LHCb upgrade II primary vertex reconstruction

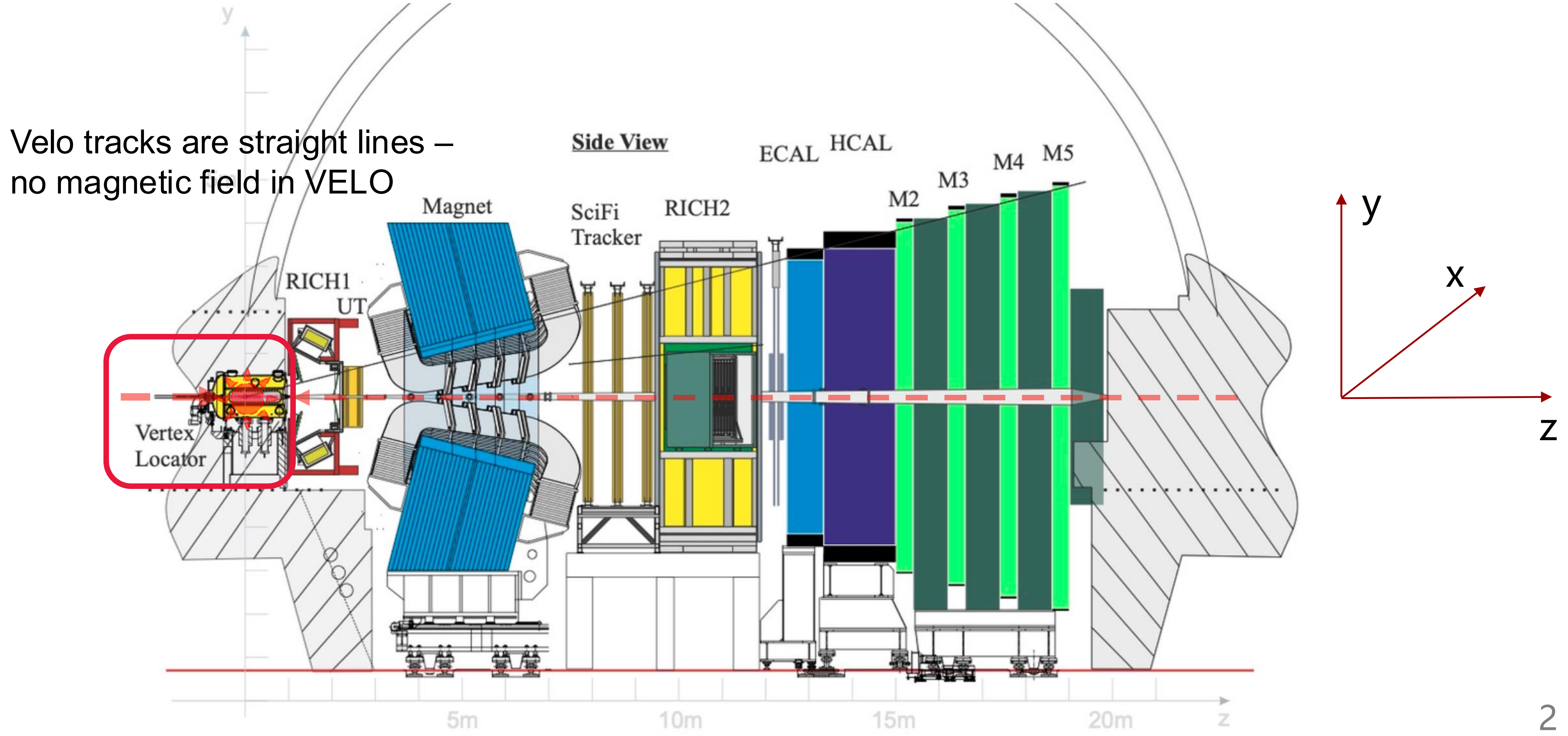
Roel Aaij, Wouter Hulsbergen, Zhihong Shen

October 24th, 2025

FASTER meeting

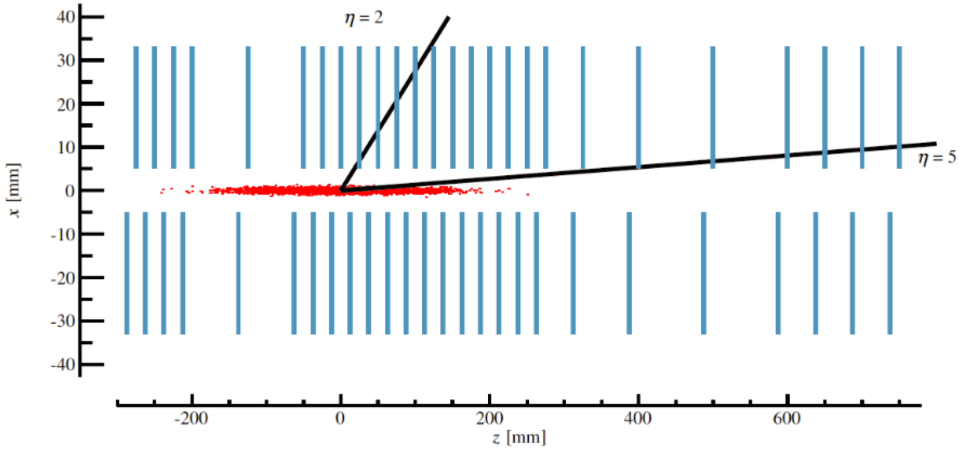
LHCb detector

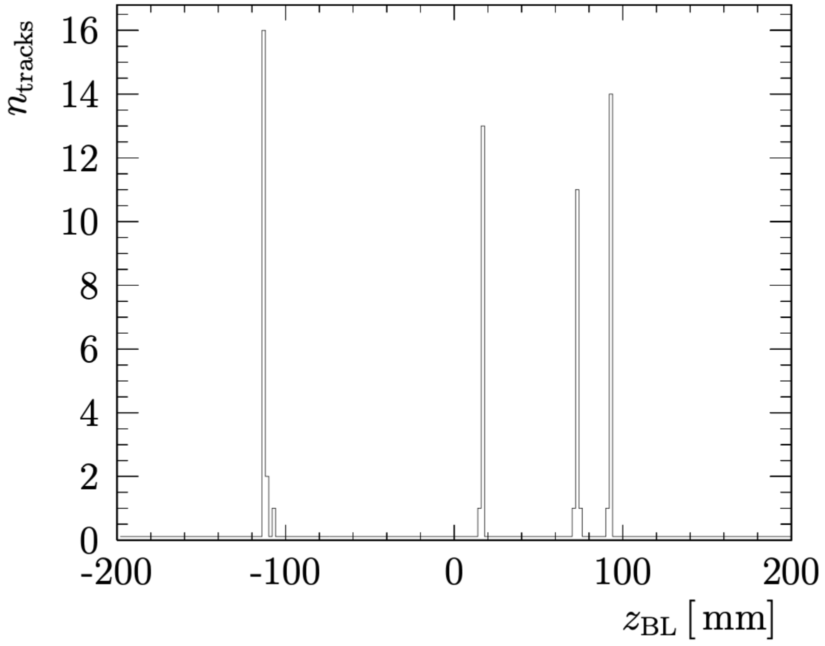
- A single-arm forward spectrometer covering $2 < \eta < 5$



Current PV reconstruction algorithm

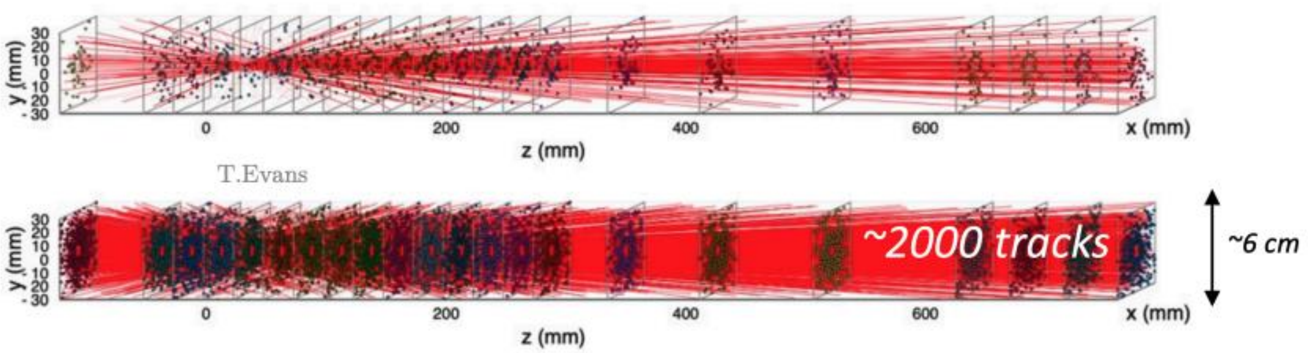
- Fast GPU-based algorithm implemented in Allen, the LHCb High-Level Trigger 1 framework
- Basic idea:
  1. Extrapolate tracks to the position of closest approach (POCA) to the known beamline
  2. Fill a histogram of POCA z-axis
  3. Identify peaks in the histogram as PV seeds.
  4. Associate tracks to each seed
  5. Perform a weighted PV fit





Challenges in LHCb Upgrade II

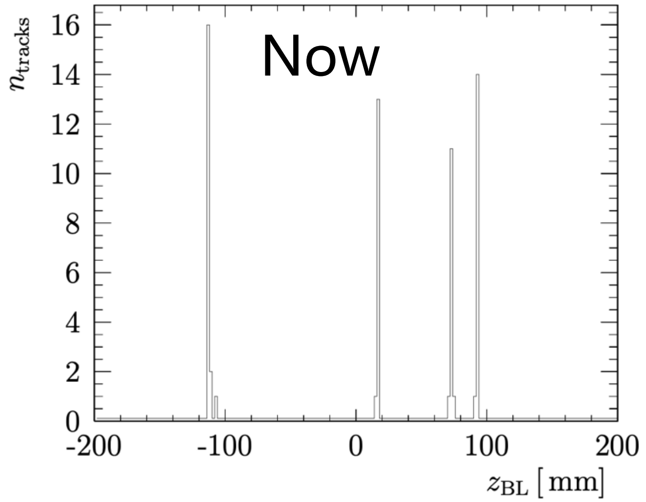
- Operating at an instantaneous luminosity of $\mathcal{O}(10^{34})\ \mathrm{cm}^{-2}\mathrm{s}^{-1}$; $\sim 7.5 \times$ larger
  - Much larger number of tracks and primary vertex (PV)

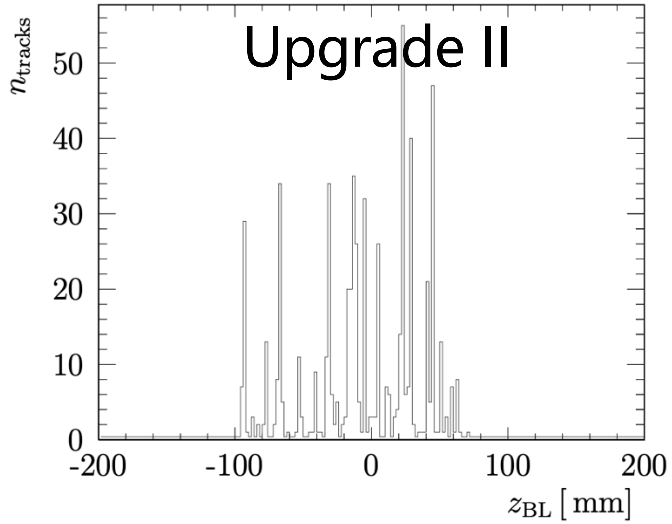


Now

Upgrade II

- What if we use current algorithm

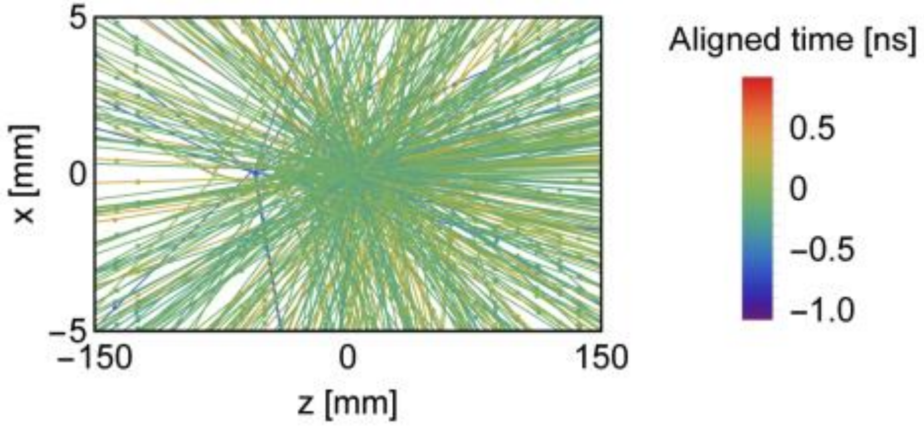




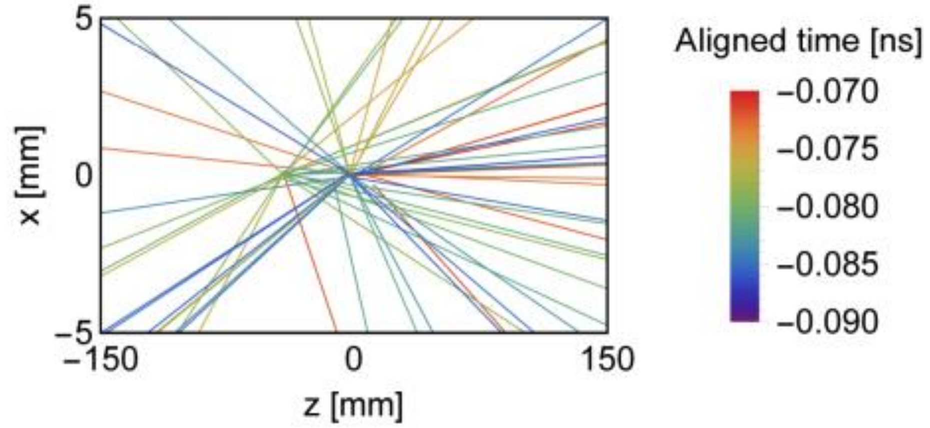
Too much overlap

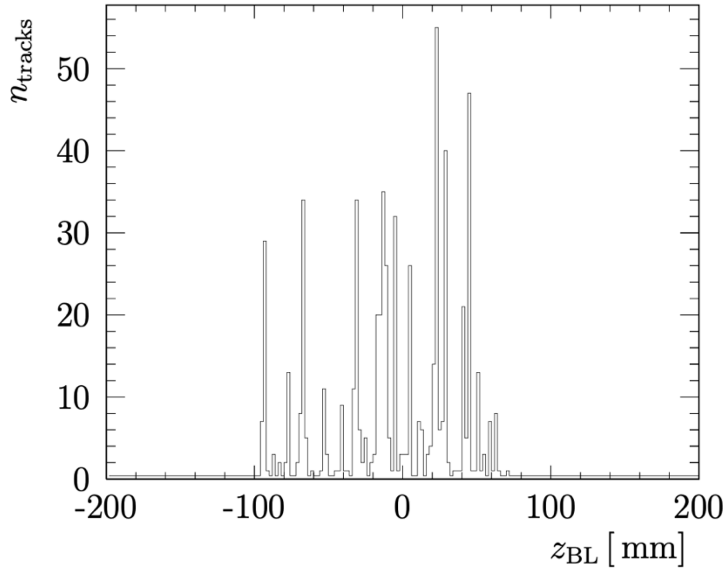
Time information needed

Within 2 ns window

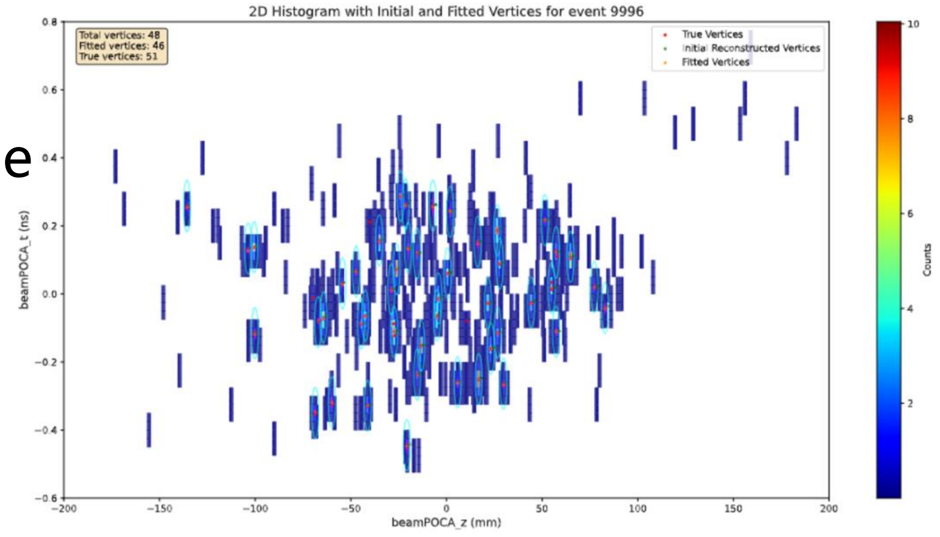


Within 2 ps window





Time



Z

Default option

- Add time dimension to current algorithm

1. Extrapolate tracks to the position of closet approach (POCA) to the known beamline
2. Fill a histogram of POCA z-axis
3. Identify peaks in the histogram as PV seeds.
4. Associate tracks to each seed
5. Perform a weighted PV fit

→

1. Extrapolate tracks to the position of closet approach (POCA) to the known beamline
2. Fill a 2D histogram of POCA z-axis and t-axis
3. Identify peaks in the histogram as PV seeds.
4. Associate tracks to each seed
5. Perform a weighted PV fit, adding time to χ^2

Preparations

- Simulation samples produced with Gauss, the LHCb simulation framework (hit time resolution set to $50\ \mathrm{ps}$)
- Transform the samples to the format that Allen (aforementioned HLT framework) needs
- Prepare cheating tracks using simulation truth information (velo tracking is under study by Justus)
- Run a simplified Kalman filter algorithm on the cheating tracks to obtain states at POCA to the beamline
- We implemented a standalone python package to study reconstruction performances under different setups
  - Use as reconstruction performance benchmark to future GPU implementations

Data flow

Pretty realistic

LHCb simulation — Generating samples containing hits and MC truth information

Allen

- Cheating tracks using MC truth
- Run a Kalman filter on cheating tracks

Ntuples with results after Kalman filter:
POCA, covariance matrix, …

As close as possible to future GPU code

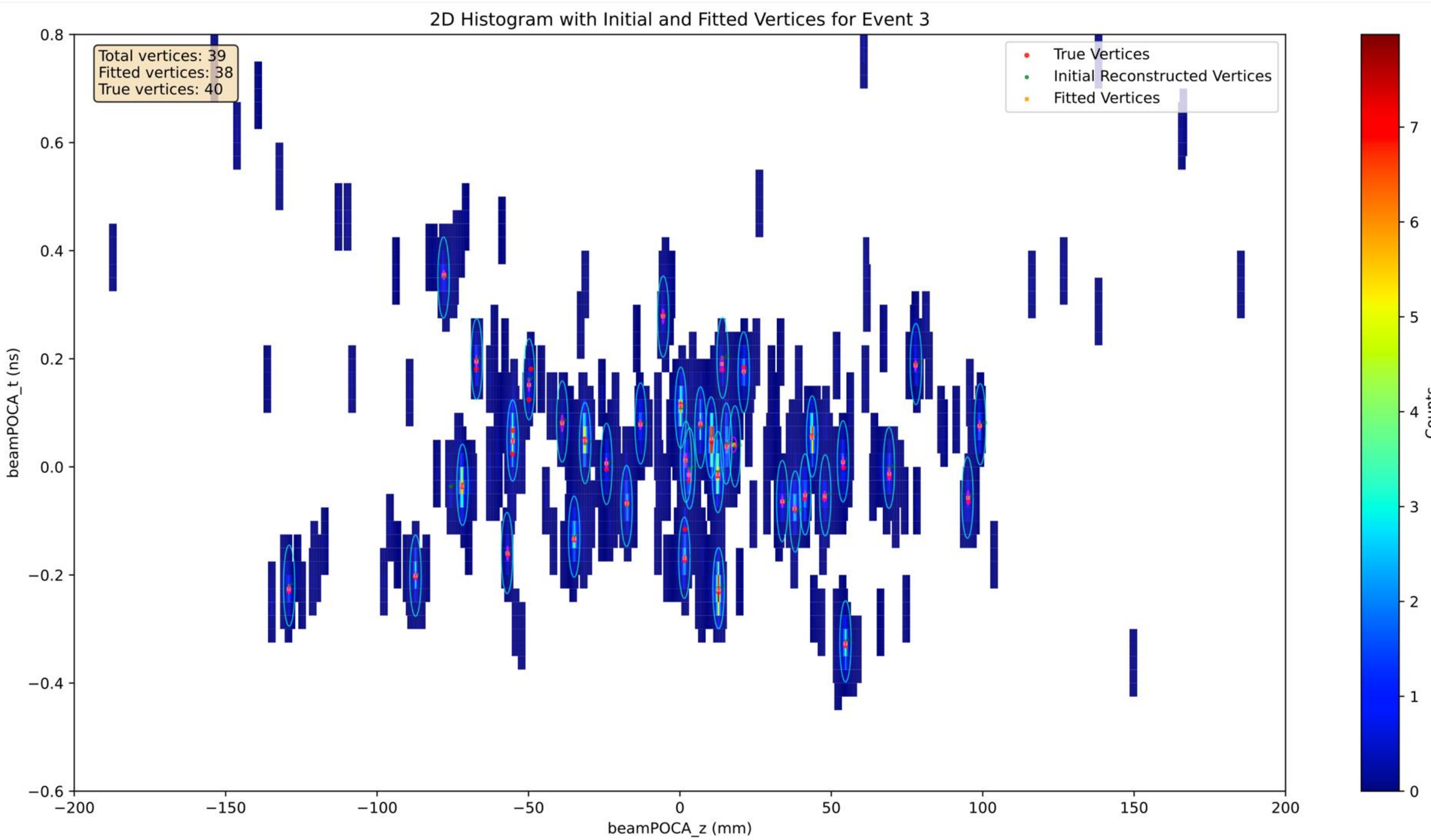
Standalone python

- Histogram filling
- Seed finding
- PV fit algorithm (in backup)
- Code to monitor the **reconstruction performance (will mentioned later)**

Code available on [gitlab]

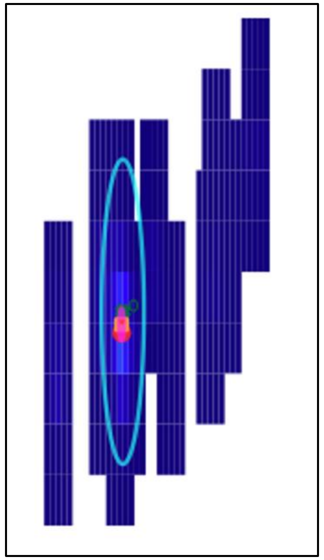
2D histogram visualization

- Apply a Gaussian smearing to the track contributions on the histogram
  - Uncertainties obtained from Kalman filter
- Consider only the contributions within a few bins around the maximum bin
- Apply a minimum number of tracks requirement
- Associate each track to the vertex seed with the smallest impact parameter (or IP significance)
- A minimum χ^2 PV fit on associated tracks



Performance study

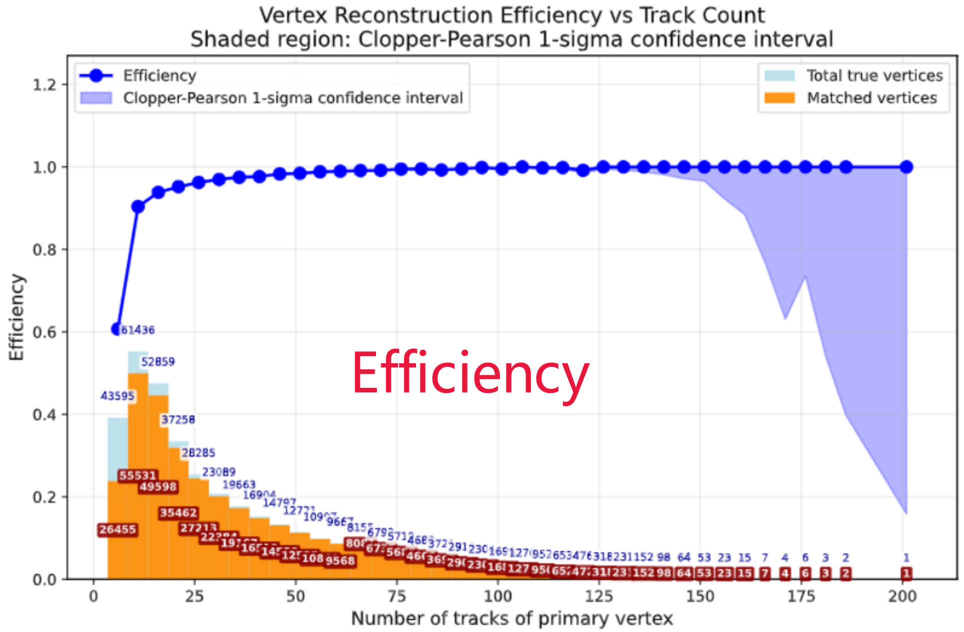
- Definition of Rec-True PV matching, metrics are very sensitive to this definition
  - Use distance in Z-T plane, if true PV in an ellipse centered the fitted PV, matched

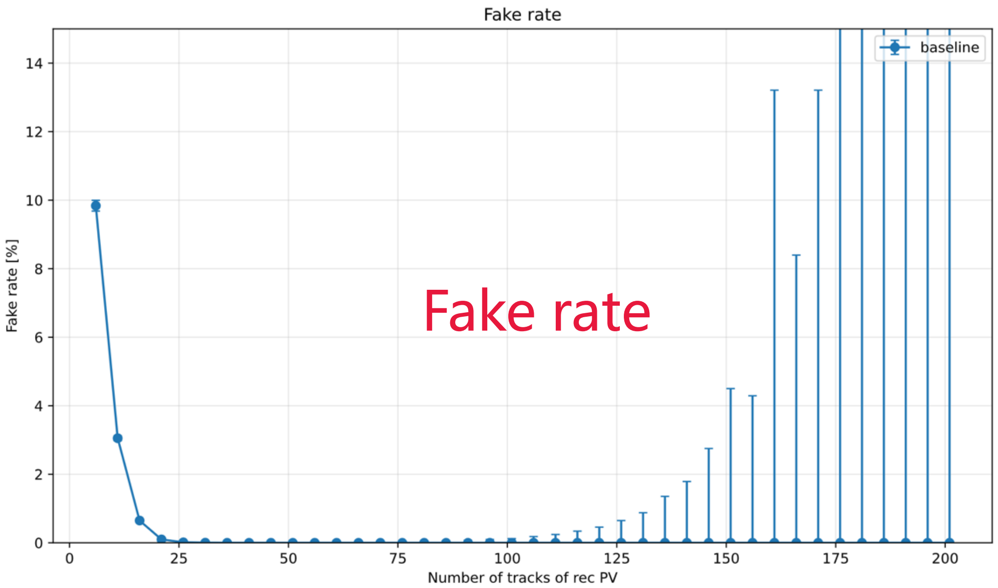


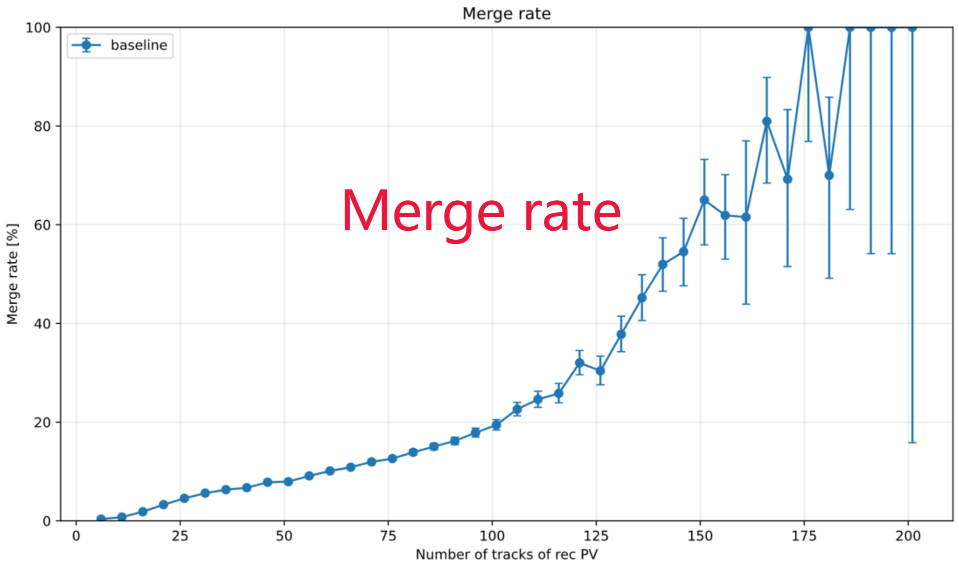
- Different metrics
  - Efficiency = $\frac{\#PV^{REC}_{matched}}{\#PV^{true}_{reconstructible}}$
  - Fake rate = $\frac{\#PV^{Rec}_{not\,match\,to\,any\,true}}{\#PV^{REC}}$
  - Merge rate = $\frac{\#PV^{REC}\,(more\ than\ 2\ true\ in\ the\ ellipse,\ but\ at\ least\ one\ not\ matched)}{\#PV^{true}_{reconstructible}}$

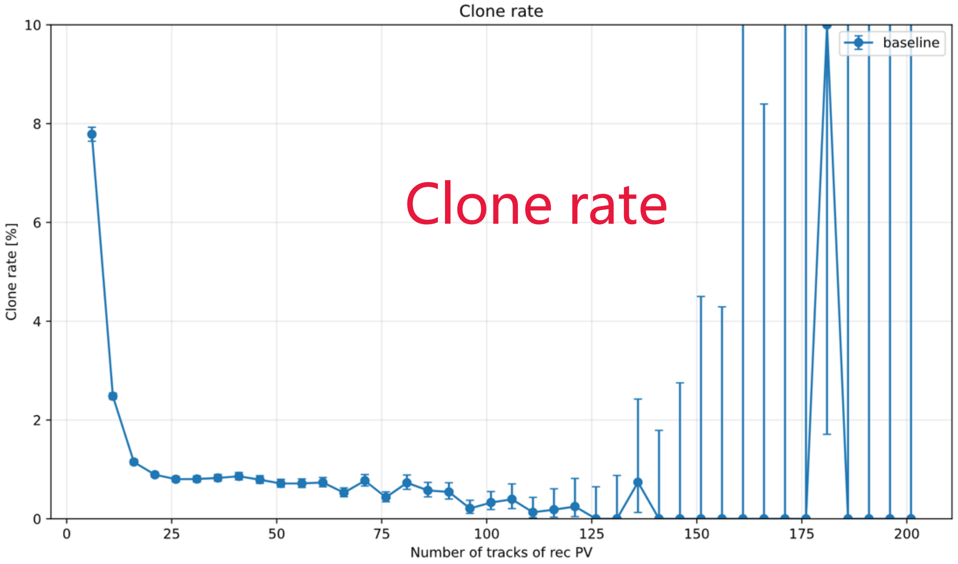
    Many Rec -> one True (closet one)
  - Clone rate = $\frac{\#PV^{REC}\,(n\ matched\ to\ one\ true,\ count\ n-1)}{\#PV^{REC}}$
  - Resolution in X, and Z

Metrics as function of number of tracks in vertex





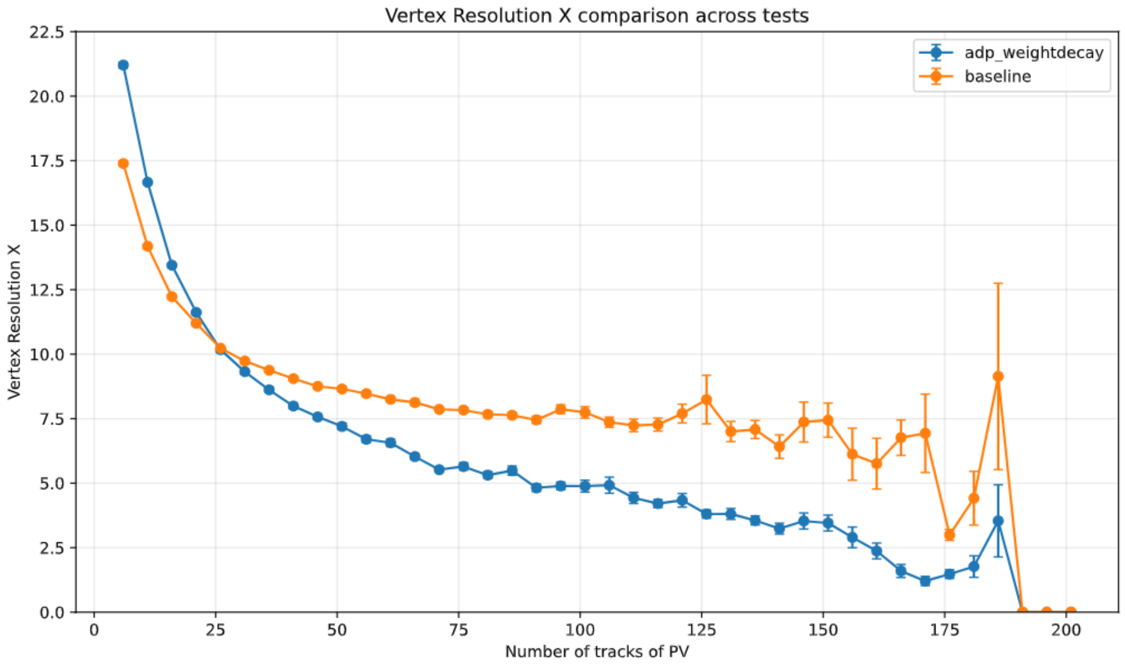




Resolution as function of number of tracks in vertex

X vertex resolution

μm



Z vertex resolution

μm

Orange: weight calculated once w.r.t seeds
Blue: weight update during iteration + weight decay

$$\chi^2 = \sum_{\text{tracks } i} w_i \, \chi_i^2$$

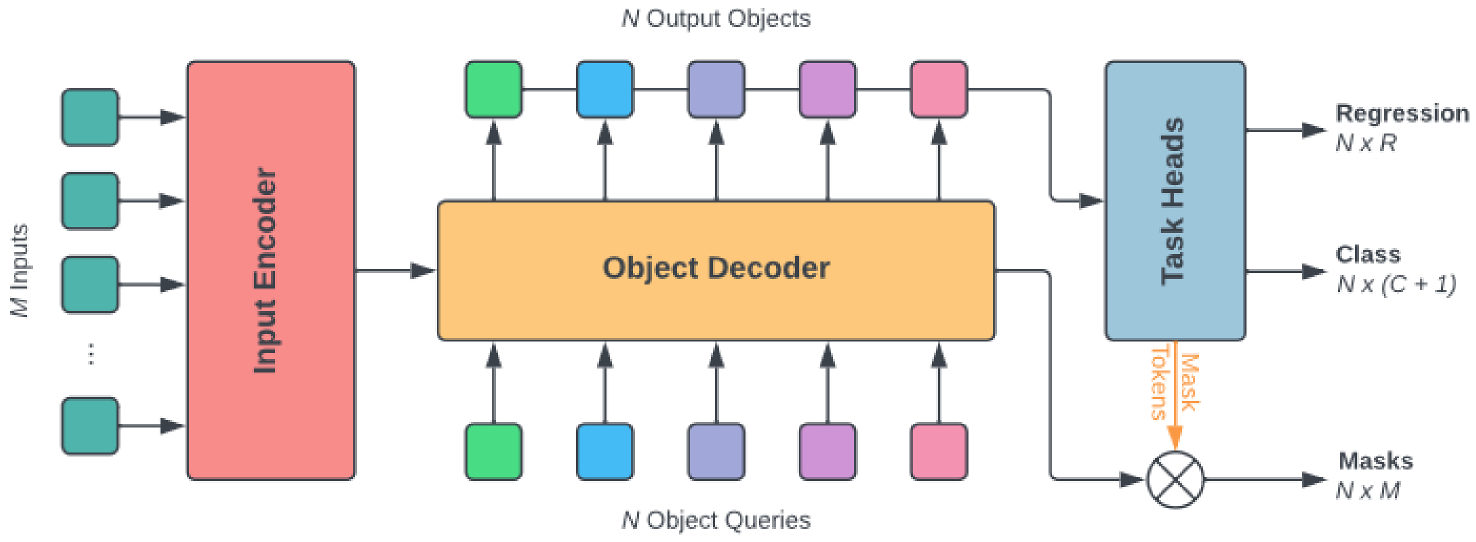
Plan

- Use a different matching definition, rather than the distance
  - Base on purity ($\frac{\#\ tracks\ from\ a\ true\ PV}{\#\ tracks\ in\ REC\ PV}$) and efficiency/recall ($\frac{\#\ tracks\ enters\ a\ REC\ PV}{\#\ tracks\ in\ a\ true\ PV}$)
  - Might be better, not rely on the ellipse size

- Implement the GPU algorithm with time information

MaskFormer

- We are also trying ML methods, hepatten package based on MaskFormer used by NIKHEF ATLAS
- MaskFormer is the the current state of the art for image segmentation
  - Suitable for many to many problem
  - Output position regression, vertex classification, and track-PV association



MaskFormer (cont.)

- Status:
  - Spend time on understanding how it works (code + paper)
  - Successfully run on Stoomboot cluster
    - Developed a module to feed our data to the framework
    - For now, no meaningful result obtained
    - More efforts are need to get preliminary result

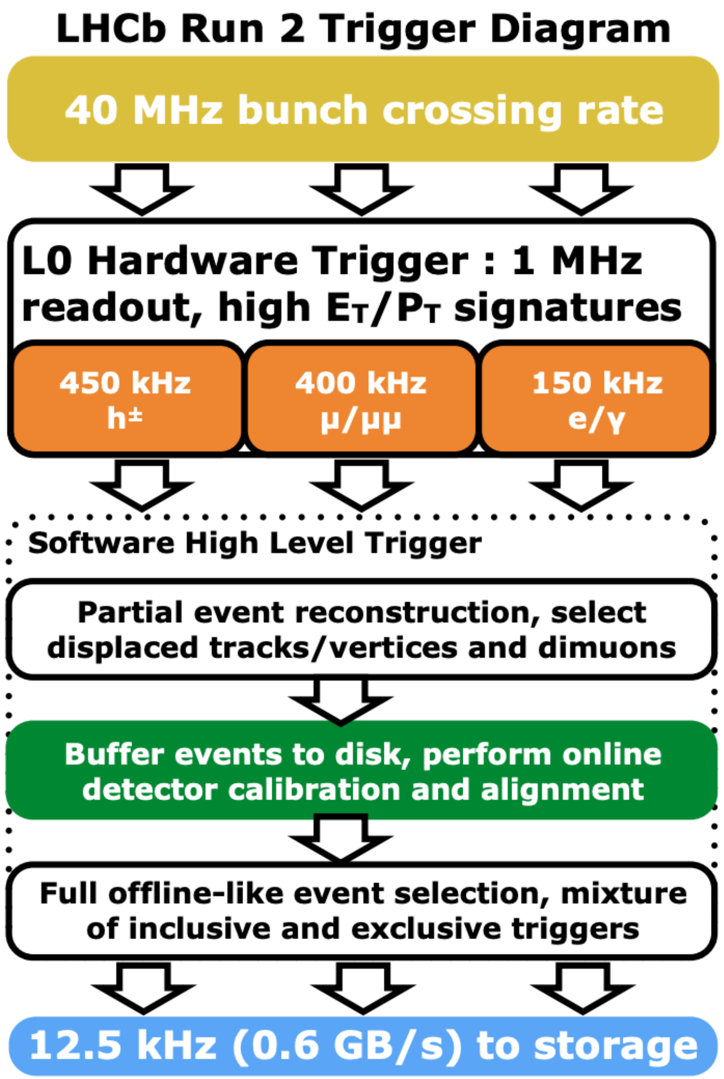
Summary

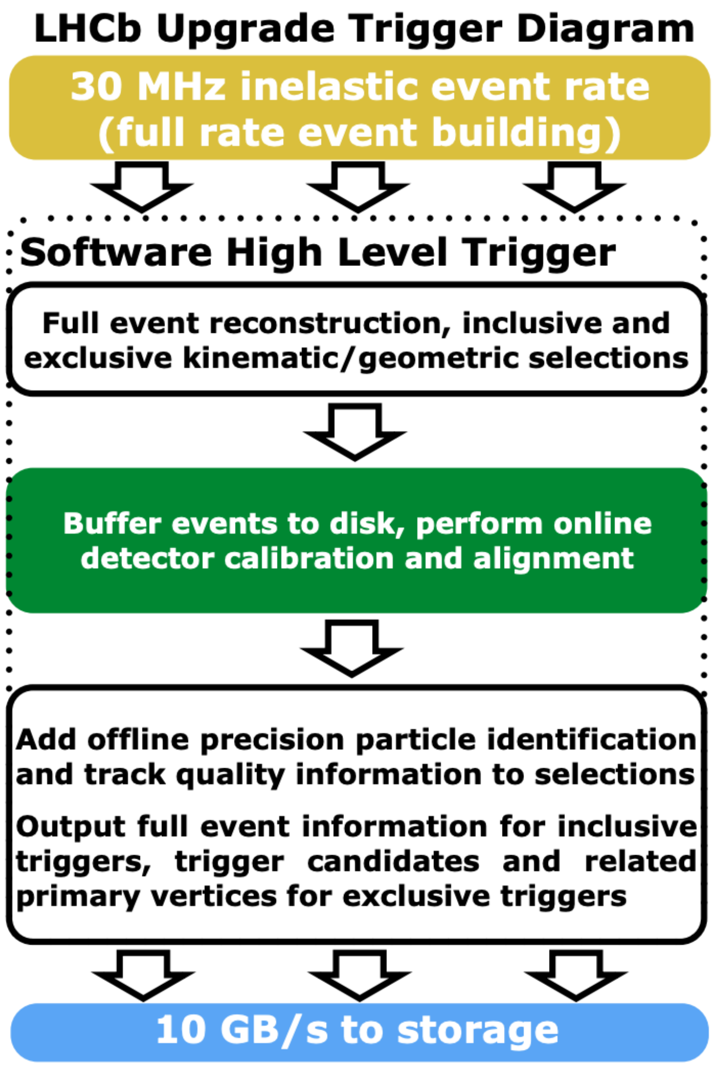
- Add time information to current LHCb PV algorithm
  - Fill 2D Z-T histogram, find peak, track-PV association, PV fit
- Obtained some performance results
- Need to implement the GPU algorithm
- We are trying to use MaskFormer

Thanks for your attention!

Backup

LHCb trigger upgrade





Details of the Vertex Fitter

- The vertex fit chi2

$$\chi^2 = \sum_{\text{tracks } i} w_i \, \chi_i^2$$

Each track

$$\chi_i^2 = r_i^T \, V_i^{-1} \, r_i$$

$$= \frac{r_x^2}{V(xx)} + \frac{r_y^2}{V(yy)} + \frac{r_t^2}{V(tt)}$$

V_i: covariance matrix at POCA state

- Residual

$$r_i = m_i - h_i(\vec{x}_{\text{vtx}}, \vec{p}_i)$$

$$r_i = \begin{pmatrix} x_{\text{trk } i} + (z_{\text{vtx}} - z_{\text{trk } i}) \cdot t_{x,\text{trk } i} - x_{\text{vtx}} \\ y_{\text{trk } i} + (z_{\text{vtx}} - z_{\text{trk } i}) \cdot t_{y,\text{trk } i} - y_{\text{vtx}} \\ \boxed{t_{\text{trk } i} + (z_{\text{vtx}} - z_{\text{trk } i}) \; t_{b,\text{ trk } i} - t_{\text{vtx}}} \end{pmatrix}$$

$$t_{b,\text{ trk } i} = \text{backward} \frac{\sqrt{1+t_x^2+t_y^2}}{c}$$

Assuming all tracks at speed of light c

- Update fitted parameters using

$$\alpha_1 = \alpha_0 - \left(\left. \frac{\partial^2 \chi^2}{\partial \alpha^2} \right|_{\alpha_0} \right)^{-1} \left. \frac{\partial \chi^2}{\partial \alpha} \right|_{\alpha_0}$$

where $\alpha = (x_{\text{vtx}},\ y_{\text{vtx}},\ z_{\text{vtx}}, \boxed{t_{\text{vtx}}})$