Advances in tracking and trigger concepts

Ivan Kisel

Goethe University, 60323 Frankfurt am Main, Germany
FIAS Frankfurt Institute for Advanced Studies, 60438 Frankfurt am Main, Germany
GSI Helmholtz Center for Ion Research, 64291 Darmstadt, Germany

Abstract. Increasing beam intensities and input data rates require to rethink the traditional approaches in trigger concepts. At the same time the advanced many-core computer architectures providing new dimensions in programming require to rework the standard methods or to develop new methods of track reconstruction in order to efficiently use parallelism of the computer hardware. As a results a new tendency appears to replace the standard (usually implemented in FPGA) hardware triggers by clusters of computers running software reconstruction and selection algorithms. In addition that makes possible unification of the off-line and on-line data processing and analysis in one software package running on a heterogeneous computer farm.

1. Introduction

Several large international research centers are focused on high-energy physics. Among them are: CERN (Geneva, Switzerland) with ATLAS, CMS, LHCb and ALICE experiments; BNL (Upton, USA) with PHENIX and STAR heavy-ion experiments; and FAIR/GSI (Darmstadt, Germany) with CBM and PANDA future experiments. Focusing on different aspects of high-energy physics and, therefore, having different structures of their detector setups, e.g. of collider or fixed-target types, the experiments confront very similar problems in the processing of experimental data.

In addition to physics governed challenges, a new challenge is brought with modern many-core computer architectures. It requires not only to use new parallel programming languages and frameworks, not only to understand the computer hardware, but in many cases to develop new methods, suitable for parallel data streams, and redesign the traditional methods, which were optimal in the past, now become inefficient running on many-cores.

2. Methods of event reconstruction

Usually the most complicated part of the data processing is the event reconstruction, which consists of four stages: (1) track finding, (2) track fitting, (3) particle identification and (4) search for short-lived particles [1]. To cope with the very large combinatorial procedure of combining one- or two-dimensional hits into five-dimensional particle trajectories, the following methods are used at the track finding stage: conformal mapping, Hough transformation, track following and cellular automaton.

The conformal mapping method (Fig. 1) images the measurement space (x, y) onto the conformal space (u, v) with the transformation:

$$u = x/(x^2 + y^2); v = y/(x^2 + y^2). \quad (1)$$

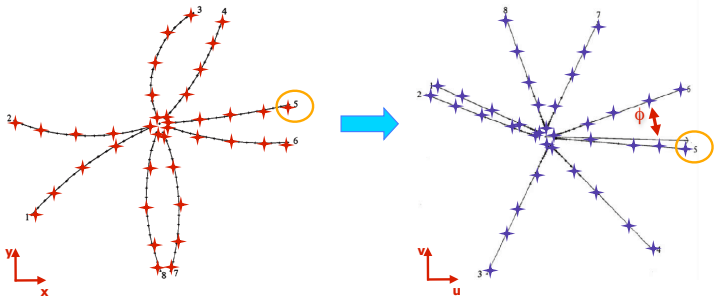

Figure 1. An illustrative representation of the conformal mapping method for particle trajectories in a cylindrical detector with homogeneous magnetic field.

Transforming a circle, which passes through the origin of the coordinate system, into a straight line, the method brings an impressive visual simplification of the problem. This makes finding of particle trajectories in a homogeneous magnetic field extremely simple. Each step of the method is easy to implement in hardware, like in an FPGA. For instance, histogramming angles of the measurements will immediately produce peaks, which correspond to the particle trajectories. Some complications of the method are: one needs to know the interaction point in advance; appropriate for uniform magnetic fields only; provides only rough track parameters estimation; suitable for finding primary tracks, but not secondaries; does not group measurements into track candidates; needs a significant amount of memory for multidimensional histogramming. Thus, the conformal mapping method is useful for implementation in hardware for simple event and trigger topologies.

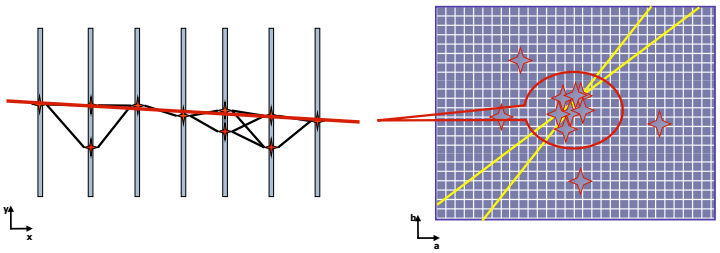

Figure 2. An illustrative representation of the Hough transformation method for a straight line trajectory.

The Hough transformation method (Fig. 2) images the measurement space (x, y) onto the parameter space (a, b) with the transformation:

$$y = a \cdot x + b; \rightarrow b = -x \cdot a + y. \tag{2}$$

Now each measurement (x, y) will determine a straight line in the parameter space. An intersection point of two lines will determine a set of track parameters (a, b). Due to non-precise measurements and effects of multiple scattering, the straight lines will not intersect each other in the same point, but produce a cluster of intersection points, that is easy to localize by histogramming. A situation becomes more complicated when noise hits are present in the detector or when several or many tracks are registered. In this case to avoid unnecessary overfilling of the histogram one can work with pairs of measurements, thus filling the histogram directly with intersection points. The Hough transformation method is a generalization of the histogramming method and is easy to implement in hardware. Some complications of the method: needs a global track model; appropriate for uniform magnetic fields only; does not include effects of multiple scattering; provides track parameters estimations without errors

estimations; does not make competition between track candidates; needs a significant amount of memory for multidimensional histogramming. As a result, similar to the conformal mapping method, the Hough transformation method is useful for implementation in hardware for simple event and trigger topologies.

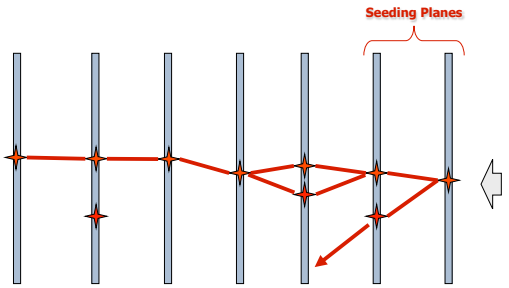

Figure 3. An illustrative representation of the track following method.

The track following method (Fig. 3) implements an intuitive approach to reproduce a particle way, i.e. to move from one measurement to another. One starts from the last detector plane (probably with the least hit density), takes a measurement and assumes that it belongs to a track. Taking another measurement on the next to the last detector plane, one gets a shortest possible track segment, called *track seed*. Further propagation of the track seed from one detector plane to another collects measurements into a track group. If several measurements are found on a detector plane within the propagation region, one creates several *branches* and follows them as separate track candidates. If a detector plane has no measurement within the region of propagation, the current track candidate is deleted, and the procedure is repeated with the next seed (or the next track candidate within the branch). During the track following one can apply the Kalman filter track fitting procedure in order to get simultaneously the estimations of the track parameters and the covariance matrix. Traditionally, development of a new experiment starts with an ideal Monte-Carlo track finder and a realistic Kalman filter track fitter, then it is easy to implement the realistic track finder as track following based on the Kalman filter, thus combining track finding and track fitting. In addition, for a developer it is psychologically easy to accept the hit-by-hit track finding approach. Some complications of the method: it is intrinsically based on a single track assumption; needs the seeding stage; efficiency is limited by the detector efficiency and the seeding efficiency; works at the hit level and forces random access to the hits; after discarding a track candidate can often repeat the same calculations; no global competition between track candidates; limited by the exponential grow of the combinatorial enumeration at high track densities. The track following method is extensively employed in the LHC experiments.

The cellular automaton method (Fig. 4), which can be regarded as a local version of the Hopfield neural network, also creates short track segments, but, contrary to the track following method, it does it between all pairs of detector planes (stage 1). Then it does not work anymore with hits, but with the created segments. Applying a track model (straight line on the illustrative figure) it introduces relations between the segments on neighboring pairs of detector planes in case they have a common hit and a small kink angle. Then it counts consecutively the neighbor segments with their possible position on a track candidate (stage 2). After the tree structure appeared, starting from the last segments with the largest position counters it collects the neighboring segments into track candidates (stage 3). At the last stage the track candidates are fitted with the Kalman filter and selected according to the χ^2-value. Local operations with data bring us to a parallel implementation of the algorithm. The implementation is staged from hits to segments and to track candidates, thus step-by-step collecting the tracking information on the fly. A second order polynomial behavior is observed at high track densities. The track competition stage is naturally included in the algorithm. Some complications of the method:

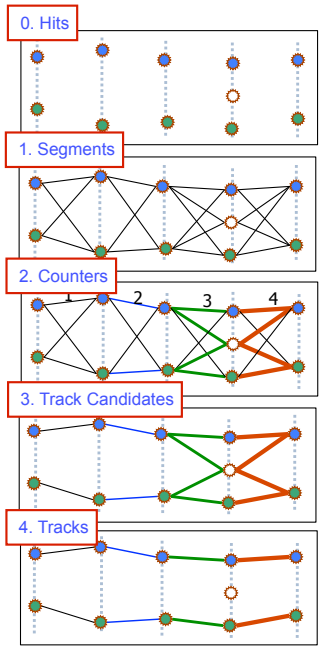

Figure 4. An illustrative representation of the cellular automaton method.

staged algorithm need synchronization after each stage; parallel implementation of the algorithm requires from a developer to have an expert skill in parallel computer architectures, which come into the market only now. The cellular automaton method is successfully used in several heavy-ion experiments.

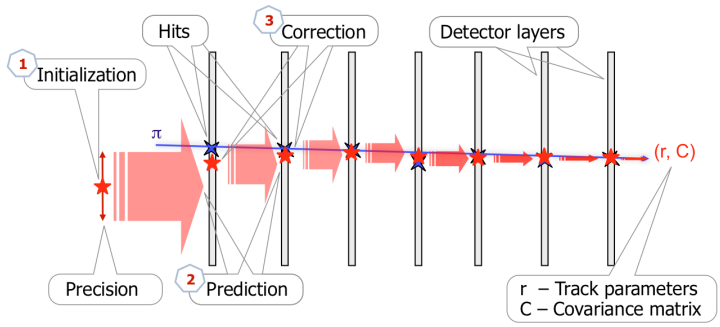

Figure 5. An illustration of the Kalman filter based track fitting algorithm.

Found tracks are then fitted using the Kalman filter method [1]. The Kalman filter (1) starts with an arbitrary initial approximation (Fig. 5); (2) adds hits one after the other; (3) refines the state vector $\vec{r}$ of track parameters and gets the optimal values of the track parameters after the last hit. There is a set of fitting procedures based on the Kalamn filter. The simplest one, described above and called track fitter, one obtains the state vector $\vec{r}$ of track parameters and the covariance matrix C in an outer (first or last) measurement. The Kalman smoother provides the state vector and the covariance matrix in an inner point of the trajectory. For some tracks of high importance for physics analysis one can apply the deterministic annealing filter (DAF), which is based on the Kalman filter and iteratively reduces weights of noise of wrongly associated hits in order to improve the quality of the track parameters estimation.

3. Many-core HPC: cores, threads and vectors

Nowadays, the complexity of event topologies lead to a necessity of using complicated reconstruction algorithms already on-line in the high-level triggers (HLT). In addition, in order to cope with high interaction rates one needs to equip an HLT farm with modern high-performance computing (HPC) hardware (Fig. 6). These are many-core central processing units (CPU) with dozens of cores and graphics processing units (GPU) with thousands of arithmetic units. For efficient use of the many-core hardware one has to implement the algorithms in a parallel manner.

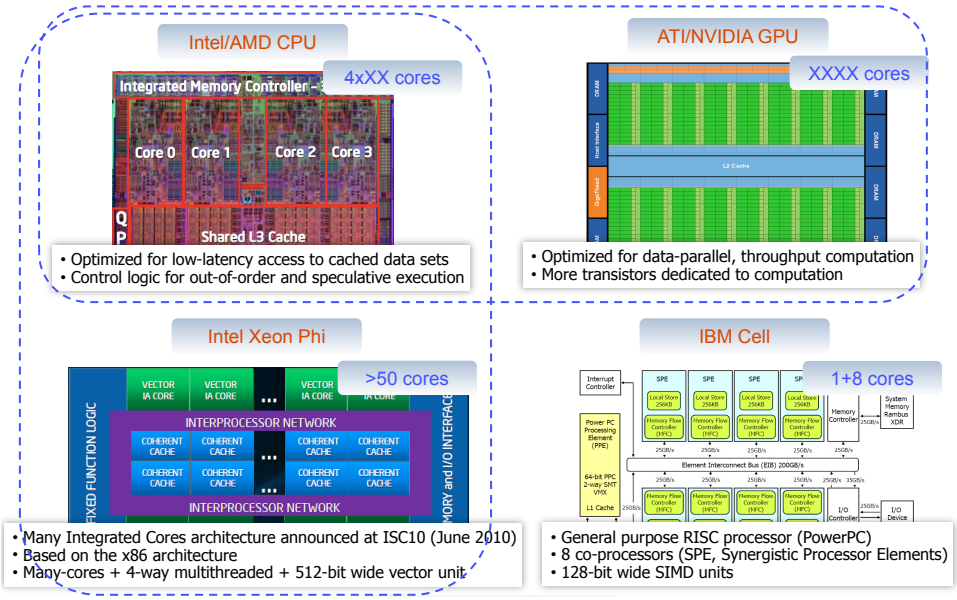

Figure 6. Future HPC systems are heterogeneous.

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 (Single Instruction, Multiple Data) 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:

$$f = 4 \text{ sockets} \cdot 8 \text{ cores} \cdot 1.3 \text{ threads} \cdot 8 \text{ SIMD} \approx 300, \tag{3}$$

which is already equivalent to a moderate computer farm with scalar single-core CPUs.

Porting an algorithm to a parallel hardware requires usually reworking of the algorithm from the ground up. It includes numerical optimization, memory optimization and parallelization, that brings another 10 to 100 or even up to 1000 speed-up factor.

Thus, the HPC architecture makes possible use of complicated reconstruction and selection algorithms already at the HLT level. In addition, on-line and off-line data reconstruction, selection and analysis become very similar or even identical, that provides enormous flexibility and compactness of a computer farm.

4. Parallel programming

Parallel programming is, first of all, a parallel formulation of an algorithm, and only then its implementation in a parallel language. 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.

At the task level parallelism one localizes independent parts of the algorithms and run them in parallel on different cores or threads with or with our synchronization between the processes. Implemented it can be using, for instance, the ITBB or OpenMP frameworks.

If the algorithm allows to organize parallel streams of data, which are processed in the same way, like fit of several tracks, these parts can be SIMDized and run without using an extra hardware, but on vectors within the same threads. For that one can use the auto-vectorization, which is provided by the compilers. Unfortunately, this brings typically (and unpredictably) about 20% of speed-up, which is almost negligible comparing to the potential factor 4/8/16. In order to reach the maximum (depending on the data level parallelism of the algorithm), one can program using the SIMD extensions directly or using the SIMD header files or the much advanced Vc library.

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

The header files provide 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 conventional debugging and testing of the code.

The OpenCL standard provides a higher level of parallel programing by writing a universal code, which can be run on different types of CPU and GPU processing units. Thus, it provides 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 example:

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

5. How to parallelize reconstruction in running experiments

Parallelization is a very non-trivial task, but parallelization of the algorithms in running experiments is a real challenge, to which we have to respond within the next years. In addition to all hardware and software complications described above, the experiment has to run stably and predictably, as this is of vital importance for the physics programme. Different experiments have chosen different strategies for parallelization of the reconstruction software (Table 1), that

ATLAS	Modify the existing code
CMS	Implement a new search algorithm
ALICE	Merge the on-line and off-line codes
STAR	Use an existing algorithm as seed finder
CBM	Develop a new code from scratch

Table 1. Strategies of parallelization in different experiments.

are the most appropriate to provide them a smooth transition to an efficient use of the many-core computer architectures [2].

ATLAS will modify the existing code by numerical and computational optimization, multi-threading and auto-vectorization. The HLT reconstruction software will be gradually integrated with the off-line code.

CMS plans to overcome the current tracking limitations of the combinatorial track following by the parallel implementation the Hough transformation algorithm.

ALICE will use the tracks, found by the cellular automaton track finder in HLT, as seeds for the off-line track following algorithm based on the Kalman filter.

STAR is decided to use the cellular automaton track finder as seed finder for the track following algorithm based on the Kalman filter in off-line and to replace in HLT the Hough transformation algorithm by the same cellular automaton track finder, thus providing mutually reproducible results in on-line and off-line data analysis.

CBM, as a future experiment, profits from the possibility to design and develop a new code from scratch, investigating different approaches. The experiment assumes full identity both in software and hardware between the off-line and on-line reconstruction, selection and analysis of data. Current track finding in CBM is based on the cellular automaton approach.

6. CBM: First Level Event Selection

To demonstrate some typical and specific features of the tracking and trigger concepts, let us consider the current status of the First Level Event Selection (FLES) [3] of the CBM experiment (see a simulated central Au-Au collision on Fig. 7). It is similar to HLT in other experiments, but since in CBM there is no previous stages of data selection, it has a different name in order to emphasize its first (and, actually, the last) level of selection. Thus, FLES combines all traditional stages of triggering.

The FLES package consists of several modules (the block diagram is shown on Fig. 8): 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 [4] using to the registered hits. The Kalman filter (KF) based track fit [5] 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 [6] 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 on-line 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

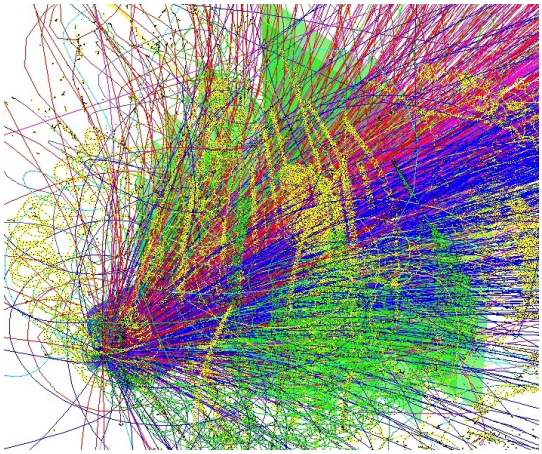

Figure 7. A simulated central Au-Au collision at 25 AGeV energy in the CBM experiment with about 1000 charged particles (different colors correspond to different types of particles).

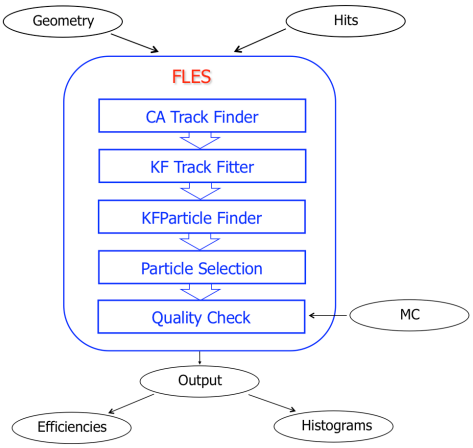

Figure 8. Block diagram of the FLES package.

to be therefore local and parallel with respect to data 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. In order to reflect the specific features of the CBM experiment, the track segments in the track finder are created from hits in each three consecutive detector 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.

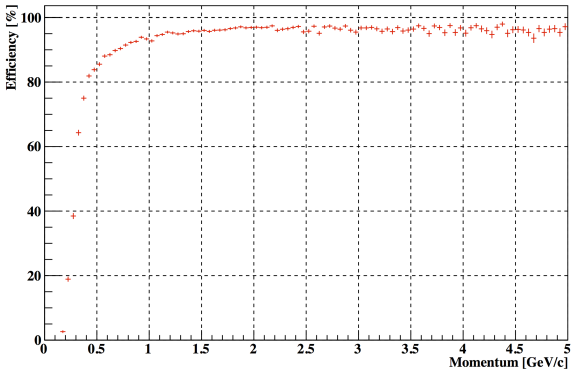

Figure 9. Efficiency of the track reconstruction for minimum bias Au-Au collisions at 25 AGeV.

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. 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 simulated collisions at 25 AGeV is presented on Fig. 9. 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 2. The tests have been performed on a server with Intel Xeon E7-4860 CPUs.

	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

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

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 [5]. 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 non-homogeneous magnetic field is done by an analytic formula, which is based on the Taylor expansion [7]. The analytic formula allows to obtain the 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 μs per track. This is an improvement by a factor 10000 with respect to the original scalar version of the algorithm [5].

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

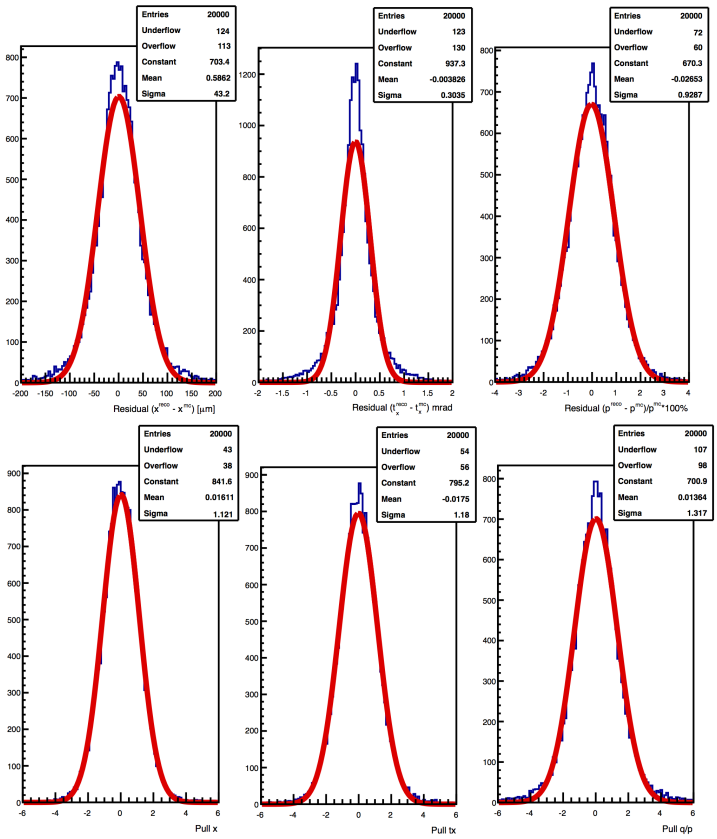

Figure 10. Residuals and pulls distributions of the x (43.2 μm, 1.12) , t_x (0.30 mrad, 1.18) and q/p (0.93 %, 1.32) track parameters.

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 Figure 10 (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 reconstruction efficiency and the low level of the combinatorial background improve significantly the event reconstruction and selection by the FLES package.

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 FLES package has been parallelized with ITBB implementing the event-level parallelism by executing one thread per logical core. Reconstruction of 1000 minimum bias Au-Au UrQMD events at 25 *A*GeV has been processed in 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. 11 shows a strong scalability for all many-core systems achieving the reconstruction speed of 1700 events per second on the 80-cores server.

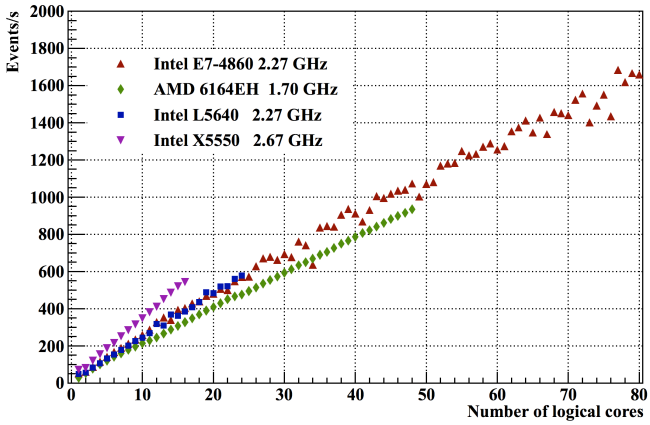

Figure 11. Scalability of the FLES package on many-core servers.

7. Summary

Modern and future many-core computer architectures open new perspectives in tracking and trigger concepts. At the same time, efficient use of the hardware requires much higher computing experience and consolidation of efforts of different experiments.

References

[1] Frühwirth R et al 2000 Data Analysis Techniques for High-Energy Physics. Second Edition, Cambridge Univ. Press

[2] Fourth International Workshop for Future Challenges in Tracking and Trigger Concepts, CERN, 28-30 November 2012

[3] Kisel I, Kulakov I and Zyzak M 2013 Standalone First Level Event Selection package for the CBM experiment *IEEE Trans. Nucl. Sci.*, vol. 60, no. 5, pp. 3703–3708

[4] Kisel I 2005 Event reconstruction in the CBM experiment *Nucl. Instr. and Meth.*, vol. A566, pp. 85–88

[5] Gorbunov S, Kebschull U, Kisel I, Lindenstruth V and Müller W.F.J 2008 Fast SIMDized Kalman filter based track fit *Comp. Phys. Comm.*, vol. 178, pp. 374–383

[6] Gorbunov S and Kisel I 2007 Reconstruction of decayed particles based on the Kalman filter CBM-SOFT-note-2007-003, GSI, Darmstadt

[7] Gorbunov S and Kisel I 2006 Analytic formula for track extrapolation in non-homogeneous magnetic field *Nucl. Instr. and Meth.*, vol. A559, pp. 148–152