Acceleration of Event-Building for Data-Driven Hybrid Pixel Detector Data

Tomáš Čelko[1,2], František Mráz[1], Petr Mánek[1,2,3], Benedikt Bergmann[2]

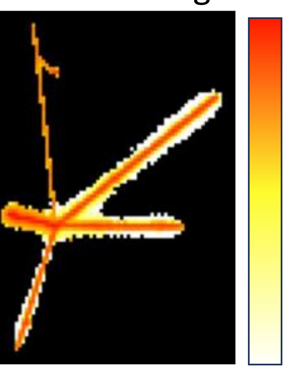
[1]Faculty of Mathematics and Physics, Charles University in Prague

[2]Institute of Experimental and Applied Physics, Czech Technical University in Prague

[3]Department of the Physics and Astronomy, University College London

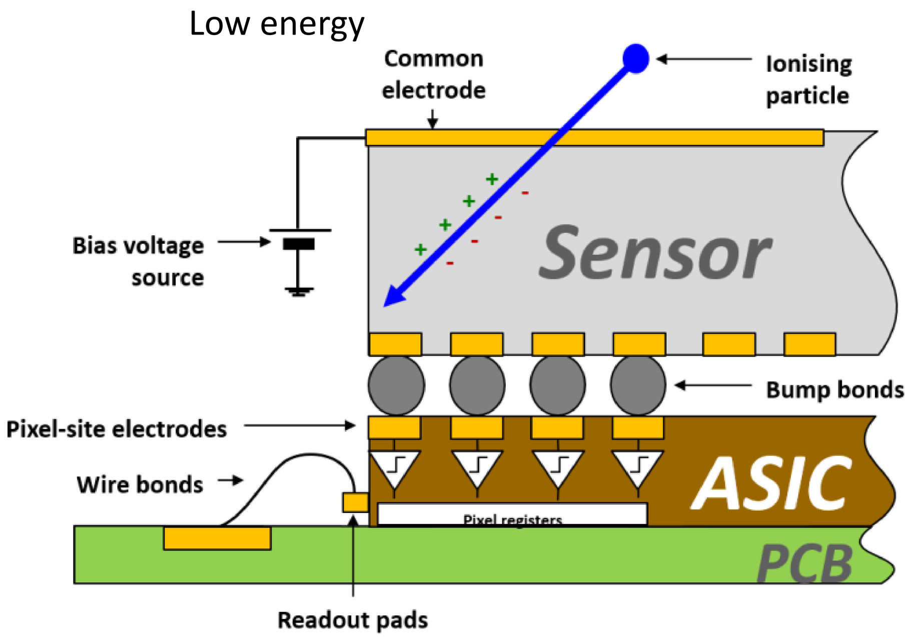
Introduction

- Hybrid pixel detectors like **Timepix3 and Timepix4** detect individual pixels hit by particles. For further analysis, individual hits from such sensors need to be grouped into spatially and temporally coinciding groups called **clusters.**



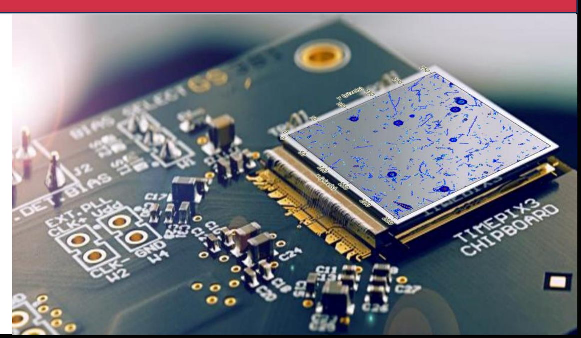
Timepix3 properties	
Pixel matrix	256×256
Pixel size	55 μm × 55 μm
Time resolution	1.56 ns
Bits per hit	48

- The Timepix3 detectors can generate up to **80 Mhit/s** (up to 640 Mhit/s with Timepix4) which is far **beyond the current capabilities** of the real-time clustering algorithms, processing at roughly **3 MHit/s.**
- Additionally, the hits from detector are **not** guaranteed to be fully **temporally ordered.**



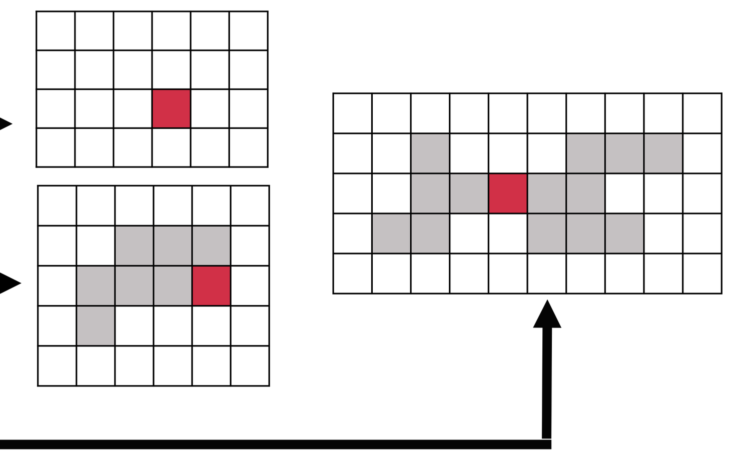
Goals

- **Accelerate** the clustering process.
- Focus on its **real-time** application.
- Selectively initiate clustering to **reduce storage** space.



Methods

```
ProcessHit(hit):
  N = findNeighboringClusters(hit)
  if (|N| == 0)
    createNewCluster(hit)
  else if(|N| == 1)
    addHitToCluster(hit, N.first)
  else
    newCluster = mergeClusters(N)
    addHitToCluster(hit, newCluster)
  outputOldClusters()
```

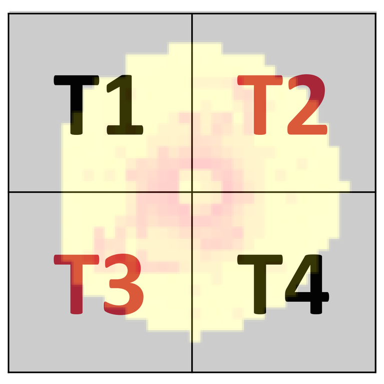


Parallel clustering performs the distributed computation of the clusters

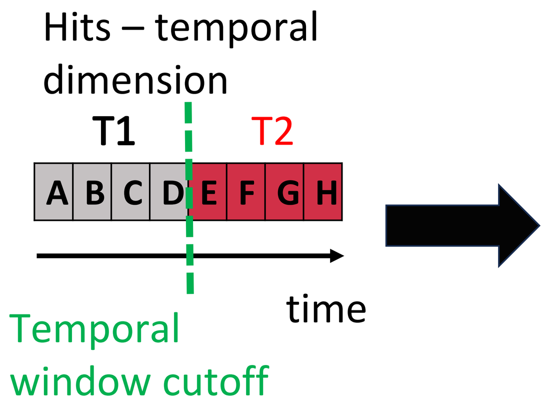
- **Step based (pipeline)** – perform individual steps of the algorithm in the parallel

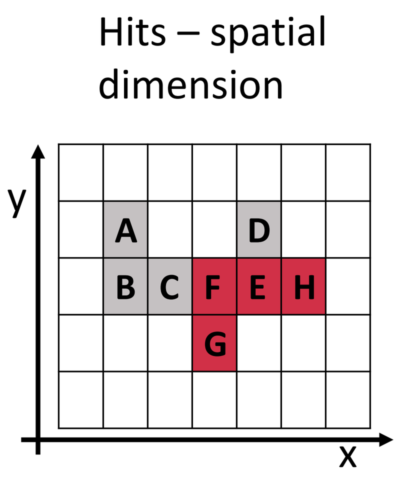
→ Reader → Calibrator → Sorter → Clusterer → Outputter

- **Data based** – split the data between workers, which can produce incomplete clusters.
  - **Spatial** – divides the area of the sensor into sectors



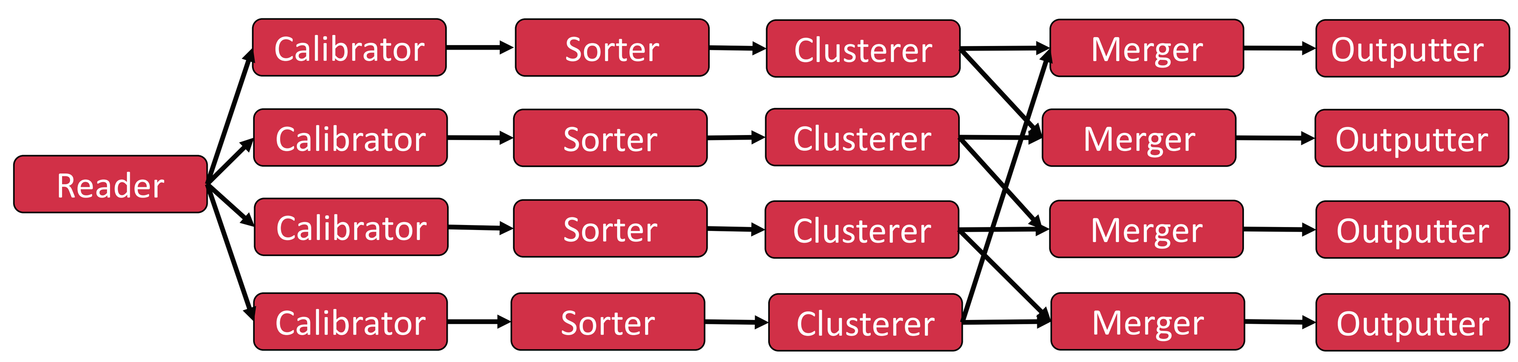
  - **Temporal** – divides the hits into time windows.





Merging incomplete clusters split by the parallelization

- Merging must be performed fast**.** A **cascade approach** is used to quickly detect complete clusters. Moreover, the **merging is parallelized**.
- With **temporal split**, the expected divided cluster rate can be **lower than 1%**, compared to nearly **2% for spatial split**.



Approximative clustering aims to exchange cluster quality for clustering speed

- **Temporal clustering** – consider only the temporal neighborhood, ignore spatial information
- **Tiled clustering** – use lower resolution of spatial matrix, effectively increasing neighborhood size and overcoming dead pixels.
- **Halo based clustering** – exploit the idea: „If two hits are spatially and temporally close, and one of them has high deposited energy, they likely belong to the same cluster"

Selective clustering monitors the hit stream and triggers clustering based on the monitored statistical features

→ Reader → Calibrator → Trigger → sorting, clustering (merging) and outputting block

- Computation of statistical features for each time window (mean and maximum for deposited energy and spatial coordinates, temporal cluster features).
- Feature differentiation with median filtering.
- There are two selective trigger approaches:
  - **Explicit** – user specifies the interesting feature ranges (DNF formula)
  - **Implicit (ML-based)** – user specifies interesting window examples and ML model is trained to trigger clustering (MLP, SVM,...)

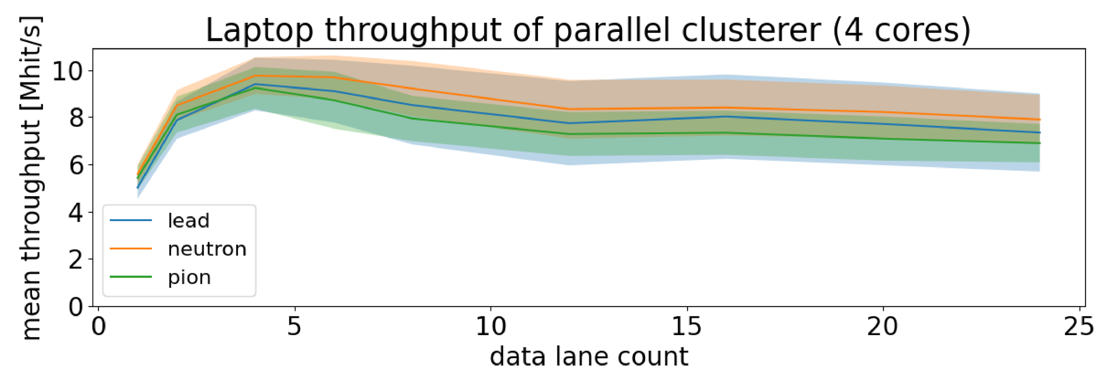
Experiments

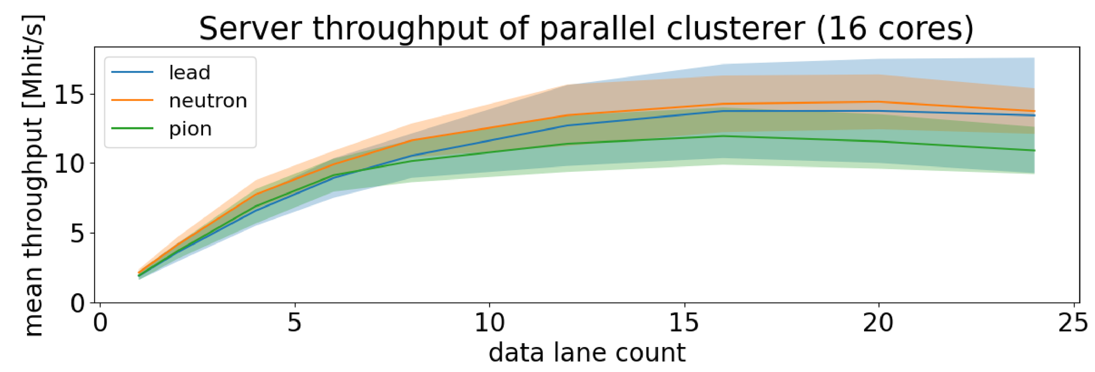
- **Validity tests**
  - Our approach was compared against the state-of-the-art fixed-window clustering method, using *IoU* metric

$$IoU(A,B) = \frac{A \cap B}{A \cup B}$$

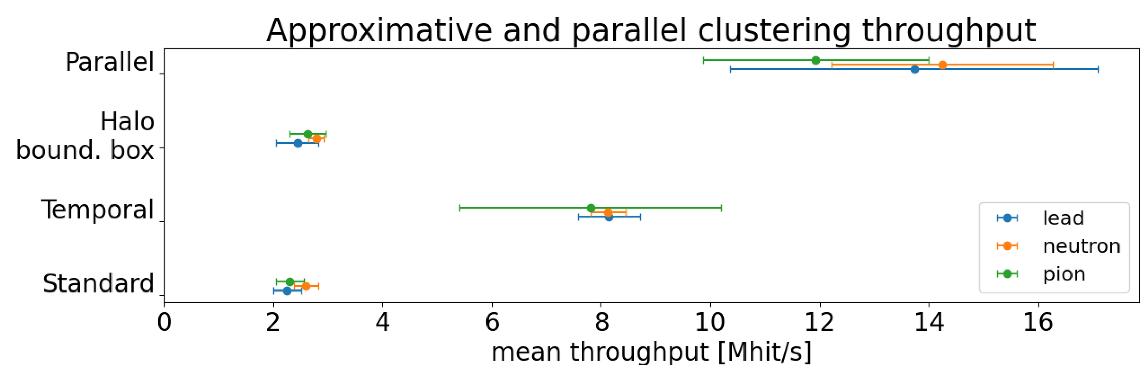
Method	Mean IoU	Std IoU
Step based parallel	0.99986	0.0001
Step and data based parallel, simple merge	0.99983	0.0002
Step and data based parallel, parallel merge	0.99983	0.0002
Halo clustering	0.856	0.025

- **Performance tests**
  - We measured the maximal speed (throughput) of the clustering methods with respect to their parameters.





- **Selective clustering test**
  - Sample task 1: Initiate clustering on nontrivial hit frequency change.
  - Solution: MLP and SVM models were trained using 200 ms time windows, the mean throughput reached **16 Mhit/s** for the laptop and **13 Mhit/s** for the server architecture. The data reduction rate approached **60%**.



  - Sample task 2: Initiate clustering on outlier window feature values.
  - Solution: Any unsupervised, outlier detection algorithm can be used. For this purpose, we chose a single class SVM, which reduced the data by nearly **90%**. This resulted in throughput increase to **25 Mhit/s** and **15 Mhit/s.**

Conclusion

- **Parallel clustering** – Despite the interdependence of different data subsets, we achieve a speed-up scaling with the number of used cores (up to 7× speedup).
- **Selective clustering** – Further, we exploited options to reduce the computational demands of the clustering by determining radiation field parameters from raw (unclustered) data features and self-initiating further clustering if these data show signs of interesting events.
- **Validation** – The proposed methods were validated and benchmarked using real-world and simulated datasets.

10th – 13th October 2023

Connecting The Dots 2023, Toulouse

Email: celko.tom@gmail.com

This research was funded by the Czech Science Foundation grant number GM23-04869M

FACULTY OF MATHEMATICS AND PHYSICS

Charles University