Spoke 2 use cases

MS10 KPI report

Pipeline for GEANT4 simulations in HPC environments, sub-project of Pipeline optimization for space and ground based experiments (PSGE)

V. Fioretti[1, 2], A. Ciabattoni[1, 2, 3]

[1] INAF OAS – Bologna, Via P. Gobetti 101, 40129 Bologna

[2] Fondazione ICSC

[3] Dipartimento di Fisica e Astronomia, Universita` di Bologna, Via P. Gobetti 93/2, 40129 Bologna

Spoke	2
WP	3
Use case short name	PSGE
Use case ID	UC2.3.6
Expected Completion	1/8/2025

1. Introduction

The Geant4 Monte Carlo toolkit [R1, R2, R3] is an open-source, C++ based, particle transport code widely used to simulate high-energy instruments and radiation shielding of Astrophysics space missions and on-ground high-energy experiments. Geant4 is the simulation tool of choice for, e.g., the currently operative ESA XMM-Newton, ASI AGILE, and the design of the proposed COSI and Athena missions. The European HORIZON2020 (AHEAD2020 [R4]) and ESA (AREMBES [R5] and EXACRAD [R6]) funded projects aimed in recent years to update and validate the Geant4 physics library, and several Geant4-based simulation frameworks (ESA GRAS [R7], MEGALib [R8]) are available for building verified user-friendly applications. The new challenge of modern large-scale simulations is handling the production of increasingly large data sets, memory and computing requirements, which bring the development of next-generation simulation frameworks to the realm of HPC computing. Starting from the 10.0 release, Geant4 allows running simulations on multiple threads at the event level, where an event is the simulation of a primary particle (and its secondaries). The memory consumption can then be reduced by sharing the objects, or parts of them, that are invariant during the event loop such as the geometry and the physics tables. Message Passing Interface (MPI) is the standard for message passing on HPC for process-level parallelisation, with several open-source implementations (e.g. OpenMPI). The G4MPI library [R9] is the only plug-and-play Geant4 interface with MPI currently available, released as part of the Geant4 example applications. G4MPI uses a master/worker model, where the master sends commands/input to workers and workers send back to master results, a common job control and submission system of HPC architectures, used e.g. in CINECA large-scale computing facilities.
Multi-threading and multi-core Geant4 applications require a dedicated HPC-ready framework with ad-hoc I/O interfaces (e.g. databases). Containerisation and modularity are the key requirements for building portable, multi-purpose applications.

1.1. Purpose

The "Pipeline for GEANT4 simulations in HPC environments" sub-project of the Spoke 2 WP3 Flagship UC2.3.6 [A1, A2] builds a multi-purpose Geant4-based framework for Monte Carlo simulations tailored for HPC environments (BoGEMMS-HPC), using the simulation of the NASA COSI (Compton Spectrometer and Imager) Anti-Coincidence System (ACS) as a test case. The project aims to apply

methodologies for multi-threading and multi-node computation in a pipeline for Geant4 simulations in HPC architectures while exploring new I/O interfaces (e.g. CAD geometries, databases).

This document describes the finalization of the BoGEMMS-HPC framework, the installation container and usage description by including the MongoDB database as an option for the simulation output, that allows simultaneous multithreading writing access. We also report the impact on the BoGEMMS-HPC computing performance of the MongoDB database.

An example for using BoGEMMS-HPC and analyze its output is provided in the ICSC Spoke 2 github repository, under the PSGE project:

https://github.com/ICSC-Spoke2-repo/PSGE/tree/main/BoGEMMS-HPC_test_case

1.2 Scope

The present document is part of the KPI2.3.4.3, which is aimed at the release of the BoGEMM-HPC framework and documentation by the end of M10.

2. Related documents

2.1. Applicable documents

[A1] "A landscape recognition in the context of Astroparticle Physics, Astrophysics and Gravitational Waves", Spoke 2 CN-HPC WP 3 Working Group

[A2] "Pipeline optimization for space and ground based experiments (PSGE)", Spoke 2 Flagship Use Case WP3, UC2.3.6

[A3] A. Ciabattoni et al., "Verification of optical processes in Geant4 for the simulation of the COSI mission: using simple slabs as test case", INAF Report, 2023

[A4] A. Ciabattoni et al., "Validation of optical processes in Geant4 for the simulation of the COSI mission: the CLAIRE simulation", INAF Report, 2023

[A5] A. Ciabattoni & V. Fioretti, "BoGEMMS-HPC validation test: comparison with standard BoGEMMS with no MT and MPI", INAF Report, 2023

[A6] V. Fioretti & A. Ciabattoni, "Pipeline for GEANT4 simulations in HPC environments, sub-project of Pipeline optimization for space and ground based experiments (PSGE)", MS7 KPI Report, March 2024 https://drive.google.com/file/d/1p6zeKekNMb4mIQ72MBVgjgtPsqp4Dr9k/view

[A7] V. Fioretti & A. Ciabattoni, "Pipeline for GEANT4 simulations in HPC environments, sub-project of Pipeline optimization for space and ground based experiments (PSGE)", MS8 KPI Report, June 2024 https://drive.google.com/file/d/1Ky3K7LP3CZdBvI_RoFxEdyr4JfOpxla9/view

2.2. Reference documents

[R1] S. Agostinelli, et al. "Geant4—a simulation toolkit", NIMPA, 506, 250 (2003)

[R2] J. Allison, et al., "Geant4 developments and applications", ITNS, 53, 270 (2006)

[R3] J. Allison, et al., "Recent developments in GEANT4", NIMPA, 835, 186 (2016)

[R4] http://ahead.astropa.inaf.it/

[R5] C. Macculi et al., ATHENA Radiation Environment Models and X-ray Background Effects Simulators (AREMBES) ESA contract No. 4000116655/16/NL/BW, 2020 https://openaccess.inaf.it/handle/20.500.12386/33732

[R6] S. Molendi et al., "Experimental Evaluation of Athena Charged Particle Background from Secondary Radiation and Scattering in Optics (EXACRAD) ESA contract n° 4000121062/17/NL/LF, 2021

[R7] https://essr.esa.int/project/gras-geant4-radiation-analysis-for-space

[R8] https://megalibtoolkit.com

[R9] A. Dotti et al., "Extending Geant4 Parallelism with external libraries (MPI, TBB) and its use on HPC resources," *2015 IEEE Nuclear Science Symposium and Medical Imaging Conference (NSS/MIC)*, San Diego, CA, USA, 2015, pp. 1-2, doi: 10.1109/NSSMIC.2015.7581867.

[R10] A. Bulgarelli et al., "BoGEMMS: the Bologna Geant4 multi-mission simulator", Proc. SPIE, 845335, 2012

[R11] V. Fioretti et al., "Monte Carlo simulations of gamma-ray space telescopes: a BoGEMMS multi-purpose application", Proc. SPIE, 91443N, 2014

[R12] J. A. Tomsick et al. for the COSI Collaboration, "The Compton Spectrometer and Imager Project for MeV Astronomy", COSI White Paper, 2021

3. MongoDB implementation

MongoDB[1] is a non-relational database that stores data in a non-tabular format (NoSQL). Using the MongoDB C++ driver in the BoGEMMS-HPC framework, the user can write the simulation output to a JSON-like BSON (binary JSON) format. Instead of tables and rows, the data is stored as documents in collections. Different types of data can be stored in the same collections. Each BoGEMMS-HPC FITS event row, that records the entrance and exit of a particle in a sensitive volume, is saved as a document in a collection. Each document is a JSON-like object containing key-value pairs. A database contains multiple collections.

MongoDB allows the threads to simultaneously access the database by using a connection pool, a cache of open, read-to-use database connections that allows sharing the same client instance among the threads.

3.1 Singularity container

- Container name: g4_11_1_HPC_v2.sif
- Built with: singularity version 3.6.4-1.el7
- Built on: Ubuntu 22.04
- List of installed software/libraries:
  - Cfitsio v3.47
  - Openmpi v4.1.5
  - SQLite
  - Geant4 v11.1
  - G4-mpi
  - MongoDB and MongoDB C++ driver (v4.0.0)

3.2 BoGEMMS-HPC installation

The MongoDB feature is turned off by default. To activate it, during the configure the user should run:

```
cmake -DMONGODBBUILD=ON <path-to>/BoGEMMS-HPC
make
```

[1] https://www.mongodb.com/

4. Simulation performance

The BoGEMMS-HPC framework computing performance when using the MongoDB output interface is tested against the baseline FITS and SQLite database on a computing node of the INAF OAS cluster (Centos Linux 7, x86_64 architecture, 2 sockets, 10 cores per socket).

The test case chosen for this evaluation is the NASA COSI anticoincidence system, where an X-ray photon is emitted towards a scintillation panel and the optical photons hitting the read-out device are collected to form the signal. The output of the performance test is the wall time and the speed-up, computed with separate tests for multi-threading (MT) and MPI multi-task (MPI) parallelism. We iteratively executed simulations for each case while incrementally increasing the number of threads/tasks utilised.

4.1 MT

Figure 1 compares the wall time and speed-up resulting from writing FITS files, including compression, SQLite and MongoDB database files, for the same number of simulated photons but changing the parallel threads from 1 to 20. With a single thread, the fastest method to write the simulated events is with an SQLite database file, with a wall-time about 40% lower. The FITS output is also slightly better than MongoDB, which likely adds the connection set-up to the server to the time of writing. MongoDB becomes 30% faster than FITS files already with two threads, and a performance that is still lower than a simple SQLite interface, but that gets closer as the number of threads increases. At 20 threads, the computing time is almost the same for the two database writing options. In terms of speed-up, increasing the threads only improves the FITS performance by about a factor 2, because the compression takes most of the writing time. Up to 6 threads, MongoDB and SQLite speed-up is the same, because the threads are still not concurrent. When they start to simultaneously access the database, in SQLite they are serialized and the speed-up bends towards a plateau while the MongoDB performance keeps improving, with a speed-up of 10x for 20 threads.

For applications with few threads, or serialized, the best output data option is SQLite, while MongoDB is the database of choice for HPC architectures.

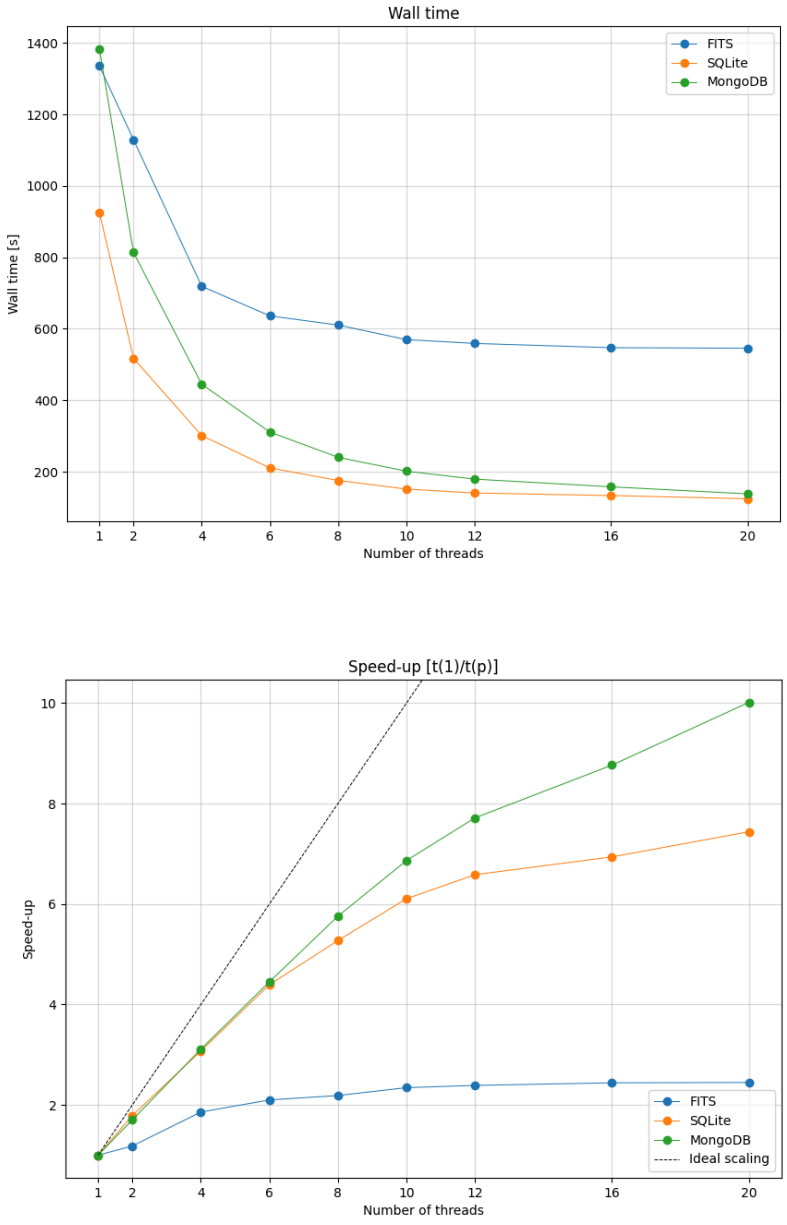

Figure 1: Multithreading wall time (top) and speed-up (bottom) as a function of the number of tasks.

4.2 MPI

In the multi-node simulation offered by MPI, the SQLite output interface, that offers the best performance for a single simulation instance, is always the fastest going from 1 to 20 tasks, with a

speed-up that is almost linear for the three options (if the FITS compression is on). MongoDB is the least favourite choice when using MPI.

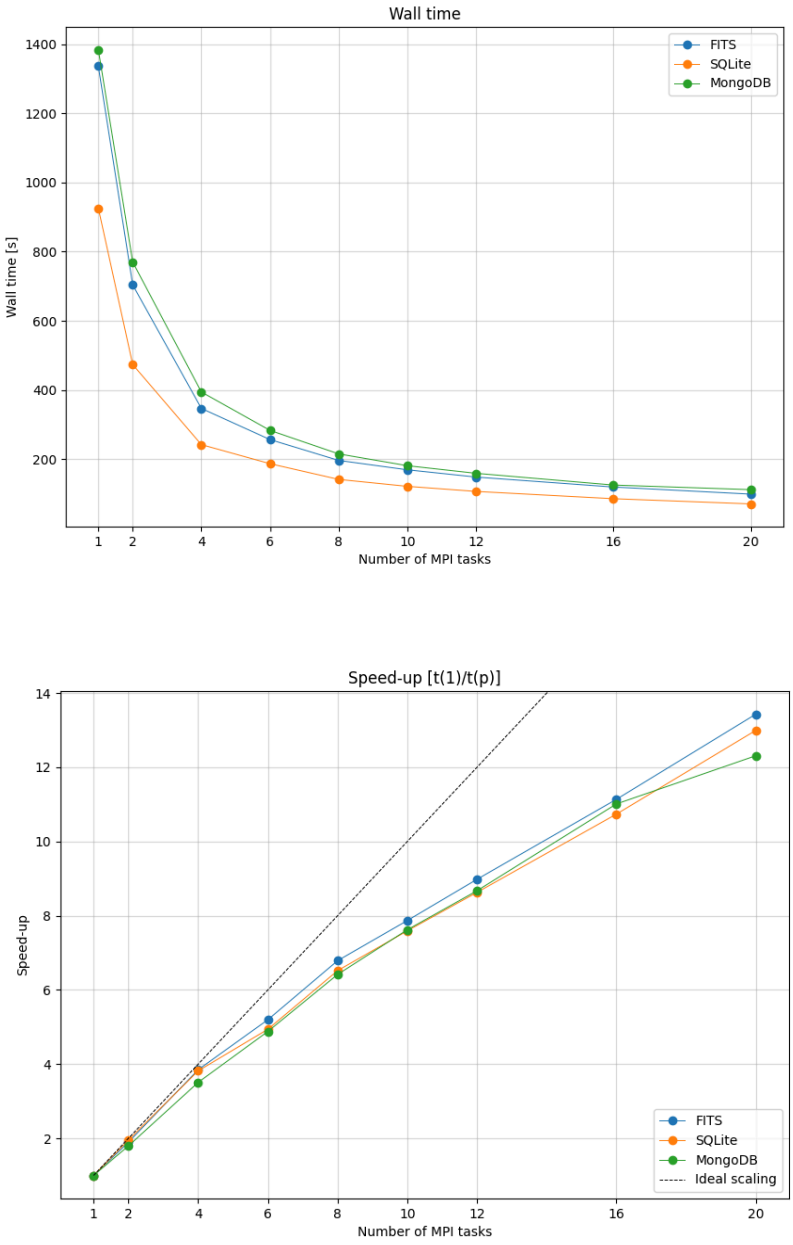

Figure 2: MPI multi-task wall time (top) and speed-up (bottom) as a function of the number of tasks.

4.3 MPI vs MT

The total simulation time going from multi-threading to multi-task is slightly lower for the latter, but we predict an even better performance of MongoDB if run on an HPC architecture where hundreds of threads are available [AD7]. The standard FITS files are instead best suited for MPI applications.

5. The BOGEMMS-HPC test case

The test case, including simulation and analysis code examples, is available in the ICSC Spoke 2 github repository: https://github.com/ICSC-Spoke2-repo/PSGE/tree/main/BoGEMMS-HPC_test_case

The test case involves the irradiation with 122 keV photons of a cesium iodide (CsI) module, contained in an aluminum casing and coupled with a photomultiplier tube (PMT). The X photons hit perpendicularly the CsI in 40 different positions. When a 122 keV photon hits the CsI, optical photons are generated inside the scintillator and part of them is collected and absorbed by the PMT. The final output is the relative response of the PMT for each position, which is proportional to the average number of optical photons detected, compared to the corresponding experimental data obtained in the laboratory.

5.1 Usage

There are two directories: 'Simulations' and 'Analysis'. 'Simulations' contains the files for performing the simulation, while 'Analysis' provides the files for analyzing the simulation output and plotting the results.

Simulation:

Files provided in 'Simulations':

- beam.mac: macro file for the 122 keV photons
- runCLAIRE.conf: configuration file for the simulation
- runCLAIRE.py: script to start the simulation
- runCLAIRE.txt: input file for runCLAIRE.py
- bogemms_container.slurm: file to start the simulation with slurm
- bogemms_container.sh: file to start the simulation with nohup
- geom/: directory with the geometry files
- phys/: directory with the physics files

- cad_files/: directory with the CAD files (for the casing and Teflon)
- config/: directory containing the coordinates of the beam positions (stored in claire_beams.dat)

Update 'runCLAIRE.txt' to define the parameters of the simulation:

- run_start and run_stop: define the initial and final run (the total number of runs will be run_stop - run_start + 1)
- csi_absl: define the CsI absorption length in mm
- refl_type: define the reflective surface type
- G4PATH: define the path where to create the simulation directory
- N_in: define the number of events (122 keV photons) to simulate
- num_threads and num_tasks: define the number of threads and MPI tasks
- cad_path: define the path to the CAD file
- job_name: define the job name (if executed with slurm)
- isslurm: define how to run the simulation (0: nohup, 1: SLURM)

If you use the SLURM scheduler, update 'bogemms_container.slurm' to match the the number of threads and/or tasks used, the correct partition and the correct paths for the container and executable.

When using MPI, the total number of events (N_in) is evenly distributed among the MPI tasks. However, N_in should be a multiple of the number of MPI tasks. If not, each task will receive N_in / num_tasks, rounded down to the nearest integer, which results in fewer total simulated events than intended.

When using multithreading, Geant4 automatically handles the distribution of events among the threads. In this case, N_in is always preserved, even if it is not a multiple of the number of threads. However, the workload distribution among the threads may be uneven.

When using both MPI and multithreading, Geant4 first distributes the events evenly among the MPI tasks. Within each MPI task, the events are then divided among the threads (whose number is specified by 'num_threads').

To start the simulation run:

$ python3 runCLAIRE.py runCLAIRE.txt

At the end of the simulation, the results are stored in the following directory:

{G4PATH}/{bgo_absl_type}/REFL_TYPE{refl_type}/{N_in}/BEAM{beam_id}/run{run_id}/

where 'beam_id' goes from 1 to 40 and 'run_id' from 'run_start' to 'run_stop'.

Analysis

In 'Analysis' you can find the following files:

- filterCLAIREPMT.py: python script to filter the simulation output
- plotCLAIREPMT.py: python script to plot the result
- config/: directory containing the file (CLAIRE_pulse.dat) with the experimental relative response for each beam position

Use filterCLAIRE.py to filter the output files from the simulation and write the relevant information to 'PMT.dat'. In 'Analysis/' run:
python3 filterCLAIREPMT.py <SIM_DIR> <run_start> <run_stop>

Example:
$ python3 filterCLAIREPMT.py ../Simulations/2000/REFL_TYPE14/1000/ 1 1

This will create a directory like:
{bgo_absl_type}/REFL_TYPE{refl_type}/{N_in}/
In this directory a file called 'PMT.dat' is created with the following columns:

- EventID: ID of the event
- N_BEAM: ID of the beam
- Edep[keV]: energy deposit [in keV] of the detected optical photon

Finally, to plot the relative response for each beam position, compared with the experimental one, run:
$ python3 plotCLAIREPMT.py <file1> <file2> ...

Example:
$ python3 plotCLAIREPMT.py 2000/REFL_TYPE14/1000/PMT.dat

6. Summary

The BoGEMMS-HPC framework will include three types of simulation output data format:

- FITS file;
- SQLite database file;
- MongoDB database collection.

They provide different writing access times depending on the type of architecture, with FITS best suited for MPI-based multi-task simulations and MongoDB for multi-threading in HPC architectures. A simulation test case, that can be used as a tutorial to run and analyse BoGEMMS-HPC, is uploaded in the ICSC Spoke 2 github repository.