EPJ Web of Conferences **226**, 03013 (2020) https://doi.org/10.1051/epjconf/202022603013
Mathematical Modeling and Computational Physics 2019

High-Performance Optimization of Algorithms Used in the BM@N Experiment of the NICA Project

Sergei Merts[1,⋆], *Sergei* Nemnyugin[2,⋆⋆], *Vladimir* Roudnev[2], and *Margarita* Stepanova[2]

[1]*Joint Institute for Nuclear Research, Joliot-Curie 6, 141980 Dubna, Moscow Region, Russian Federation*
[2]*Saint-Petersburg State University, University emb. 7–9, 199034 Saint-Petersburg, Russian Federation*

Abstract. Results of high-performance optimization of BmnRoot software modules are presented. The BmnRoot package used in the BM@N experiment of the NICA project plays a crucial role in the simulation and event reconstruction so its performance should be maximized to make the data processing efficient. Results of performance analysis on representative testcases are given and bottlenecks are localized. Most suitable approaches to BmnRoot optimization are chosen and numerical estimates of the scalability of the parallelized modules for event reconstruction are presented.

1 Introduction

The BmnRoot software package is used in the BM@N experiment [1] of the NICA project to solve a great many different tasks. The main problems to be solved and the logical structure of the package are shown in Fig. 1. Both simulation and track reconstruction problems may be solved by running multiple independent modules with different typical times of execution (Fig. 2) [2]. For example, the event reconstruction may take as long as several seconds per event, depending on the type of the colliding particles, the beam energy, the collision centrality and other parameters. Event simulation with realistic Monte-Carlo generators is also time-consuming. Processing the tens of millions of events may take significant time. Very large samplings must be produced by event generators to get reliable results, so any kind of performance-oriented improvement not only of the particles beam control [3], but of the simulation and reconstruction algorithms and their implementation is of the utmost importance.

A systematic approach to the performance-oriented optimization should take into account various aspects of the problem: 1) availability of a high-performance computing platform; 2) appropriate computational models; 3) the choice of efficient algorithms; 4) optimal software implementation; 5) usage of high-performance software libraries; 6) careful tuning of compiler optimizations; 7) optimization based on dynamic analysis of the application; 8) employing parallel programming techniques. The present study is devoted to the performance analysis and optimization of the algorithms used in the BM@N experiment of the NICA project and is based on some of the above mentioned aspects.

2 Performance bottlenecks of the BmnRoot software

The complexity of the BmnRoot package, the variety of the execution paths and their dependence on input parameters makes the dynamic performance analysis a necessity. For this purpose an instrumen-

⋆e-mail: sergey.merts@gmail.com
⋆⋆e-mail: s.nemnyugin@spbu.ru

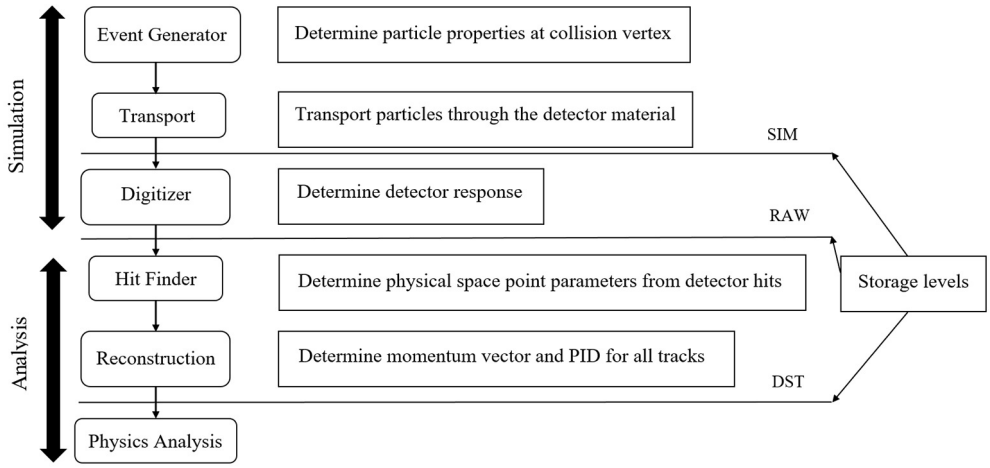

Figure 1. Logical structure of the BmnRoot software

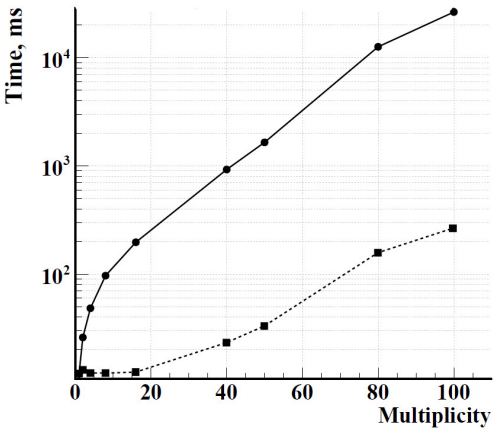

Figure 2. Time consumed by the BmnRoot: 1) • per event; 2) ■ per track reconstruction

tation of source or binary files by functions that have access to hardware or system counters should be performed. In our study the performance analysis has been done using three approaches:

- direct timing for some modules of the BmnRoot package which is implemented by insertion calls of standard timers in the source code;
- usage of Google Performance Tools [4] for automatic localization of the most time consuming functions (“hotspots” of the program);
- dynamic analysis of the BmnRoot modules by other software tools.

The results which have been obtained with all three approaches are consistent with each other.

The analysis used the following testbenches and testcases.

Testbench 1. CPU: Intel(R) Core(TM) i5-2400 @ 3.10GHz (4 core, no hyperthreading). RAM: 16 Gigabytes. OS: Linux (Ubuntu).

Table 1. Hotspots of the BmnRoot simulation modules

Function and/or module	Time, sec
sincos	399
Trandom::Gauss	369
DeadZoneOfStripLayer::IsInside	365
TRandom3::Rndm	232
deflate	167
BmnGemStripModule::AddRealPointFull	121

Table 2. Hotspots of the BmnRoot reconstruction modules

Function and/or module	Time, sec
BmnCellAutoTracking::CellsConnection	239
inflate	48
BmnKalmanFilter::RK4Order	22
BmnNewFieldMap::FieldInterpolate	17
BmnNewFieldMap::IsInside	12
BmnKalmanFilter::TransportC	10

Testbench 2. CPU: Intel Xeon E-2136 @ 4.5GHz Turbo (6 cores with hyperthreading). RAM: 32 Gigabytes. OS: Linux (Ubuntu).

Testcase 1. Simulation with the BOX generator. Sampling size 5000 events for hotspot analysis (macros run_sim_bmn.C).

Testcase 2. Simulation with the LaQGSM generator. 5000 events for hotspots/1000 events to study scalability (macros run_reco_bmn.C).

Testcase 3. Reconstruction for the LaQGSM generator. Sampling sizes: 5000 events for hotspot analysis and 4000 events for the study of scalability and quality assurance (collisions of Ar and Pb nuclei with energy 3.2 Gev/Nuclon, only the tracking in the inner detectors (Silicon + GEM) is included in the reconstruction, macros run_reco_bmn.C).

Some of the results of hotspot analysis are given in tables 1 and 2 (testbench 2 and testcases 2-3). It can be seen from Tab. 1 that the most time-consuming hotspots of the BmnRoot simulation part are system functions that may not be modified. As a consequence we have focused our attention on the track reconstruction modules [5].

One of the most significant hotspots of the BmnRoot package is the event reconstruction by Kalman filtering which is a *de facto* standard in particle trajectory reconstruction [6]. Other hotspots are the functions that deal with the magnetic field map. An advanced microarchitecture hotspot analysis has also revealed multiple inefficiencies in the code: data dependencies, inefficient use of pipelines and so on.

3 High-performance optimization of the BmnRoot

We have tested various gcc compiler optimization options for both the simulation and the reconstruction parts of the BmnRoot. The tests involved complex `-O2` and `-O3` level optimizations, aggressive vectorization, loops autoparallelization, profile-guided optimization etc. No significant effect was obtained, which is a consequence of the source code structure.

OpenMP parallelization was performed for the CellsConnection function. Implementation of the threadsafe parallelization required a modification of the algorithm used in the function. Its correctness

was ensured by the Quality Assurance module. The scalability of the parallelized version is presented in Fig. 3. More efficient threadsafe parallelization of BmnRoot reconstruction modules requires a deeper modification of the reconstruction algorithm.

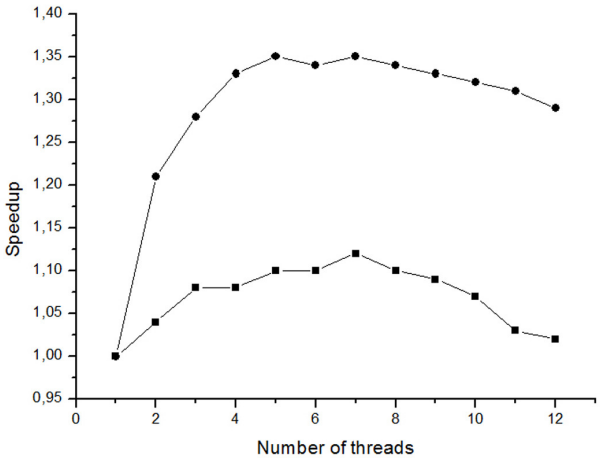

Figure 3. Speedup versus number of threads: 1) ■ 400 events; 2) • 4000 events.

4 Conclusion

In this article we presented the results of systematic analysis of the BmnRoot software package with respect to the performance optimization. We have identified bottlenecks in both the simulation and the reconstruction modules of the BmnRoot software package. We have performed a partial parallelization of the reconstruction module, studied scalability of the parallelized version and observed up to 35 percent performance improvement. Further improvements of efficiency and scalability of the optimized BmnRoot modules require much deeper modification including a revision of the numerical algorithms being used. Hybrid programming for the General Purpose Graphics Processing Units and vectorization should also be analysed for their applicability to the BmnRoot package.

Acknowledgements

This work is supported by Russian Foundation for Basic Research grant 18-02-40104 mega. We are also grateful to the Physics Educational Center of the Research Park of the Saint-Petersburg State University for support of educational projects related to the subject of the present study.

References

[1] D. Baranov, M. Kapishin, T. Mamontova, et al., KnE Energ. Phys. 3, **291**, 43 (2018)
[2] K. Gertsenberger, S.P. Merts, O.V. Rogachevsky, et al., Eur. Phys. J. A, **52**, 214 (2016)
[3] O.A. Malafeyev, S.A. Nemnyugin, *25-th Russian Particle Accelerator Conference, Proceedings*, St. Petersburg, (2016) p. 437
[4] *Google Performance Tools*, `https://github.com/gperftools/gperftools`
[5] P. Batyuk, D. Baranov, S. Merts, O. Rogachevsky, EPJ Web of Conferences **204**, 07012-1–7 (2019)
[6] R. Fruhwirth, Nucl. Instr. and Meth. in Phys. Res. A **262**, 444 (1987)