Contribution of Saint Petersburg University group in development of BmnRoot

S.NEMNYUGIN

SAINT-PETERSBURG STATE UNIVERSITY

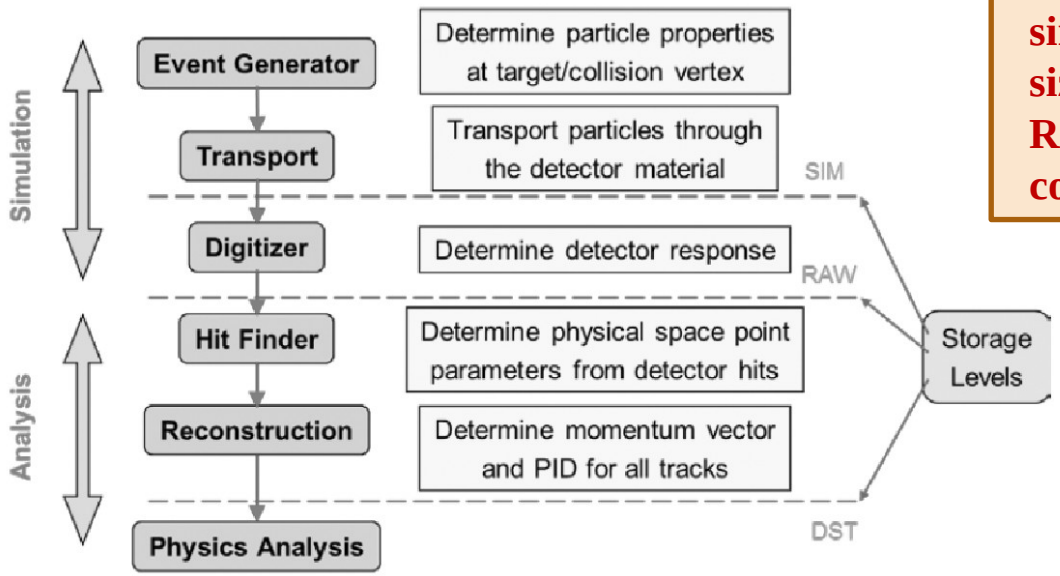
Outline

- **BmnRoot** framework
- Optimization levels
- Multithreaded simulation
- **PROOF** for reconstruction
- Memory errors in **BmnRoot**
- **BmnRoot** code optimization
- Summary

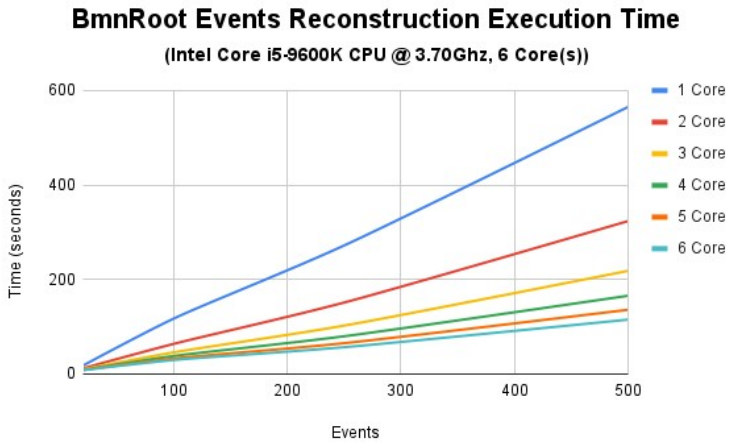
Work is supported by Russian Foundation for Basic Research grant 18-02-40104 mega.

BmnRoot framework

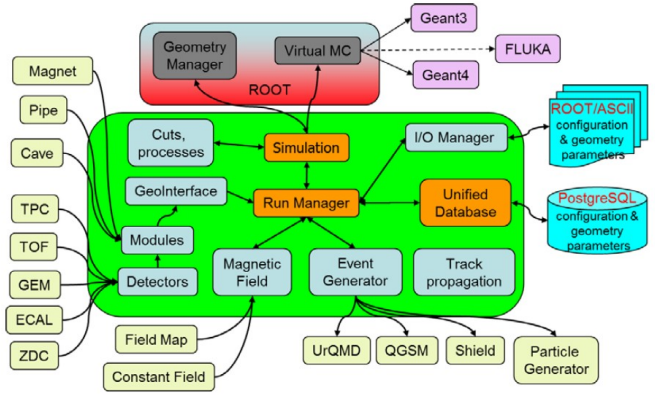
- ✓ **BmnRoot** framework (used in BM@N NICA) is based on the **FairRoot** and **FairSoft** software packages (GSI, Darmstadt).
- ✓ Module structure with a lot of modules for simulation and analysis, hundreds of thousands lines of code.
- ✓ Simulation: description of configuration and beam parameters,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



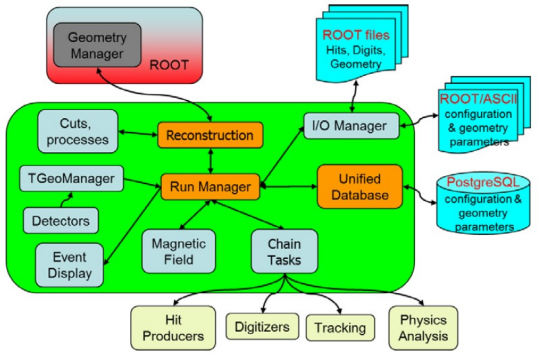
Reasonable Monte-Carlo simulation needs for large sampling size.
Realistic simulations are time-consuming.



BmnRoot reconstruction time vs nEvents



BmnRoot simulation modules



BmnRoot reconstruction modules

Optimization

Levels of optimization:

1. External optimization on the base of distributed computing (data parallelism). Multithreaded **Geant4**. **PROOF**.
2. Internal optimization – removing 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.

Geant4 multithreading

Event-level parallelism.
Multicore CPU. Thread-safe.

HiEn physics events are independent => data parallel => multitask parallelism, may be implemented on computing clusters and multicore CPU. Thread-safety!

Optimization of simulation

Implementation of multithreaded Geant4 in BmnRoot framework

Implementation of multithreading in BmnRoot affects levels: **Geant4**, **Geant4** VMC, VMC in **ROOT**, **FairRoot**, **BmnRoot**.

Implementation steps:

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

Debugging and benchmarking.

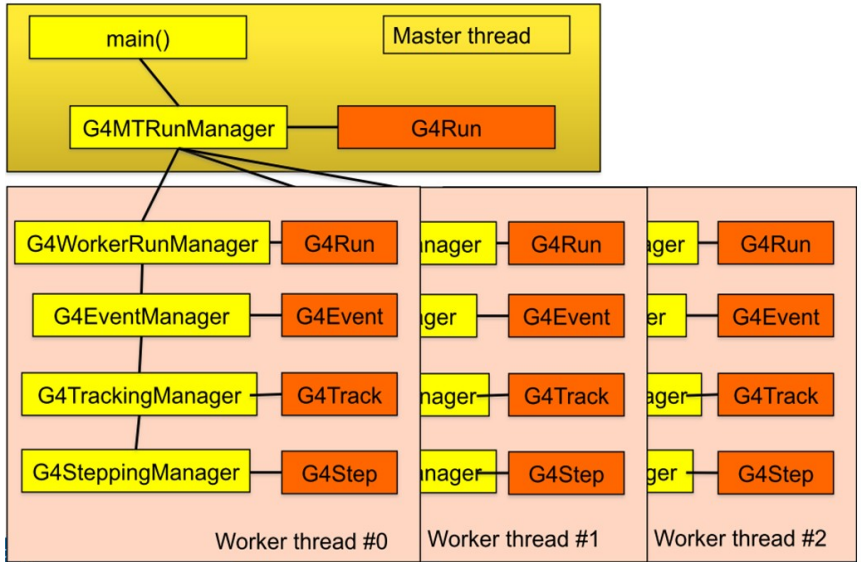
Evaluation of scalability and efficiency of **Geant4** MT+**BmnRoot**.

STEPANOVA M., DRIUK A.

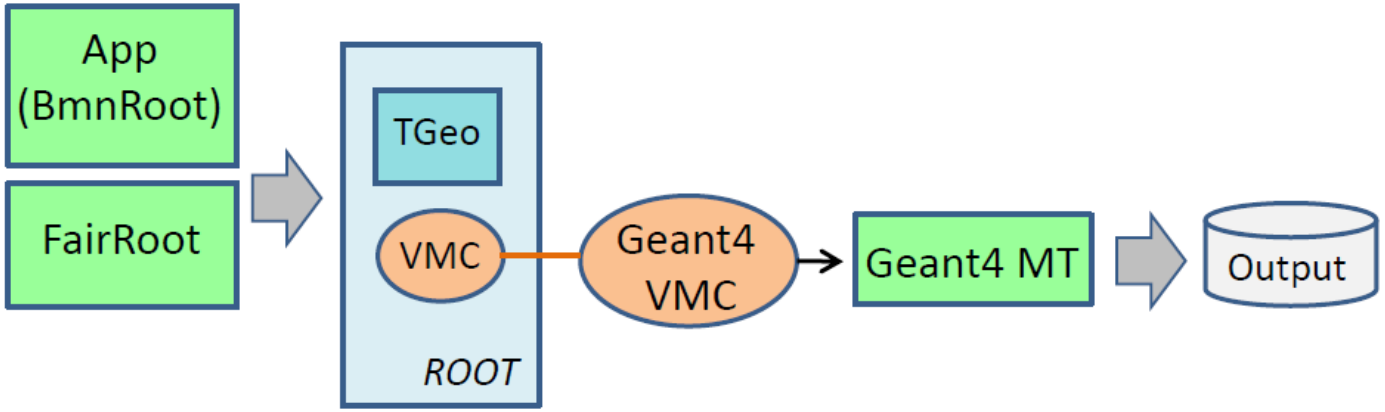
Multithreaded paradigm of Geant4 MT (since 10.03, 2013)

(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 standard (pthreads).
- Thread-local storage, TLS.
- Heterogeneous parallelism supported (MT+MPI, Intel® TBB).



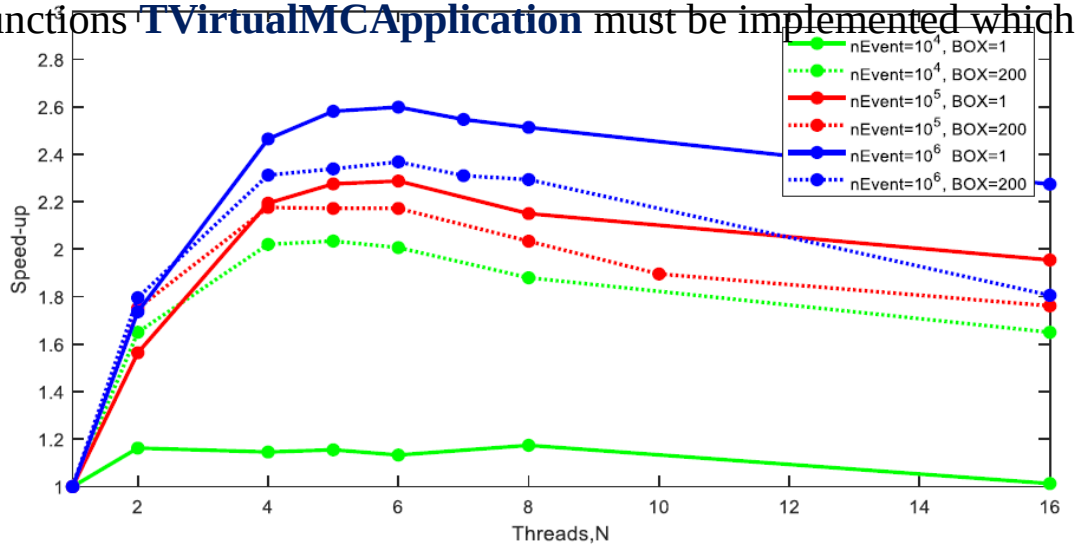
- **Virtual Monte Carlo (VMC)** – common interface for various transport codes used for simulation of detector subsystems of BmnRoot.
- **Geant4 VMC** – implementation of Virtual Monte Carlo for **Geant4**.
- **Geant4 VMC MT** (since 3.0 issued in 2014) and is based on the same approach to multithreading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



Implementation of MT in BmnRoot required following modifications (I Hrivnacova. EPJ Web of Conferences 214, 02018 (2019))**:**

- 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 adding control of I/O operations – **ROOT** output for each thread.
- Classes **FairRunSim, FairStack, FairRootManager** and **FairLogger** have been modified.
- For migration of apps inherited from **FairRoot** base classes new functions **TVirtualMCApplication** must be implemented which are used for cloning of the application and its objects in workers.

Package	Min version (MT+)	Now used	Last version
Geant4	10.03.p01	10.05.p01	10.07.p02
Geant4 VMC	3.0	4.0p1	5.3
ROOT	6.09.04	6.16.00	6.24.06
FairRoot	18.2.0	18.2.0	18.6.4



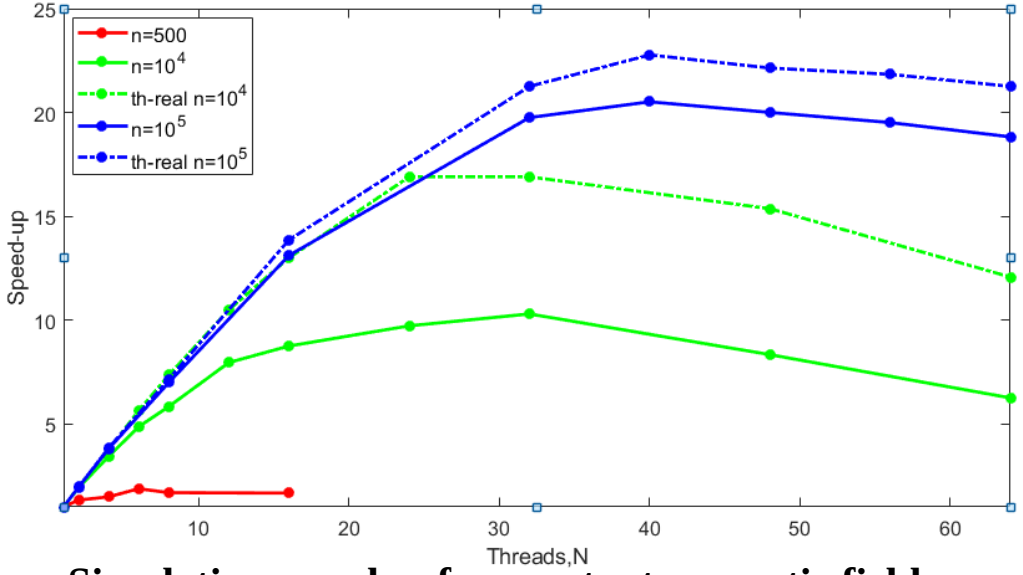
Scalability of simulation with BOX Generator (Rutherford exp.)

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

BmnRoot modifications for MT:

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

In general case - performance degradation. Reason – magnetic field read from a file, interpolation, etc.



Simulation speedup for constant magnetic field

Geometry : CAVE MAGNET STS ECAL ZDC. BOX=(2110,10)

Optimization of reconstruction

MYASNIKOV A.

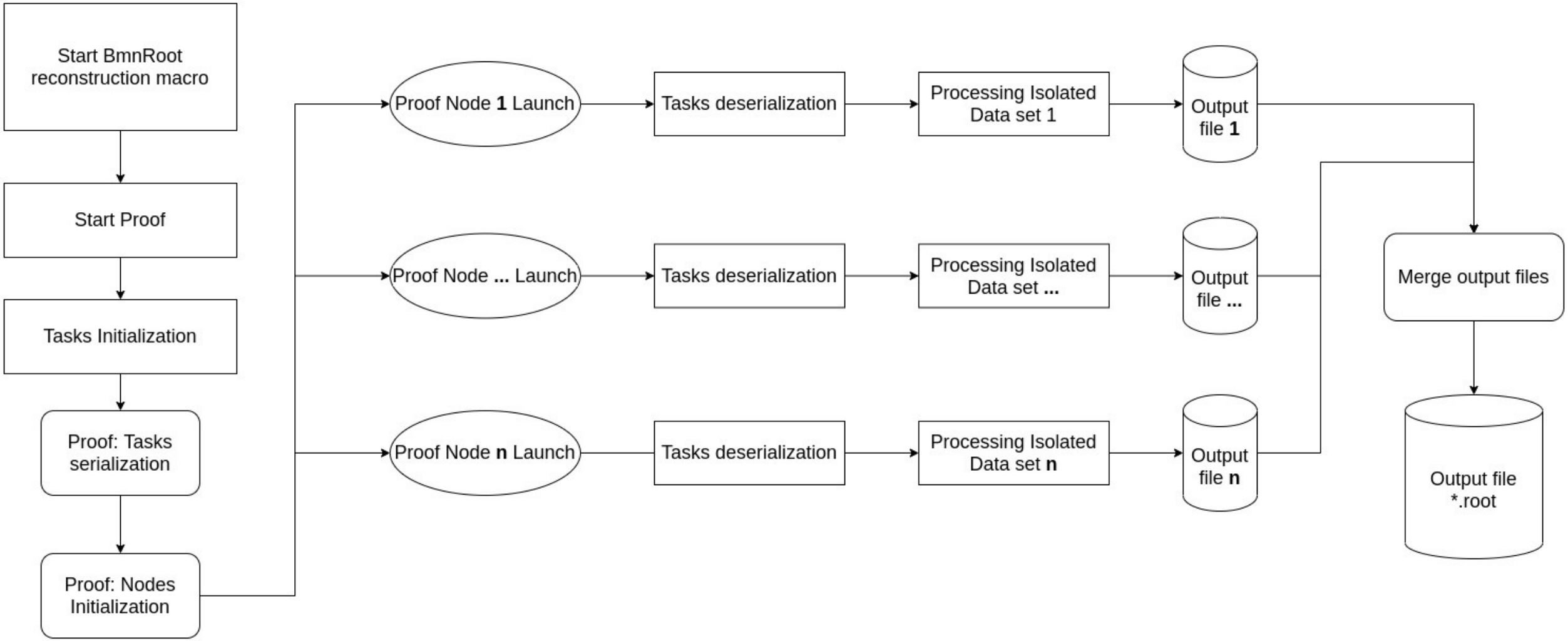
PROOF tool is applied for the BmnRoot reconstruction macro.

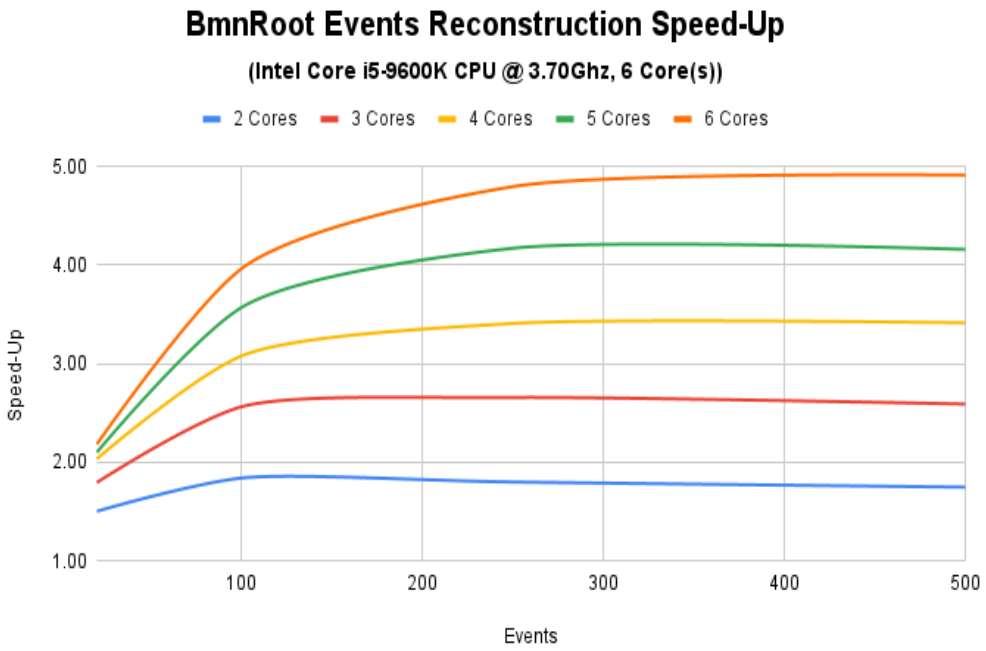
PROOF supports following types of computing architectures: shared memory (multicore CPU), distributed memory (clusters), data processing in GRID.

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

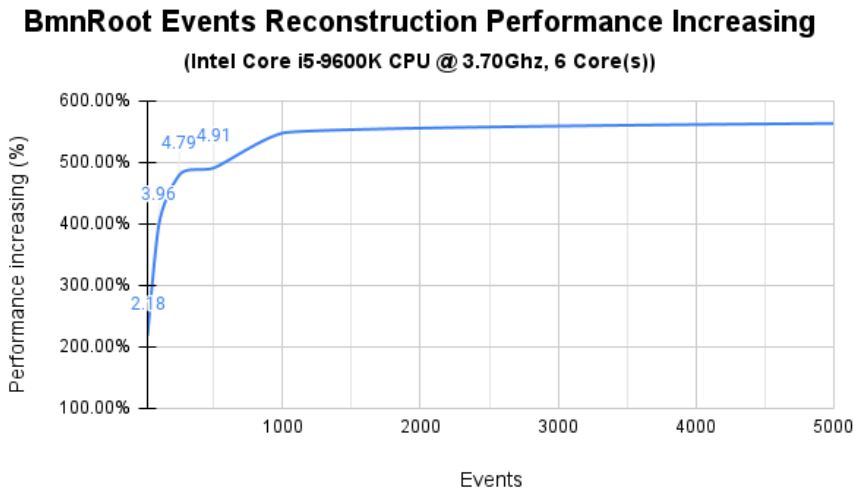
Following steps were taken for integration **PROOF** in **BmnRoot** events reconstruction:

- 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.


Start BmnRoot reconstruction macro
Start Proof
Tasks Initialization
Proof: Tasks serialization
Proof: Nodes Initialization
Proof Node 1 Launch
Proof Node ... Launch
Proof Node n Launch
Tasks deserialization
Tasks deserialization
Tasks deserialization
Processing Isolated Data set 1
Processing Isolated Data set ...
Processing Isolated Data set n
Output file 1
Output file ...
Output file n
Merge output files
Output file *.root

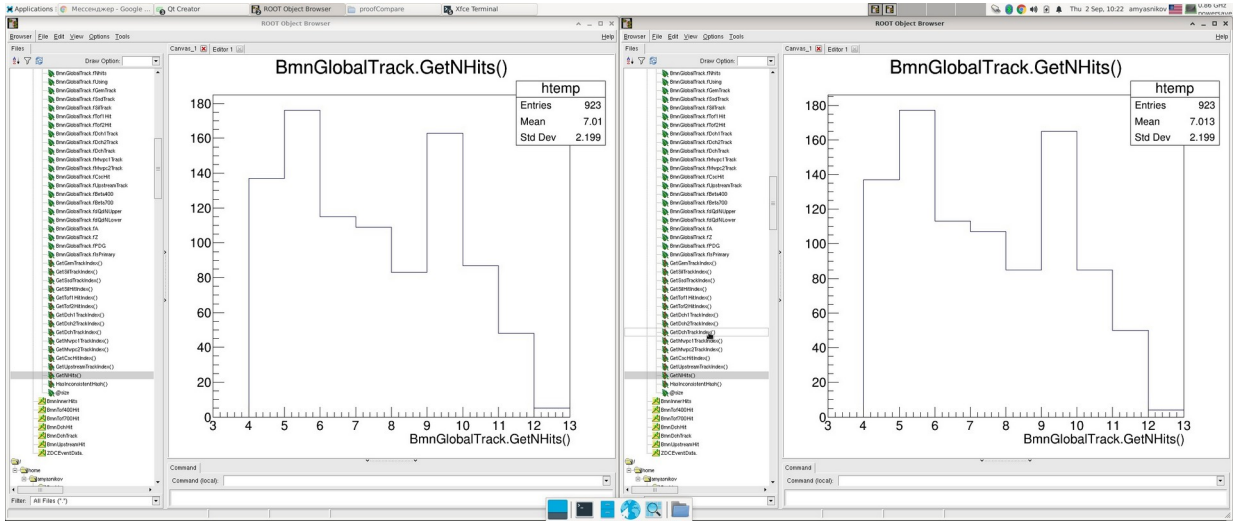


Maximum speedup is observed when reconstructing at least a thousand events - 5.64.



Testbench

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



Quality Assurance for parallel version

Search and removal of memory errors in BmnRoot

DRIUK A.

Analysis configuration:

1.Valgrind 3.15.

2. OS Ubuntu 20.04.2 LTS x86_64.

3. Compiler gcc 9.3.

Memory leaks in reconstruction procedure have been localized in the following places: **BmnInnTrackerAlign.cxx**, **BmnMwpcHitFinder.cxx**, **BmnTof1HitProducer.cxx**, **UniDbDetectorParameter.cxx**, **BmnDchHitProducer.cxx**, **UniDbRun.cxx**. **BmnMwpcGeometrySRC.cxx**, **BmnFileSource.cxx**.

The classes **BmnTof2Raw2DigitNew.cxx**, **BmnTof2Raw2DigitNew.h**, **BmnTofGeoUtils.h**, **BmnTofHitProducer.h** have been fixed for correction work in Release mode. The incorrect access to an element outside the array resulted in segmentation fault. Also, an attempt of double deletion of **Tlist** in **run_reco_bmn.C** fixed (see Fig). It was caused by a global variable with the same name as the local one.

```
root [1] .q
Error in <TList::Clear>: A list is accessing an object (0x55bdf1198150) already deleted (list name =
 TList)
Error in <TList::Clear>: A list is accessing an object (0x55bdf2212a10) already deleted (list name =
 TList)
Error in <TList::Clear>: A list is accessing an object (0x55bdf131bbe0) already deleted (list name =
 TList)
Error in <TList::Clear>: A list is accessing an object (0x55bdef8812f0) already deleted (list name =
 TList)
```

Fig. Double free of Tlist

3. The cross links were found and fixed in classes **BmnGemStripDigitizer.h**, **BmnGemStripHitMaker.cxx**, **BmnSiliconDigitizer.cxx**, **BmnSiliconHitMaker.cxx** and **BmnInnTrackerAlign.cxx**

a) Alignment corrections moved into database.

b) Macro to upload corrections into database added.

4. Macros **run_sim_bmn.cxx** and **run_sim_src.cxx** for Valgrind in **macro/profiling/** were updated according to the current version of **run_sim_bmn(src).C**. Now, everyone can check simulation and reconstruction procedure in the profiling mode. Analysis of that macros at the current version didn't show significant memory leaks.

Optimization of the BmnRoot code. Simulation modules

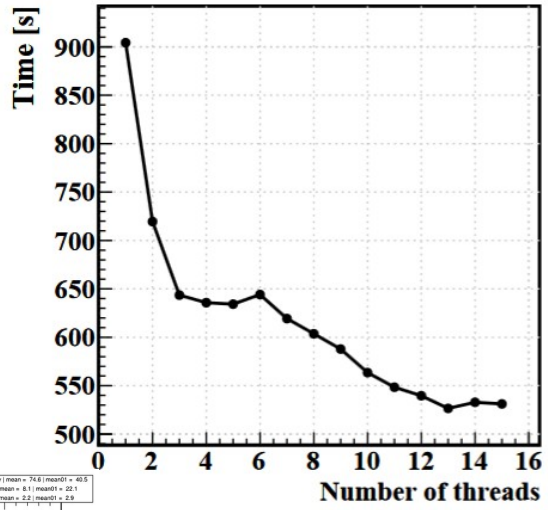
IUFRYAKOVA A., NEMNYUGIN S.

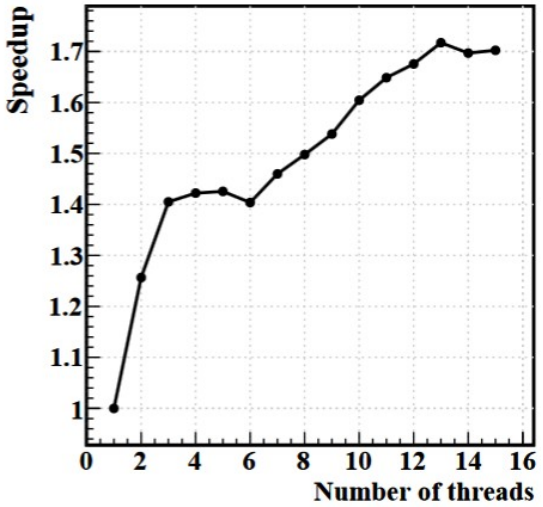
- ✓ Performance optimization (parallelization of most time-consuming hotspots).
- ✓ Tests of correctness and scalability of optimized code (Quality Assurance).

BmnGemStripDigitizer

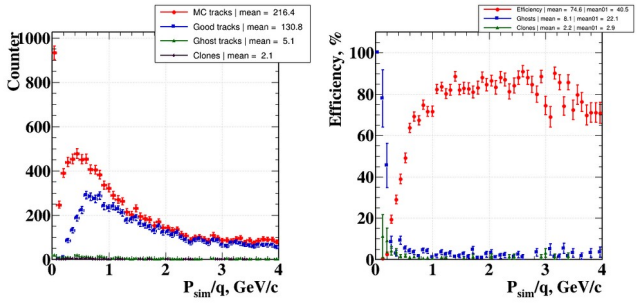
```
…
FairMCPoint* GemStripPoint;
Int_t NNotPrimaries = 0;
#pragma omp parallel
#pragma omp for schedule(dynamic)
for (UInt_t ipoint = 0; ipoint < fBmnGemStripPointsArray->GetEntriesFast();
ipoint++) {
GemStripPoint = (FairMCPoint*) fBmnGemStripPointsArray->At(ipoint);
…
```

OpenMP parallelization

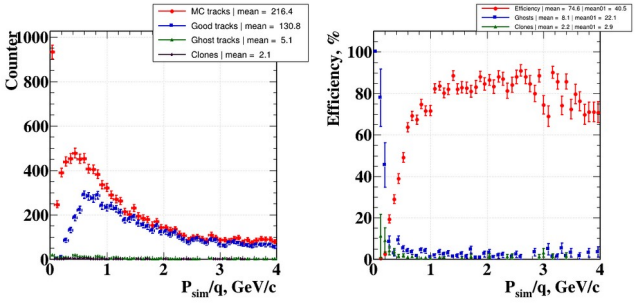




Scalability of the BmnRoot simulation parallelized with OpenMP



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 optimization of magnetic field interpolation and extrapolation.
- Optimization of memory access patterns.
- Optimization of Kalman filter software implementations.

```
#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 plans:

- Hybrid technologies (CPU+accelerator: GPGPU).

Summary

- Performance studies are performed and performance bottlenecks of **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 operations with magnetic field including data input.
- It is planned to perform scalability study of **PROOF** version of reconstruction on the cluster and integrate **PROOF** into reconstruction in SRC.
- Memory access errors have been removed in significant part of **BmnRoot** code.

.

Thank you for attention