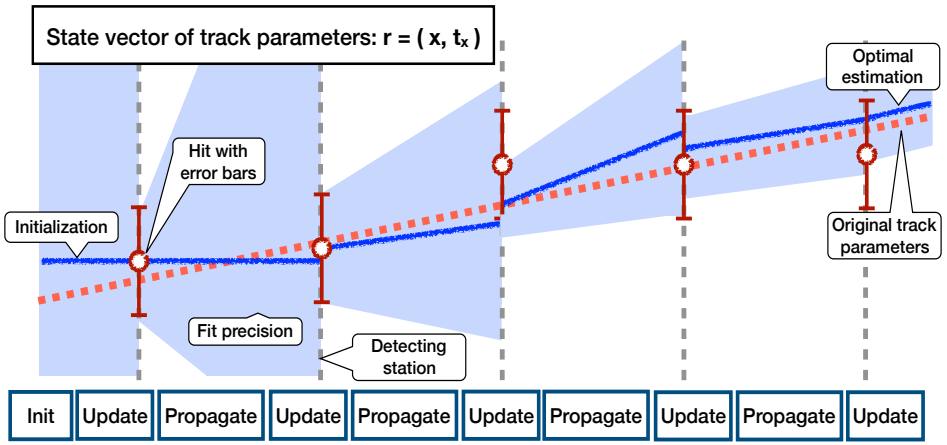
CPU/GPU Workflows

G.Kozlov, A.Redelbach

KF Track Fitting - benchmark results

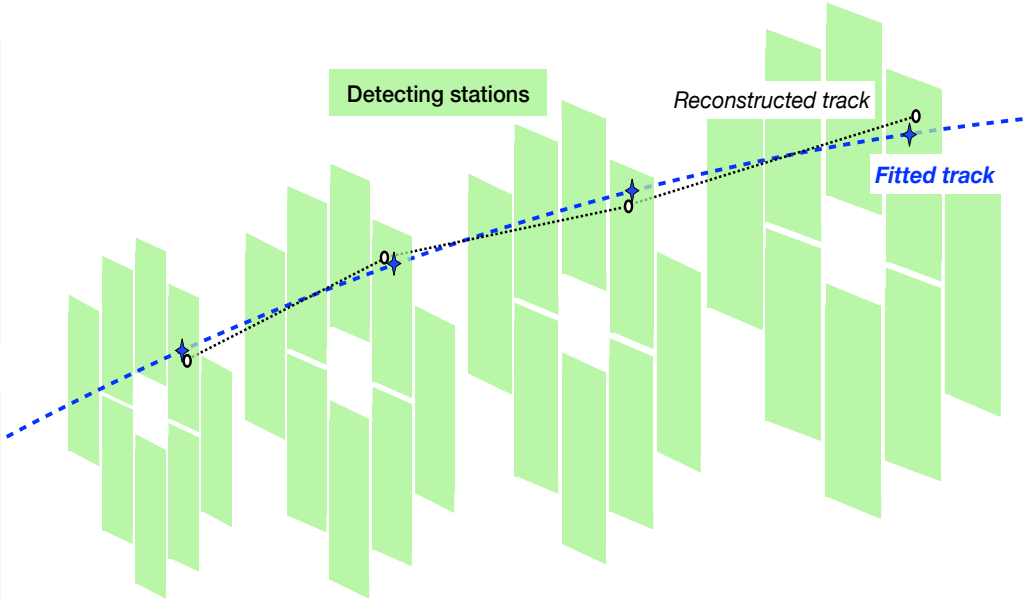
Track fitting with KF:

- determines the **track parameters** by sequentially adding hits with updating the **state vector** and **covariance matrix**;
- independently fits each individual track;
- floating point calculations;
- has great potential for parallelization at both the thread and data levels.



Parallel calculations:

- data level: scalar, SSE, AVX2, AVX-512 (headers and Vc);
- task level CPU: OpenMP (CPU affinity);
- task level GPU: OpenCL.



Input data:

- CBM STS simulated reconstructable tracks;
- 8 hits long tracks only;
- flexible dataset size to evenly fill all threads.

Hardware for benchmarks

CPU (×2):

- Intel Xeon Gold 6130 (Skylake, server);
- Total cores (HT threads): 16 (32);
- Base (max) frequency: 2.10 (3.70) GHz;
- Cache: 22 MB L3 cache;
- Instruction set extension: SSE 4.2 (128 bit), AVX, AVX2 (256 bit), AVX-512 (512 bit).

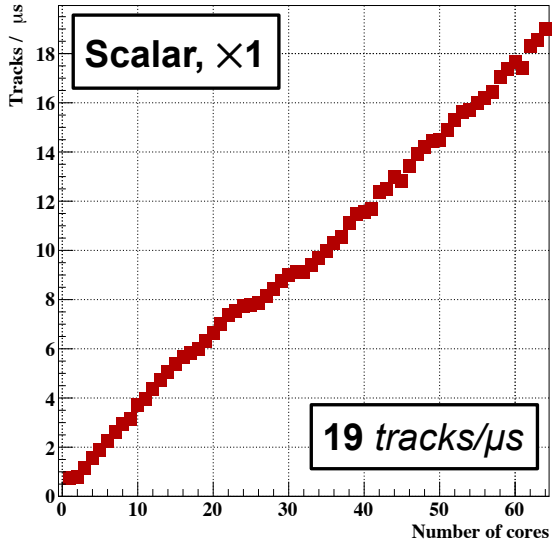
AMD GPU:

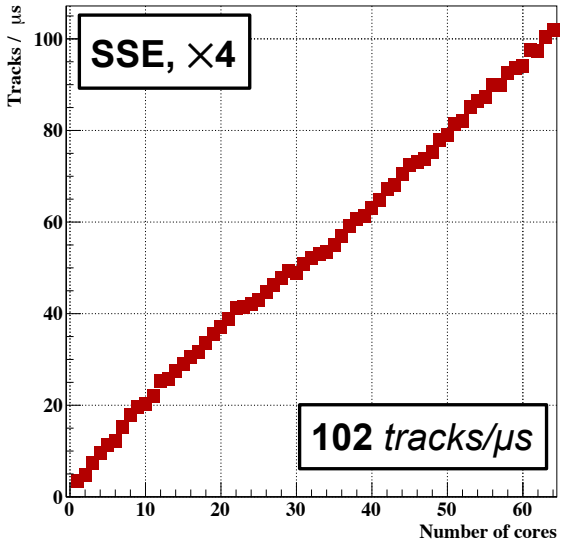
- AMD Radeon VII (Vega20);
- Compute Units (threads per CU): 60 (64);
- Clock rate: 1450 (1800) MHz;
- Memory: 16 GB @ 1000 MHz.

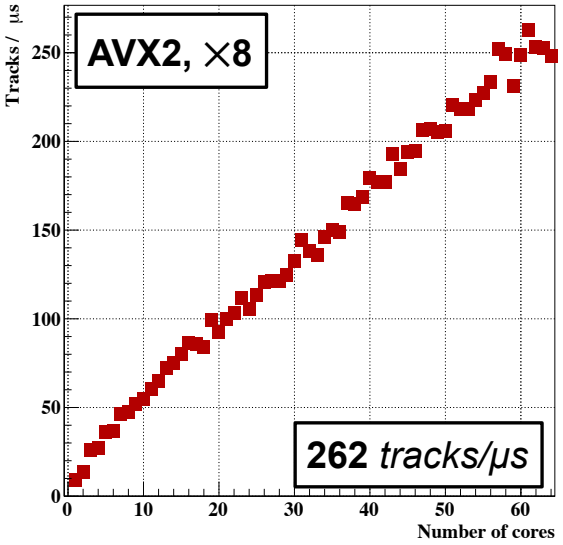
NVIDIA GPU:

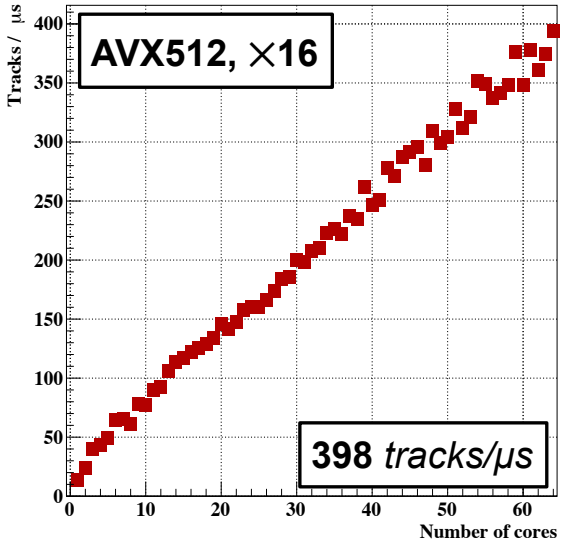
- NVIDIA GeForce RTX 2080 Ti;
- Compute Units (threads per CU): 68 (32);
- Clock rate: 1350 (1545) MHz;
- Memory: 11 GB @ 1750 MHz.

SIMD speed up and CPU scalability





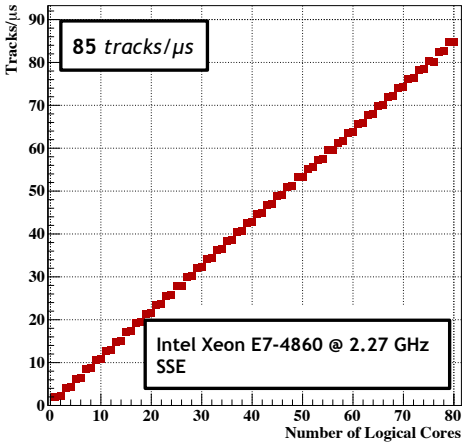




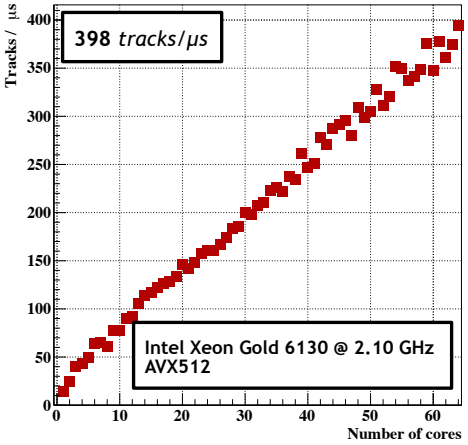
- The shape of the graphs is close to linear.
- The speed of SIMD calculations is higher than expected due to better utilization of the cache.
- The execution time of AVX instructions is less stable than SSE or scalar ones, especially when using a large number of threads.
- The maximum speed up factor relative to scalar calculations is 21.

CPU and GPU scalability and comparison with old benchmark results

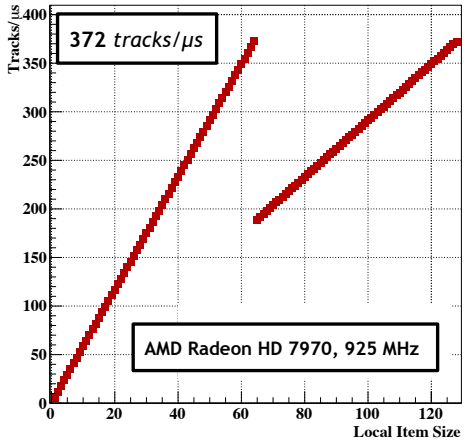
CPU

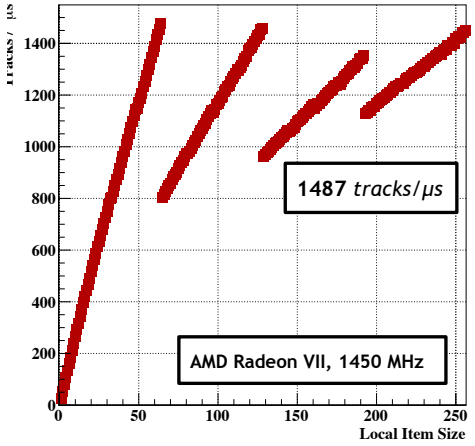


Up to 6 times faster with the same number of threads

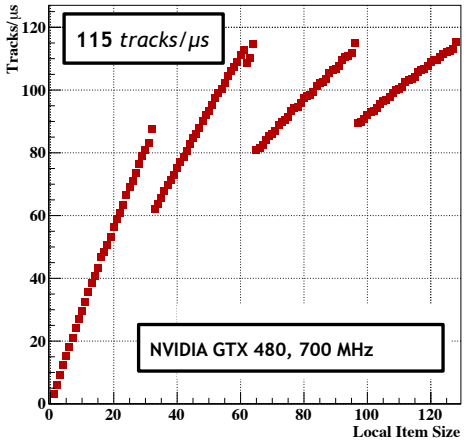


AMD GPU

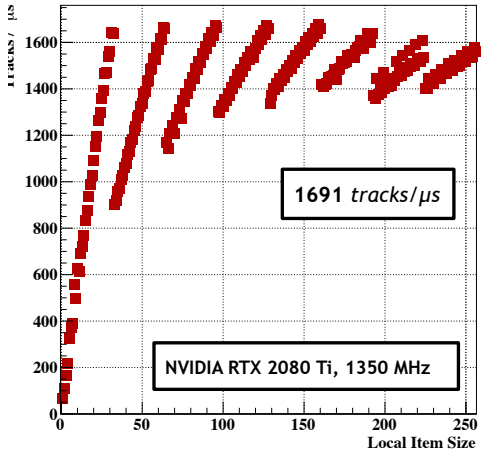




NVIDIA GPU



Track Fitter calculation speed growth is in line with GPU performance.



Next steps

CBM **L1 Track Finder** is the main algorithm for time-based particle trajectory reconstruction in the STS and MVD detectors of the CBM experiment.

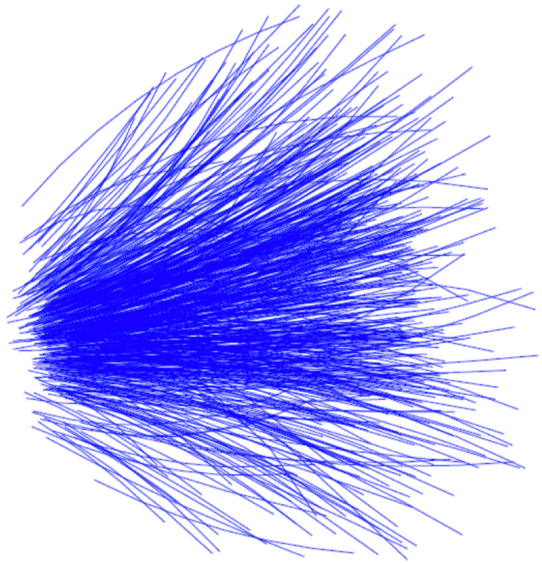
- Fast and efficient 4D reconstruction.
- Cellular Automaton based.
- Highly parallelizable and vectorizable.

Current status:

- parallelized on CPU with OpenMP;
- SIMDized with Vc - SSE intrinsics only;
- GPU parallelization is required.

Future plans:

- GPU implementation / common CPU/GPU solution:
  - extending the functionality of the XPU framework by integrating SIMD intrinsics;
  - implementation and benchmarking of XPU-based SIMD track fitting within the framework of the L1 tracker
  - GPU parallelization and preparation of a common CPU/GPU version of the tracker based on the XPU framework



Frameworks and computing resources

1. Parallelization: **XPU** is a lightweight C++ library for GPU software development (author Felix Weiglhofer).

- Abstraction layer for specific GPU API/compiler.
- Compiles code as CUDA, HIP or regular C++ with OpenMP.
- Provides optimized GPU algorithms with CPU fallback.
- Collects timing data and manages memory allocation.
- Device code is compiled once for each available backend. Device selection occurs at runtime.

https://github.com/fweig/xpu

The integration of SIMD instructions will make it possible to fully utilize the potential of CPUs. This will ensure that the code is consistent across devices and that one can use either version if needed.

The developments are directly related to both efficient CBM tracking and FIDIUM milestones.

Frameworks and computing resources

2. Portability: **Singularity** is a container platform that allows one to create and run containers that package up pieces of software in a portable and reproducible way.

A prepared singularity container must include at least:

- all improved or developed program code;
- installed and configured Cbmroot framework;
- benchmarks.

3. Computing resources:

Testing and benchmarking of the developed and upgraded reconstruction software should be done using high-performance computing clusters such as:

- **GSI VIRGO**;
- **Goethe HLR**.

Event reconstruction is associated with the processing of large amounts of data. In order to ensure a fast transfer of input data and results during track reconstruction benchmarking, the **FAIR data lake** should be used.

Conclusion and next steps

- Benchmarking of the Kalman filter based Track Fitter was carried out within the Milestone 1 of the FUDIUM project.
- Comparison with the old results showed a performance increase in line with the processing power of the devices.
- Graphics cards can achieve significantly greater acceleration than CPUs.
- The next step is to prepare a general CPU/GPU version of the CBM L1 Track Finder using the XPU library.
- Functionality of the XPU library will be extended to utilize SIMD instructions. L1 track fitter should become a test platform.
- Singularity containers will provide portability of the code and benchmarks.
- Data transfer will be organized in the FAIR data lake.
- GSI VIRGO and Goethe HLR high performance clusters will be used for benchmarking.

Backup: Source code

SIMD KF Track Fitter (on Google.Drive):

- OpenMP version for CPU:

https://drive.google.com/file/d/1AM582g4on6AuH6JVJKK1QVhYNvJ6Lyvj/view?usp=sharing

https://goo.su/On5bI5q

- OpenCL version for GPU:

https://drive.google.com/file/d/1fhft9Idc-ixYgQoIcoGlsJa5uuDbxJtm/view?usp=sharing

https://goo.su/JzOY6