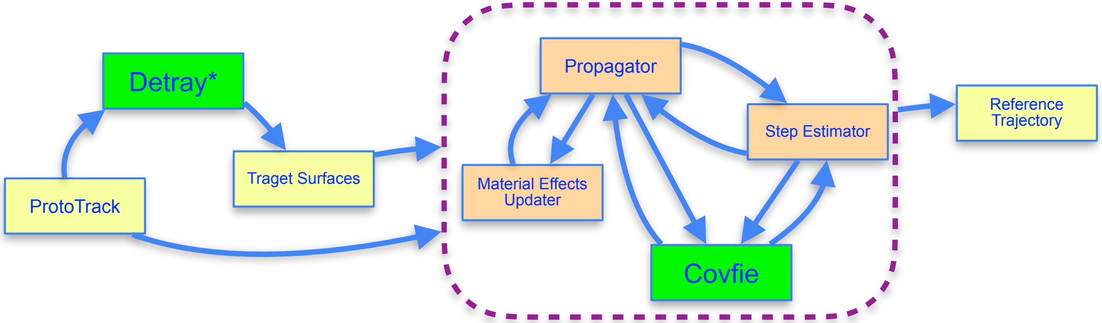
A Kalman Filter with a reference trajectory is a different way of linearising the track fit

- Mathematically it is totally sound, it is just a different way of linearising the track model
- The convergence of this KF track fit depends on the precision of the starting parameters $\mathbf{p}_0$
- If $\mathbf{p}_0$ is too far off, the linear term is not sufficient and/or the Reference Trajectory misses the right sensor surfaces
- In DELPHI, I did iterate the track fit once to improve convergence
- To ensure that all relevant surfaces are "on" the Reference Trajectory, consider overlapping sensors even in case of "near misses"

Alternative KF Track Finding Implementation

In practical terms, the starting parameters **p_0** are obtained from the ProtoTrack

- Hence we start with the normal output of the Track Seeding step
- We can use Detray (or a simple cone approach like Igor) to define the list of Target Surfaces
- Propagator loops over the (sorted) list of traget surfaces to build the Reference Trajectory **$f(p_0)$**
- Result is a much more linear algorithm flow:

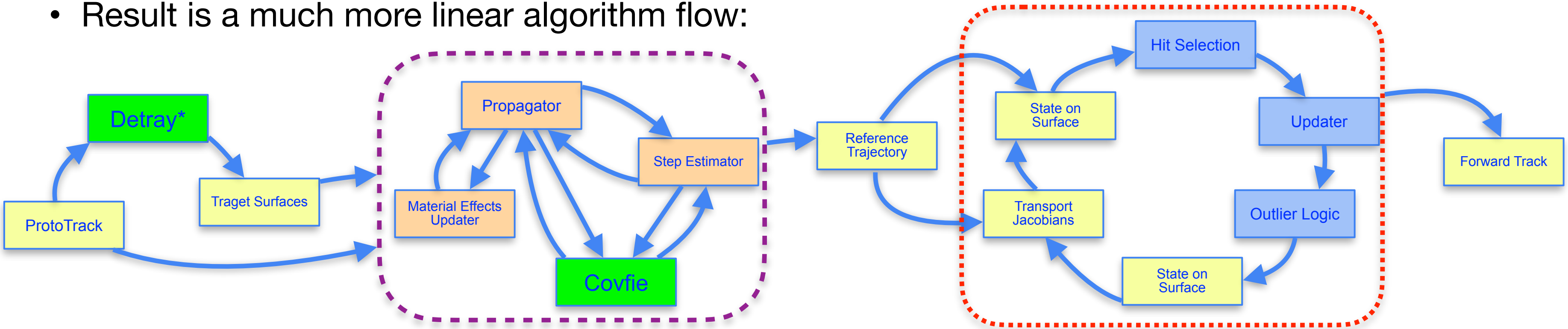


*Andi has an idea on how to do such a surface search in Detray

Alternative KF Track Finding Implementation

In practical terms, the starting parameters p_0 are obtained from the ProtoTrack

- Hence we start with the normal output of the Track Seeding step
- We can use Detray (or a simple cone approach like Igor) to define the list of Target Surfaces
- Propagator loops over the (sorted) list of traget surfaces to build the Reference Trajectory **$f(p_0)$**
- Result is a much more linear algorithm flow:



Adding the KF Track Finder now, it becomes a loop over surfaces on the Reference Trajectory

- Uses the Transport Jacobians as a replacement of the call to the Propagator !

*Andi has an idea on how to do such a surface search in Detray

Alternative KF Track Finding Implementation

Resulting algorithmic flow looks much more linear

- It builds fully on existing TRACC software, most of the required code exists
- What changes is the calling sequence and the mathematical approach to the KF

Each of the steps in the chain can be individually parallelised on the GPU

- Building the list of Target Surfaces using Detray
- Using the Propagator to build the Reference Trajectory
- The KF Track Finder using the Reference Trajectory

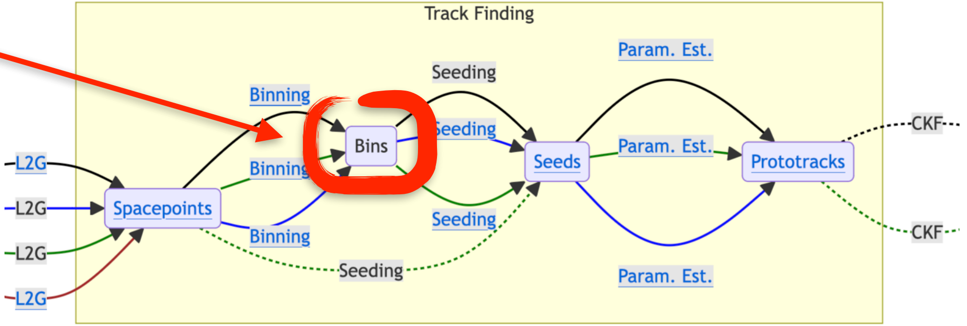
The Propagator is still the most involved piece of code, but disentangled from the Detray

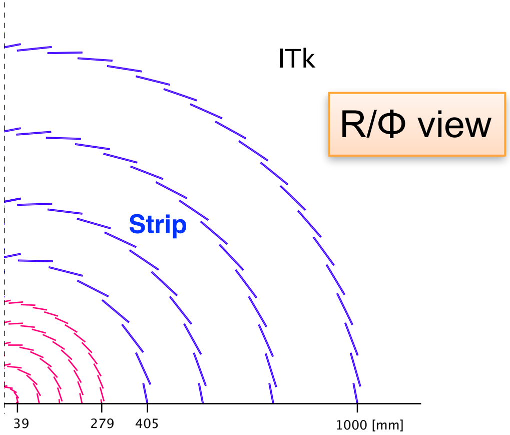
- Note, the material effects are handled at this stage too !
- Multiple scattering enters in the transport of the covariance using the Jacobians
- Energy loss enters as a change in the curvature of the Reference Trajectory

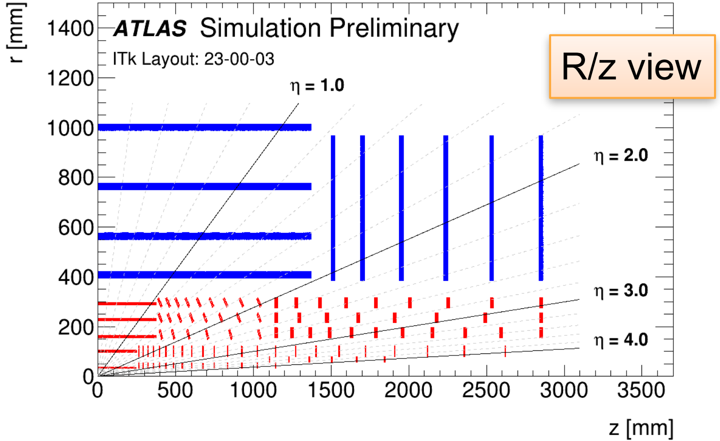
On Parallelising...

One final remark on the how to parallelise the algorithmic calls in the chain

- The Track Seeding iterates over **Bins** !
- All tracks in a Bin cross a similar number of surfaces, so parallel loops make sense
- Tracks in the neighbouring Bins in R/ϕ as well
- Bins are a natural sorting for Prototracks, Reference Trajectories ...







Questions ?