# Online 4-Dimensional Reconstruction of Time-Slices in the CBM Experiment

Valentina Akishina and Ivan Kisel

***Abstract*—The main focus of the CBM experiment (FAIR, Darmstadt, Germany) is the measurement of very rare probes, which require interaction rates of up to 10 MHz. It makes it mandatory to perform the full online event reconstruction at the first level trigger and to operate with huge data rates of up to 1 TB/s. CBM will have a continuous beam without bunch structure, that means collisions may overlap in time, making the traditional event-based approach not applicable. It requires full online event reconstruction and selection to be done in 4D, including time. The standalone First Level Event Selection (FLES) package has been created for the CBM experiment. It contains all reconstruction stages: track finding, track fitting, short-lived particles finding and event selection. For track reconstruction the Cellular Automaton (CA) method is used, that allows to resolve tracks from a time-slice in event-corresponding groups. The algorithm is intrinsically local and the implementation is both vectorized within a core and parallelized between CPU cores. The CA track finder shows a strong scalability on many-core systems. The speed-up factor of 10.6 was achieved on a CPU with 10 physical cores using hyper-threading.**

***Index Terms*—Data processing, elementary particles, high performance computing, pattern recognition.**

## I. Introduction

THE CBM (Compressed Baryonic Matter) experiment [1] at the upcoming FAIR accelerator (GSI, Darmstadt, Germany) aims to explore the phase diagram of strongly interacting matter at the highest net baryon densities by investigating nuclear collisions from 2 to 45 GeV energy per-nucleon ($A$GeV). One of the most promising observables carrying information on the early stage of a collision are rare probes measurements (e.g. charmonium), which require unprecedented statistics for this energy range and, thus, collision rates up to 10 MHz. Taking into account the multiplicity of charged particles in a heavy-ion collision (up to 1000, see Fig. 1), one should expect data flow rate of 1 TB/s. Such a huge data rate makes it mandatory to select interesting events online with a reduction factor of about three orders of magnitude in order to meet the data recording rate of 1 GB/s.

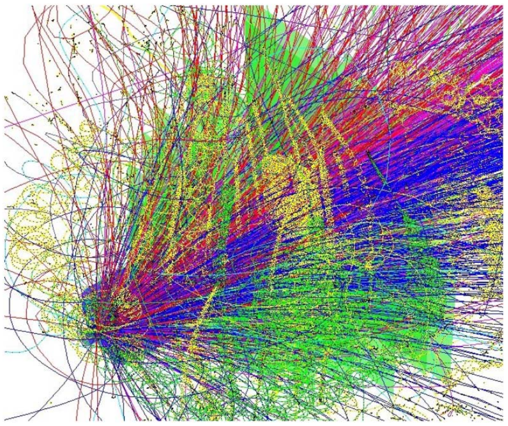


Fig. 1. Tracks in a central Au-Au event at 25 $A$GeV collision energy in the CBM experiment. On average there are about 1000 tracks of charged particles.

Moreover, CBM will operate on a continuous beam without bunch structure. As a result, collisions may overlap in time, making the traditional event-based approach not applicable. That requires full online event reconstruction and selection not only in space, but also in time, so-called 4D event reconstruction and selection.

The event building and filtering is implemented using the First Level Event Selection (FLES) package [2] running online on a dedicated multiprocessor computer farm. That requires the package to be fast, precise and suitable for online data processing in order to use the full potential of many-core computer architectures.

The FLES software is being developed as a platform and operating system independent package, which includes several modules of the reconstruction chain: track finding, track fitting, short-lived particles finding and event selection. The input data is distributed within the FLES farm in a form of so-called time-slices, whose time length is proportional to the computing power of a processing node.

The Cellular Automaton (CA) track finder [3] is used to reconstruct tracks of charged particles inside a time-slice. The reconstruction of each time-slice is performed in parallel between cores within a CPU, thus minimizing communication between CPUs. After all tracks of the whole time-slice are found and fitted in space and time (4D), they are collected into clusters of tracks originated from common primary vertices,

Manuscript received June 16, 2014; revised December 12, 2014; accepted May 17, 2015. Date of current version December 11, 2015. This work was supported by the Hessian Ministry of Science and Art, Hessian LOEWE Initiative through the Helmholtz International Center for FAIR (HIC for FAIR), HGS-HIRe, GSI F&E, BMBF Verbundforschung and EU-FP7 HadronPhysics2.

V. Akishina is with the Uni-Frankfurt—Goethe University, Frankfurt am Main 60325, Germany, and Frankfurt Institute for Advanced Studies (FIAS), Frankfurt am Main 60438, Germany (e-mail: V.Akishina@gsi.de).

I. Kisel is with Uni-Frankfurt—Goethe University, Frankfurt am Main 60325, Germany, and also with the Frankfurt Institute for Advanced Studies (FIAS), Frankfurt am Main 60438, Germany, and Helmholtz Center for Heavy Ion Research (GSI), Darmstadt 64291, Germany (e-mail: I.Kisel@compeng.uni-frankfurt.de).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNS.2015.2498099

0018-9499 © 2015 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

Authorized licensed use limited to: Gesellschaft Fuer Schwerionen-Forschung mbh. Downloaded on February 17,2020 at 13:52:03 UTC from IEEE Xplore. Restrictions apply.

which then are fitted, thus identifying 4D interaction points registered within the time-slice. Secondary tracks are associated with primary vertices according to their estimated production time. After that short-lived particles are found. The last stage of the FLES package is selection of events according to the requested trigger signatures.

The paper describes the track finding stage of the event reconstruction, namely the reconstruction of time-slices at high interaction rates up to 10 MHz.

## II. Cellular Automaton Track Finder

The general problem of finding tracks of charged particles out of dense and contaminated with noise detector measurements is a non-trivial task, which is often considered to be the most challenging and time-consuming phase of the event reconstruction. One of the reasons for that is a huge amount of combinatorial combinations, that grows with increased track density and has to be considered by a track finder in order to combine one- or two-dimensional measurements into five-dimensional tracks. Unfortunately, the exponential growth of the combinatorial enumeration at high track densities usually makes it impossible to consider all combinations within a reasonable time. However, a solid solution for the combinatorial optimization is provided by the CA track finder algorithm [3]–[7].

The CA method (Fig. 2), being a local one, suppresses combinatorial enumeration [8] by building short segments at the first stage before starting the main combinatorial search (step 1 in Fig. 2). These track segments, so-called cells, have a higher dimensionality, than measurements have. After this stage is finished the CA track finder never goes back to processing hits information again, working only with created track segments instead. Taking into account a track model, the method searches for neighboring cells, which share a hit in common and have the same direction within some error, and, thus, potentially belong to the same track. During this neighbors search the track finder also estimates a possible position of a segment in a track (step 2). Beginning with the first station the track finder goes to the last station moving from one neighbor to the next one assigning to each segment a counter, which stores number of neighbors to the left. Starting with the segment of the largest position counter the track finder follows a chains of neighbors collecting segments into a track candidate (step 3). As a result one gets a tree structure of track candidates. In the last stage (step 4) the competition between the track candidates takes place: only the longest tracks with the best $\chi^2$-value sharing no hits in common with better candidates are to survive.

As one can conclude from the CA track finder strategy the major part of the algorithm is intrinsically local, since working only with data within a small neighborhood region at each particular moment. In addition to that, the algorithm transforms the tracking information step-by-step to a higher consolidation extent: moving from hits to segments, from segments to candidates, from candidates to tracks. Thus, the information processed and analyzed once by the track finder is stored in a new form for the next stage with no need to read it again later. This optimizes memory access, since no data is read or processed twice.

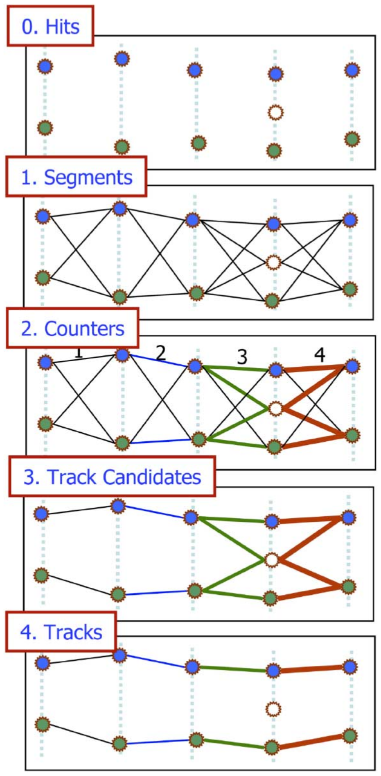


Fig. 2. Simplified scheme of the Cellular Automaton track finding algorithm. Tracking stations are shown by dashed lines. Hits, which correspond to two different particles, are shown as blue and green circles, noise hit is shown as white circle. Track segments are depicted by lines, color of a line corresponds to its position on a track.

The algorithm features, mentioned above, make it suitable for a parallel implementation in order to be able to fully utilize the power of many-core computer architectures.

## III. Track Finding at High Track Multiplicities

Since the CBM experiment will operate at extremely high interaction rates, different collisions may overlap in time, leaving no possibility to separate them in a trivial way. Thus, the need to analyze so-called time-slices, 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 the 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 many-core computer architectures. For implementing in-event level parallelism these events do not have enough sources of parallelism (like hits, segments, track candidates for different reconstruction stages) in order to be reconstructed on 10 or more CPU cores simultaneously.

As a first step on a way towards the time-slice reconstruction we introduce a group 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) Au-Au minimum bias events at 25 $A$GeV ignoring such information as event number or time measurements (Fig. 3). One can notice that with such an approach we create hits on

Authorized licensed use limited to: Gesellschaft Fuer Schwerionen-Forschung mbh. Downloaded on February 17,2020 at 13:52:03 UTC from IEEE Xplore. Restrictions apply.

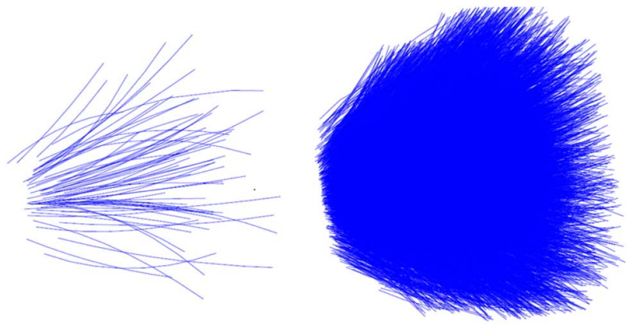


Fig. 3. Reconstructed tracks in a minimum bias event (left) and in a packed group of 100 minimum bias events (right), 109 and 10 340 tracks on average respectively.

the minimum bias event level, so we produce fake combinatorial space points in double-sided strip detectors within original events only, not within the whole group.

The groups were treated by the CA track finder as regular events 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 the track multiplicity. Here we define the efficiency as a ratio of reconstructed and reconstructable tracks. A track is considered as reconstructable, if it has at least 4 or more consecutive Monte-Carlo points. By definition, a reconstructed track is assigned to a generated particle, if at least 70% of its hits have been produced by this particle. A generated particle is regarded as found, if it has been assigned to at least one reconstructed track. A reconstructed track is called ghost, if it is not assigned to any generated particle according to the 70% criterion.

As one can see in Fig. 4 high momentum primary tracks (RefPrim), that have a particular physical importance, are reconstructed with an excellent efficiency of about 96%, which varies within less than 2% up to hundred events in a group. 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 the multiple scattering in the detector material. It changes within a 6% window in the 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.

However, not only the efficiency, but also the speed of the reconstruction algorithm is crucial for successful performance of the CBM experiment. We have studied the processing 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. 5). The results show that the dependence is perfectly described with a second order polynomial. This is a promising result, if one keeps in mind the exponential growth of combinatorics with increasing track multiplicity. 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.

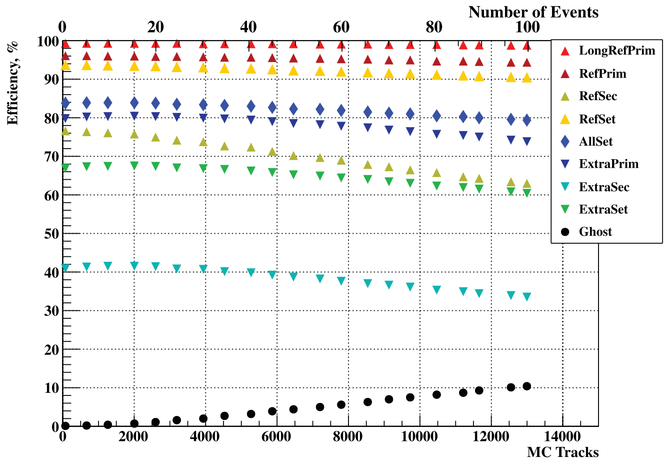


Fig. 4. Track reconstruction efficiencies and ghost rate for different sets of tracks versus track multiplicity.

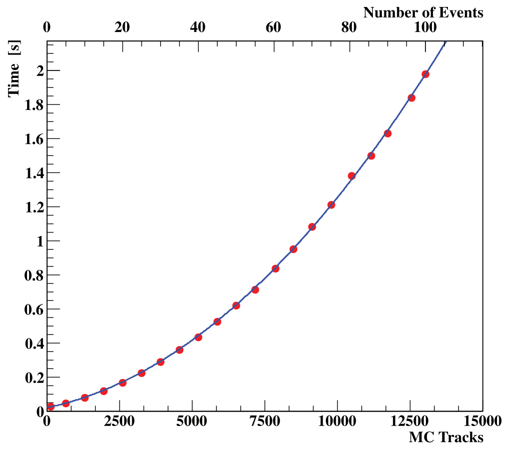


Fig. 5. The CA track finder time needed to reconstruct groups of minimum bias events without time information with respect to the track multiplicity. The dependence is fitted with a second order polynomial.

## IV. In-Event Parallelism for the CA Track Finder

The vectorized, but sequential in terms of cores usage, version of the CA track finder [2] was taken as a starting point for the development of a parallel version using the Open Multi-Processing (OpenMP) technique [9]. OpenMP is an API, which supports multi-platform shared-memory parallel programming. It provides an interface for implementing parallelisation between cores in a user application. The user prompts OpenMP, which section of the code should be run in parallel, marking the section with a preprocessor directive, defines number of threads (independent streams of instructions) before the section is executed and the computations are divided between the threads. 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 a CPU from sending a thread to other cores during runtime, and, thus, affecting the parallelisation efficiency, we use the POSIX library [10] to set a permanent thread to core affinity. In order to prevent non-optimal usage of NUMA architecture memory, like processing data allocated on a certain CPU by another CPU, a function *mlockall* was used. This function allows the user to prevent the data migration in the memory by causing

Authorized licensed use limited to: Geselleschaft Fuer Schwerionen-Forschung mbh. Downloaded on February 17,2020 at 13:52:03 UTC from IEEE Xplore. Restrictions apply.

all of the pages mapped by the address space of a process to be memory resident until unlocked or until the process exits [10].

The goal was to make a parallel implementation of the CA algorithm keeping the same efficiencies and having the stable track reconstruction result regardless of a number of executing threads. However, certain parts of the code, being essentially sequential, had to be significantly rewritten in order to introduce parallelism.

Parallel implementation requires certain features from an 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 a parallel section should always be thread-safe, so that 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 some level of synchronization into the threads execution. Synchronization usually slows down the speed of a program, since threads have to wait for each other or exchange results of their work, so we tried to minimize usage of this mechanism. Also memory optimization and the data structures scheme become essential in the case of parallel programming: it is often crucial for the speed that one uses the fastest memory only. So certain work with data structures optimization in order to fit the memory size is necessary.

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

As it was mentioned in the introduction, the algorithm consists of several logical parts (see Table I). 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 4%, and in addition during 3.4% of the reconstruction time we prepare information for the next iteration. All steps of the algorithm were parallelized 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 hyper-threaded logical cores) are presented in Fig. 6.

TABLE I
FRACTION OF THE TOTAL EXECUTION TIME FOR DIFFERENT STEPS OF THE CA TRACK FINDER ALGORITHM IN A SEQUENTIAL RUN

| Algorithm step | % of total time |
|---|---|
| Initialization | 2.0 |
| Triplets construction | 90.4 |
| Tracks construction | 4.1 |
| Final stage | 3.4 |

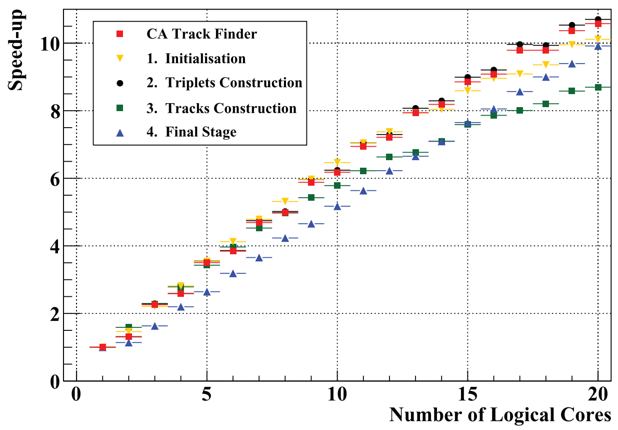


Fig. 6. Speed-up factor due to parallelisation for different steps and the full algorithm on Intel Xeon E7-4860 CPU with 10 physical cores and hyper-threading for the case of 100 minimum bias events grouped.

Some steps have a better speed-up for a higher number of cores due to less thread synchronization needed. The algorithm shows a linear scalability. In addition to a speed-up factor of 10, which reflects the total number of physical cores, one expects an extra 25%–30% increase of the speed of the algorithm due to hyper-threading, that theoretically gives a speed-up factor of about 13 on such a CPU. The achieved speed-up factor is 10.6 for the full CA track finder reconstruction algorithm on the Intel Xeon E7-4860 CPU with 10 physical cores with hyper-threading.

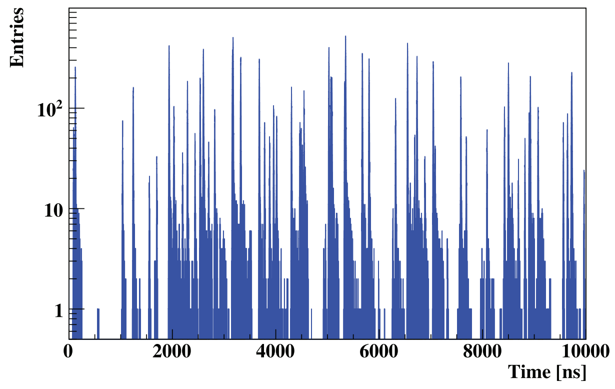
## V. Reconstruction of Time-Slices

After 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 is an implementation of time measurements.

In order to introduce time measurements into the reconstruction procedure the event start time was assigned to each minimum bias event in a 100 events group during the 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 the event start time plus a time shift due to the time of flight from a collision point to a detector station. This time of flight differs for each hit. 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 time measurements we can use the time information in the CA track finder. Here we do not allow to build triplets out of hits, which time difference is greater than $3.5\sigma$ of

Authorized licensed use limited to: Gesellschaft Fuer Schwerionen-Forschung mbh. Downloaded on February 17,2020 at 13:52:03 UTC from IEEE Xplore. Restrictions apply.



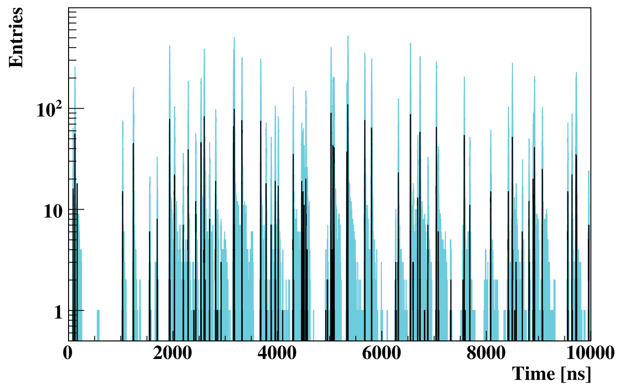


Fig. 7. Part of a time-slice with 100 minimum bias events. The upper picture: with blue color the distribution of hit time measurements in the time-slice is shown. The picture below: with light blue color the initial distribution of hit measurements is shown (same as in the upper picture), black color shows time measurements of the reconstructed tracks.

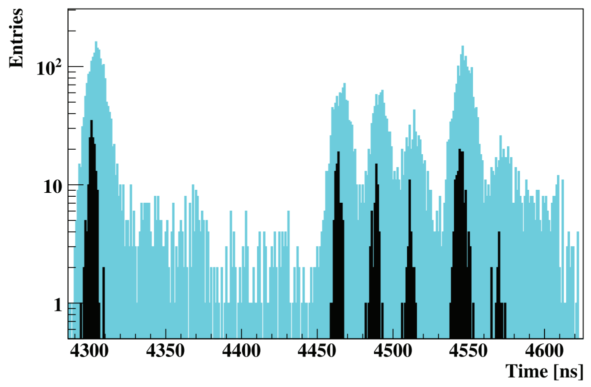


Fig. 8. Zoomed-in part of Fig. 7: black reconstructed track groups are well resolved on the blue background of overlapped initial hits.

the detector time resolution. It is a justified assumption, 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 the regular way described above. After the reconstruction we assign to each track a time measurement, which is calculated as an average of the hit time measurements.

The initial distribution of hits measurements representing the complexity of defining event borders in a time-slice at the interaction rate of $10^7$ Hz is shown in the upper part of Fig. 7 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 the lower part of Fig. 7. The reconstructed tracks clearly represent groups, corresponding to events, which they originate from. Even in the area of the most severe overlap (Fig. 8) the time-based CA track finder allows to resolve tracks from different events in time.

## VI. Conclusion

The standalone FLES package for the CBM experiment contains all reconstruction stages: track finding, track fitting, short-lived particles finding and event selection.

For the most time-consuming part of the reconstruction procedure the Cellular Automaton track finder is used. The efficiency of the algorithm proved to be stable with respect to the track multiplicity up to the extreme case of reconstruction of 100 minimum bias events at once without use of time information. The reconstruction time dependence on the track multiplicity in these conditions behaves as a second order polynomial, leaving a room for further improvement after adding time information into the track finding routine.

The CA track finder is both vectorized and parallelized. The algorithm shows strong scalability on many-core systems. The speed-up factor of 10.6 was achieved on a CPU with 10 physical cores using hyper-threading.

The event-based CA track finder was adapted for the time-slice-based reconstruction, which is a requirement in the CBM experiment for the event building. The 4D CA track finder allows to resolve hits from different events overlapping in time into event-corresponding clusters of tracks. The FLES package is ready for the 4D reconstruction of time-slices in the CBM experiment.

## References

[1] The CBM Collaboration, Compressed baryonic matter experiment GSI, Darmstadt, Tech. Stat. Rep., 2005, 2006 update.
[2] I. Kisel, I. Kulakov, and M. Zyzak, "Standalone first level event selection package for the CBM experiment," *IEEE Trans. Nucl. Sci*, vol. 60, pp. 3703–3708, Oct. 2013.
[3] I. Kisel, "Event reconstruction in the CBM experiment," *Nucl. Instr. Meth. A.*, vol. 566, pp. 85–88, 2006.
[4] A. Glazov, I. Kisel, E. Konotopskaya, and G. Ososkov, "Filtering tracks in discrete detectors using a cellular automaton," *Nucl. Instr. Meth. A.*, vol. 329, pp. 262–268, 1993.
[5] I. Kisel, V. Kovalenko, and F. Laplanche *et al.*, NEMO Collaboration, "Cellular automaton and elastic net for event reconstruction in the NEMO-2 experiment," *Nucl. Instr. Meth. A.*, vol. 387, pp. 433–442, 1997.
[6] I. Abt, D. Emeliyanov, I. Kisel, and S. Masciocchi, "CATS: A cellular automaton for tracking in silicon for the HERA-B vertex detector," *Nucl. Instr. Meth. A.*, vol. 489, pp. 389–405, 2002.
[7] I. Abt, D. Emeliyanov, I. Gorbounov, and I. Kisel, "Cellular automaton and Kalman filter based track search in the HERA-B pattern tracker," *Nucl. Instr. Meth. A.*, vol. 490, pp. 546–558, 2002.
[8] R. Mankel, "Pattern recognition and event reconstruction in particle physics experiments," *Rep. Prog. Phys.*, vol. 67, pp. 553–622, 2004.
[9] The OpenMP API specification for parallel programming [Online]. Available: http://openmp.org/wp
[10] POSIX threads programming [Online]. Available: https://computing.llnl.gov/tutorials/pthreads

Authorized licensed use limited to: Geselleschaft Fuer Schwerionen-Forschung mbh. Downloaded on February 17,2020 at 13:52:03 UTC from IEEE Xplore. Restrictions apply.