High-performance optimization of simulation and reconstruction modules in the BM@N software at the NICA

A V Driuk¹, S P Merts², S A Nemnyugin¹, V A Roudnev¹, M M Stepanova¹ and A A Iufryakova¹

¹ Saint Petersburg State University, University emb. 7–9, 199034 Saint-Petersburg, Russian Federation
² Joint Institute for Nuclear Research, Joliot-Curie 6, 141980 Dubna, Moscow Region, Russian Federation
E-mail: s.nemnyugin@spbu.ru

Abstract. Results of performance study of the BmnRoot software are presented. The BmnRoot software package plays crucial role in the Baryonic Matter at Nuclotron (BM@N) experiment of the NICA project. It is used for simulation and analysis of experimental data. Various approaches to performance optimization are discussed including algorithmic improvements and parallelization in simulation and reconstruction modules. Results of performance analysis are given and bottlenecks are localized. Results of efficiency and scalability studies for different approaches are also presented.

1. Introduction

The BmnRoot software package plays an important role in the BM@N (Baryon Matter at Nuclotron) experiment [1] (figure 1) of the NICA project (Joint Institute of Nuclear Research, Dubna) for simulation, event reconstruction and analysis of experimental data. BmnRoot consists of many modules including: Monte-Carlo event generators, geometry and run managers, tools for description of detector subsystems, track and hit finders [2], magnetic field mappers etc. Logical structure of the BmnRoot package is presented in figures 2 and 3 [3].

Simulation and track reconstruction are CPU time-consuming tasks. Dependence of CPU time on event multiplicity with BmnRoot is presented in figure 4. Event simulation with realistic Monte-Carlo generators and reasonable sample sizes (more than $10^5$ events) is also very time-consuming. There are various approaches to the performance-oriented optimization: 1) algorithmic optimization; 2) compiler optimizations; 3) parallelization of the BmnRoot modules. In the present paper the results of performance analysis and optimization both of the BmnRoot simulation and reconstruction modules are presented.

2. Optimization of the BmnRoot simulation modules

Multi-module structure of the BmnRoot package, large amount of source code, dependence of execution paths on input parameters significantly complicate optimization. In the present study dynamic analysis was used to localize the "hotspots" (parts of code with maximum consumption
Figure 1. BM@N setup. Detector subsystems, target and analysing magnet.

Figure 2. BmnRoot simulation modules.

Figure 3. BmnRoot reconstruction modules.

of CPU time) of simulation and reconstruction modules. The most significant hotspots are listed in figure 5.

Figure 4. CPU time by the BmnRoot reconstruction modules: 1) dotted line – per event; 2) solid line – per track reconstruction.
Testbench (simulation and reconstruction):

- CPU: Intel Xeon E-2136 @ 4.5GHz Turbo (6 cores). Memory (RAM): 32 GB 2666 MHz.
- OS: Ubuntu 16.04.

Testcase (simulation):

- Simulation with DQGSM (DubnaQuark-Gluon-String-Model) generator.
- 1000–5000 events.

OpenMP is used for parallelization of BmnGemStripDigitizer. Example of multithreaded code is given below:

```cpp
FairMCPoint* GemStripPoint;
Int_t NNotPrimaries = 0;
#pragma omp parallel
#pragma omp for schedule(dynamic)
for (UInt_t ipoint = 0; ipoint < fBmnGemStripPointsArray->GetEntriesFast(); ipoint++) {
    GemStripPoint = (FairMCPoint*) fBmnGemStripPointsArray->At(ipoint);
    ...
}
```

Scalability of parallelized BmnRoot simulation code is presented in figure 5. More detailed hotspot analysis displays inefficiencies in the code: loop dependencies, non-efficient use of pipelines and so on.
3. Optimization of the BmnRoot reconstruction modules

The modified track reconstruction algorithm which is used in the BmnRoot now is multistage and consists of the following steps:

- Search for high momentum tracks.
- Search for high momentum tracks with low efficiency.
- Search for low momentum tracks with inefficiency.

Performance tests of reconstruction modules have been performed both on simulated and experimental data. Testcases are based both on Monte-Carlo and experimental data from Run 7 at BM@N with Argon beam and Aluminium target. Reconstruction time in all these cases is too big and processing of a single portion of experimental data (hundreds of thousands and millions of events) may require days and weeks of computations.

Reconstruction speedup for change of build mode of gcc compiler optimizations is presented in table 1. Small code improvements (more efficient addressing, replacement of small arrays by variables, more efficient programming of arithmetical expressions etc.) lead to small performance improvement (percents of CPU time). Other kinds of the GCC compiler optimization: aggressive vectorization, autoparallelization of loops, profile-guided optimization, loops optimization etc. seems to be not very efficient.

Details of CPU time consumption for the reconstruction: Si+GEM detectors track finder: 45% of total CPU time, global matching: 21%, vertex finder: 19%. As it follows from figure 7 some hotspots belong to the BmnField module responsible for load of the analyzing magnet field measured in nodes of 3D Cartesian lattice. Most time-consuming operations are piece-wise linear interpolation between lattice nodes and field extrapolation outside known values. Algorithmic optimization of BmnFieldMap is based on replacement of linear-piecewise interpolation by constant-piecewise one. In this case the calculation for 8 nodes of the elementary cell is not necessary and the computational complexity of the field interpolation is reduced.

Table 1. Reconstruction time for change of build mode

| Testcase | Debug, sec | Release (O2 optimization level), sec |
|----------|------------|-------------------------------------|
| Monte Carlo | 1          | 0.3                                 |
| Experiment | 6          | 0.7                                 |

Figure 6. Scalability of the BmnRoot simulation parallelized with OpenMP.
Performance tests for the algorithmic optimization leads to the following results. Build in Debug mode (without compiler optimization) reduces total execution time by 10%, with build in O2-optimization mode reduces execution time by 4%. Execution time of the BmnField is 7% from total reconstruction time.

Example of track finder OpenMP parallelization is given below.

```cpp
BmnInnerTrackingRun7::FindTracks_4of4_OnLastGEMStations() {
  // Fit of dX vs X for different stations {p0, p1, sigma}
  // (from qgsm simulation)
  const Int_t nxRanges = 8;
  const Int_t nyRanges = 5;
  ...
  vector<BmnTrack> candidates;
  vector<BmnTrack> sortedCandidates;
  Int_t nThreads = THREADS_N;
  vector<vector<BmnTrack>> candsThread(nThreads);
  clock_t t0 = 0;
  Int_t threadNum;
  Int_t sH8 = sortedHits[8].size();
  #pragma omp parallel if(sH8 > 100) num_threads(nThreads)
  #pragma omp for // schedule(static,1)
  for (Int_t ii = 0; ii < sH8; ++ii) {
    BmnHit* hit8;
    hit8 = sortedHits[8].at(ii);
    ...
  }
  ...
```

Reasonable scalability of the OpenMP version of reconstruction is not yet achieved. Possible reasons of low efficiency are following. In many cases the number of loops iterations is zero, so the efficiency of OpenMP-parallelization is low. The most significant hotspot points to the Kalman filter realization, so it should be optimized first.
4. Conclusion
The results of performance analysis of the BmnRoot software package are presented. Hotspots of the simulation and the reconstruction modules of the BmnRoot are localized for common cases of usage. Parallelization of the simulation module and various kinds of optimization including algorithmic one have been performed for the BmnRoot reconstruction. Results of performance benchmarking are given and problems to be solved on the way to further optimization are formulated.

Acknowledgments
Acknowledgments 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] Baranov D, Kapishin M, Mamontova T et al. 2018 KnE Energ. Phys. 3 291 43
[2] Nemnyugin S A, Roudnev V A, Stepanova M M and Usov D P 2019 Proceedings of the 27th International Symposium Nuclear Electronics and Computing (NEC’2019) 2507 pp 397-401
[3] Gertsenberger K, Merts S P, Rogachevsky O V et al. 2016 Eur. Phys. J. A 52 214
[4] Fruhwirth R 1987 Nucl. Instr. Meth. Phys. Res. A 262 444