EPJ Web of Conferences **95**, 01007 (2015)
DOI: 10.1051/epjconf/20159501007

Scientific and high-performance computing at FAIR

Ivan Kisel[1,2,3]

[1] *Goethe University, Grueneburgplatz 1, 60323 Frankfurt am Main, Germany*
[2] *FIAS Frankfurt Institute for Advanced Studies, Ruth-Moufang-Str. 1, 60438 Frankfurt am Main, Germany*
[3] *GSI Helmholtz Center for Heavy Ion Research, Planckstr. 1, 64291 Darmstadt, Germany*

Abstract. Future FAIR experiments have to deal with very high input rates, large track multiplicities, make full event reconstruction and selection on-line on a large dedicated computer farm equipped with heterogeneous many-core CPU/GPU compute nodes. To develop efficient and fast algorithms, which are optimized for parallel computations, is a challenge for the groups of experts dealing with the HPC computing. Here we present and discuss the status and perspectives of the data reconstruction and physics analysis software of one of the future FAIR experiments, namely, the CBM experiment.

1 Introduction

The CBM (Compressed Baryonic Matter) experiment [1] is an experiment being prepared to operate at the future Facility for Anti-Proton and Ion Research (FAIR, Darmstadt, Germany). Its main focus is the measurement of very rare probes, that requires interaction rates of up to 10 MHz. Together with the high multiplicity of charged particles produced in heavy-ion collisions, this leads to huge data rates of up to 1 TB/s. Most trigger signatures are complex (short-lived particles, e.g. open charm decays) and require information from several detector sub-systems.

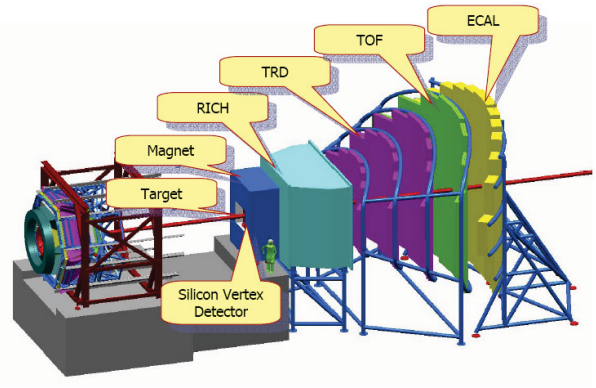

Figure 1. CBM – a future heavy-ion experiment at FAIR.

The First Level Event Selection (FLES) package [2] of the CBM experiment is intended to reconstruct the full event topology including tracks of charged particles and short-lived particles. The FLES package consists of several modules: track finder, track fitter, particle finder and physics selection. As an input the FLES package receives a simplifed geometry of the tracking detectors and the hits, which are created by the charged particles crossing the detectors. Tracks of the charged particles are reconstructed by the Cellular Automaton (CA) track finder [3] using to the registered hits. The Kalman filter (KF) based track fit [4] is used for precise estimation of the track parameters. The short-lived particles, which decay before the tracking detectors, can be reconstructed via their decay products only. The KF particle finder, which is based on the KFParticle package is used in order to find and reconstruct the parameters of short-lived particles by combining already found tracks of the long-lived charged particles. The KFparticle finder also selects particle-candidates from a large number of random combinations. In addition, a module for quality assurance is implemented, that allows to control the quality of the reconstruction at all stages. It produces an output in a simple ASCII format, that can be interpreted later as efficiencies and histograms using the ROOT framework. The FLES package is platform and operating system independent.

The FLES package in the CBM experiment will be performed on-line on a dedicated many-core CPU/GPU cluster. The FLES algorithms have to be therefore intrinsically local and parallel and thus require a fundamental redesign of the traditional approaches to event data processing in order to use the full potential of modern and future many-core CPU/GPU architectures. Massive hardware parallelization has to be adequately reflected in mathematical and computational optimization of the algorithms.

One of the efficient features supported by almost all modern processors is the SIMD (Single Instruction, Multiple Data, vector operations) instruction set. It allows to pack several data values into a vector register and to work with them simultaneously getting a factor more calculations per clock cycle. Therefore the reconstruction routines have been revised in order to use SIMD.

In addition, the reconstruction algorithms have been parallelized between cores using the Intel Threading Building Blocks package (ITBB), that provides a scalable event-level parallelism with respect to the number of hardware threads and CPU cores.

2 Many-core computer architectures: cores, threads and vectors

Modern high-performance computing (HPC) nodes are equipped with central processing units (CPU) with dozens of cores and graphics processing units (GPU) with thousands of arithmetic units (Fig. 2).

To illustrate the complexity of the HPC hardware, let us consider a single work-node of an HLT computer farm, a server equipped with CPUs only. Typically it has 2 to 4 sockets with 8 cores each. In case of Intel CPUs, each core can run in parallel 2 hardware threads (processes), that increases the calculation speed by about 30%. The arithmetic units of CPUs operate with vector registers, which contain 4 (SSE), 8 (AVX) or 16 (MIC) data elements. Vectors realize the SIMD paradigm, that means they apply an operation to a vector as a whole, giving a speed-up factor of 4/8/16 with respect to the same operation, but with a scalar. In total, a pure hardware potential speed-up factor of a host is:

$$\mathrm{f} = 4 \text{ sockets} \times 8 \text{ cores} \times 1.3 \text{ threads} \times 8 \text{ SIMD} \approx 300,$$

which is already equivalent to a moderate computer farm with scalar single-core CPUs. In order to investigate the HPC hardware and to develop efficient algorithms we use different nodes and clusters in several high-energy physics centers over the worlds (see Tab. 1) ranging from dozens to thousands of cores.

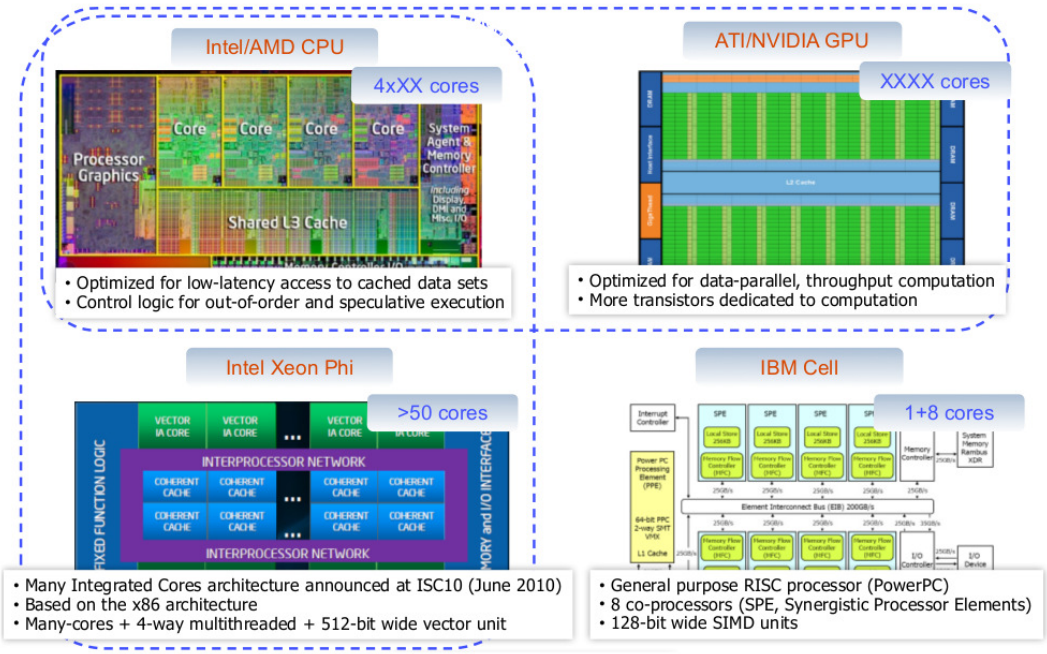

Figure 2. Future high-performance computing systems are heterogeneous many-core CPU/GPU compute nodes.

Table 1. List of some heterogeneous HPC nodes, used in our investigations.

Location		Architecture	(Nodes·)sockets·cores·threads·SIMD	Cores
CERN	Switzerland	AMD 6164HE	4·12·1·4	192
GSI	Germany	Intel E7-4860	4·10·2·4	320
JINR	Russia	Intel E5-2650+AMD HD 7970	2 8 2 8+2 32 16 4	256+4960
BNL	USA	Intel E5-2680+Intel Phi 5120D	2·8·2·8+2·60·4·16	256+7680
FIAS	Germany	Intel E5-2600+Intel Phi 7120	2 8 2 8+2 61 4 16	256+7808
ITEP	Russia	AMD 6272	(100·)2·16·1·4	12800

3 Parallel programming

The hardware provides us two levels of parallelization: a task level parallelism working with cores and threads, and a data level parallelism working with SIMD vectors. Both levels are implemented in the reconstruction algorithms. The parts of the algorithms with parallel streams of data, like fit of several tracks, are SIMDized and run on vectors providing a speedup factor up to 4/8/16. For SIMDization we have developed special header files, which overload the SIMD instructions inlining the basic arithmetic and logic functions. An illustrative example of a simple code for calculation of a polynomial function of the first order, which is written using SSE instructions, is:

```
__m128 y = _mm_add_ps(_mm_mul_ps(a,x),b);
```

The same function, but implemented using the header file, recovers the scalar-like form:

```
fvec y = a*x + b;
```

with overloading in the SIMD header file:

```
friend fvec operator+( const fvec &a,
                       const fvec &b ) {
       return _mm_add_ps(a,b); }
friend fvec operator*( const fvec &a,
                       const fvec &b ) {
       return _mm_mul_ps(a,b); }
```

As a further evolution of the header files, the Vc library implements in addition to vertical operations with full vectors also horizontal operations with elements of a single SIMD vector in order to manipulate with data within the vector. Random access to array elements is implemented with the gather and scatter functionality. To create an API for conditional execution, all functions and operators of the vector classes are able to take a mask argument optionally. The Vc library automatically determines the platform and chooses the corresponding instruction set during the compilation.

The Vc library is now a part of the CERN ROOT framework, that makes it available for physics analysis by default.

At the task level parallelism we localize independent parts of the algorithms and run them in parallel on different cores or threads with or without synchronization between the processes. Parallelization between cores is done using the Intel Threading Build- ing Blocks (ITBB) and the Open Multi-Processing (OpenMP) techniques.

The OpenCL standard provides a higher abstraction level for the parallel programing. It allows to write a universal code, which can be run on different types of CPU and GPU processing units, thus proving a portable and efficient access to heterogeneous computer platforms. The OpenCL standard supports both vectorization and parallelization between cores of CPUs and GPUs. The vectorized code in OpenCL looks similar to the previous tools:

```
float4 y = a*x + b.
```

In order to be flexible and efficient with respect to the modern many-core computer architectures we develop the algorithms in a portable form and using advantages of the languages and frameworks mentioned above. Within the KF track fit library we have reached 72.2% efficiency of hardware utilization.

Some HEP algorithms we use also in the HPC practical course to give the students a feeling of real-life problems.

4 Kalman Filter (KF) track fit library

Searching for rare interesting physics events, most of modern high energy physics experiments have to work under conditions of still growing input rates and regularly increasing track multiplicities and densities.

High precision of the track parameters and their covariance matrices is a prerequisite for finding rare signal events among hundreds of thousands of background events. Such high precision is usually obtained by using the estimation algorithms based on the Kalman filter (KF) method. In our particular case, the KF method is a linear recursive method for finding the optimum estimation of the track parameters, grouped as components into the so-called state vector, and their covariance matrix according to the detector measurements.

The Kalman filter based library for track fitting includes following tracking algorithms (Fig. 3):

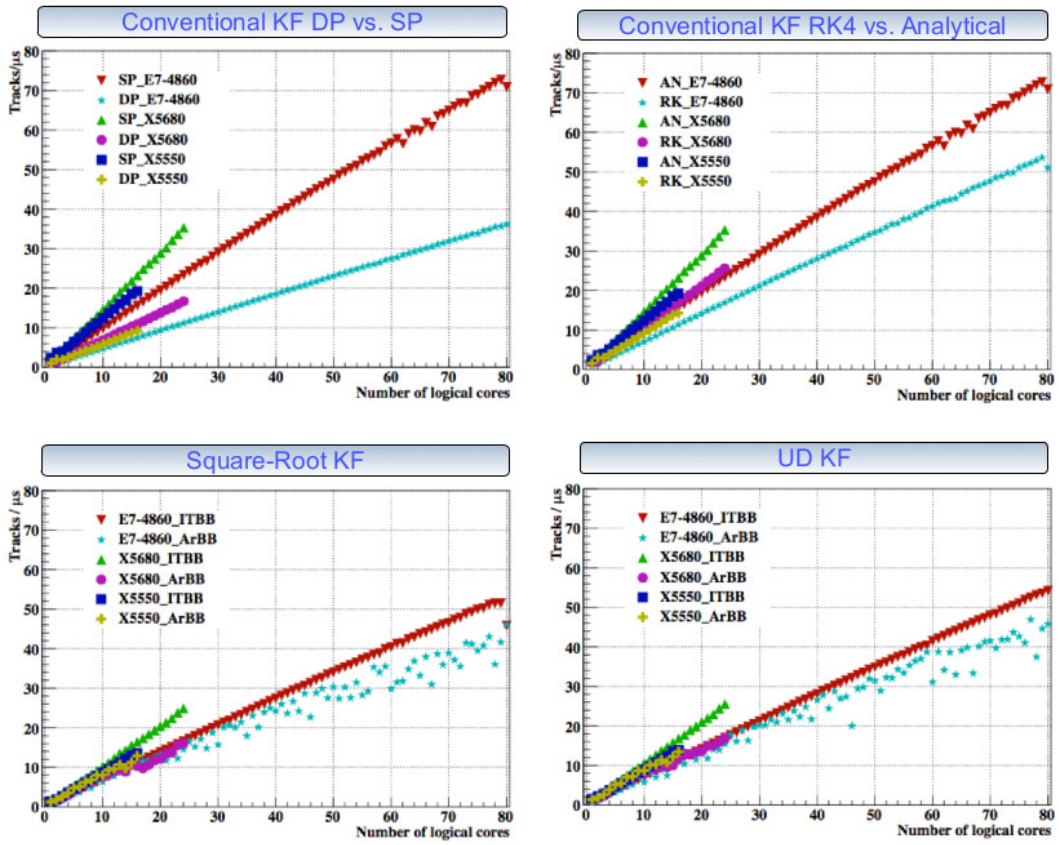

Figure 3. Scalability and performance comparison of different algorithms of he Kalman Filter track fit library.

- track fit based on the conventional Kalman filter;
- track fit based on the square root Kalman filter;
- track fit based on the UD Kalman filter;
- track smoother based on the listed above approaches and
- deterministic annealing filter based on the listed above track smoothers.

High speed of the reconstruction algorithms on modern many-core computer architectures can be accomplished by:

- optimizing with respect to the computer memory, in particular declaring all variables in single precision;
- vectorizing in order to use the SIMD instruction set and
- parallelizing between cores within a compute node.

Several formulations of the Kalman filter method, such as the square root KF and the UD KF, increase its numerical stability in single precision. All algorithms, therefore, can be used either in double or in single precision.

Table 2. Track reconstruction efficiency for minimum bias and central collisions.

	Efficiency, %	
	mbias	central
All tracks	88.5	88.3
Primary high-p tracks	97.1	96.2
Primary low-p tracks	90.4	90.7
Secondary high-p tracks	81.2	81.4
Secondary low-p tracks	51.1	50.6
Clone level	0.2	0.2
Ghost level	0.7	1.5
Reconstructed tracks/event	120	591
Time/event/core	8.2 ms	57 ms

The vectorization and parallelization of the algorithms are done by using of: header files, Vc vector classes, Intel TBB, OpenMP, Intel ArBB and OpenCL.

The KF library has been developed and tested within the simulation and reconstruction framework of the CBM experiment, where precision and speed of the reconstruction algorithms are extremely important.

5 Cellular Automaton (CA) track finder

Every track finder must handle a very specific and complicated combinatorial optimization process (see Fig. 4 with a simulated Au-Au collision), grouping together one- or two-dimensional measurements into five-dimensional tracks.

In the Cellular Automaton (CA) method first (1) short track segments, so-called cells, are created. After that the method does not work with the hits any more but instead with the created track segments. It puts neighbor relations between the segments according to the track model here and then (2) one estimates for each segment its possible position on a track, introducing in such a way position counters for all segments. After this process a set of tree connections of possible track candidates appears. Then one starts with the segments with the largest position counters (3) and follows the continuous connection tree of neighbors to collect the track segments into track candidates. In the last step (4) one sorts the track candidates according to their length and χ^2- values and then selects among them the best tracks.

Efficiency of the track reconstruction for minimum bias Au-Au UrQMD (Ultra relativistic Quantum Molecular Dynamics) simulated collisions at 25 AGeV for different sets of tracks and ratios of clones (double found) and ghost (wrong) tracks are shown in Tab. 2. The test have been performed on server with Intel Xeon E7-4860 CPUs.

The majority of signal tracks (decay products of D-mesons, charmonium, light vector mesons) are particles with momentum higher than 1 GeV/c originating from the region very close to the collision point. Their reconstruction efficiency is, therefore, similar to the efficiency of high-momentum primary tracks that is equal to 97.1%. The high-momentum secondary particles, e.g. in decays of K^0_s and Λ particles and cascade decays of Ξ and Ω, are created far from the primary vertex, therefore their reconstruction efficiency is lower – 81.2%. Significant multiple scattering of low-momentum tracks in the material of the detector system and large curvature of their trajectories lead to lower reconstruction efficiencies of 90.4% for primary tracks and of 51.1% for secondary low momentum tracks. The total efficiency for all tracks is 88.5% with a large fraction of low-momentum secondary tracks. The levels of clones (double found tracks) and of ghost (wrong) tracks are 0.2% and 0.7% respectively.

The reconstruction efficiency for central events is also given in the Table in order to show the stable behavior of the CA track finder with respect to the track multiplicity.

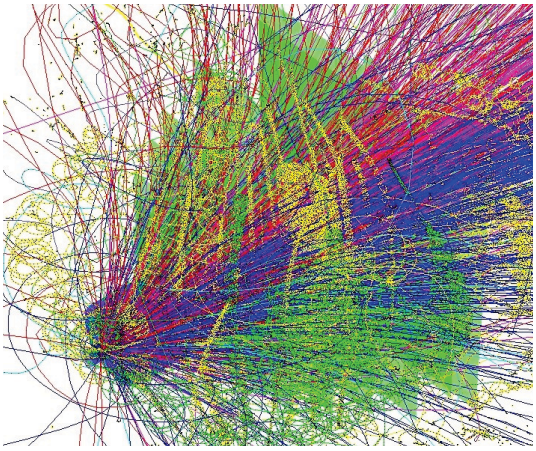

Figure 4. A simulated central Au-Au collision at 25 AGeV energy with about 1000 charged particles (different colors correspond to different types of particles).

The high track finding efficiency and the track fit quality are crucial, especially for reconstruction of the short-lived particles, which are of the particular interest for the CBM experiment. The reconstruction efficiency of short-lived particles depends quadratically on the daughter track reconstruction efficiency in case of two-particle decays. The situation becomes more sensitive for decays with three daughters and for decay chains. The level of a combinatorial background for short-lived particles depends strongly on the track fit quality. The correct estimation of the errors on the track parameters improves distinguishing between the signal and the background particle candidates, and thus to suppress the background. The ghost (wrong) tracks usually have large errors on the track parameters and therefore are easily combined with other tracks into short-lived particle candidates, thus a low level of ghost tracks is also important to keep the combinatorial background low. As a result, the high track reconstruction efficiency and the low level of the combinatorial background improve significantly the event reconstruction and selection by the FLES package.

6 Track finding at high track multiplicities

Since the CBM experiment will operate at extremely high interaction rates, different collisions may overlap in time. Thus, the need to analyze so-called timeslices, which contain information from a number of collisions, rather than isolated events arises. The need to work with time-slices instead of events is triggered not only by physical circumstances, but also is encouraged by computing hardware reasons. Not only minimum bias events, but even central events were proved to be not big enough in order to be processed in parallel on a modern many-core computer architectures. For implementing in-event level parallelism these events do not have enough sources of parallelism in order to be reconstructed on 20 or more CPU cores simultaneously.

As a first step on a way towards the time-slice reconstruction we introduce a container of packed minimum bias events with no time information taken into account. To create such a group we combine space coordinates of hits from a number (from 1 up to 100) AuAu minimum bias events at

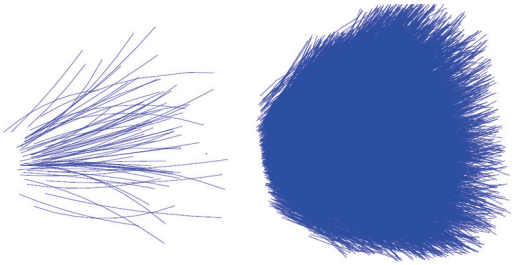

Figure 5. Reconstructed tracks in a minimum bias event (left) and in a packed group of 100 minimum bias events (right), 109 and 10 340 tracks on average, respectively.

25 AGeV ignoring such information as event number or time measurements (Fig. 5). The group was treated by the CA track finder as a regular event and the reconstruction procedure was performed with no changes. Varying the number of minimum bias events in a group we have studied the track reconstruction efficiency dependence with respect to track multiplicity. As one can see in Fig. 6, high momentum primary tracks (RefPrim), that have particular physical importance, are reconstructed with excellent efficiency of about 96%, which varies within less than 2% up to a hundred events grouped. If we include secondary tracks (RefSet) the efficiency is a bit lower – 93.7%, since some secondary tracks originate far from the target. This value varies within 3% for the extreme case of 100 minimum bias events grouped. The efficiency for low momentum tracks is 79.8% (ExtraPrim) due to multiple scattering in detector material. It changes within 6% window in case of the largest track multiplicities. The ghost fraction remains at acceptable level (less than 10%) up to the highest track multiplicities. Thus, the CA track finder is proved to be stable with respect to the high track multiplicities.

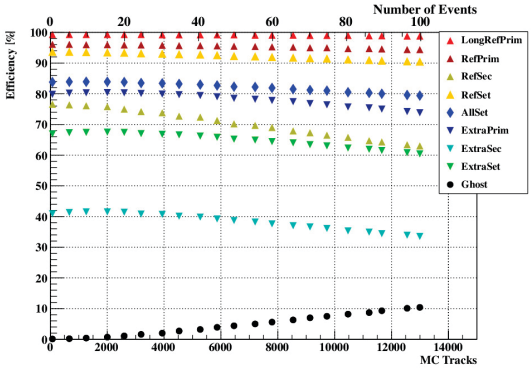

Figure 6. Track reconstruction efficiencies and ghost rate for different sets of tracks versus track multiplicity.

However, not only efficiency, but also a speed of the reconstruction algorithm is crucial for successful performance in case of CBM. We have studied the time, that the CA track finder needs to reconstruct a grouped event as a function of the number of Monte-Carlo tracks in a group (Fig. 7). The results show that the dependence is perfectly described with a second order polynomial. This is a remarkable result, if one keeps in mind the exponential growth of combinatorics with the track multi-

plicity. This dependence can be improved further and turn into a linear one, which corresponds to the case of event-based analysis, after introducing time measurements into the reconstruction algorithm.

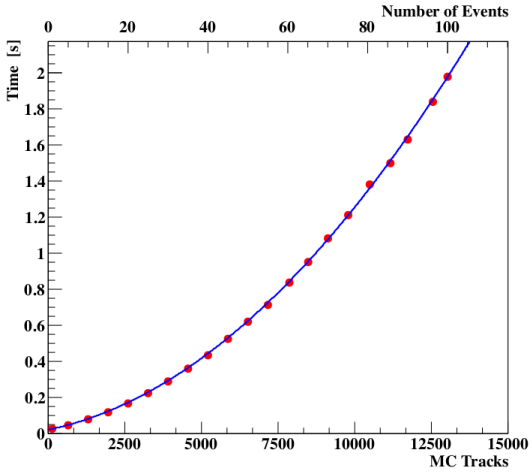

Figure 7. The CA track finder time needed to reconstruct groups of minimum bias events without time information with respect to the track multiplicity. The dependence is fitted with a second order polynomial.

7 In-event parallelism of the CA track finder

The SIMDized, but sequential in terms of core usage, version of the CA track finder was taken as the starting point for developing a parallel version with the use of the Open Multi-Processing (OpenMP) technique. By default the threads are allocated to processors by the runtime environment, which takes into account different factors, like the processor usage or the machine load. In order to prevent the CPU from sending a thread to other cores during runtime, which can affect parallelization efficiency, we use the Pthreads interface for setting a permanent thread to core affinity. The goal was to make parallel implementation of the CA algorithm keeping the same efficiencies and having the stable track reconstruction result regardless of number of executing threads.

Parallel implementation requires certain features of the algorithm. First of all, in order to get correct results, parallel iterations should not have loop dependencies, that means that the result of one parallel iteration should be independent from other parallel iterations, running at the same time. Second, one has to keep in mind that the parallel section should always be thread-safe, so that the shared data structures are used in a manner that guarantees safe simultaneous execution by multiple threads at the same time. This can be achieved by allocating local data structures for each thread and summing up results of their work afterwards or introducing synchronization into threads execution.

An important issue while making parallel implementation is to keep in mind a certain computer architecture. The optimization and testing of the parallel CA track finder was performed on a server with 4 Intel Xeon E7-4860 processors. Each processor has 10 physical cores with hyper-threading. It is an example of so-called NUMA (Non-Uniform Memory Access) architecture, that means that the memory access time for the server depends on the memory location relative to the processor. CPUs can communicate and exchange data between each other, but it takes longer time. Thus, the decision was taken in order to avoid processors communication to send one time-slice to a single CPU for

Table 3. Track reconstruction efficiency for minimum bias and central collisions.

Algorithms step	% of total time
Initialization	2.0
Triplets construction	90.4
Tracks construction	4.1
Final stage	3.4

reconstruction, not to the whole node. This way such an architecture can be filled with 4 time-slices reconstructed in parallel.

The CA algorithm consists of several logical parts (see Tab. 3). First, a short (2% of the total execution time) initialization, when we prepare hit information for tracking, takes place. The main and the most time consuming part of triplet construction takes about 90% of the sequential execution time. Out of triplets we construct tracks, that takes about 4and in addition 3.4for the next iteration. All steps of the algorithm were parallelized inside the event, using different sources of parallelism in each step: hits in the initialization and final stages, triplets for the major part, track candidates for the track construction step. In order to have enough sources of parallelism to fill a whole CPU, a group of 100 minimum bias events was processed. The resulting speed-up factors for different steps as well as for the full algorithm within one CPU (20 hyperthreaded logical cores) are presented in Fig. 8.

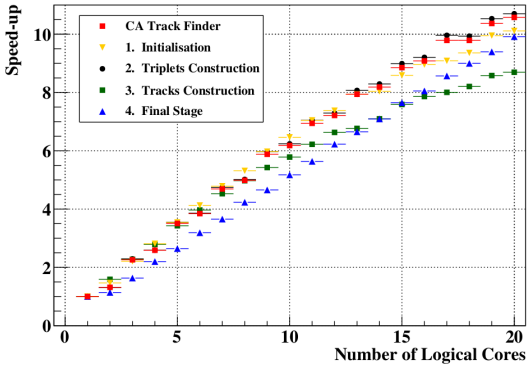

Figure 8. Speed-up factor due to parallelization for different steps and the full algorithm on Intel Xeon E7-4860 CPU with 10 physical cores and hyper-threading for the case of 100 minimum bias events grouped.

Some steps have a better speed-up for higher number of cores due to less thread synchronization needed. The algorithm shows linear scalability. Due to hyperthreading one can expect a speed up factor of about 13 on such a CPU in the ideal case. The achieved speed-up factor is 10.6 for the full CA track finder reconstruction algorithm on a CPU with 10 physical cores with hyper-threading.

8 4-Dimensional time-based event building

Since resolving different events is a non-trivial task in the CBM experiment, the standard reconstruction routine will include an event building, the process of defining exact borders of events within a time-slice and grouping tracks into even-corresponding clusters, which they originate from. For this task an efficient time-based tracking is essential. Since the CA track finder proved to be fast and stable

with respect to the track multiplicity, the next step towards the time-slice based reconstruction would be the implementation of time measurements.

In order to introduce a time measurement into the reconstruction procedure to each minimum bias event in a 100 events group an event start time was assigned during simulation phase. The start time was obtained with the Poisson distribution, assuming the interaction rate of 10^7 Hz. A time stamp we assign to a certain hit consists of this event start time plus the time shift due to the time of flight, which is different for all hits. In order to obtain a time measurement for a hit we then smear a time stamp according to a Gaussian distribution with a sigma value of the detector resolution of 5 ns.

After introducing the time measurement we can use the time information in the CA track finder. We do not allow to build triplets out of hits, which time difference is greater than 3σ of the detector time resolution. It is a very good approximation, since the time of flight between the detector planes is negligible in comparison to the detection precision. Apart from that, we perform the reconstruction procedure in a regular way. After the reconstruction we assign to each track a time measurement, which is calculated as an average of its hits measurements.

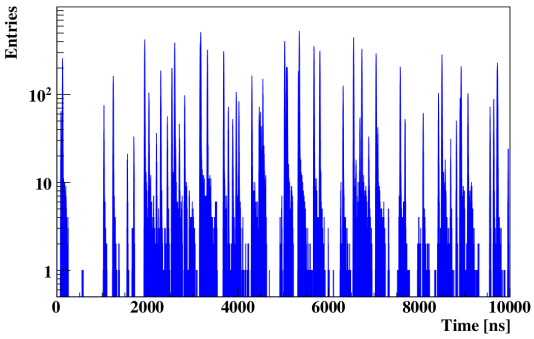

Figure 9. Part of a time-slice with 100 minimum bias events. With blue color the distribution of hit time measurements in a time-slice is shown.

The initial distribution of hits measurements representing the complexity of defining event borders in a time-slice at interaction rate of 10^7 Hz is shown in Fig. 9 with blue color. The resulting distribution of reconstructed track measurements (black color), as well as the distribution of initial hit measurements (light blue color), one can see in Fig. 10. The reconstructed tracks clearly represent groups, which correspond to events, which they originate from. Even in the area of the most severe overlap (Fig. 11) the time-based CA track finder allows to resolve tracks from different events in time.

9 KF Particle Finder – a common package for reconstruction of short-lived particles

Today the most interesting physics is hidden in the properties of short-lived particles, which are not registered, but can be reconstructed only from their decay products. A fast and efficient KF Particle Finder package, based on the Kalman filter (hence KF) method, for reconstruction and selection of shortlived particles is developed to solve this task. A search of more than 50 decay channels has been currently implemented. The package doesn't require any specific information about the geometry of an experiment, therefore it is implemented as a common package for and tested on the CBM, PANDA, ALICE and STAR experiments.

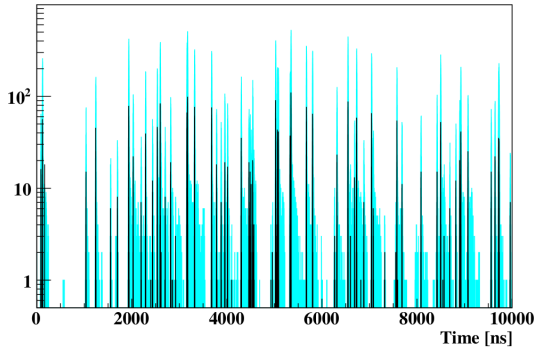

Figure 10. Part of a time-slice with 100 minimum bias events. With light blue color the initial distribution of hit measurements is reproduced, black color shows time measurements of reconstructed tracks.

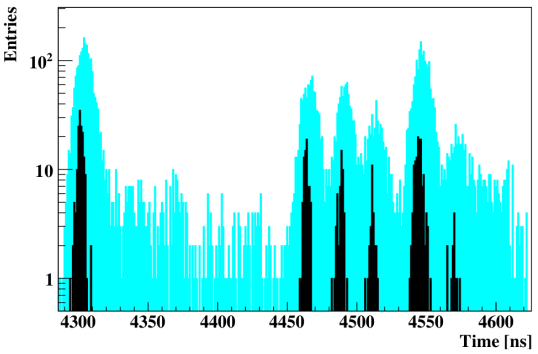

Figure 11. Event building: black reconstructed track groups are well resolved on the blue background of overlapped initial hits.

In the package all registered particle trajectories are divided into groups of secondary and primary tracks for further processing. Primary tracks are those, which are produced directly in the collision point. Tracks from decays of resonances (strange, multi-strange and charmed resonances, light vector mesons, charmonium) are also considered as primaries since they are produced directly at the point of the primary collision. Secondary tracks are produced by the short-lived particles, which decay not in the point of the primary collision and can be clearly separated. These particles include strange particles (K_s^0 and Λ), multi-strange hyperons (Ξ and Ω) and charmed particles ($D_0, D^{\pm}, D_s^{\pm}$ and Λ_c). After that tracks are combined according to the block diagram in Fig. 12. The package estimates the particle parameters, such as decay point, momentum, energy, mass, decay length and lifetime, together with their errors. The package has a rich functionality, including particle transport, calculation of a distance to a point or another particle, calculation of a deviation from a point or another particle, constraints on mass, decay length and production point. All particles produced in the collision are reconstructed at once, that makes the algorithm local with respect to the data and therefore extremely fast. KF Particle Finder shows a high efficiency of particle reconstruction. For example, for the CBM

experiment efficiencies of about 15% for Λ and 5% for Ξ^- with AuAu collisions at 35 AGeV are achieved together with high signal-to-background ratios (1.3 and 5.9 respectively).

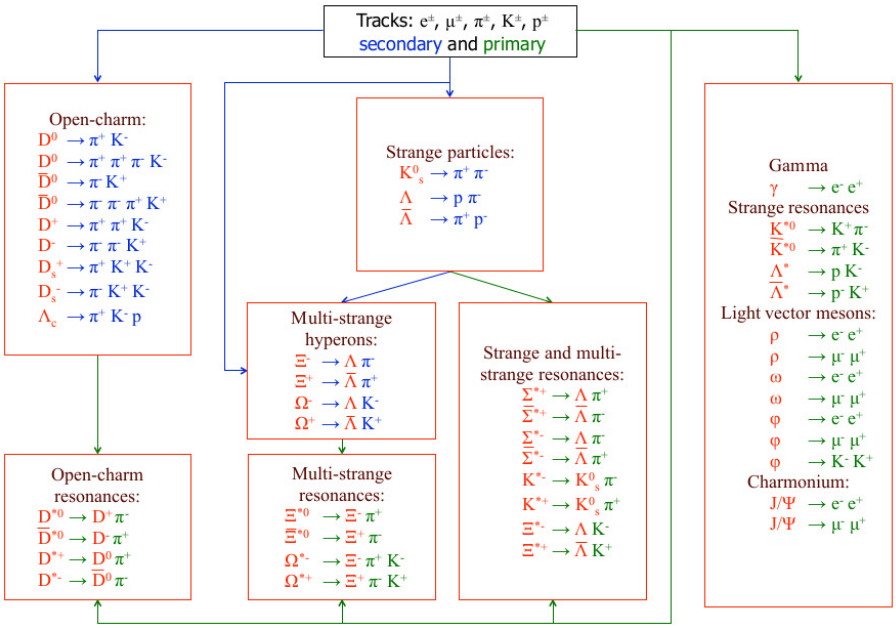

Figure 12. Block diagram of the KF Particle Finder package. The particle parameters, such as decay point, momentum, energy, mass, decay length and lifetime, together with their errors are estimated using the Kalman filter method.

In order to utilize all resources of modern computer architectures and to achieve the highest possible speed the KF Particle Finder is vectorized based on the SIMD instructions. Also, the package has been parallelized at the level of cores and demonstrates a strong linear scalability on many-core servers with respect to the number of cores. For example, the scalability of the package using 100 UU minimum bias collisions of the STAR experiment per thread on two Intel Xeon E5-2680 CPUs with 32 logical cores and one Intel Xeon Phi accelerator with 240 threads running simultaneously is shown in Fig. 13. For the Intel Xeon Phi the scalability has shown for running 1, 2, 3 and 4 threads per each physical core.

The core of the KF Particle Finder package is used in the CBM experiment for processing of simulated data and in the ALICE experiment – for real data. The package is under installation in the PANDA and STAR experiments. Such simultaneous use of the package in the running ALICE and STAR experiments and in the future FAIR experiments CBM and PANDA provides reach functionality and operate reliability of the KF Particle Finder.

10 FLES – a standalone First Level Event Selection package for the CBM experiment

The First Level Event Selection (FLES) package of the CBM experiment is intended to reconstruct online the full event topology including tracks of charged particles and short-lived particles. The

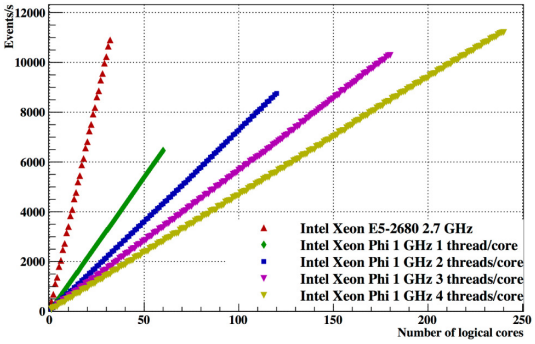

Figure 13. Scalability of the KF Particle Finder package on two Intel Xeon E5-2680 CPUs with 32 logical cores and one Intel Xeon Phi accelerator with 240 logical cores with minimum bias UU collisions at 200 AGeV in the STAR experiment.

FLES package consists of several modules (the block-diagram is shown on Fig. 14): CA track finder, KF track fitter, KF Particle Finder and physics selection. In addition, a quality check module is implemented, that allows to monitor and control the reconstruction process at all stages. The FLES package is platform and operating system independent.

The FLES package is portable to different many-core CPU architectures. The package is vectorized using SIMD instructions and parallelized between CPU cores. All algorithms are optimized with respect to the memory usage and the speed.

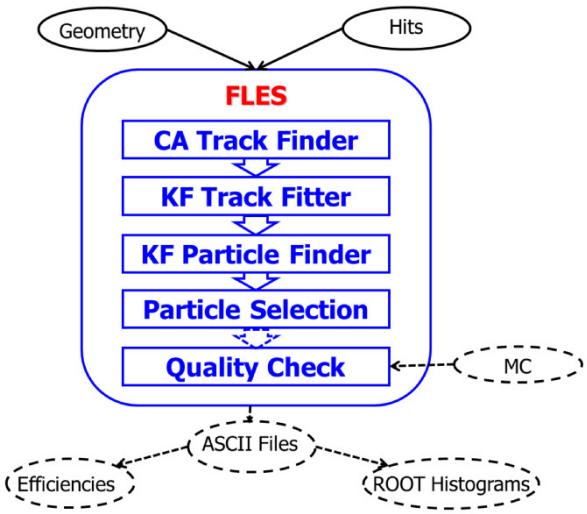

Figure 14. Block-diagram of the FLES package, which consists of several modules: track finder, track fitter, particle finder and physics selection.

Four servers with Intel Xeon E7-4860, L5640 and X5550 processors and with AMD 6164EH processor have been used for the scalability tests. The AMD server has 4 processors with 12 physical cores each, in total 48 cores. All Intel processors have the hyperthreading technology, therefore each physical core has two logical cores. The most powerful Intel server has 4 processors with 10 physical cores each, that gives 80 logical cores in total.

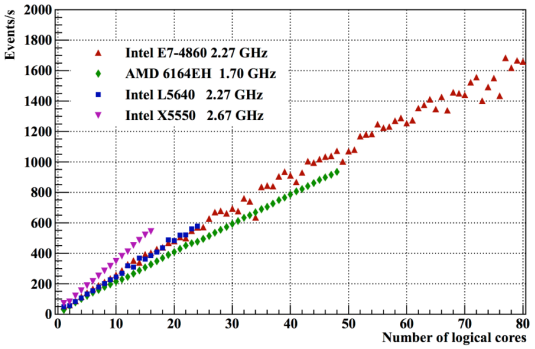

Figure 15. Scalability of the FLES package on many-core servers with 16, 24, 48 and 80 logical cores.

The FLES package has been parallelized with ITBB implementing the event-level parallelism by executing one thread per one logical core. Reconstruction of 1000 minimum bias Au-Au UrQMD events at 25 AGeV has been processed per each thread. In order to minimize the effect of the operating system each thread is fixed to a certain core using the pthread functionality provided by the C++ standard library. Fig. 15 shows a strong scalability for all manycore systems achieving the reconstruction speed of 1700 events per second on the 80-cores server.

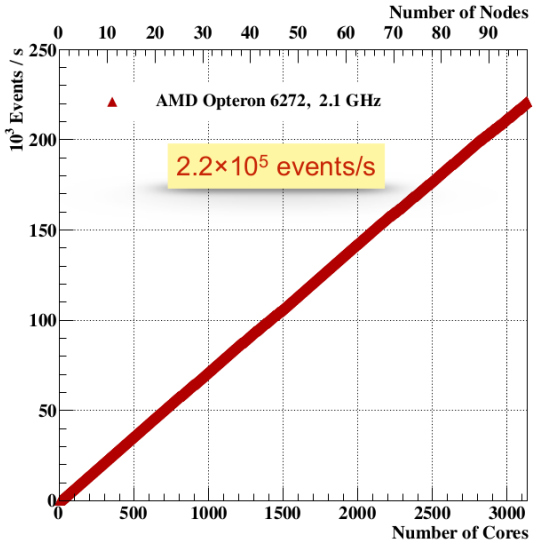

Figure 16. Scalsaiability of FLES, package on 3 200 cores of the FAIR-Russia HPC cluster (ITEP, Moscow).

The FLES package in the CBM experiment will be performed for the on-line selection and the offline analysis on a dedicated many-core CPU/GPU farm. The farm is currently estimated to have a compute power equivalent to 60 000 modern CPU cores. Fig. 16 shows the scalability of the FLES package on a many-core computer farm with 3 200 cores of the FAIR-Russia HPC cluster (ITEP, Moscow).

11 Summary

The challenges in the data reconstruction and physics analysis of the CBM experiment, discussed in the paper, are typical not only for the FAIR experiments, but for all modern and future experiments at LHC and other research centers in the world. One can expect a higher level of consolidation between the experiments in the cooperative work of their expert groups in order to find an optimal way in developing of new or restructuring of the existing reconstruction and analysis packages and frameworks.

References

[1] The CBM Collaboration, Compressed Baryonic Matter Experiment, Tech. Stat. Rep., GSI, Darmstadt, 2005; 2006 update

[2] I. Kisel, I. Kulakov and M. Zyzak, IEEE Trans. Nucl. Sci. **60**, 3703–3708 (2013)

[3] I. Kisel, Nucl. Instr. and Meth. A **566** 85–88 (2006)

[4] S. Gorbunov, U. Kebschull, I. Kisel, V. Lindenstruth, and W.F.J. Müller, Comp. Phys. Comm., **178**, 374–383 (2008)