FIAS in TA5 – WP4

Short overview

A. Redelbach

Outline

- Summary of planned contributions
- Short overview of some relevant projects
- First plans for WP4

→ More details in the focus of coming meetings

Overview: Some other FIAS projects

Experiments studying heavy ion collisions:
ALICE (CERN), STAR (BNL), CBM (FAIR)

ALICE

Event-level processing (EP)
Administration of EP nodes
GPU acceleration of reconstruction code
High-level trigger system
Ensure resource-efficiency

STAR

Cellular automaton tracker
Kalman-Filter particle finder for decay signatures
Using the same algorithms for online and offline processing
Starting calibration procedure as soon as data become available

CBM

First-level event selector:
Timeslice interval building from all subdectors
Event building and selection of „interesting“ events
Triggers in software instead of hardware

→ **Focus on real-time processing and high-performance online reconstruction**

See also related PUNCHLunches:
Nov 11 2021, Feb 24 2022

Contributions to work programme

WP2:
D-TA5-WP2-1 (31 May 2022): Curation & metadata schemes for dynamic filtering.

D-TA5-WP2-5 (01 Mar 2026): Algorithms for massively parallel real-time sorting, clustering and pattern recognition on specialised hardware.

WP4:
D-TA5-WP4-3 (30 Jun 2025): Caching strategies for processing a set of benchmark files with the evaluated efficiencies and latencies.

D-TA5-WP4-4 (30 Jun 2026): Definition and initial implementation of an efficient real-time data processing framework

Focus: Dynamic caching

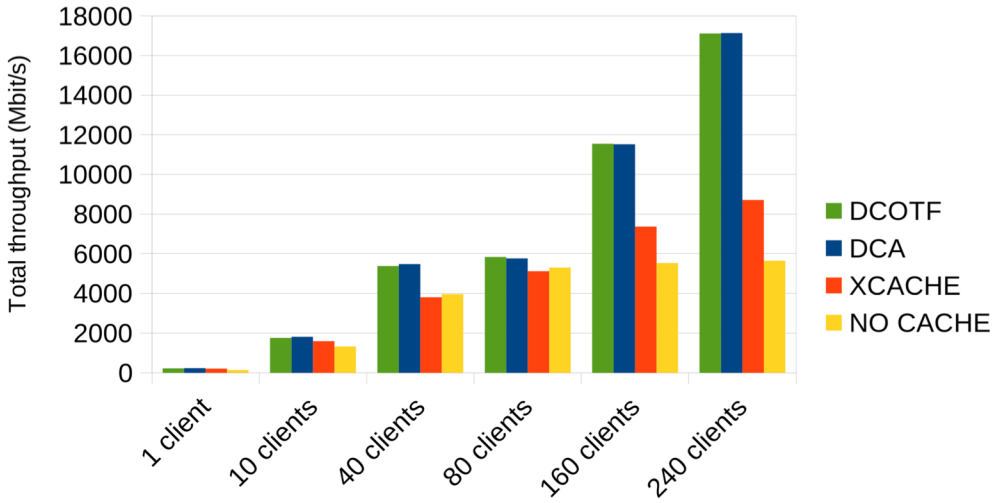
- Contributions within ErUM_Data project – cooperation with GSI
- XRootD
  - high performance, scalable fault tolerant access to data repositories
  - modular software package utilizing plug-ins
  - built-in disk based caching plug-in setting a XRootD caching proxy server
  - reduce network traffic through WAN while concurring requests can be redirected internally to the cache
- XRootD plugins
  - **XCache**: bottleneck caused by XCache server
  - **XCache with Direct Cache Access**: client has direct access to cache via a shared file system
  - **Disk caching on the fly**: Enabling local file handling via POSIX interface
- Scalability studies
- Performance studies utilizing high-bandwidth WAN between Goethe HLR and GSI
- Integration of (singularity) container system in SLURM scheduler used at GSI

Dynamic caching: Benchmark tests

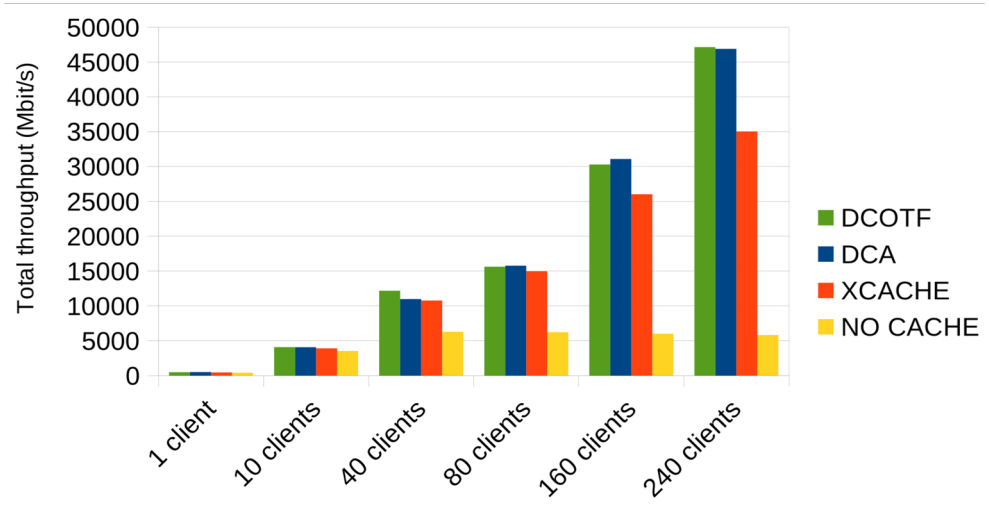
- Local cache location under BeeGFS filesystem at Goethe-HLR with infiniband fabric
- Caching node: 2x (20 core, 40 thread) CPUs, 192GB RAM, EDR link (100Gbit/s)
- “No cache” measurements with old router for 10Gbit/s link
- 10530 MB / 25990 events AOD data (CPU intensive analysis tests)
- 31457 MB / 17290 event ESD data (data intensive analysis tests)
- 1, 10, 40 and 80 client on single node; 60 clients on 2 nodes, 240 clients on 3 nodes
- Sample analysis reads a root branch from data

→Similar performance DCOTF and XCache DCA:

→AOD-level in total 17Gbit/s; ESD-level in total 47Gbit/s

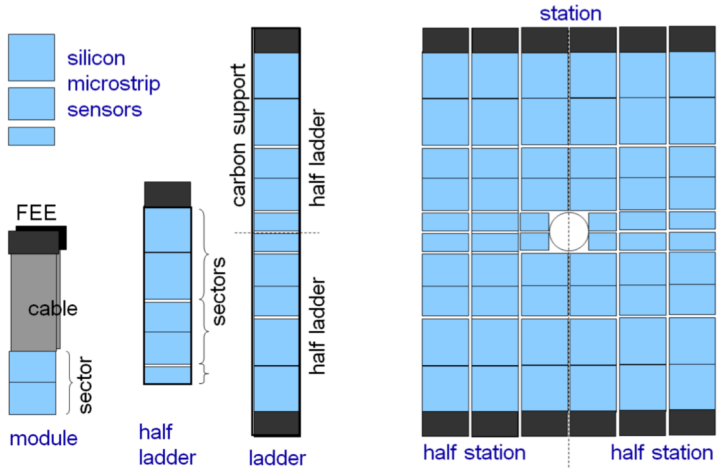


ALICE AOD data with singularity container at Goethe-HLR



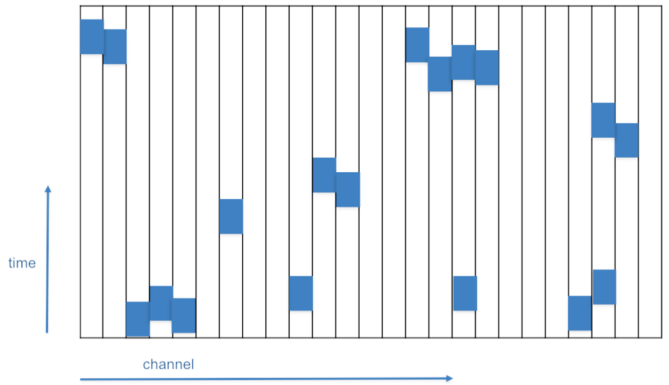
ALICE ESD data with singularity container at Goethe-HLR

Focus: Parallelization of local reconstruction



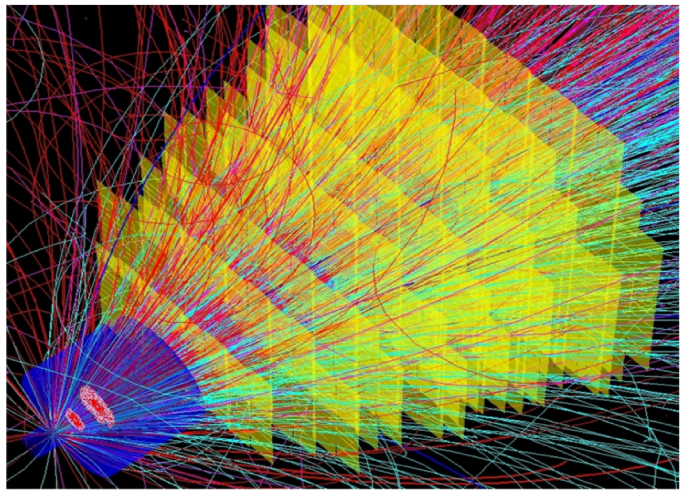
STS (Silicon Tracking Station) detector in CBM:

Reconstruction of
- Clusters
- Hits
- Tracks

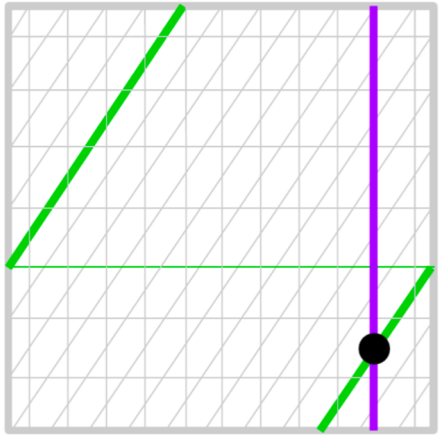


Cluster finding

Algorithm used for finding clusters for digital measurements(digis) in the STS



Tracks from a central 25 AGeV Au+Au collision overlaid with the STS tracking stations.

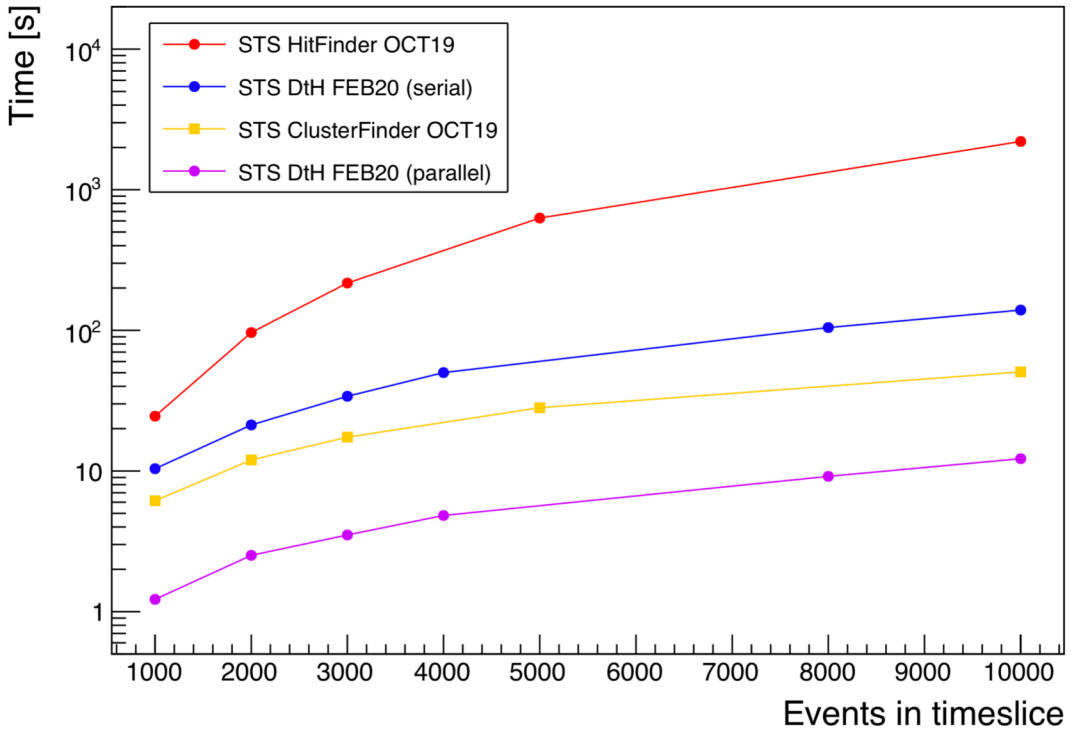


Hit finding

Intersection of frontside and backside strips gives reconstructed hit coordinate

Parallelization of local reconstruction in STS

- basic algorithm used for finding clusters for digital measurements(digis) in the STS
- STS cluster finder algorithm is linear in the number of underlying measurements
- Subsequent process of determining STS hits reflecting the original interaction points of traversing particles
- Optimization:
  - Combination of clusters at frontside and backside performed only for subset of clusters within relevant time window
  - Data-level parallelisation at the level of (independent) STS modules
  - Parallelisation on CPU using OpenMP
- Result: data-level parallelization at the level of STS modules for clusters and hits implemented



Runtimes for STS reconstruction for different number of min. bias events per timeslice of Au-Au collisions
New StsDigisToHits (DtH) task for serial and parallel execution at node with Intel Xeon Gold 6130 processor.

Parallelization using GPUs

- sorting of STS digis wrt. time based on parallel merge sort algorithm (on level of modules)
- channel-wise cluster finder approach: assigns one STS module to each GPU threadblock and one channel to each thread
- digi-wise cluster finding assigns one STS digi to a thread taking (advantage of coalesced memory access)
- calculation of the cluster properties analyzed for channel-wise or digi-wise thread mapping
- Some of the calculations relevant for determination of hits have been pre-drawn to the calculation of clusters

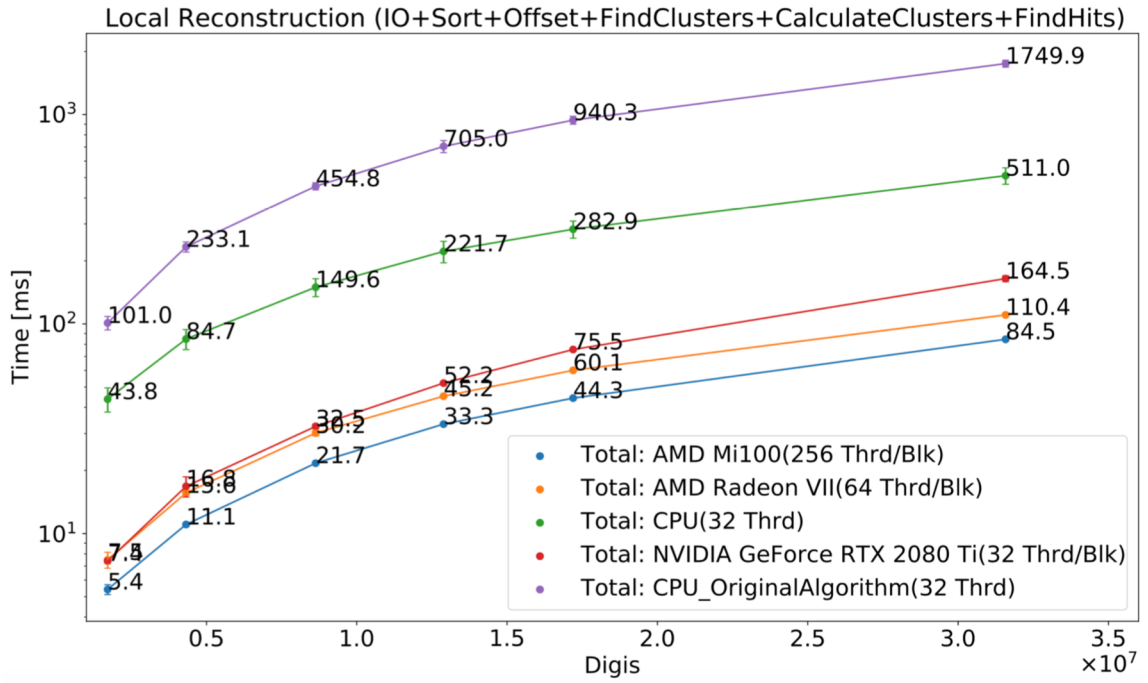
→best performance observed as a combination of channel-wise cluster finding and digital-wise cluster calculations including pre-drawn cluster widths

→Speedup factor 6

→speedup factor of 3 compared to original algorithms for local STS reconstruction

→Utilizes XPU framework

Type	ComputeUnits	Clock	Performance (SP)	Bandwidth
Intel(R) Xeon(R) Gold 6130 (x2)	16 Cores (32Threads)	2.1 GHz		119 GB/s
Nvidia RTX 2080Ti	48 CU (32 Thrd/Blk)	1635 MHz	13.8 TFLOPs	616 GB/s
Amd Vega 20 (Radeon VII)	60 CU (64 Thrd/Blk)	1400 MHz	14,2 TFLOPS	1024 GB/s
Amd Mi100	120 CU (256 Thrd/Blk)	1502 MHz	23,1 TFLOPS	1228 GB/s



Master thesis Kilian Hunold

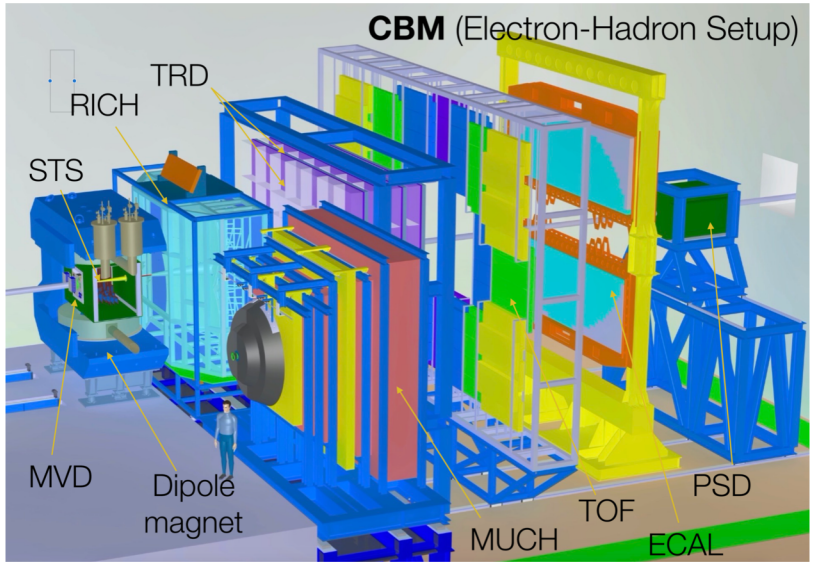
First plans for WP4

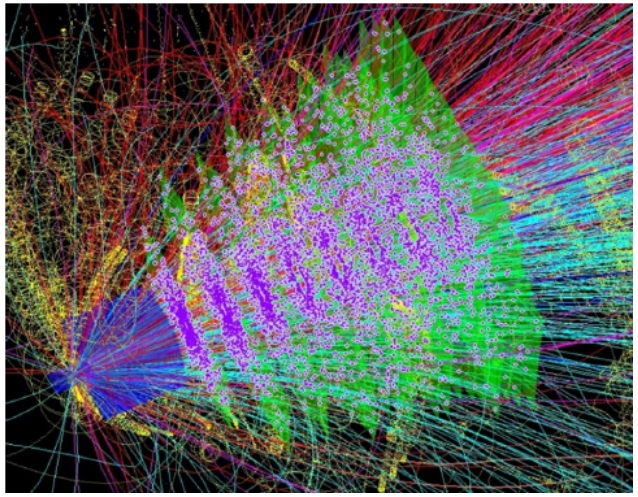
- Present and structure some of the solutions already available
  - Efficient caching strategies: XRootD, Xcache, Disk caching on the fly
  - Data processing frameworks: FairMQ, FairRoot, FLESNet, workflows related to reconstruction
  - Data aggregation (FLESNet at CBM)
  - Asynchronous message passing API to be used in a multi-process topology: FairMQ
  - Framework for building source code on different GPU devices and CPU: XPU
- Not to re-invent the wheel

- Requirements analysis
- Basis for Postdoc position (to be filled beginning of 2023)
- Structure projects for students

BACKUP

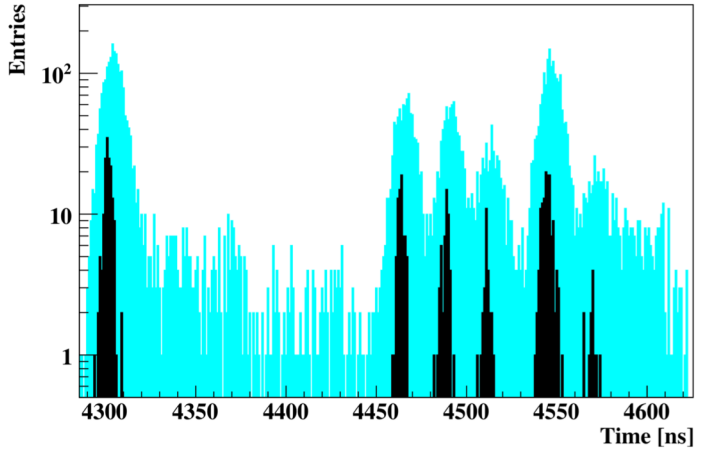
CBM detector and FLES system





- Fixed target heavy ion experiment at FAIR
- Complex trigger signatures
- Extreme reaction rates of up to 10 MHz and track densities up to 1000 tracks

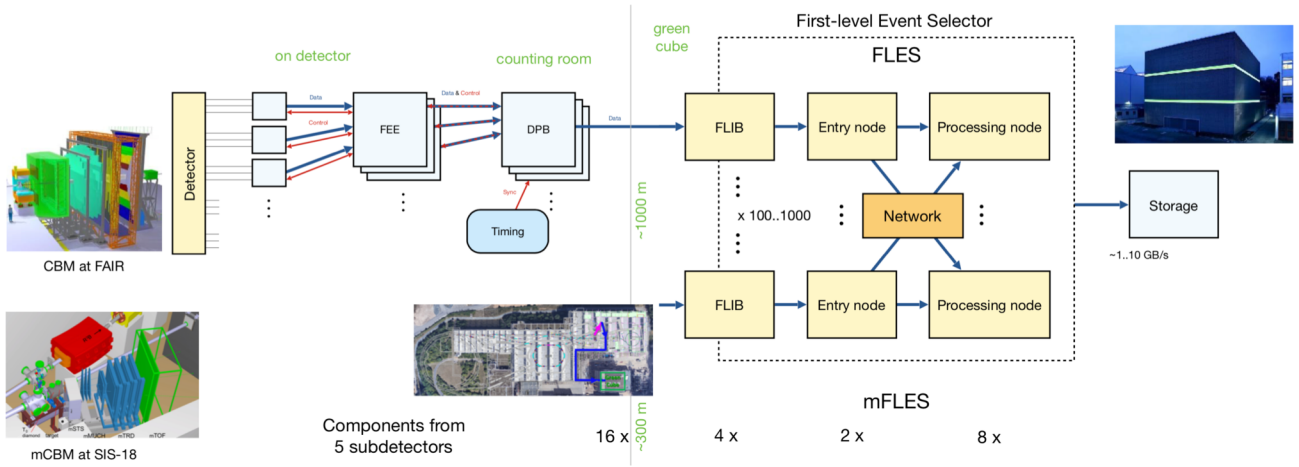
Requirements for fast online processing / reconstruction:
→ Total input data rate of 1 TByte/s
→ Self-triggering free-streaming readout electronics



STS hit time measurement in the time-slice with one hundred min. bias AuAu UrQMD events at 25 AGeV at 10 MHz (light blue), fitted track time (black color)

https://doi.org/10.1051/epjconf/201817301002

Workflows: CBM FLES and data combination



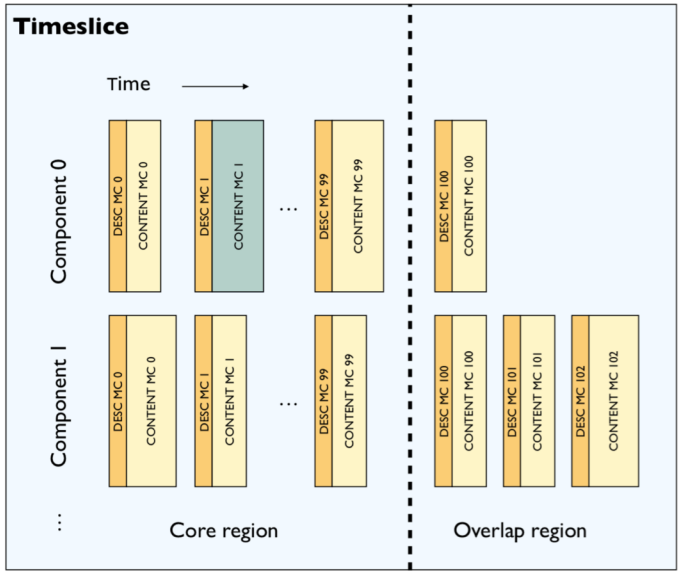
mCBM prototype experiment 2020

Timeslice building: combine matching time intervals from all input links to one "timeslice" (processing interval)

Distribute different timeslices to different processing nodes

Timeslice is self-contained

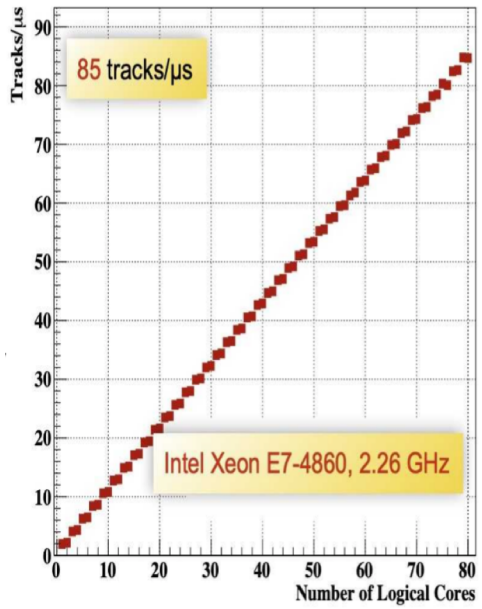
No network communication required during reconstruction and analysis

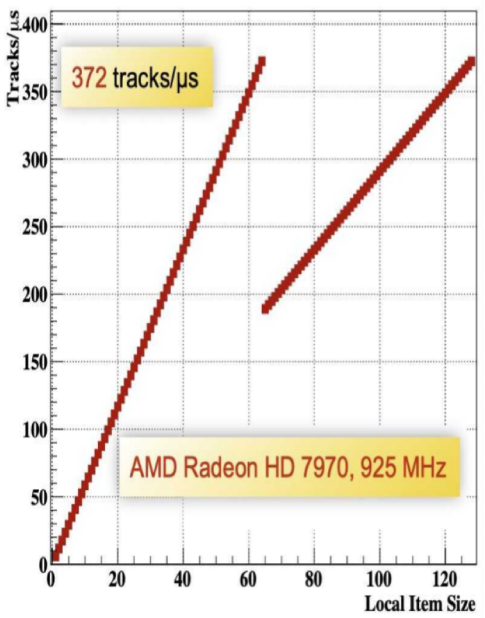


Timeslice
- Two-dimensional indexed access to microslices
- Overlap allows limited timing calibration in front-end
- Interface to online reconstruction software

Microslice
- Timeslice substructure
- Constant in experiment time
- Allow overlapping timeslices

Scaling of workflows for efficient pattern recognition





https://indico.desy.de/event/33453/contributions/118429/attachments/71950/91950/Kisel%20PUNCH%20Meeting%2024.02.2022.pdf