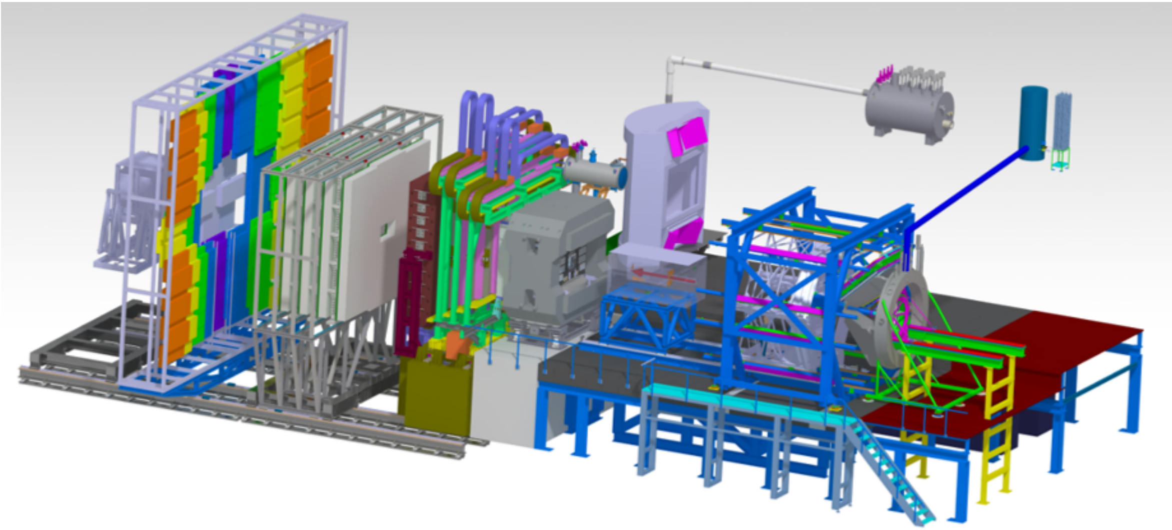
CHEP Conference, October 2024, Krakow

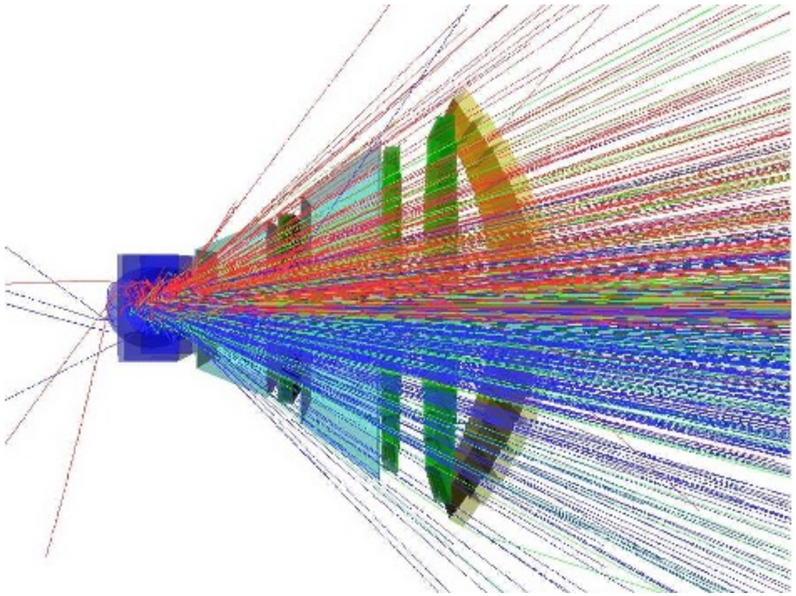
FREE-STREAMING ONLINE TRACKING IN CBM

SERGEY GORBUNOV [1]
SERGEI ZHARKO [1]
VALENTINA AKISHINA [1,2]

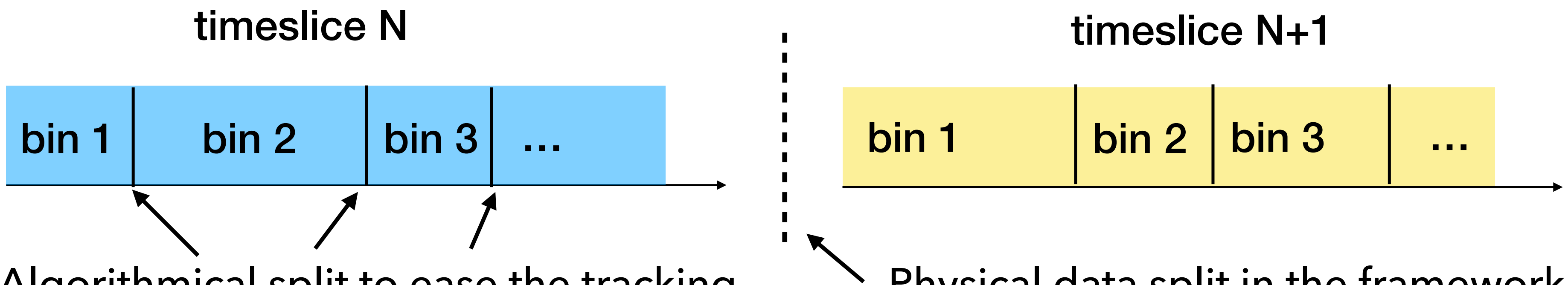
[1] GSI Helmholtzzentrum für Schwerionenforschung
[2] FIAS Frankfurt Institute for Advanced Studies



- 10^7 heavy ion collisions/sec
- up to 700 charged particles/collision
- free-streaming tracking
- online event selection
- operation begins in 2028, test setup is already running



The data is received in large timeslices



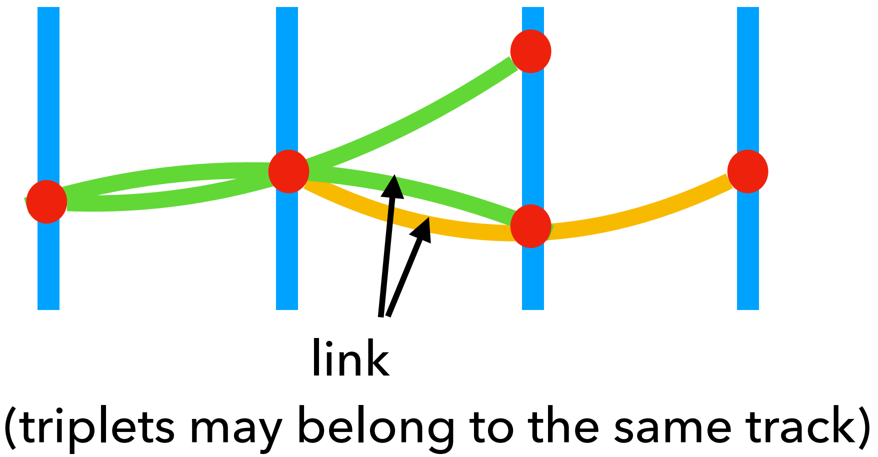
Algorithmical split to ease the tracking.
The time bins must be merged.

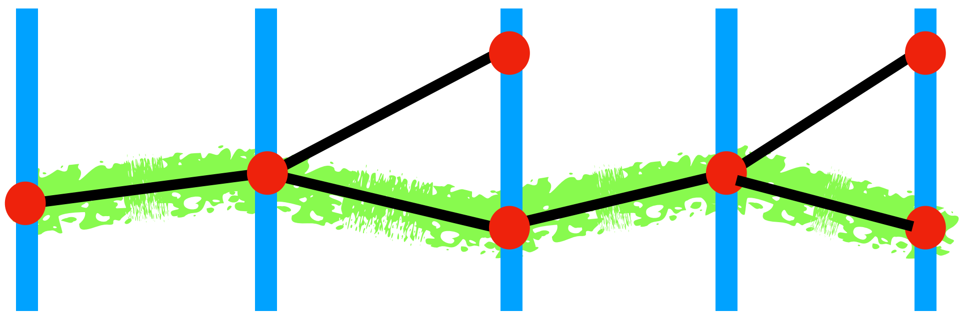
Physical data split in the framework.
Timeslices can’t be merged.

Our approach to process the free-streaming data:

Cut data in small time bins and process them with existing event-by-event tracking.

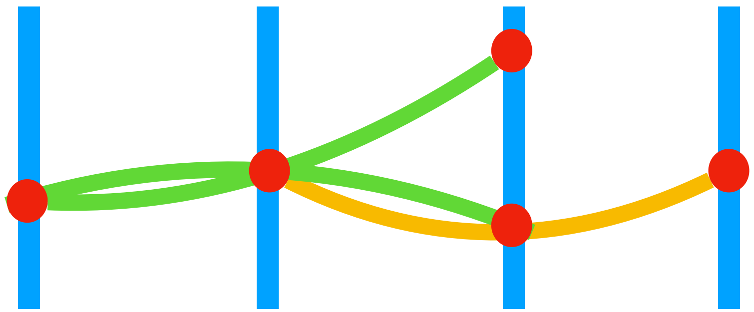
- account for the hit time when matching the hits
- properly treat the time bin borders





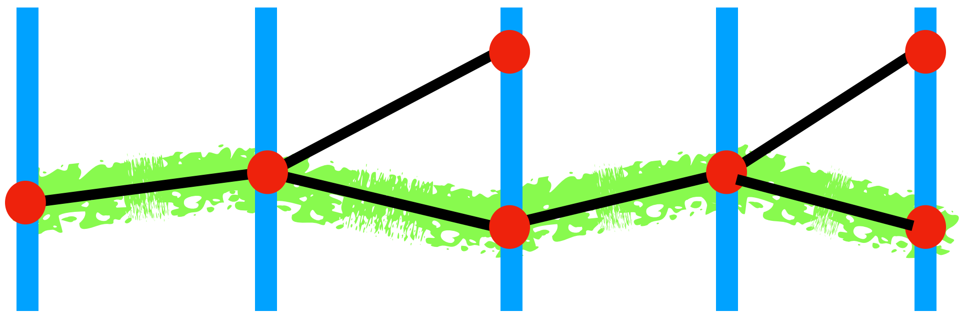
Iterations

1. Create triplets on every 3 consecutive stations
2. Link triplets pair-wise into combinatorial trees
3. Look for good tracks within the trees



Triplet construction:

- Estimate search windows with the Kalman Filter (== track following)
- to be replaced with polynomial approximation / NN



Search in combinatorial trees:

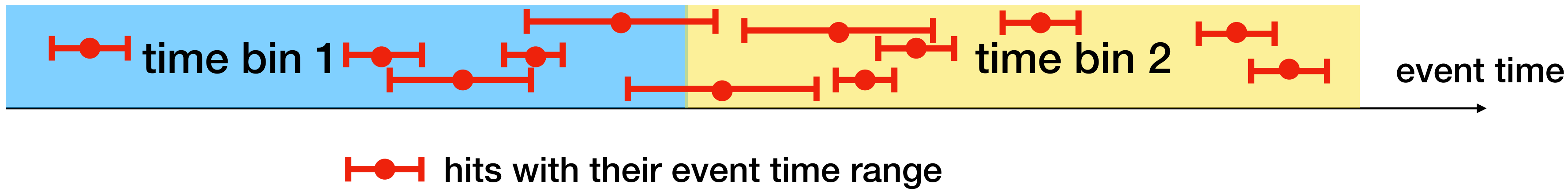
- Length and sum of triplet-match χ^2 for selection of the best track candidate
- Kalman Filter fit of the best candidate for the final quality check

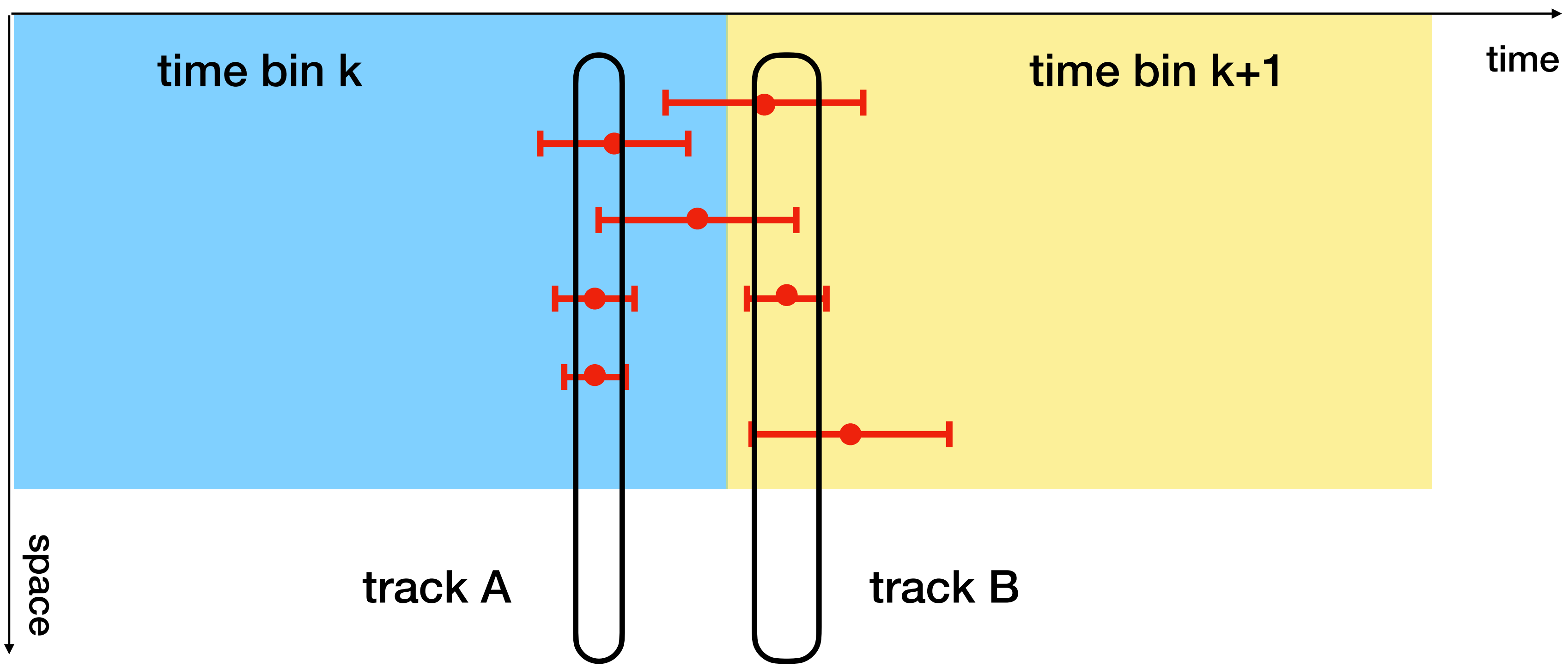
Time sliding window algorithm to perform tracking in time bins

Large timeslices are split into small time bins

Each time bin is processed as one super-event w/o data sorting in time

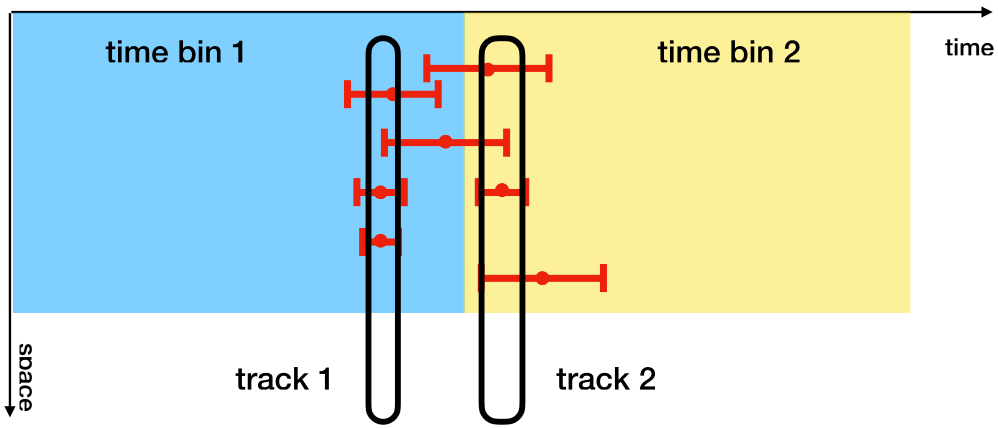
consider hit „event time range“ for the overlaps



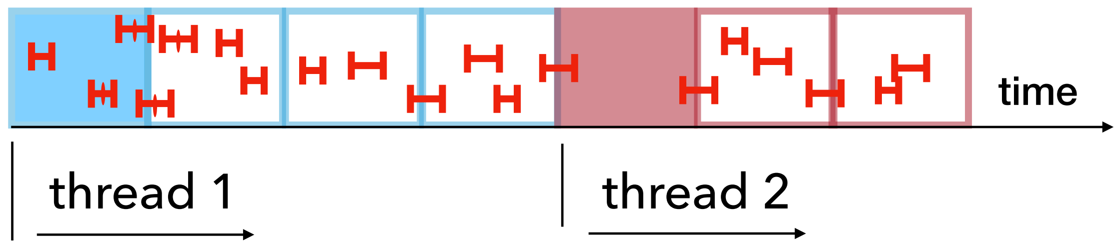


Algorithm for the time bin k:

- store only tracks that have at least one hit not overlapping with the next time bin
- remove hits that belong to the reconstructed tracks



multithreading is straightforward



Features:

- no extra track merging routine at the border
- hits that overlap with the border are processed twice
- easy multithreading
- robust to the extreme cases where some hits belong to several time bins
- hits need only be approximately time-ordered to identify their time bins

Algorithm to read slightly time-disordered data w/o sorting

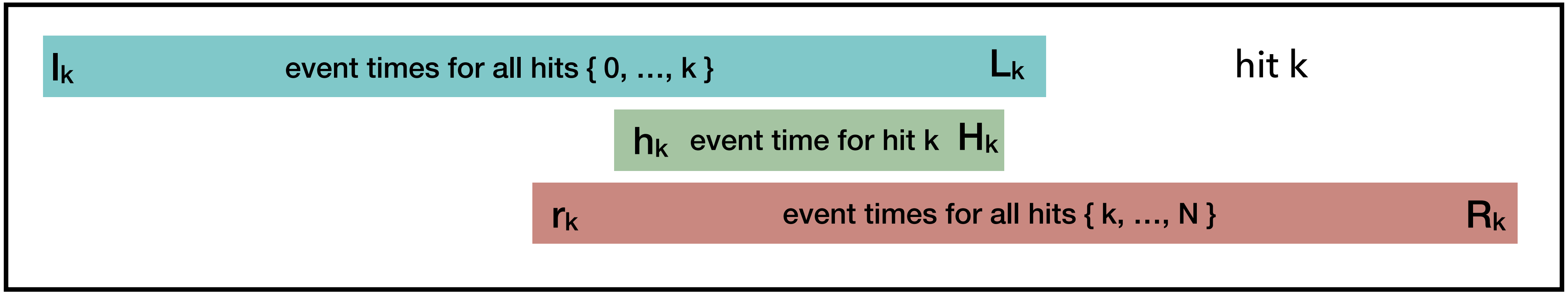
Assumption: the data are more or less sorted in time on the large scale!

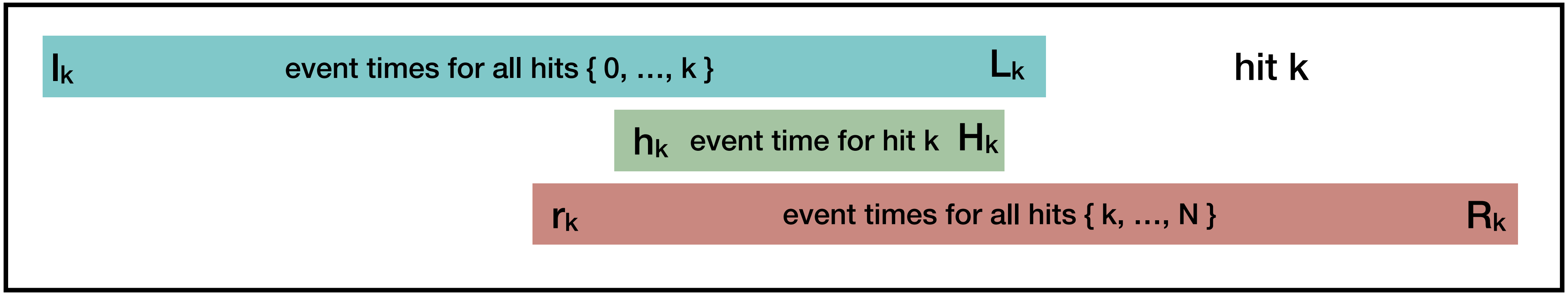
input data array

hit k
- hit time
- time error
- …

auxiliary array || to the input

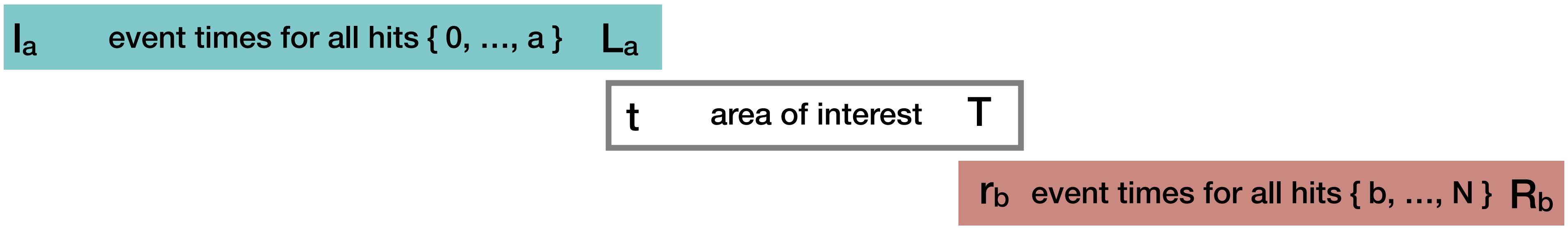
hit k
- [h_k, H_k] – event time range for the hit k
- [l_k, L_k] – aggregated event time range for all the hits { 0, …, k }
- [r_k, R_k] – aggregated event time range for all the hits { k, …, N }

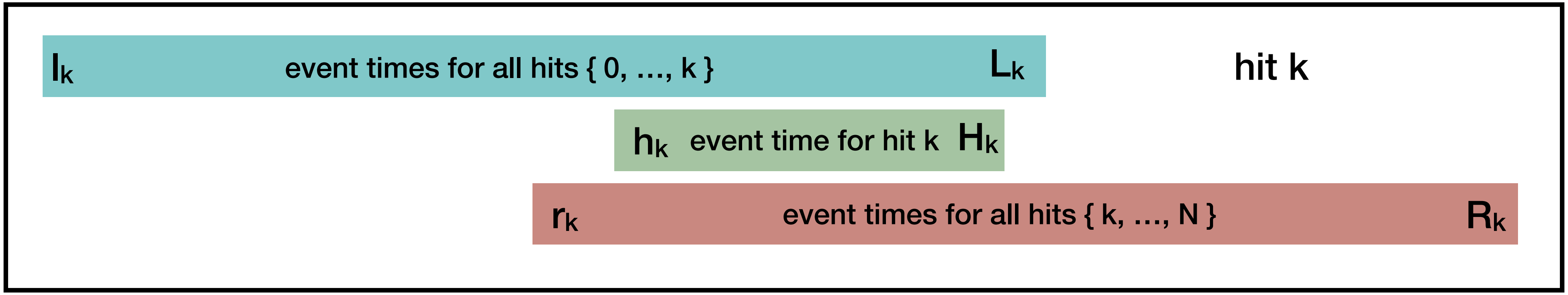




Now, to collect all the hits that overlap with time area [t, T]:

- start at the first hit a with ($t <= L_a$)
- stop at the last hit b with ($r_b <= T$)





Features:

- calculating the values: 2 passes over the input data array
- no overhead when the input is perfectly sorted. The data is read continuously while the time-sliding window is moving.
- a slight disorder in time => maybe a few extra reads at the time bin edges
- no problem with the overlapping event time regions (when hits can not be perfectly sorted)

<u>No more requirement for the input to be sorted in time!</u>

To match hits in time one needs to include the time to the track model.

To propagate the time along the trajectory one needs to know the velocity:

- In the physics analysis:
  v can be calculated from the mass hypothesis and the measured momentum.

- In the combinatorial tracking:
  no mass and/or no momentum is known.
  One can not just use average values because velocity variations should be considered.

Our solution: Make the track velocity a free parameter of the track model.

CBM track model for 4D fit

track = { x, y, tx, ty, q/p, time, 1/v } at z Propagation: Δtime = (1/v) * ΔL

Options to set the velocity:

1. v = from a priori known range
2. v = from hits
3. v = from the measured q/p and the mass hypothesis
   (v and q/p can be bind together after the fit via cov. matrix)

Features:

- accurate estimation of the time error
- time propagation is (almost) linear in z
- one can use all three above options together

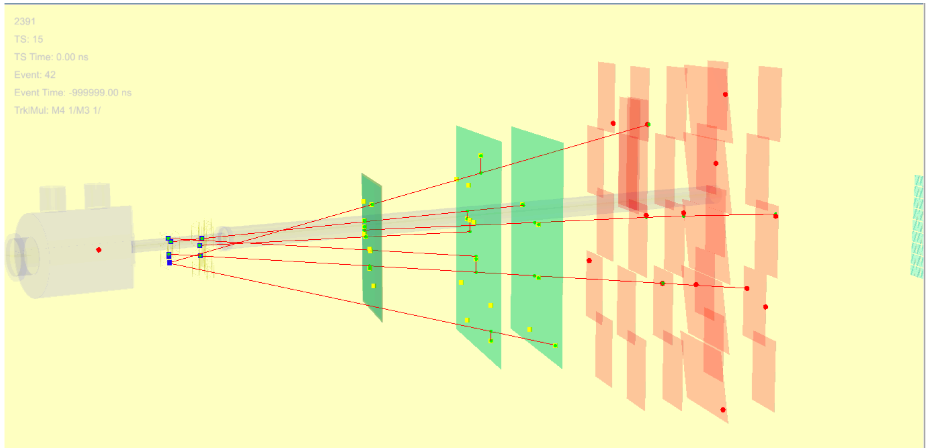
- stable efficiency up to 1MHz,
- degradation at 10 MHz due to the readout dead time

- comparable speed:
  - 3D: 8.2ms / event
  - 4D: 8.5ms / event

1000 AuAu UrQMD minimum bias events at 10 AGeV STS channel dead time 800 ns

Track type	E-by-E	0.1MHz	1MHz	10MHz
Long high-p primary	99.5	99.8	99.8	99.7
All tracks	91.4	92.7	92.1	85.3
Primary high-p	96.5	97.7	97.1	90.0
Primary low-p	91.6	93.2	92.6	86.9
Secondary low-p	63.8	66.5	66.2	59.7
Clone	1.3	0.7	0.7	1.38
Ghost	1.2	1.0	1.1	2.6
True hits per reco track	90.5	92.1	91.9	90.4
Hits per MC track	6.84	6.83	6.78	6.36

Reconstructable track — has at least 4 consecutive mc points, p > 0.1 GeV/c
Clone — more than one track obtained for one simulated particle
Reconstructed track — purity >= 70%, Ghost — purity < 70%



Test setup:

- designed to test the detectors
- real-life problems: acceptance gaps, noise, misalignment
- tracking with different PID detectors

Beamtime 2024:

- the tracker was running smoothly all the time with the required speed*
- it finds many tracks - also long tracks - with no calibration or alignment
- it produces QA plots

TRACKING FEATURES

- process time slices with no-merging sliding window
- algorithm to read the data without time sorting
- track model with the explicit velocity

PLANS

- Speedup the triplet construction by replacing the Kalman Filter track following with polynomial search windows
- port the code to GPU (work in progress)

THANK YOU!