Performance analysis of a cellular automaton algorithm to solve the track-reconstruction problem on a multicore...

Article *in* Physics of Particles and Nuclei Letters · March 2013

DOI: 10.1134/S1547477113020088

CITATIONS
2

READS
56

4 authors, including:

Victor V. Ivanov
Joint Institute for Nuclear Research
231 PUBLICATIONS **1,626** CITATIONS

SEE PROFILE

ISSN 1547-4771, Physics of Particles and Nuclei Letters, 2013, Vol. 10, No. 2, pp. 162–170. © Pleiades Publishing, Ltd., 2013.
Original Russian Text © I.S. Kulakov, S.A. Baginyan, V.V. Ivanov, P.I. Kisel, 2013, published in Pis'ma v Zhurnal Fizika Elementarnykh Chastits i Atomnogo Yadra, 2013, No. 2(179), pp. 253–267.

COMPUTER TECHNOLOGIES IN PHYSICS

Performance Analysis of a Cellular Automaton Algorithm to Solve the Track-Reconstruction Problem on a Multicore Server at the Laboratory of Information Technologies, Joint Institute of Nuclear Research

I. S. Kulakov[a, b], S. A. Baginyan[c], V. V. Ivanov[c], and P. I. Kisel[c]

[a]*Goethe University, Frankfurt am Main, Germany*
[b]*Shevchenko National University, Kiev, Ukraine*
[c]*Joint Institute for Nuclear Research, Dubna*

Abstract—The results of tests for the track-reconstruction efficiency and the speed of the algorithm and its scalability with respect to the number of cores of the server with two Intel Xeon E5640 CPUs (in total 8 physical or 16 logical cores) are presented and discussed.

DOI: 10.1134/S1547477113020088

INTRODUCTION

The high-intensity heavy-ion beams that will be produced by the accelerators at the Facility for Antiproton and Ion Research (FAIR), together with the Compressed Baryonic Matter (CBM) experiment, now in preparation, offer outstanding possibilities for studying baryonic matter at superhigh densities and moderate temperatures under laboratory conditions [1]. The CBM physics program is aimed at studying the structure and behavior of baryonic matter at densities comparable to those in the center of neutron stars. The program includes (1) setting a phase boundary between the hadronic and partonic matter, (2) determining the critical end point, and (3) searching for indications of the origin of chiral symmetry reconstruction at high pure baryonic densities.

Figure 1 presents a diagram of the CBM experimental facility intended for muon decay detection. In this figure, immediately behind the target in the superconducting dipole magnet, there is a vertex track detector consisting of two planes of Micro-Vertex Detector (MVD) monolithic micropixel detectors and eight Silicon Tracking System (STS) silicon microstrip detectors. The vertex detector is designed to reconstruct the tracks of charged particles and restore, with maximum accuracy, the secondary vertices at high track density. The information from the vertex detector will also be used to determine the momenta of charged particles to better than 1%. The MUon CHamber (MUCH) muon station includes alternating layers of iron absorbers (to suppress hadrons and extract muons) and coordinate detectors. The MUCH detectors will be used to reconstruct the paths of muons being registered. These tracks will be "connected" with the tracks reconstructed by the vertex detector, making it possible to determine the momenta of the muons being detected. The Transition Radiation Detector (TRD) is meant for (1) track reconstruction and (2) electron and positron identification under a dominating pion background. A time-of-flight measuring system (TOF) is used for the identification of hadrons, while electrons and photons will be identified with an Electromagnetic CALorimeter (ECAL).

The recognition of the paths of charged particles using the vertex detector is the key problem for the task of event reconstruction in the CBM experiment (GSI, Germany). High event multiplicity (up to 1000 secondary tracks in each nucleus–nucleus collision), intense

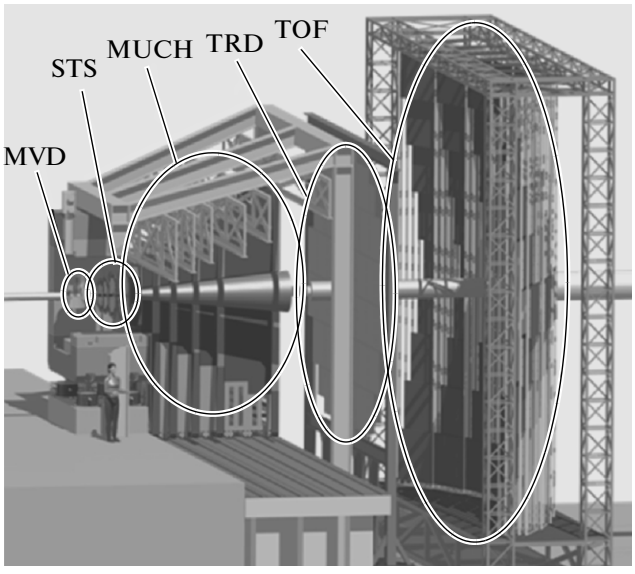

Fig. 1. Diagram of the CBM experimental setup for detecting muon decay.

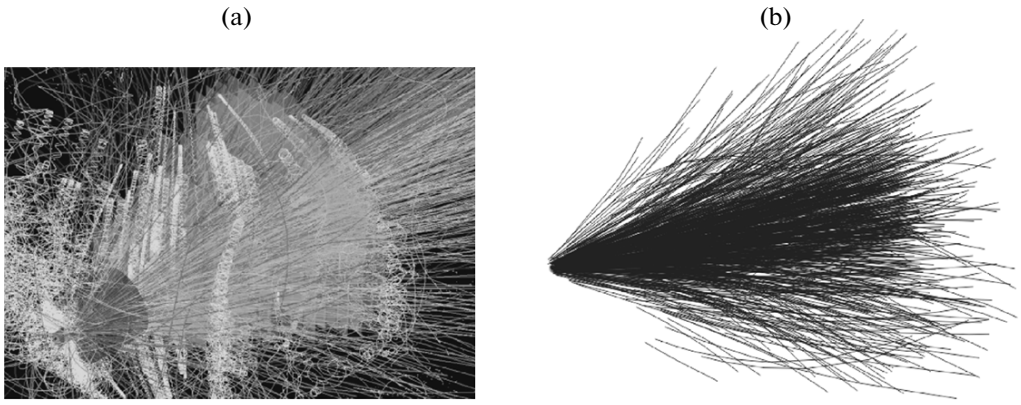

Fig. 2. An example of a typical CBM event generated by the Monte Carlo method for central Au + Au collision at an energy of 25 GeV/nucleon: (a) a generated event in the MVD and STS detectors; (b) a reconstructed event containing 729 tracks.

background (up to 85% background readings in the STS detector), an inhomogeneous magnetic field, and the necessity of reconstructing all events in real time (up to 10^7 events per second) required not only the development of new approaches to solve the problem under study, but also the maximal utilization of the potential of CPU/GPU modern multicore architectures.

As an example, Fig. 2b shows an event that is typical of the CBM and generated by the Monte Carlo (MC) method in the MVD and STS detectors using the GEANTS package [13] in a CBM-ROOT environment [14] for a central Au + Au collision at an energy of 25 GeV/nucleon.

In the present paper, we report the results of testing the track-reconstruction algorithm in the STS detector of the CBM experiment on the Laboratory of Information Technologies, Joint Institute of Nuclear Research (JINR LIT), multicore server with two Intel Xeon E5640 CPUs (in total 8 physical or 16 logical cores). A cellular automaton (CA) and a Kalman filter are at the heart of this algorithm, which allows for parallel event processing both in the optimal use of all the server cores and in data exchange.

1. THE CELLULAR AUTOMATON METHOD FOR SOLVING TRACK-RECONSTRUCTION PROBLEMS

A physical model of the path of a charged particle in the Cartesian coordinate system adopted in the CBM experiment (0 is the origin of coordinates; the OZ axis coincides with the incident direction of particle beam) and with known z_i coordinates of the STS stations is described by a vector of state (x_i, y_i, t_{xi}, t_{yi}, q/p). Here, x_i, y_i are the coordinates of the point where the particle intersects the plane of the ith STS detector (in what follows, these will also be referred to as hits); t_{xi}, t_{yi} are the slopes of the track to the OZ axis; and q/p is the particle charge to momentum ratio. The reconstruction of tracks registered in one event of the CBM experiment involves the selection of hits relating to the paths of individual particles and the formation of the corresponding vectors of state.

A simplified diagram illustrating the operation of the CA algorithm for track reconstruction is shown in Fig. 3. The figure depicts an event in the STS detector comprised of six coordinate stations which contains only two tracks. The CA algorithm includes two successive stages [2–5].

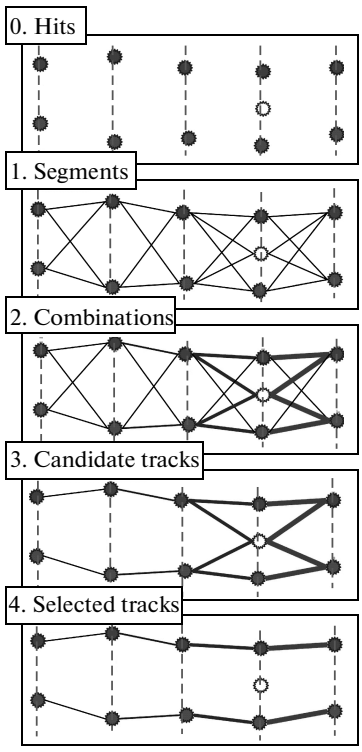

Fig. 3. Simplified diagram illustrating the operation of the CA algorithm for track reconstruction: (0) measured hits, (1) formation of track segments, (2) combination of track segments with the neighboring ones, (3) isolation of candidate tracks, and (4) selection of best candidate tracks.

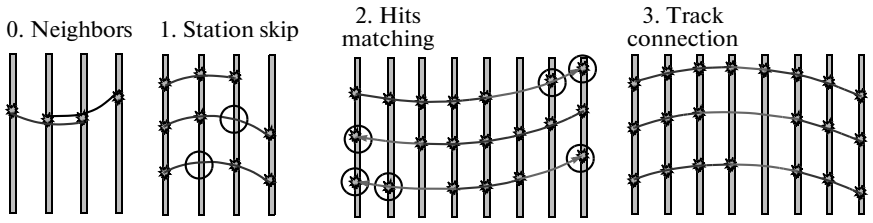

Fig. 4. Diagram illustrating the operation of the CA algorithm for the formation of candidate track: (0) neighboring triplets, (1) three different types of triplets, (2) "matching" of additional hits for the reconstructed candidate track, and (3) combination of candidate tracks.

1.1. First Stage: The Formation of Track Segments (Triplets)

Hits are used as input information in the CA algorithm (see Fig. 3). In the first step, the hits form track segments (triplets) that include the hits of three neighboring STS stations. In Fig. 3 this procedure is readily illustrated by a simplified model of a track segment containing only two tracks (1 in Fig. 3).

To minimize the search of hits at the formation of triplets, both geometrical and physical constraints imposed by the experiment are used. For instance, with CM simulation, the CBM experiment has established that the great majority of registered particles escape from the region of the primary vertex (the point of interaction between the beam and the target) at small angles to the incident beam direction. Moreover, the particles carrying useful information on the physical processes going on in nucleus–nucleus collisions have momenta higher than 0.1 GeV/c. Because of this, particles with momenta less than 0.1 GeV/c are excluded from consideration.

Every triplet contains hits from three neighboring STS stations. Because one hit includes (x, y) coordinates of the point where the particle intersects the plane of a particular station, the hits from the three STS stations form a set of six measurements, making it possible to uniquely determine all five parameters that describe the track element being considered, as well as estimate χ^2 deviations of the real measurements from the reconstructed track. Further analysis showed that the track χ^2 estimate obtained in this manner makes it possible to reject most random triplets.

1.2. Second Stage: The Formation of Candidate Tracks

In the second stage, for all triplets formed for an event being examined, the track segments are combined with the neighboring track segments to construct candidate tracks (see Fig. 3, 2–4).

Going from left to right, the CA algorithm first finds for each triplet its neighbors (triplets that, in accordance with the track model, may be continuations of the track being considered). The neighboring triplets must have two common hits and momenta moduli coinciding within the limits of errors (see Fig. 4, 0).

As neighbors are being established for each triplet, the so-called level of the triplet is formed (a quantity equal to the number of triplets adjacent to the one under consideration from the left and forming a continuous chain of triplet neighbors). The level of a triplet locates it on the track in the chain of neighbors: the greater the value it takes, the longer the track this triplet can construct.

Knowledge of triplet levels allows us to quickly and easily bind them together into full-valued candidate tracks. Beginning with the triplet of the maximum level, we connect it to the neighboring ones with a level rigorously less by one. The latter are in turn also connected with the corresponding neighbors. We continue with this procedure until we reach triplets having no neighbors to the left. A treelike structure of potential candidate tracks is formed in this way, from which the best track is selected using the χ^2 criterion.

Therefore, a candidate track is constructed for every maximum-level triplet. All the candidate tracks are then analyzed by pairs to reveal common hits in them. When the candidate tracks have a common hit, preference is given to the candidate with the least χ^2. The procedure of the final selection of the reconstructed tracks is based on sorting by the χ^2 criterion. Thereafter, starting with the best, the candidate tracks are denoted as reconstructed ones and all the hits related to these tracks are marked as used. If a regular candidate track contains used hits, it is rejected. After all candidates with maximum length have been considered, the same procedure is applied to candidate tracks containing fewer neighbors, and so on.

The two-step procedure described above is applied to every event in three passes: (1) the search for tracks from fast ($p > 0.5$ GeV/c) quasi-primary (emitted from the target region) particles, (2) the search for tracks from slow ($0.1 < p < 0.5$ GeV/c) quasi-primary particles, and (3) the search for tracks from secondary particles with arbitrary momentum. After each pass, all measurements pertaining to the tracks found are tagged and excluded from further consideration. With such an approach, a successive slackening of the criteria in the search for tracks causes the number of hits to

fall. This provides a fast and reliable track reconstruction at high event multiplicities and measurement densities in the CBM vertex detector.

During the construction and selection of candidate tracks, some limitations are used that allow a substantially reduced number of wrongly reconstructed tracks. To allow for the inefficient operation of some stations of the vertex detector, the CA algorithm includes additional possibilities (Fig. 4, 1, 2, 3): (1) specifying the triplets, making it possible to skip one inefficiently working station; (2) the extrapolation of candidate tracks through inefficient stations and the collection of additional hits; and (3) the connection of the particle-track elements reconstructed independently of each other.

Figure 2 shows the result of processing an MC-generated event for a central Au + Au collision at an energy of 25 GeV/nucleon: the reconstructed event (b) contains 729 tracks.

2. THE KALMAN FILTER METHOD IN THE PARTICLE TRACK-RECONSTRUCTION PROBLEM

The approximation (fitting) of measurements of the charged particle track by the least-square technique (global fitting) in ideal conditions, namely, with a homogeneous magnetic field and with no multiple scattering, yields the best estimate for the parameters of the model of the particle path. However, the situation drastically changes in the case of an inhomogeneous magnetic field and multiple scattering, which contributes substantially to the measurement error:

$$\xi = \xi_{\text{er}} \cup \xi_{\text{ms}}, \tag{1}$$

where ξ_{er} is the measurement error of the detector; ξ_{ms} is the error due to multiple scattering.

In this case it is convenient to take advantage of the well-proven Kalman filter, which is used to find an optimal estimate of the parameters for a linear discrete dynamic system [7, 8] describing the process under study. In our case, the track fitting procedure may be presented as an evolution process of the indicated system [9], where the vector of true values of the track-model parameters $\mathbf{p}_k = (x^k, y^k, t_x^k, t_y^k, q/p)^T$ in the track passing through the kth detector acts as a system state vector in the kth moment of time, while the hits registered by the kth detector act as a system-measurement vector:

$$\mathbf{m}_k = (x^k, y^k, z^k)^T. \tag{2}$$

Thus, the track is represented as a discrete linear dynamic system described by the system evolution equation

$$\mathbf{p}_k = F_k \mathbf{p}_{k-1} + \omega_k, \tag{3}$$

and the equation for the projection of the space of the track-model parameters into the space of measurements

$$\mathbf{m}_k = H_k \mathbf{p}_k + \xi_k. \tag{4}$$

Here, F_k is the so-called transition matrix, which describes the relation between the parameters of the track model in the $(k-1)$th detector and the parameter values in the kth detector; H_k is the matrix of projection of the track model into the measurement space. In our case, the strip location and matrix $H_k = \{\sin\phi, \cos\phi, 0, 0, 0\}$, where ϕ is the strip angle to the OX axis and ω_k and ξ_k are errors introduced into the process by multiple scattering and the measurement system of the kth detector, respectively, is used as the measurement. The random vectors ω_k and ξ_k are independent of each other and have a normal distribution $N(0, Q_k)$ and $N(0, V_k)$, where Q_k and V_k are covariant matrices of vectors ω_k and ξ_k, respectively.

Let us introduce the following notations:

(i) $\tilde{\mathbf{p}}_k$ is an estimate for the vector of system parameters at the kth step; in fact, it is a mathematical expectation of the vector $\mathbf{p}_k$, provided that k measurement vectors are already known, $E(\mathbf{p}_k|\mathbf{m}_k, \mathbf{m}_{k-1}, \ldots, \mathbf{m}_1)$;

(ii) $\tilde{\mathbf{p}}_k^{k-1}$ is the predicted value of the system parameter vector at the kth step when the estimate of the parameter vector at the $(k-1)$th step is known: $\tilde{\mathbf{p}}_k^{k-1} = F_k \tilde{\mathbf{p}}_{k-1}$;

(iii) $C_k = \text{cov}(\tilde{\mathbf{p}}_k - \mathbf{p}_k)$ is the covariant matrix of error in the estimate of the system parameter vector at the kth step;

(iv) $C_k^{k-1} = \text{cov}(\tilde{\mathbf{p}}_k^{k-1} - \mathbf{p}_k)$ is the covariant matrix of error in the predicted vector.

The Kalman filter procedure makes it possible to obtain the estimated value for the system parameter vector at the kth step, $\tilde{\mathbf{p}}_k$, based on the estimate of the parameter vector at the $(k-1)$th step, $\tilde{\mathbf{p}}_{k-1}$, and measurement vector $\mathbf{m}_k$ at the kth step.

The basic formulas we need (in the sequence given below) to construct the Kalman filter algorithm are as follows:

(1) $\tilde{\mathbf{p}}_k^{k-1} = F_k \tilde{\mathbf{p}}_{k-1}$;

(2) $C_k^{k-1} = F_k C_{k-1} F_k^T + Q_k$;

(3) $K_k = C_k^{k-1} H_k^T (H_k C_k^{k-1} H_k^T + V_k)^{-1}$, where K_k is the Kalman transformation matrix, which transfers error from the space of measurements to the parameter space;

(4) $\tilde{\mathbf{p}}_k = \tilde{\mathbf{p}}_k^{k-1} + K_k(\mathbf{m}_k - H_k \tilde{\mathbf{p}}_k^{k-1})$ is just the basic formula of one-step Kalman predictor;

(5) $C_k = (I - K_k H_k) C_k^{k-1}$, here I is unity matrix;

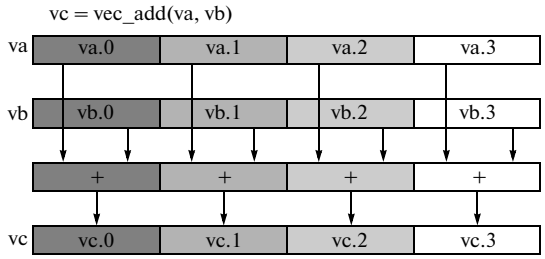

Fig. 5. Illustration of the simultaneous execution of one operation on four numbers at once.

(6) $\chi^2_k = \chi^2_{k-1} + (\mathbf{m}_k - H_k\tilde{\mathbf{p}}_k^{k-1})^T \times (H_k C_k^{k-1} H_k^T + V_k)^{-1} (\mathbf{m}_k - H_k\tilde{\mathbf{p}}_k^{k-1})$.

The Kalman filter represents an iteration process, and to initiate it, starting from the available a priori information and/or MC simulation performed, initial values are determined for $\tilde{\mathbf{p}}_0$ and C_0. The following cycle is then organized:

(1) $\tilde{\mathbf{p}}_k^{k-1} = F_k\tilde{\mathbf{p}}_{k-1}$;

(2) $C_k^{k-1} = F_k C_{k-1} F_k^T + Q_k$;

(3) $K_k = C_k^{k-1} H_k^T (H_k C_k^{k-1} H_k^T + V_k)^{-1}$, here K_k is the Kalman converter;

(4) $\tilde{\mathbf{p}}_k = \tilde{\mathbf{p}}_k^{k-1} + K_k(\mathbf{m}_k - H_k\tilde{\mathbf{p}}_k^{k-1})$;

(5) $C_k = (I - K_k H_k) C_k^{k-1}$

over all STS stations, beginning with the first and up to the last one: $k = 1, \ldots, n$.

After completing the calculations with the Kalman filter, at the algorithm output, we have an optimal approximation of charged particle track in the inhomogeneous field of the dipole magnet along with reconstructed momentum.

3. COMPUTER TECHNOLOGY AND PARALLEL COMPUTATIONS

Present-day computer technology permits for simultaneously performing many operations, i.e., paralleling computations within one computing environment: a supercomputer, multiprocessor/multicore server, and even an individual PC or notebook. These offer a means for considerably (10, 100, and more times) reducing the astronomical time needed to do certain time-consuming computational algorithms.

At present there are three main classes of computer architectures that fulfill different principles of performing operation and data exchange:

(1) Single Instruction, Single Data (SISD) stream, one instruction for one scalar quantity; all operations are executed sequentially.

(2) Single Instruction, Multiple Data stream (SIMD), one instruction for many data; used for the simultaneous execution of one operation with many data.

(3) Multiple Instruction, Multiple Data (MIMD) many instructions for many data; several instructions for many data are executable simultaneously.

Modern computers are supported by multithreading and hyperthreading technology, which enables the computational process to be immediately distributed among several virtual processors (cores) and the computational potential of each core to be used to best advantage, which is equivalent to utilizing two logical cores on one physical core.

3.1. SIMD Architectures

The key element of SIMD architectures is a vector register, which permits a simultaneous (parallel) execution of operations on several numbers (data). Therefore, computations with SIMD architectures are often called vector ones. The starting data on which operations are executed are represented by words divisible by one byte; they most often contain 4 bytes. Before any operation on numbers, they should be entered (packed) in the vector register [6, 10]. One of the main characteristics of the SIMD architecture is the number of words processed simultaneously. In particular, with a 28-bit vector register, one can carry out operations simultaneously on four 32-bit real numbers at once with ordinary accuracy (Fig. 5).

SIMD architectures are used in almost all existing processors. In Intel processors, SIMD architectures for operation with integers and floating-point numbers are realized in MultiMedia eXtension (MMX) and Streaming SIMD Extensions (SSE) technology.

MMX technology is a set of SIMD instructions developed by Intel and brought to fruition in 1997 on Pentium MMX microprocessors. In the MMX microprocessors, eight new vector registers, each being a 64-bit integer, are added to the architecture of the preceding processor.

The SSE technology was introduced in 1999 in Pentium III processors. The SSE adds eight new 128-bit registers, known as XMM registers. Every register contains four 32-bit floating-point numbers with ordinary accuracy. The SSE2 technology introduced into Pentium IV added to these very 128-bit XXM registers new mathematical instructions for real numbers with double accuracy (64 digits) and 8-/16-/32-bit integer numbers. Therefore, the SSE2 enabled programmers to apply mathematical operations on practically all numbers in SIMD architectures (from 8-bit integer to 64-bit real number). The SSE3 technology is a further development of SSE2: DSP oriented mathematical instructions are added, such as the addition and subtraction of many numbers that are kept in one vector register. The SSE4 technology is a fourth-generation SSE2 designed for new microarchitectures developed by Intel.

There are four different versions of SSE vector registers: (1) 168-bit words with or without a sign, (2) 816-bit words with or without a sign, (3) 432-bit integer numbers, and (4) 432-bit floating-point numbers.

The SEE set includes cache control instructions, which make it possible to improve the cache efficiency when working with data flows. Starting with the SSE2 technology and up, one can operate with 64-bit real numbers with double accuracy, as well as exchange data between the vector registers and base memory.

The track-recognition problem uses SIMD technology in the stage of the construction of triplets and their approximations. This turned out to be possible because every triplet is processed independently of the other triplets and the processing is performed by one algorithm: the Kalman filter. The parameters of track sections and measurements are packed up into relevant SIMD format and then specially developed functions are applied to them, which occur parallel to all stages of the Kalman filter (correction of the track parameters when the next measurement is included; extrapolation; and taking account of multiple scattering).

The application of SIMD architectures to the track-reconstruction problem allows one to compute with the Kalman filter about four times faster and with the cell automaton algorithm about two times faster.

3.2. Multithreading

Multithreaded programming has long been used to speed up the operation of the programs. In doing so, certain computations are separated into a particular thread. The thread takes its own registers of the processor and has its own stack (part of the memory containing the address for the return to subprocedures and local changes in these subprocedures). In this case, the address space of the thread being considered is common to other threads of the processor. The threads are started independently of one another and, like the processes in the operating system, none of the threads knows about the existence and state of the others [11]. However, by using common memory, one can ensure that all threads work efficiently on one and the same problem.

Another important benefit of distributing one task among several threads is the possibility of avoiding dead time in readout–recording operations. Reading out of the base memory is a very time-consuming procedure that may take up to several hundreds of steps; in this case, the program stops the computations and waits for the necessary information. Therefore, the processor may be idle for quite a long time. However, having started a few threads simultaneously, we can save all registers for the thread being now involved in some other operations (for instance, reading out of the base memory) and pass on to computations for another thread.

To use the described approach, the track-reconstruction algorithm was parallelized in part of the search for triplets. For this purpose, the algorithm sections answering particular stages of triplets search, namely, finding one-/two-/three-readout fragments, were isolated from the common cycle and divided into three respective parts which are performed in succession. Each of the indicated parts includes a cycle for STS detectors. Particular iterations of these cycles were made independently, enabling their simultaneous initiation in several threads. A search for triplets takes up 84% of the algorithm operation time; therefore, theoretically (using a large number of processors), parallelizing allows the execution of the algorithm to happen about six times faster.

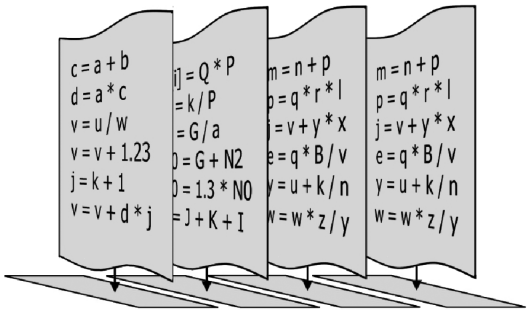

Fig. 6. Illustration of the simultaneous execution of operations using multithreading technology.

However, another approach proved to work better: it looks for tracks in individual events on several threads, with only one thread intended for one event. To execute the track-recognition programs using multithreading technology, a set of procedures from the Intel Threading Building Block (ITBB) library was used [12].

4. DISCUSSION OF THE RESULTS

The main quantity that characterizes the operation of the track-reconstruction algorithm is its efficiency. To assess this efficiency, we introduce the notion of the found track of the particle (the reconstructed track) and the notion of the particle that can potentially be found (the reconstructible particle). The part of the set of the potentially reconstructible particles which is found by the algorithm is called the search algorithm efficiency. The set of potentially reconstructible particles is defined by the problems and acceptance of the detector in which the search is carried out. Under the conditions of central collisions at energies of about 25 GeV/nucleon in the CBM detectors MVD and STS, it is impossible to distinguish combinations of three readings of one particle from those accidentally located along a path similar to the physical one. What is more, the number of the latter combinations is a few orders of magnitude larger than the number of the former ones. A similar problem occurs for tracks with a momentum of 0.1 GeV/c: these have a large curvature and are emitted from the sensitive volume of the detector, passing few stations. Because of this, to estimate the track-reconstruction algorithm, it was

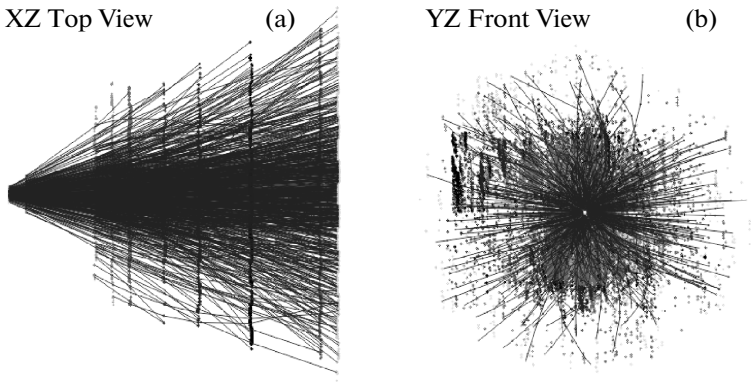

Fig. 7. Hits and reconstructed tracks for one central Au + Au collision at an incident nucleus energy of 25 GeV/nucleon: the number of reconstructed tracks is equal to 729. (a) Top view and (b) front view. Gray points denote the hits corresponding to the places of passage of a reconstructible particle; black points are background hits. The reconstructed tracks are shown with blue polygonal lines (a color version of the figure is available at www1.jinr.ru).

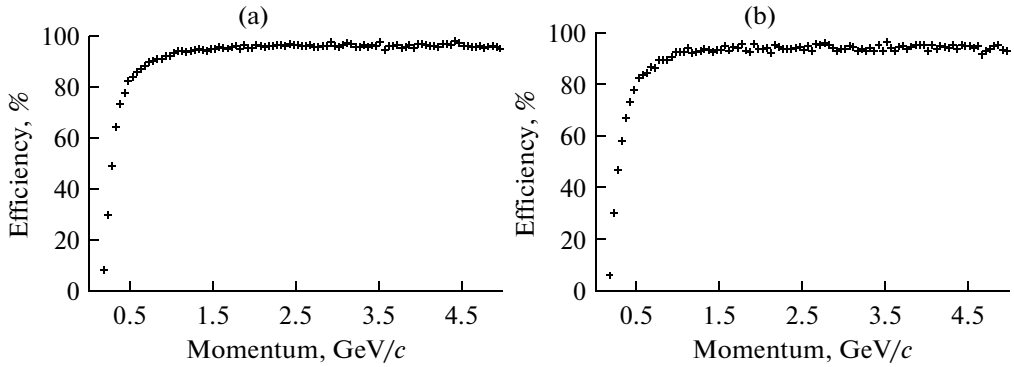

Fig. 8. Track-reconstruction efficiencies as a function of particle momentum for mixed (a) and central (b) events.

decided to treat only those particles that intersect four or more detector stations, as well as have a momentum above 0.1 GeV/c, as potentially reconstructible.

The algorithm was tested by reconstructing 100 MC-generated Au + Au central collisions at an incident nucleus energy of 25 GeV/nucleon, which corresponds to the most intricate scenario (because of the number of produced secondary particles) for the track-reconstruction problem. As an example, Fig. 7 shows the result of a reconstruction for one central event.

It should be noted that the vast majority of nucleus–nucleus collisions in the CBM experiment will be peripheral. According to the estimates made, the portion of the central events will be within 1%. In connection with this, we have also generated and reconstructed 1000 events with a random impact parameter. In what follows, we shall refer to such (minimum bias) events as mixed ones.

Reconstruction efficiencies for different categories of tracks, level of clones and ghosts, number of reconstructed tracks, and algorithm execution times

Parameter	Mixed	Central
Efficiency, %		
Fast primary	97.4	95.5
Slow primary	88.1	86.7
Fast secondary	81.0	78.3
Slow secondary	47.9	45.1
All tracks	88.1	86.3
Level, %		
Clones	0.3	0.4
Ghosts	1.6	4.4
Number of reconstructed tracks	153	717
Time per event, ms	25	220

4.1. Efficiencies of Track Reconstruction

The track-reconstruction efficiencies according to the momentum of the particle being registered for mixed and central events are presented in Fig. 8 and the table.

For a sophisticated treatment of the operation of the track-reconstruction algorithm, all particles and tracks were separated into several categories. First, the MC-generated particles were divided: (1) according to

momentum into fast ($p > 1$ GeV/c) and slow ($0.1 < p < 1$ GeV/c); (2) according to production place into primary, which correspond to particles produced in a primary vertex, and secondary, which resulted from the decay of short-lived particles. The reconstructed track is associated with some particle when no less than 70% of the track readings were formed by interaction of this particle with the detector. A particle is treated as found if it was associated with at least one reconstructed track. When more than one reconstructed track is assigned to one particle, all additional tracks are considered clones. A reconstructed track is considered found incorrectly and called ghost when no real particle corresponds to it.

Most (signal) tracks of interest—products of the decay of D mesons, J/ψ-, ψ'-, φ-, and ρ particles—are high-energy and emerge from the region of a primary vertex, so the efficiency of their search is defined by the category of fast "primary" particles. The share of such particles in mixed events constitutes 97%. The secondary fast tracks may result from decays of K_s^0, Λ, and from cascade decays of Ξ and Ω. Such tracks are produced away from the target; they intersect fewer stations and the position of the target cannot be used as additional information for them. As a consequence, their search efficiency is slightly less than for "primary" particles (81%). The secondary low-energy particles may emerge from decays of Ξ and Ω hyperons. The track reconstruction of slow primary particles is important for reducing the background level in the reconstruction of decays of vector mesons and J/φ by the electron channel. Electrons resulting from Dalitz decays of (π^0) and γ are one component of such a background. Multiple scattering in the detector material and the large curvature of the path also lead to additional losses on the reconstruction of slow particles: the efficiencies for primary and secondary particles are, respectively, equal to 88% and 48%.

The track-reconstruction efficiency averaged over all the particle categories amounted to 88%, with the share of clones 0.3% and ghosts 2%.

The central events are much superior to the mixed ones both in the number of produced particles and in reading density in the STS detector. As a result, they are characterized by lower efficiencies of correctly reconstructed tracks and larger percent of incorrectly found tracks. The track-reconstruction efficiency averaged over 100 central events comprised 86%; the share of ghosts ran as high as 4%.

4.2. Scalability of the Track-Reconstruction Algorithm

As noted above, in the CBM experiment it is planned to perform a total reconstruction of events in real time. Therefore, the speed of execution of the algorithm for reconstructing the tracks registered in the STS detector is important, as is the algorithm efficiency. At present, in the development of special-purpose high-performance servers, the trend is towards an increase in number of processors, as well as towards an increased number of nuclei at a constant clock frequency. In connection with this, it is essential that the event processing speed should increase in proportion to the increasing number of nuclei; in our case, this is the speed of track reconstruction.

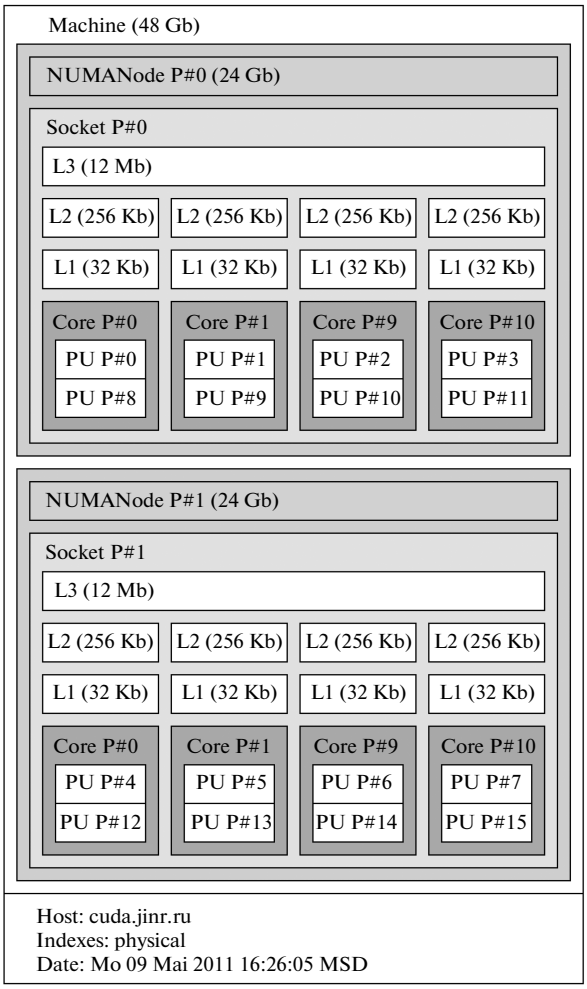

Fig. 9. Structure of the server cuda.jinr.ru: distributions of the random access memory among the processors (pink) and cache memory for the first, second, and third levels (white). Two logical cores correspond to each physical core (dark gray) (see the colored version of the figure at www.ljinr.ru).

An eight-core server cuda.jinr.ru (LIT JINR) is used for testing the time characteristics of the algorithm (Fig. 9). The server has two Intel Xeon E5640 CPUs, each including four cores with a frequency of 2.66 GHz and 12-Mbyte third-level cache memory. The random access memory of 45 Gbyte is distributed equally between the two processors. Hyper-threading permits using each physical core in the form of two logical cores.

Average times spent on one event with the algorithm started on one core of the server are shown in the table. These turned out to be equal to 25 and 220 ms for mixed and central events, respectively.

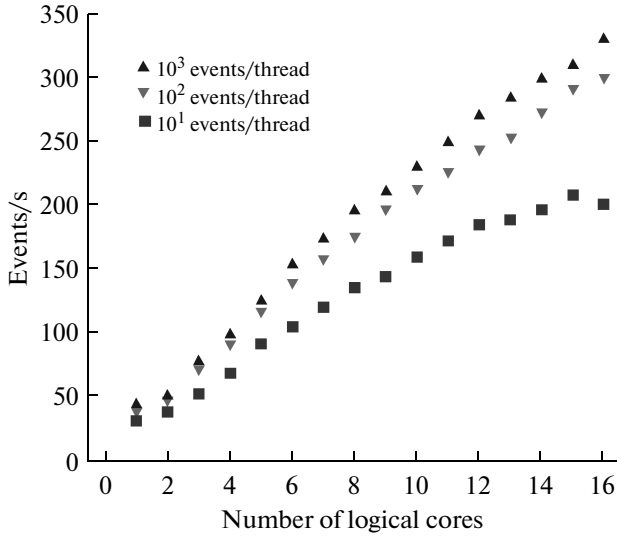

Fig. 10. Scalability of the track-reconstruction algorithm for mixed events.

The problem of track reconstruction was run in parallel on logical cores varying in number. Each core performed track reconstruction for 10, 100 and 100 mixed events. The library of Intel Threading Building Blocks (TBB) was used to organize parallel computations involving many cores. The obtained results of track scalability are given in Fig. 10. It is seen that, for a small number of events, extra expenses (overheads) are needed for the distribution of processes among the cores. The groups with larger numbers of events show the required linear scalability of the algorithm.

CONCLUSIONS

Based on the simulated events generated with GEANT3 [13] in the CBM-ROOT environment [14], efficiency and performance have been estimated for the algorithm of reconstructing the paths of charged particles on the multicore server at the LIT JINR. The track-reconstruction algorithm has its origins in the cellular automaton concept and Kalman filter method. Despite the high event multiplicity, intense background, and inhomogeneous magnetic field, the algorithm has shown a high efficiency of track reconstruction (96–97%) at a low level of incorrectly found tracks (2–4%). High-speed event processing with one core has been achieved, which on average is equal to 220 ms per one central event and 25 ms per one mixed event. It has been shown that the number of the processed events grows almost linearly as the number of nuclei involved in processing increases.

ACKNOWLEDGMENTS

The authors are grateful to I.V. Kisel for consultations and useful discussions and D.V. Belyaev and A.M. Raportirenko for technical support and assistance in conducting studies on the cuda.jinr.ru server.

REFERENCES

1. B. Friman, et al., "The CBM Physics Book," Lect. Notes in Phys. **814**, 980 (2011).
2. I. Kisel, "Event Reconstruction in the CBM Experiment," Nucl. Instrum. Methods Phys. Res., Sect. A **566**, 85–88 (2006).
3. M. P. Bussa, et al., "Application of a Cellular Automaton for Recognition of Straight Tracks in the Spectrometer DISTO," Comp. Math. Appl. **34** (7/8), 695–701 (1997).
4. M. P. Bussa, et al., "Application of CA and NN for Event Recognition in Experiments DISTO and STREAMER," Nucl. Instrum. Methods Phys. Res., Sect. A **389**, 208–209 (1997).
5. S. Gorbunov and I. Kisel, "Analytic Formula for Track Extrapolation in Non-Homogeneous Magnetic Field," Nucl. Instrum. Methods Phys. Res., Sect. A **559**, 148–152 (2006).
6. S. Gorbunov, et al., "Fast SIMDized Kalman Filter Based Track Fit," Comput. Phys. Commun. **178**, 374–383 (2008).
7. A. V. Balakrishnan, *Kalman Filtering Theory* (Mir, Moscow, 1988) [in Russian].
8. R. E. Kalman and R. S. Bucy, "New Results in Linear Filtering and Prediction Theory," J. Basic Engineering **82**, 35–40 (1960).
9. "Application of Kalman Filtering to Track and Vertex Fitting," Nucl. Instrum. Methods Phys. Res., Sect. A **262**, 444–450 (1987).
10. IA-32 Intel Architecture Optimization reference Manual, Intel (2005).
11. J. Reinders, Intel Threading Building Blocks. O'Reilly Media Inc. (2007).
12. Intel Threading Building Blocks, http://www.threadingbuildingblocks.org.
13. GEANT - Detector Description and Simulation Tool. CERN Program Library, Long Write-up W5013. 1995.
14. D. Bertini, et al., "The FAIR Simulation and Analysis Framework," Proc. of Intern. Conf. on Computing in High Energy and Nuclear Physics, Victoria, BC, Canada, 2007.

SPELL: 1. parallelized, 2. parallelizing