# Standalone First Level Event Selection Package for the CBM Experiment

Ivan Kisel, Igor Kulakov, and Maksym Zyzak

***Abstract*—The main focus of the CBM experiment (FAIR, Germany) is the measurement of very rare probes at interaction rates of up to 10 MHz. The experiment will operate with a data flow of up to 1 TB/s and requires full online event reconstruction and selection. This is a task of the First Level Event Selection (FLES).**

**A standalone FLES package has been developed for the CBM experiment. It contains modules for all reconstruction stages: track finding, track fitting, short-lived particle finding and selection. Reconstruction of about 50 types of particle decay channels is currently implemented. The algorithms are local with respect to data and their implementation is both vectorized (SIMD) and parallelized between CPU cores.**

**For the track reconstruction the Cellular Automaton (CA) and the Kalman filter (KF) algorithms are used, that allows to achieve a high track reconstruction efficiency of up to 97% and a track parameters quality with 1% momentum resolution. The KF particle finder has a high efficiency with an optimal signal to background ratio. The FLES package shows a strong scalability on many-core systems and a processing speed of 1700 events per second on an Intel based computer with 80 cores. The investigation was done based on simulated minimum bias Au-Au UrQMD collisions at 25 $A\mathrm{GeV}$ with a realistic detector response.**

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

## I. Introduction

THE CBM (Compressed Baryonic Matter) experiment [1] is an experiment being prepared to operate at the future Facility for Antiproton 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 (up to 1000 particles in central collisions, see Fig. 1) 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. The data acquisition is thus being designed in a free-running fashion, without a hardware trigger. Event reconstruction and selection will be performed online on a dedicated many-core CPU/GPU computer farm.

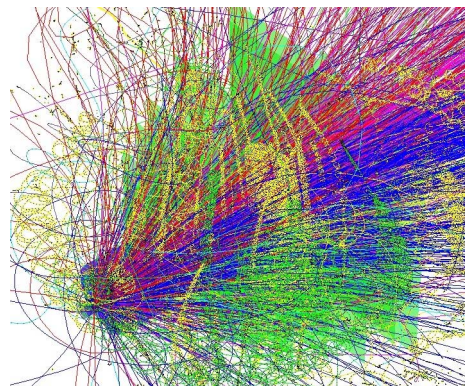

Fig. 1. A simulated central Au-Au collision at 25 $A\mathrm{GeV}$ energy in the CBM experiment with about 1000 charged particles (different colors correspond to different types of particles).

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 FLES package consists of several modules (the block diagram is shown on Fig. 2): track finder, track fitter, particle finder and physics selection. As an input the FLES package receives a simplified 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 [2] using to the registered hits. The Kalman filter (KF) based track fit [3] 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 [4] 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 KF particle finder also selects particle-candidates from a large number of random combinations. 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 in the CBM experiment will be performed for the online selection and the off-line 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. The FLES algorithms have to be therefore local and parallel with respect to data and thus require a fundamental redesign of

Manuscript received July 01, 2012; revised October 24, 2012; accepted May 20, 2013. Date of publication July 01, 2013; date of current version October 09, 2013. 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.

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).

I. Kulakov and M. Zyzak are with Uni-Frankfurt — Goethe University, Frankfurt am Main 60325, Germany, and Frankfurt Institute for Advanced Studies (FIAS), Frankfurt am Main 60438, Germany (e-mail: I.Kulakov@gsi.de; M.Zyzak@gsi.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.2013.2265276

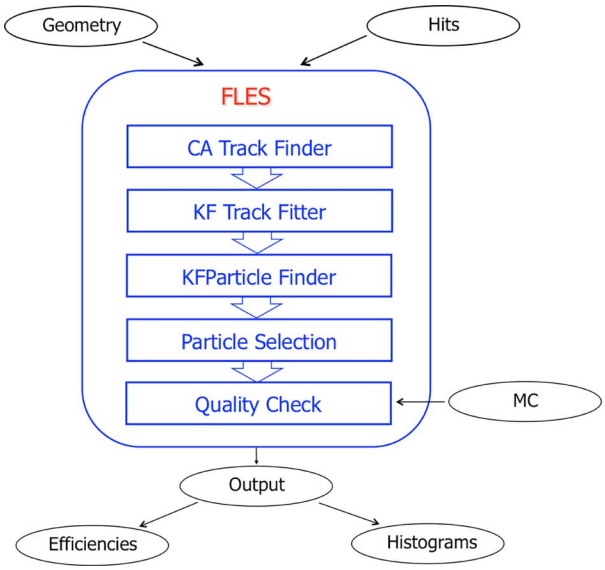


Fig. 2. Block diagram of the FLES package.

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 data values into a vector register and to work with them simultaneously getting 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) [5], that provides a scalable event-level parallelism with respect to the number of hardware threads and CPU cores.

## II. Parallelism in the FLES Package

The SIMD instruction set provides possibility to perform one operation on several values simultaneously. Majority of modern CPUs have 128 bit SIMD registers allowing to pack 4 single precision floating point values into one vector register. The newest CPUs have already 256 bit registers that corresponds to 8 values, and processors with 16 values wide registers are also under development. That makes utilization of SIMD registers by the FLES package extremely important. All modules of the FLES package are vectorized (SIMDized) using specially developed headers, that keeps the reconstruction algorithms and the low-level code implementation separated.

The header files overload the SIMD instructions implementing the operands and inlining the basic arithmetic and logic functions, that makes the code compact and easy readable. An 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

```
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); }
```

in the header file.

Use of the header files provides a simple and flexible SIMD implementation with respect to different CPU architectures. It keeps the program code in a scalar-like form, while the CPU specific SIMD extensions are hidden in the header files, which can be chosen automatically depending on the CPU type. Also a header file with the true scalar implementation exists, that allows to make the standard debugging and testing of the code.

Vectorization using the SIMD instruction set is implemented at all stages of the FLES package. In the CA track finder all operations with track segments, track candidates and tracks are vectorized. Vectorization is especially important in the Kalman filter track fit, which is extensively used in the CA track finder in order to reliably estimate parameters of the track segments and quality of the track candidates. Since the tracks are independent, they can be fitted in parallel by the same algorithm after packing the corresponding track parameters of four tracks into SIMD registers (see more details in [3]). They are packed such that, for instance, $x$-coordinates of four tracks are represented as a vector of four elements $(x_i, x_{i+1}, , x_{i+2}, x_{i+3})$. Operating with data in such a way, the data processing remains identical both for scalar and vector representations.

Modern computer architecture developments are currently focused on increasing the number of cores rather than the frequency of the processing units, thus forcing the users to exploit the MIMD (Multiple Instructions – Multiple Data) architecture. Here the algorithms have to be parallelized between cores in an efficient and scalable way.

The Intel Threading Building Blocks package (ITBB) [5] is used in the FLES package for parallelization between cores. ITBB is a C++ template library, which offers a rich functionality for parallel execution on several cores. In order to access multiple processors, groups of operations are treated here as tasks. An ITBB program thus creates, synchronizes and destroys graphs of dependent tasks. Tasks are then executed with respect to the graph dependencies.

Using the ITBB parallelization, the FLES package demonstrates a strong many-core scalability at the event-level parallelization, as will be reported in Section VII.

## III. Cellular Automaton Track Finder

Every track finder must handle a very specific and complicated combinatorial optimization process, grouping together one- or two-dimensional measurements into five-dimensional

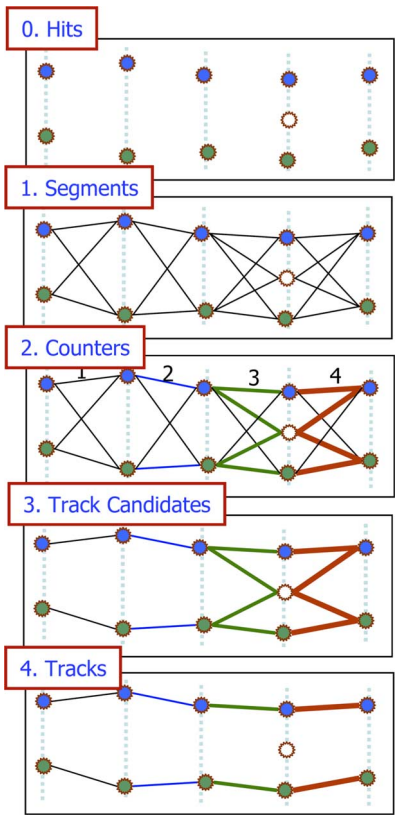


Fig. 3. A simplified illustration of the Cellular Automaton based track finding algorithm. Here the tracking stations are shown by the vertical dashed lines, hits of two different particles are shown by the blue and green circles, the noise hit is shown by the empty circle. Track segments are shown by the solid lines with their thickness and color corresponding to a possible position of a segment on a track.

tracks. The Cellular Automaton (CA) method profits from building up cells, i.e., short track segments with a higher than measurements dimensionality, before starting the combinatorial search [2]. The method is intrinsically local working with data only in a neighborhood. It provides a steady consolidation of the track information identified in the course of the algorithm evolution. In addition, the CA based algorithms can be parallelized in order to be implemented on modern and future many-core CPU/GPU computer architectures.

In the CA method (Fig. 3) 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 $\chi^2$-values and then selects among them the best tracks.

Some additional modifications are done in the FLES package in order to reflect the specific features of the CBM experiment.

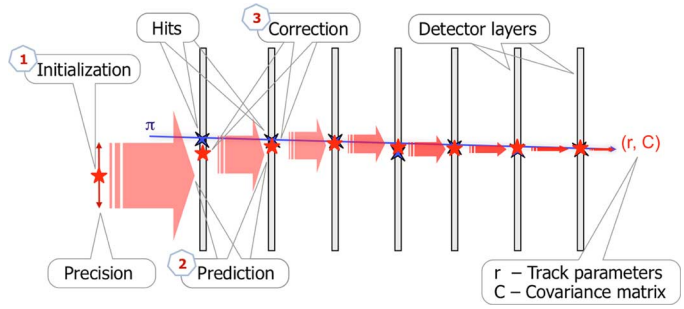


Fig. 4. An illustration of the Kalman filter based track fitting algorithm.

The track segments are created from hits in each three consecutive stations to reconstruct the particle momentum. To recover a possible detector inefficiency, hits in the track segments can be also separated by one inefficient station. The track finding procedure is organized in several iterations to make the reconstruction fast and reliable in presence of a high track density: at the first iteration only high-momentum primary tracks are reconstructed, at the second one — low-momentum primary tracks, and then — all other tracks. After each iteration all used hits are removed from further consideration, thus significantly reducing the combinatorics, which is extremely high due to the use of double-sided strip detectors.

## IV. Kalman Filter Track Fitter

The estimation (or fit) of the track parameters and their errors is done by the Kalman Filter (KF) method [3]. The Kalman filter (1) starts with an arbitrary initial approximation (Fig. 4); (2) adds hits one after the other; (3) refines the state vector and gets the optimal values of the track parameters after the last hit.

The Kalman filter is intensively used in the combinatorial part of the CA track finder, therefore its fast implementation on modern CPU/GPU computer systems and stability in single precision are crucial.

Starting from the idea of using the SIMD unit of modern processors, the KF track fitting algorithm was examined aiming to increase the speed of the CA track finder as a part of the FLES event reconstruction. After the detailed optimization of the memory utilization and the numerical analysis, the KF algorithm had been vectorized [3]. To optimize the memory usage, a magnetic field approximation is used for particle propagation instead of the full magnetic field map, which takes about 70 MB and therefore does not fit into the CPU cache memory. The magnetic field is approximated with a polynomial function of fifth order at each detector station. During the fit of a track the field behavior between the stations is approximated with a parabola taking field values at the three closest measurements along the track. To stabilize the fit, an initial approximation of the track parameters is done by the least square estimator assuming a one-component magnetic field. The first measurement is processed in a special way, which increases the numerical stability of the method in single precision: the equations were simplified analytically using a special form of the initial covariance matrix. The track propagation in the nonhomogeneous magnetic field is done by an analytic formula, which is based on the Taylor expansion [6]. The analytic formula allows to obtain the

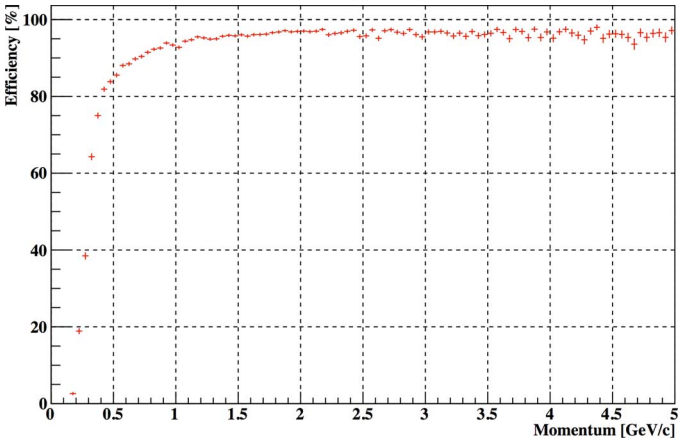


Fig. 5. Efficiency of the track reconstruction for minimum bias Au-Au collisions at 25 $A$GeV.

TABLE I
TRACK RECONSTRUCTION EFFICIENCY FOR MINIMUM BIAS AND CENTRAL EVENTS

| | 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 |

same track fit quality as the standard fourth order Runge-Kutta method, while being 40% faster. Operator overloading has been used to keep flexibility of the algorithm with respect to different CPU/GPU architectures. All these changes have increased the processing speed of the SIMD KF track fit algorithm down to 1 $\mu$s per track. This is an improvement by a factor 10000 with respect to the original scalar version of the algorithm [3].

## V. PERFORMANCE OF THE TRACK RECONSTRUCTION

For evaluation purposes a reconstructed track is assigned to a generated particle, if at least 70% of its hits have been caused by this particle [2]. A generated particle is regarded as found, if it has been assigned to at least one reconstructed track. If the particle is found more than once, all additionally reconstructed tracks are regarded as clones. A reconstructed track is called a ghost, if it is not assigned to any generated particle according to the 70% criterion.

Efficiency of the track reconstruction for minimum bias Au-Au UrQMD (Ultra relativistic Quantum Molecular Dynamics) simulated collisions at 25 $A$GeV is presented on Fig. 5. In addition the track reconstruction efficiencies for different sets of tracks and ratios of clones (double found) and ghost (wrong) tracks are shown in Table I. The tests have been performed on a 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 $\Lambda$ particles and cascade decays of $\Xi$ and $\Omega$, 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.

The CBM experiment is an experiment with a forward geometry along $Z$-axis and, therefore, has a typical set of tracks parameters: $x$ and $y$ track coordinates at a reference $z$-plane, $t_x = \tan\theta_x$ and $t_y = \tan\theta_y$ are the track slopes in the $XZ$- and $YZ$-planes, $q/p$ is an inverse particle momentum, signed according to the charge of a particle.

Residuals of the track parameters are determined as a difference between the reconstructed parameters and their true Monte-Carlo values. The normalized residuals (pulls) are determined as the residuals normalized by the estimated errors of the track parameters. In the ideal case these should be unbiased and Gaussian distributed with width of 1.0. Thus the pull distributions provide a measure of the track fit quality.

The residuals and the pulls for all track parameters are calculated at the first hit of each track. The distributions for the $x$, $t_x$ and $q/p$ parameters together with their Gaussian fits are shown on Fig. 6 (the results for $y$ and $t_y$ are similar). All distributions are not biased with pulls widths close to 1.0 indicating correctness of the fitting procedure. The slight deviations from 1.0 are caused by several assumptions made in the fitting procedure, mainly in the part of the detector material treatment. The $q/p$ pull is the widest being the most sensitive to these simplifications.

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

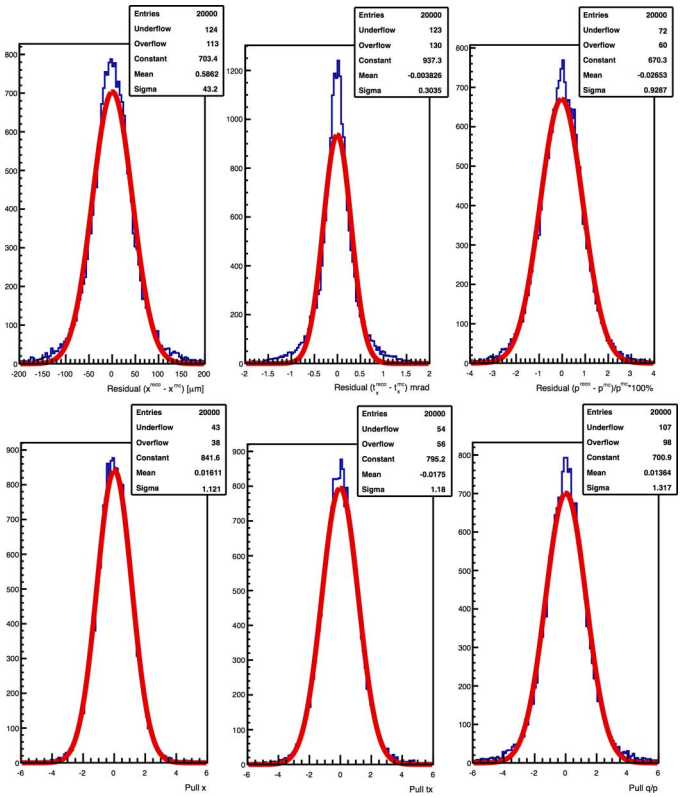


Fig. 6. Residuals and pulls distributions of the $x$ (43.2 $\mu$m, 1.12) , $t_x$ (0.30 mrad, 1.18) and $q/p$ (0.93%, 1.32) track parameters.

reconstruction efficiency and the low level of the combinatorial background improve significantly the event reconstruction and selection by the FLES package.

## VI. Reconstruction of Short-Lived Particles With the KF Particle Finder

Reconstruction of short-lived particles, which are not registered in the detectors, but can be reconstructed by their daughter products, is important for the success of the whole experiment. The KFParticle [4] software package for the reconstruction of short-lived particles has been developed such that all parameters of the particles are provided in the most suitable form for the final physics analysis. The state vector of the parameters contains three coordinates ($x$, $y$ and $z$), three components of the momentum ($p_x$, $p_y$ and $p_z$) and the energy ($E$). These parameters are traditionally used in physics for description of a moving particle, that makes the KFParticle package geometry and experiment independent. KFParticle is based on the Kalman filter method, therefore it provides an estimation of the particle parameters together with the covariance matrix. The package provides also an easy access to important physical quantities of a particle such as mass, decay length, lifetime, momentum, transverse momentum, rapidity, etc. together with their errors. The errors are calculated based on the covariance matrix of the state vector of the particle parameters. The KFParticle package treats all particles (long-lived as well as short-lived) in the same way, therefore the reconstruction of decay chains is straightforward within the package.

The fit of a short-lived particle is implemented iteratively. The KFParticle package starts with an initial approximation of the secondary vertex, adds particles one after another, refines the state vector and gives the optimal values after the last particle. If the initial approximation is not known precisely enough, the procedure can be repeated several times using the obtained state vector as the initial approximation for the next iteration.

Since the speed of the algorithms is crucial for the FLES package, KFParticle has been also fully vectorized and implemented in single precision. To optimize the memory usage in the package, the magnetic field approximation is used for the particle propagation instead of the full magnetic field map in the same way as for the Kalman filter (Section IV).

Based on the KFParticle package the KF particle finder has been developed. At the first stage all tracks of the charged particles, which are found by the CA track finder, are divided into two groups: primary and secondary tracks. Strange, multi-strange and open-charm particles are constructed from the secondary tracks taking into account the corresponding mass assumptions for the tracks. Combining the obtained strange and open-charm particles with the primary tracks, strange, multi-strange and open-charm resonances are constructed. The primary tracks with the corresponding mass hypothesis are combined into strange resonances (which decay into charged long-lived particles), light vector mesons and charmonium. The search of in total about 50 decay modes of short-lived particles is currently implemented in the KF particle finder. Taking into account the topology of the decay and $\chi^2$ criteria, the particle candidates are then selected and stored.

The search for the short-lived particles is done in one go, thus minimizing access to the memory. Together with the optimization and vectorization of the code this allows to achieve a high speed even in a presence of a huge combinatorics. The speed of the KF particle finder per core on a server with Intel Xeon E7-4860 CPUs is 1.4 ms per minimum bias Au-Au collision and 10.5 ms per central Au-Au collision at 25 $A$GeV.

The KF particle finder package provides the reconstruction efficiency and the signal to background (S/B) ratio for the reconstructed decays. For instance, for 1000 minimum bias Au-Au UrQMD events at 25 $A$GeV the reconstruction efficiency (normalized on $4\pi$) for the $K_s^0$ meson is 11.3% with the S/B ratio 1.15 and for the $\Lambda$ hyperon — 9.2% and 2.14 respectively. The mass distributions of $K_s^0$ and $\Lambda$ particles are shown on Fig. 7. Since the online version of the KFParticle package differs only in details of the parallel implementation from the off-line version, they both produce, therefore, consistent results for other decay channels.

## VII. Scalability of the FLES Package

Four servers with Intel Xeon E7-4860, L5640 and X5550 processors and with AMD 6164EH processor have been used for the scalability tests (Table II). The AMD server has 4 processors with 12 physical cores each, in total 48 cores. All Intel processors have the hyper-threading 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.

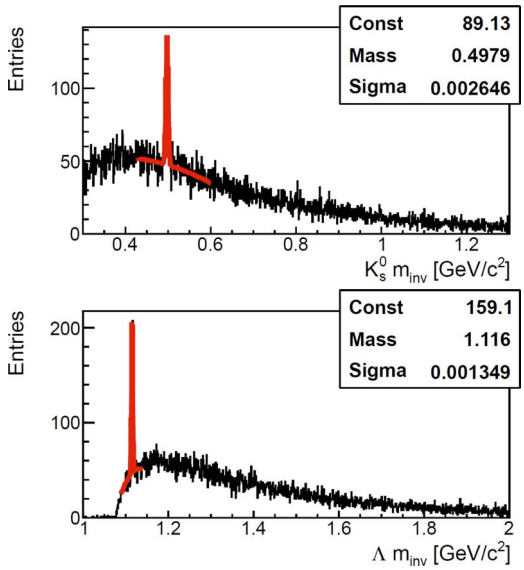


Fig. 7. Mass distributions of $K^0_s$ and $\Lambda$ candidates for 1000 minimum bias Au-Au UrQMD collisions at 25 $A$GeV.

TABLE II
SERVERS FOR THE SCALABILITY TESTS

| Processor | Clock, GHz | L3 Cache, MB | Number of CPUs | Total number of cores |
|---|---|---|---|---|
| Intel E7-4860 | 2.27 | 24 | 4 | 80 |
| AMD 6164EH | 1.7 | 12 | 4 | 48 |
| Intel L5640 | 2.27 | 12 | 2 | 24 |
| Intel X5550 | 2.66 | 8 | 2 | 16 |

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 $A$GeV 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. 8 shows a strong scalability for all many-core systems achieving the reconstruction speed of 1700 events per second on the 80-cores server.

## VIII. CONCLUSION

The standalone FLES package has been developed for the CBM experiment. It contains track finding, track fitting, short-lived particles finding and physics selection. The Cellular Automaton and the Kalman filter algorithms are used for finding and fitting tracks, that allows to achieve a high track reconstruction efficiency up to 97% and the track parameters quality with

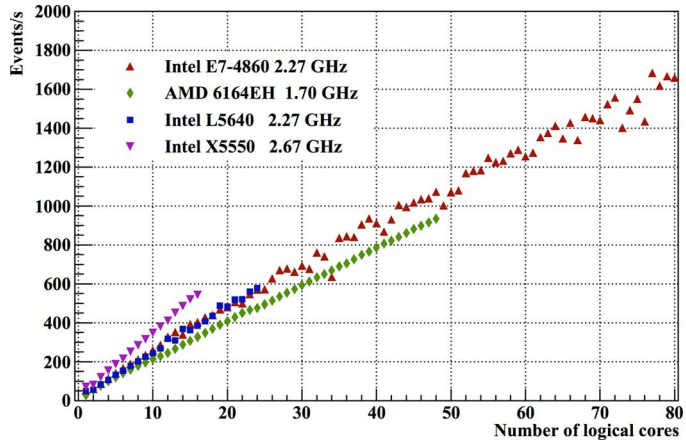


Fig. 8. Scalability of the FLES package on many-core servers.

1% of the momentum resolution. Reconstruction of about 50 decay channels of short-lived particles is currently implemented in the KF particle finder. The package shows a high reconstruction efficiency with an optimal signal to background ratio. For instance, the reconstruction efficiencies of 11.3% and 9.2% are achieved for $K^0_s$ and $\Lambda$ particles with the signal to background ratios of 1.15 and 2.14 respectively.

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. The FLES package shows a strong scalability on the many-core CPU systems and the processing and selection speed of 1700 events per second on a server with 80 Intel cores.

## ACKNOWLEDGMENT

The authors would like to thank Dr. I. Vassiliev for his valuable advices during the development the KF particle finder.

## REFERENCES

[1] The CBM Collaboration, "Compressed baryonic matter experiment," Tech. Stat. Rep., GSI, Darmstadt, Germany, 2005.
[2] I. Kisel, "Event reconstruction in the CBM experiment," *Nucl. Instrum. Methods Phys. Res. A*, vol. 566, pp. 85–88, 2006.
[3] S. Gorbunov, U. Kebschull, I. Kisel, V. Lindenstruth, and W. F. J. Müller, "Fast SIMDized Kalman filter based track fit," *Comp. Phys. Commun.*, vol. 178, pp. 374–383, 2008.
[4] S. Gorbunov and I. Kisel, "Reconstruction of decayed particles based on the Kalman filter," GSI, Darmstadt, Germany, 2007, CBM-SOFT-note-2007-003.
[5] Intel Threading Building Blocks Reference Manual, Intel Corporation.
[6] S. Gorbunov and I. Kisel, "Analytic formula for track extrapolation in non-homogeneous magnetic field," *Nucl. Instrum. Methods Phys. Res. A*, vol. 559, pp. 148–152, 2006.