Implementation of high-performance computing technologies in the BmnRoot framework

NEMNYUGIN S.A., MERTS S.P., DRIUK A.V., IUFRYAKOVA A.A., MYASNIKOV A.A.,

STEPANOVA M.M.

SAINT-PETERSBURG STATE UNIVERSITY

JOINT INSTITUTE OF NUCLEAR RESEARCH

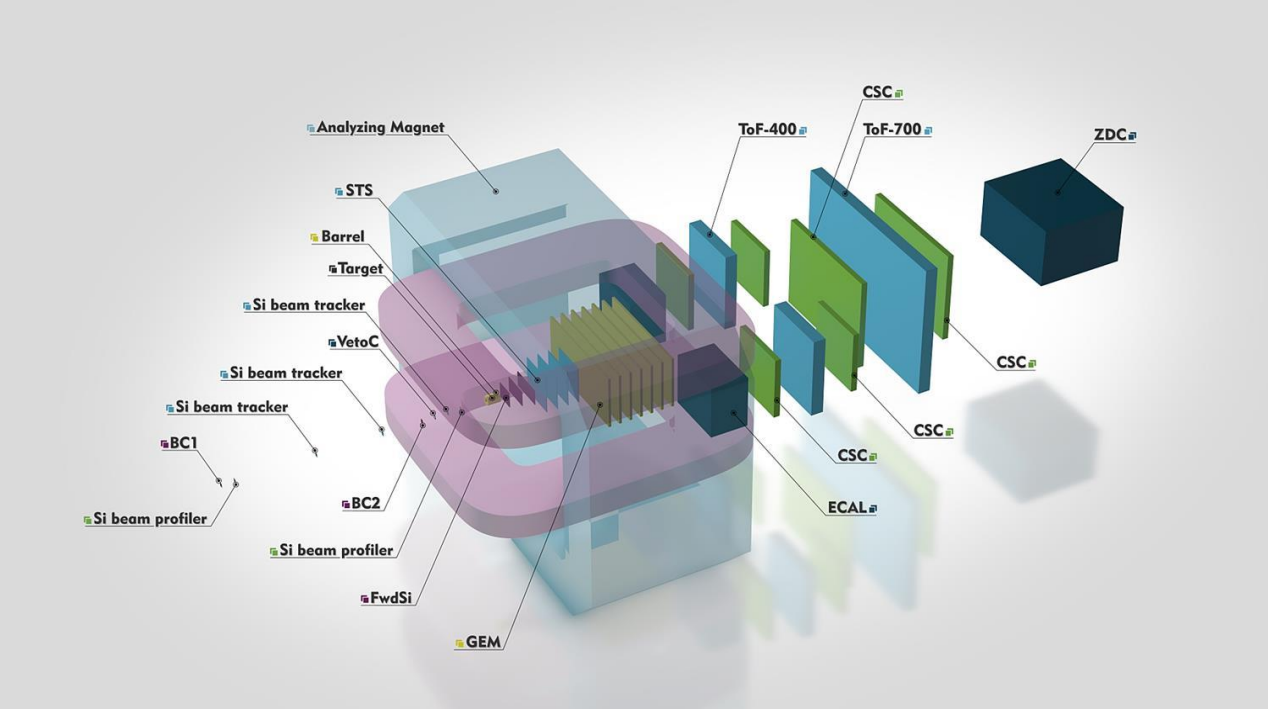
Outline

- Baryon Matter at Nuclotron (BM@N) experiment.
- BmnRoot framework.
- Optimization levels.
- Multithreaded Geant4 for simulation.
- PROOF for reconstruction.
- Optimization of the BmnRoot code.
- Conclusion

Work is supported by Russian Foundation for Basic Research grant 18-02-40104 mega.

BM@N NICA experiment

NICA - **N**uclotron based **I**on **C**ollider f**A**cility, Joint Institute for Nuclear Research, Dubna.



BM@N – **Baryon M**atter at **N**uclotron experiment. Heavy ion collisions with fixed targets. 7 runs have been performed.

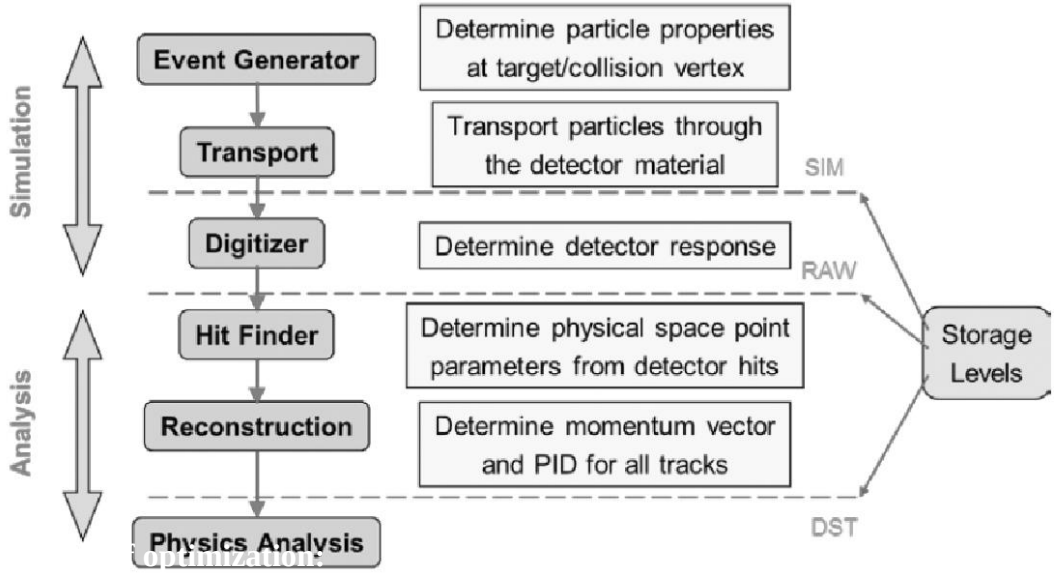
Beam: Ar. Kr, C, d, energy 2.9 – 4.5 GeV.

Target: Al, Cu, C, Pb, Sn, H2 (liquid) etc.

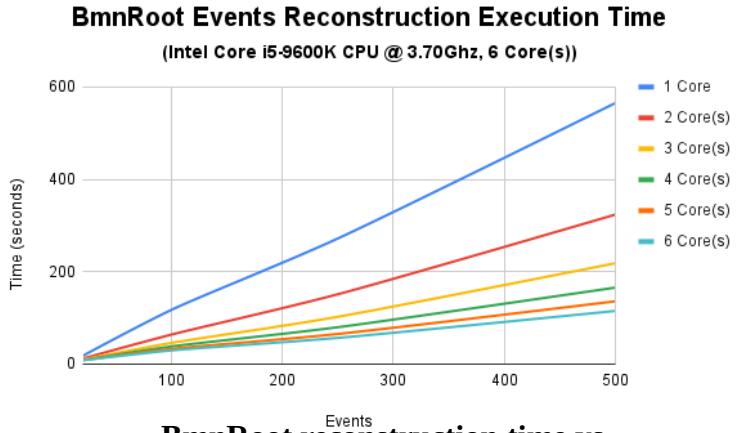
Setup includes detector subsystems, magnet, electronics.

BmnRoot framework

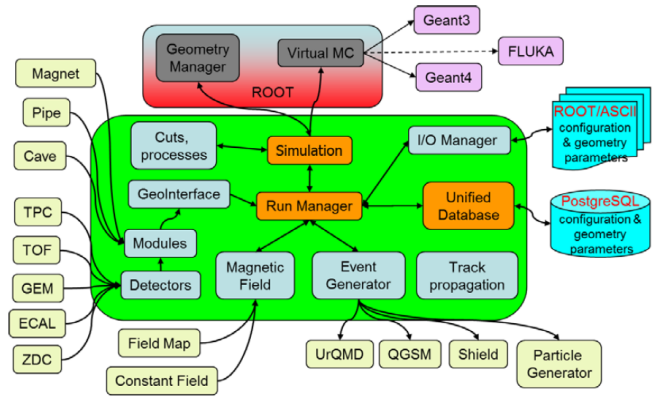
- ✓ BmnRoot framework (BM@N NICA) is based on the FairRoot and FairSoft software packages (GSI, Darmstadt).
- ✓ Module structure with a lot of modules for simulation and analysis, hundreds of thousands lines of code.
- ✓ Simulation: setup configuration, beam parameters, Monte-Carlo event generators (BOX, QGSM, UrQMD, SHIELD), Virtual Monte-Carlo, transport codes (Geant3, Geant4, Fluka), magnetic field maps etc.
- ✓ Reconstruction: setup configuration, all detector subsystems, beam parameters, magnetic field accounting, matching (local/global) etc.
- ✓ Simulation and reconstruction performance should be improved as much as possible.



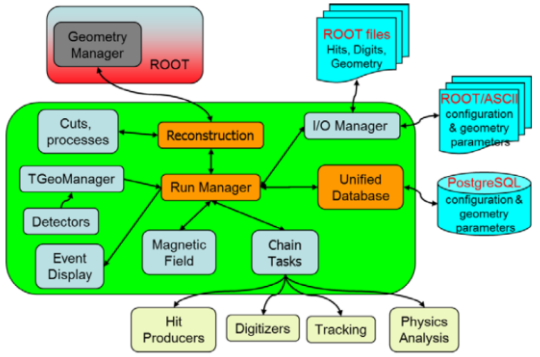
Reasonable Monte-Carlo simulation needs for large sampling size.
Realistic simulations are time-consuming.



BmnRoot reconstruction time vs events multiplicity



BmnRoot simulation modules



BmnRoot reconstruction modules

Optimization

Levels of optimization:

1. External optimization on the base of distributed computing (data parallelism). Multithread Geant4. PROOF.
2. Internal optimization – extinction of «hotspots». Vectorization (compiler vectorization, intrinsics, libraries), parallelization (OpenMP, MPI, CUDA. OpenCL, OpenACC) etc.

ROOT Data Analysis Framework

PROOF (Parallel ROOT Facility)

Part of ROOT (CERN & MIT).

The PROOF system allows:

- parallel analysis of trees in a set of files;
- parallel analysis of objects in a set of files;
- parallel execution of scripts

on parallel computing systems.

Events are independent => data parallel => multitask parallelism, may be implemented on computing clusters. Multicore CPU. Thread-safe.

Geant4 multithreading : event--level parallelism

Event-level parallelism.

Multicore CPU. Thread-safe.

Optimization of simulation

Implementation of multithreaded Geant4 in BmnRoot framework.

Implementation of multithreading in BmnRoot affects various levels: Geant4, Geant4 VMC, VMC в ROOT, FairRoot, BmnRoot

Implementation steps:

Search for C++ classes of BmnRoot which are influenced by MT Geant4 and their modification to work efficiently and thread safely in multithreaded regime of simulation.

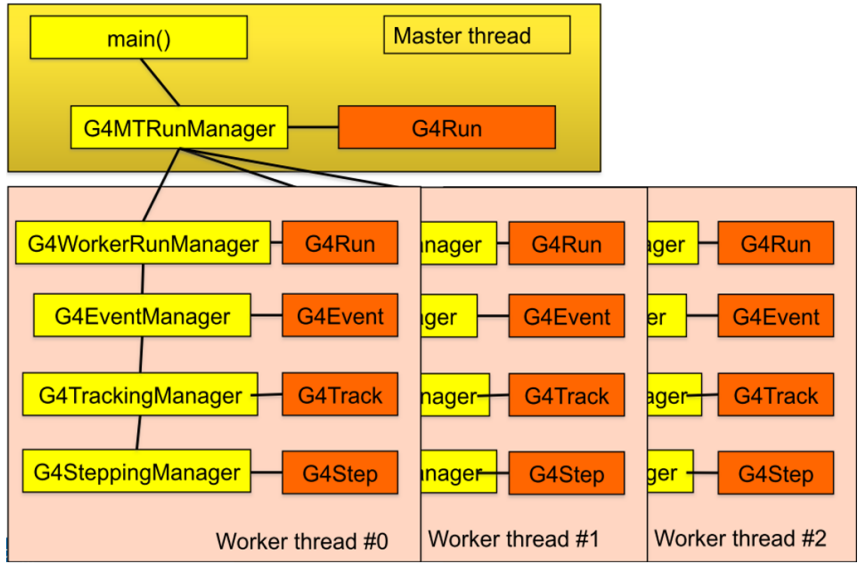
Debugging and benchmarking.

Evaluation of scalability and efficiency of MT Geant4+BmnRoot.

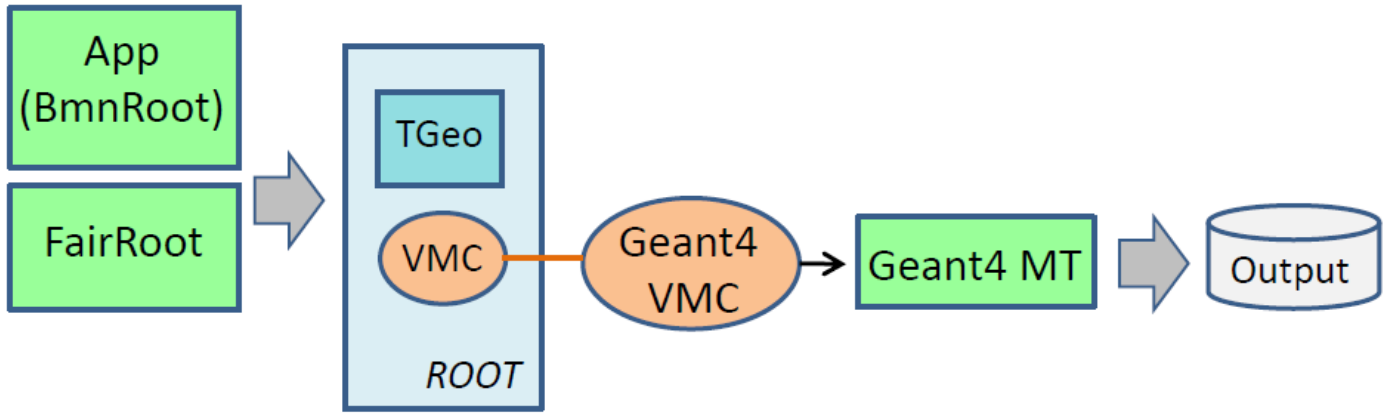
Multithreaded paradigm of MT Geant4 (since 10.03, 2013)

(geant4.cern.ch)

- Event-level parallelism, master-workers model.
- **G4MTRunManager** - workers scheduling.
- Instances of the **G4WorkerRunManager** class – events simulation in threads.
- Parallelism of GEANT4 is based on POSIX.
- Thread-local storage, TLS.
- Heterogeneous parallelism supported (MT+MPI, Intel® TBB).



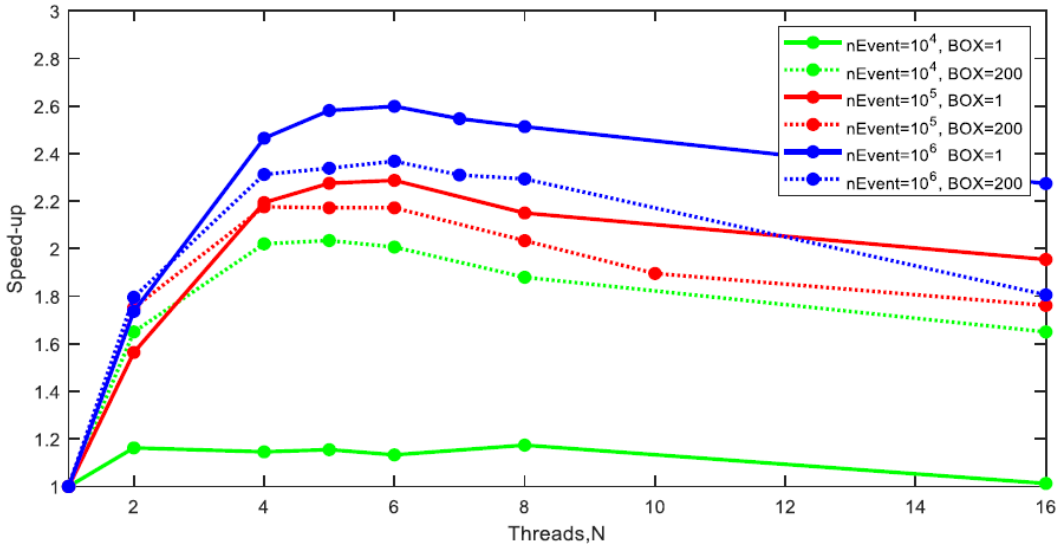
- Virtual Monte Carlo (VMC) – common interface for various transport codes used for simulation of detector subsystems of BmnRoot.
- Geant4 VMC – implementation of Virtual Monte Carlo for Geant4.
- First Geant4 VMC MT (3.0) issued in 2014 and is based on the same approach to multithreading as Geant4 MT.



Implementation of MT in BmnRoot requires following modifications (Fair packages are opensource):

- Class **FairMCApplication**: methods **CloneForWorker(), InitOnWorker()** are added.
- Class **FairModule**: new virtual function **FairModule* CloneModule() const;** is added. Its redefinition in MT regime is necessary for objects cloning on workers.
- Class **FairRootManager** is modified by control of I/O – ROOT output for each thread.
- Classes **FairRunSim, FairStack, FairRootManager** and **FairLogger** have been modified.
- For migration of apps new functions **TVirtualMCApplication** must be developed which are used for cloning of the application and its objects on workers.

Min (MT+)	Now used		
GEANT4	10.03.p01	10.05.p01	10.7.1
Geant4VMC	3.0	4.0.p1	
ROOT	6.09.04	6.16.00	6.22.08
FairRoot	18.2.0	18.2.0	v18.6.3



Scalability of simulation with BOX Generator

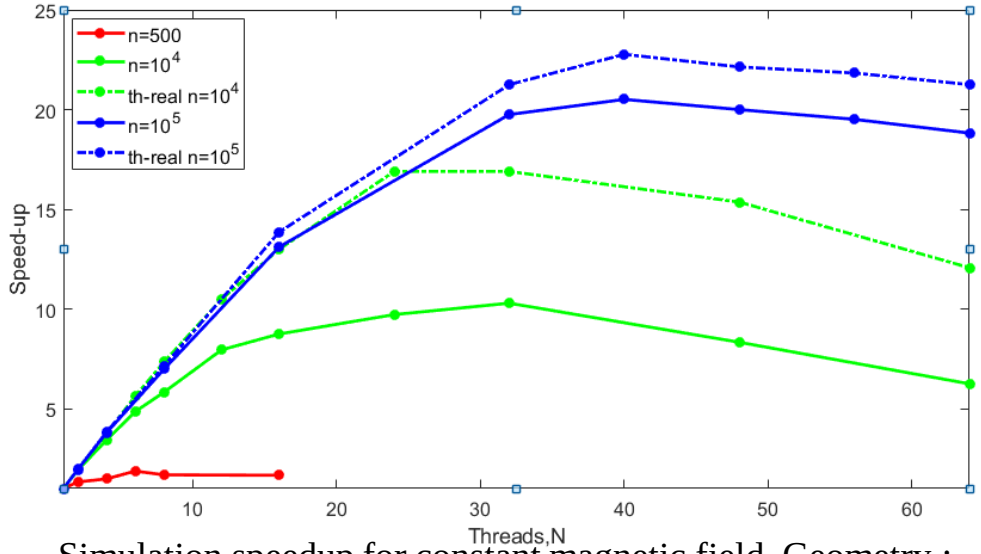
Computing node of NCX-cluster (LHEP)
2x Intel Xeon E5-2697a, DDR4-2400, 2.6GHz, 32(64) cores, 512G RAM, CentOS7, EOS Storage

BmnRoot framework. Simulation.

BmnRoot modifications for MT:

- In software realizations of detector subsystems (BmnFD, BmnBd, BmnCSC, BmDch, BmnEcal, BmnSilicon BmnMWPC, CbmSts, BmnTOF, BmnTOF1, BmnZdc, BmnPsd,BmnRecoil) functions are included: **FairModule* CloneModule() const;** as well as copying constructors.
- Class CbmSts: methods for registration of particles in threads are added.
- Class CbmStack: cloning of objects in threads is realized.

But in general case… we see performance degradation! Reason – magnetic field read, mapping etc.!



Simulation speedup for constant magnetic field. Geometry : CAVE MAGNET STS ECAL ZDC. BOX=(2110,10) .

BmnRoot framework. Reconstruction

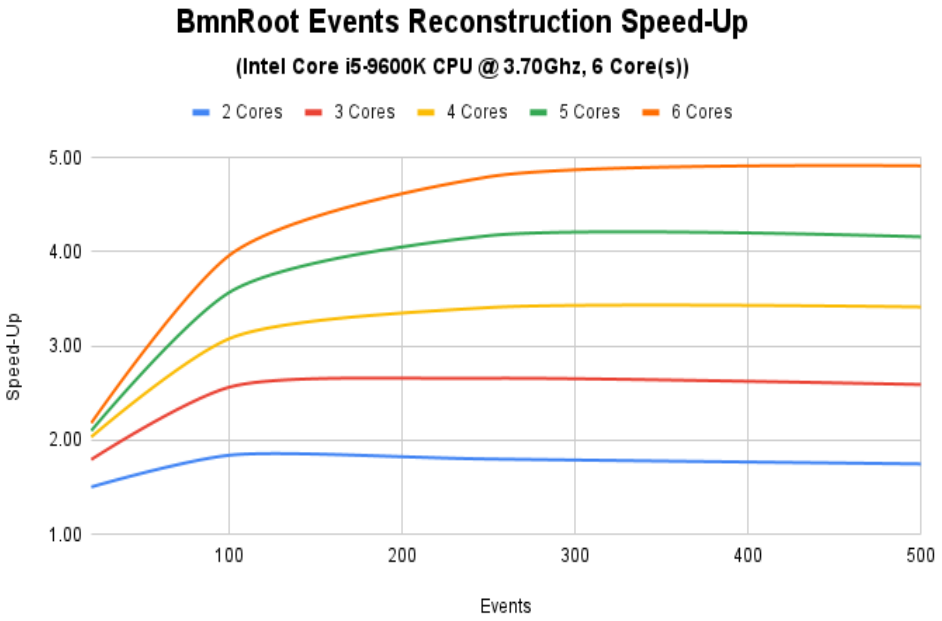
PROOF tool is applied for the BmnRoot reconstruction macro.

PROOF supports following types of computing architectures: shared memory (multicore CPU), distributed memory (cluster), data processing in GRID.

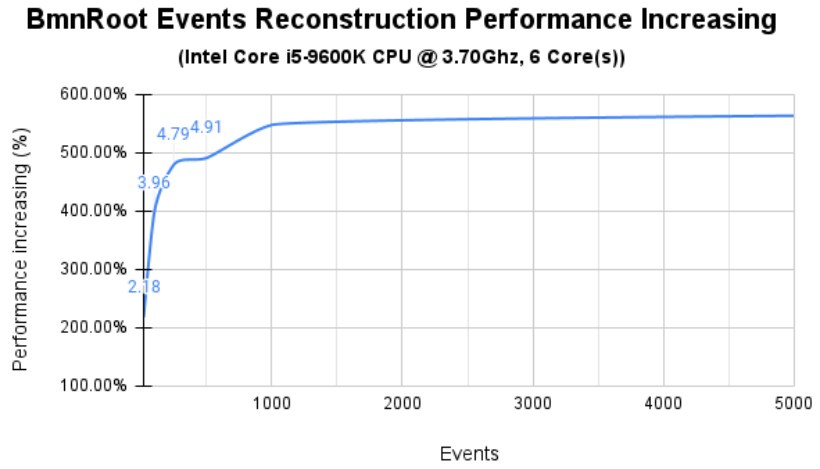
The report presents the results of the work on acceleration of the event reconstruction via the PROOF package on multicore CPU.

Following steps were taken for integration PROOF in BmnRoot events reconstruction:

- Serialization problems have been solved for members of classes inherited from **FairTask** for correct packing by **TStreamer** class.
- Constructors of classes inherited from **FairTask** were modified for correct unpacking by the **TStreamer** class.
- Non-Root libraries were bundled as the libBmnRoot.par package for distribution across nodes.



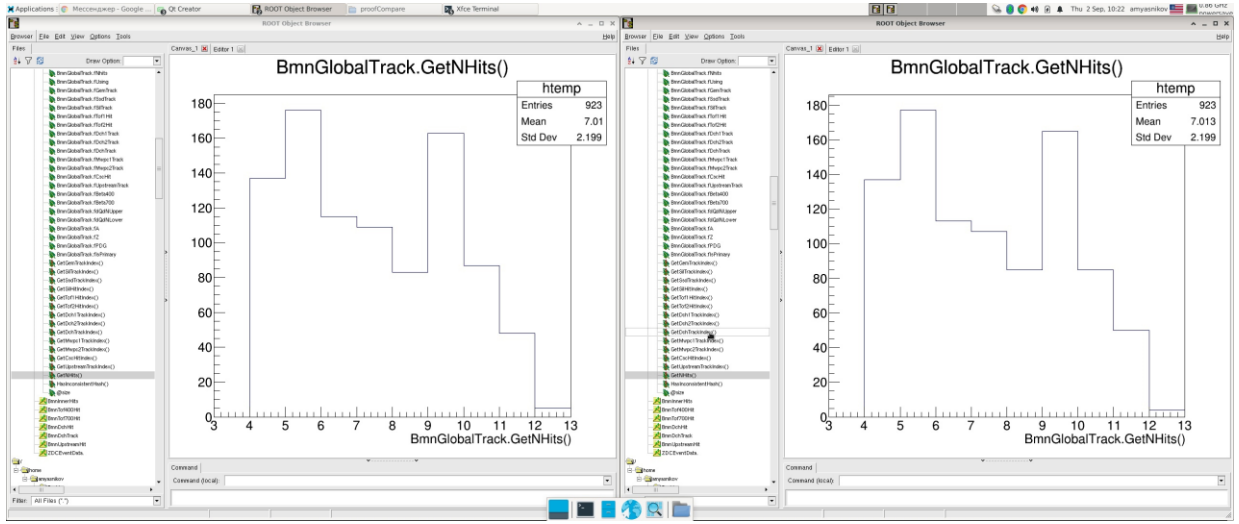
Maximum speedup is observed when reconstructing at least a thousand events - 5.64.



Testbench:

Intel Core i5-9600K CPU (6 Cores, 32Gb RAM.

Measurements were made for chains of 20,100,250,500.



Quality Assurance for parallelized version

Optimization of the BmnRoot code. Simulation modules

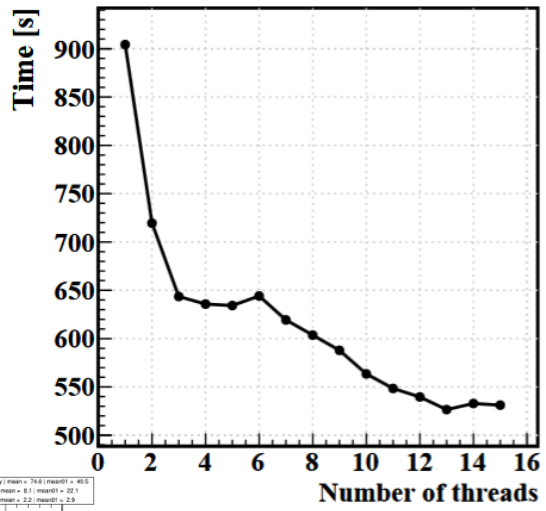
✓Performance optimization (parallelization of most time-consuming hotspots).

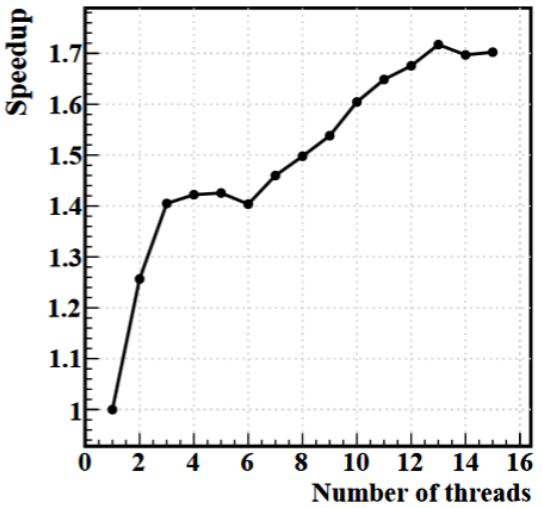
✓Tests of correctness and scalability of optimized code (Quality Assurance).

BmnGemStripDigitizer

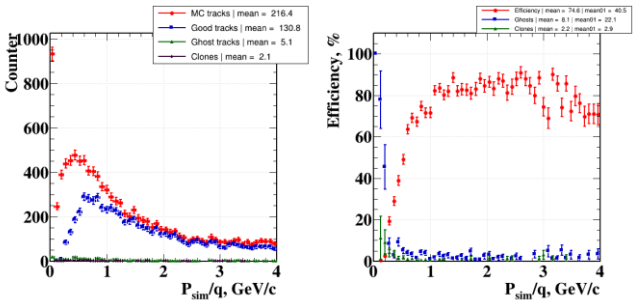
```
…
FairMCPoint* GemStripPoint;
Int_t NNotPrimaries = 0;
#pragma omp parallel
#pragma omp for schedule(dynamic)
for (UInt_t ipoint = 0; ipoint < fBmnGemStripPointsArray->GetEntriesFast();
ipoint++) {
GemStripPoint = (FairMCPoint*) fBmnGemStripPointsArray->At(ipoint);
…
```

OpenMP parallelization

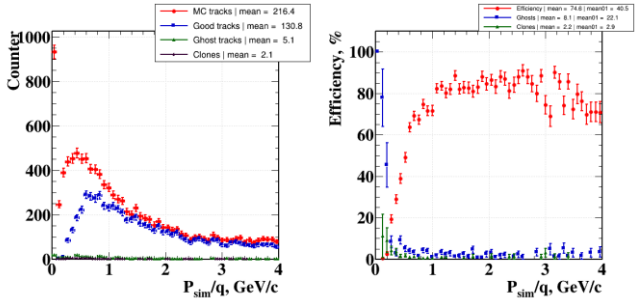




Scalability of the BmnRoot simulation parallelized with OpenMP



QA - 1 thread



QA - 4 thread

Quality Assurance for simulation

Performance problems of the BmnRoot. Reconstruction modules

Very slow
- ✓ **Monte Carlo data 1 sec/event**
- ✓ **Experimental data 6 sec/event**
- ✓ **One file (200 000 event) up to 2 weeks**

Testbench

CPU: Intel Xeon E-2136 @ 4.5GHz Turbo (6C 2xHT, L3 Cache 8MB)
RAM: 32GB 2666MHz DDR4
OS: Ubuntu 16.04.6 LTS

Testcase

Simulation data with DQGSM generator
1000-5000 events.
Experimental data: Run 7 at BM@N, Argon beam, Al target.
Macro run_reco_bmn.C

Details of CPU Time Consumption
Si+GEM Track Finder: 45%
Global Matching: 21%
Vertex Finder: 19%

Analysis summary

A lot of hotspots belong to the BmnField module – load of the analyzing magnet field:

- 3D Cartesian lattice;
- piecewise linear interpolation between lattice nodes;
- extrapolation outside known values.

HOTSPOTS

Function / Call Stack	CPU Time ▼	Module
clock	66.450s	libc.so.6
BmnKalmanFilter::RK4Order	34.481s	libBmnData.so.0.0.0
BmnNewFieldMap::FieldInterpolate	23.124s	libBmnField.so.0.0.0
TArrayF::At	22.950s	libBmnField.so.0.0.0
inflate	20.673s	libz.so.1
BmnKalmanFilter::TransportC	17.638s	libBmnData.so.0.0.0
BmnNewFieldMap::IsInside	15.992s	libBmnField.so.0.0.0
TArray::BoundsOk	15.566s	libBmnData.so.0.0.0
BmnFieldMap::Interpolate	15.226s	libBmnField.so.0.0.0
std::vector<double, std::allocator<doub	13.862s	libBmnData.so.0.0.0
operator new	12.704s	libstdc++.so.6
std::vector<double, std::allocator<doub	12.222s	libBmnDst.so.0.0.0
BmnKalmanFilter::RK4TrackExtrapolat	11.896s	libBmnData.so.0.0.0
std::vector<double, std::allocator<doub	11.128s	libBmnData.so.0.0.0
TGeoVoxelFinder::GetNextCandidates	10.982s	libGeom.so.6.16
__pow	10.944s	libm.so.6
std::__fill_n_a<double*, unsigned long	10.074s	libBmnData.so.0.0.0
TGeoVoxelFinder::GetCheckList	9.832s	libGeom.so.6.16
__GI_	8.701s	libc.so.6
std::vector<double, std::allocator<doub	8.420s	libBmnData.so.0.0.0
std::vector<BmnLink, std::allocator<Bn	7.352s	libSilicon.so.0.0.0
TGeoNavigator::SearchNode	6.766s	libGeom.so.6.16

Hotspots of the BmnRoot reconstruction modules

BmnRoot framework. Code optimization

In progress:

- Compiler optimization (including autovectorization by compiler gcc 10.2, pragmas of vectorization and optimization control).
- "By hand" vectorization with intrinsics, vector data types etc.
- Algorithmic interpolation of magnetic field interpolation and extrapolation.
- Optimization of memory access patterns.
- Optimization of Kalman filter software realizations.
- "Cleaning" of the code (Valgrind, Intel® Inspector): correction of memory access errors and multithread realization problems: memory leakages, data races, access to non-initialized variables etc.

```
#pragma GCC optimize("O3")
#pragma GCC target("avx2")
#pragma GCC optimize("unroll-loops")
#pragma GCC ivdep
…
```

```
#include <x86intrin.h>
…
__m256d xxx = _mm256_loadu_pd(&cIn[i]);
__mm256_storeu_pd(&cIn_tmp[i], yyy);
…
```

Future:

- Hybrid technologies (CPU+accelerator: GPGPU, FPGA, …).
- New programming technologies (DPC++, …).

Conclusion

- Performance studies are performed and performance bottlenecks of the BmnRoot simulation and reconstruction modules are revealed.
- Multithreading functionalities of Geant4 and PROOF are incorporated into BmnRoot software package.
- Next problems should be solved: more thread safety, modification of detectors software realizations for MT, optimization of magnetic field data input and so on.
- It is planned to perform scalability study on the cluster and integrate PROOF into the reconstruction of SRC events (part of BM@N NICA experiment).

.

Thank you for attention!