GEANT4 Montecarlo simulations: a multithread approach

Bruno Luigi Martino[*] [1], **Simone Lotti**[2],**Ugo Zannoni**[3], **Giorgio Patria**[4]

[1] *CNR-IASI: Institute for Systems Analysis and Computer Science*
Via dei Taurini 19, 00185 Roma, Italy
bruno.martino@iasi.cnr.it
[2] *IAPS/INAF: Institute for Space Astrophysics and Planetology*
Via Fosso del Cavaliere 100, 00133 Roma, Italy
simone.lotti@iaps.inaf.it
[3] *IAPS/INAF: Institute for Space Astrophysics and Planetology*
Via Fosso del Cavaliere 100, 00133 Roma, Italy
ugo.zannoni@iaps.inaf.it
[4] *ITIS G. Galilei*
Via Conte Verde, 51, 00185 Roma, Italy
giorgio.patria@gmail.com

In this paper we will show the hardware and software implementation of the gforcr computing cluster operating in the LCD laboratory at the IAPS / INAF headquarters in Rome. The computing system is used to perform Monte Carlo simulations using the GEANT4 toolkit in multithread mode. The simulations are designed to evaluate and reduce the background of the microcalorimeter on board the ATHENA mission whose launch is expected in 2018.

The Golden Age of Cataclysmic Variables and Related Objects IV
11-16 September, 2017
Palermo, Italy

[*]Speaker.

1. The context

Athena (**A**dvanced **T**elescope for **H**igh **EN**ergy **A**strophysics) is ESA's next large X-ray observatory class mission concept, developed from April to December 2011. On 27 June 2014, Athena was selected as the second L-class mission in ESA's Cosmic Vision 2015-25 plan: the telescope project was born out of the reformulation of several older missions. This is the biggest mission of X-ray astronomy in the next 30 years.

Particles depositing energy in the energy bands of the Athena detectors produce a background that decreases the instrument sensitivity.

There are no experimental data about the non-X-ray background level experienced by X-ray microcalorimeters in L2, and since ATHENA can be the first X-ray large mission placed there, we estimate it with detailed Monte Carlo simulations using the GEANT4 [**?**].

For any satellite operating in the X-ray band the background is composed of an internal particle component and a diffuse com- ponent. The former is generated by particles traveling through the spacecraft, releasing energy inside the detector, also creating swarms of secondary particles (mostly electrons) along the way.

The standard approach is to exploit Monte Carlo simulations to predict and analyze the background components.

2. The importance of simulations

Computational physics has been said to constitute a third way of doing physics, comparable to theory and experiment [**?**]. What is the role of theory or simulation in physics today? In the last half century we have turned to computer simulations as a very powerful way of providing detailed and essentially exact information about physics related problems. A (computer) simulation applies mathematical methods to the analysis of real-world problems and predicts what might happen depending on various actions/scenarios. The simulation activities in the field of high energy physics are very useful for:

- understanding signatures of new physics models
- elementary Particle reconstruction, calibration
- demonstration of physics analysis methods
- detector design and optimization

Simulations must be done when:

- Doing the actual experiments is not possible
- The cost in money, time, or danger of the actual experiment is prohibitive
- Various alternatives are examined
- The system does not exist yet!

HEP detectors design and optimization is complicated, huge and very expensive; simulations are very useful to study the interactions particle/detectors, to optimize design and cost/benefits ratio and to compute the background contamination and signal efficiency.

3. Geant4

A Montecarlo simulation is a numerical simulation method which uses sequences of random numbers to solve complex problems e.g. mathematics, physics and chemistry. Other numerical methods usually need a mathematical description of the system (e.g. partial differential equations). MC assumes the system is described by probability density functions [**?**]. The main reasons why random numbers are used are:

- some variables vary randomly in time
- there is neither analytical solutions nor analysis available
- problems of high dimension such as integrals in multi-dimensions

The simulation of the interactions between particles and matter in studies for developing HEP detectors requires long computing times. A powerful tool used by the scientific community to perform this task is GEANT4. GEANT4 is a framework which provides interfaces for integration with user programs: every simulation is a compilable C++ program. It does not offer a command line interface or a GUI to define the geometry; the user supplies the geometry, list of physics processes, initial states, definition of particles. GEANT4 provides a wide variety of physics models of particle interactions with matter and supplies the information about particles passing through user-defined inspection points named detectors.

The categories of physical processes are:

- standard electromagnetic
- low energy electromagnetic
- hadronic

All the particle decays and interactions are handled by processes; each particle has its own list of applicable processes. A list of processes for each kind of particles to use in the simulation is called physics list. Particles can interact with materials, geometrical boundaries and fields; even their transportation is a process. There is also a shower parameterization process which can take over from transportation.

4. Build a GEANT4 simulation

The step needed to build a new simulation are the following:

- creation of the *main ()* function
- description of detector

- creation of user action classes
- setup the user interface
- selection of the physics processes
- decide the beam and generate primary event

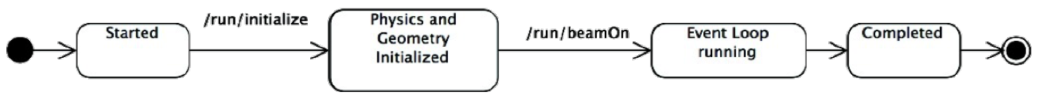


Figure 1: GEANT4 application flow

As explained before, Geant4 does not provide the *main ()* function because is a toolkit; the main is part of the user application. Inside the *main ()* function, the user must instantiate **G4RunManager** (or derived class) and notify the **G4RunManager** user classes.

The mandatory classes within a complete simulation are: **G4VUserDetectorConstruction** (experimental set-up description), **G4VUserPhysicsList** (physics you want to activate selection), **G4VUserPrimaryGeneratorAction** (primary events generation).

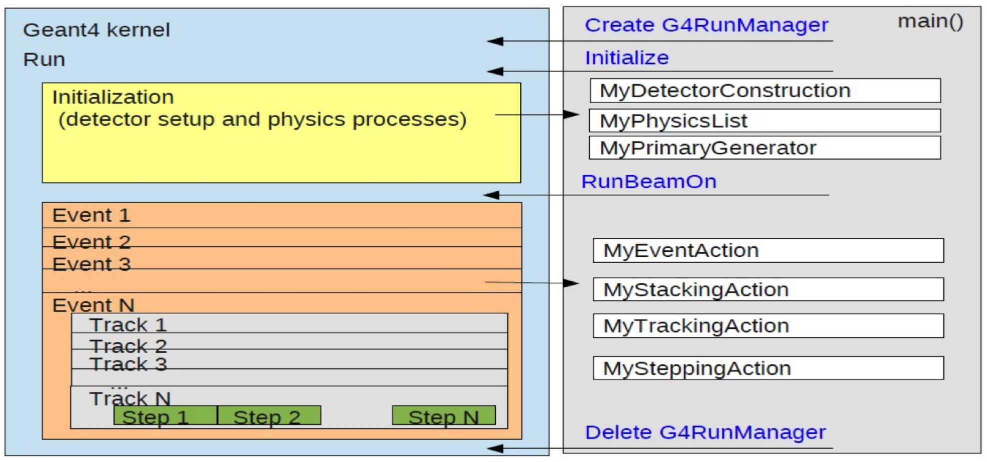


Figure 2: GEANT4 application structure

5. GFORCR Beowulf cluster

Beowulf cluster denotes a computer cluster of standard desktop computers which are interconnected using a local area network (LAN) with freely available software installed and shared among them [?]. The most important point is that the computers that are part of the cluster are no longer used as workstations, but are used as nodes in a High Performance Computing (HPC) system. The choice of having more physical machines dedicated to clustering purposes is not always the best solution because a virtualization environments make it easier to implement and is able to remove the hardware dependencies [**?**].

While a virtual machine abstracts away the hardware, container abstraction happens at the operating system level. Each VM runs not just a full copy of an operating system, but a virtual copy of all the hardware that the operating system needs to run. This is one of the major reasons why containers are often used for running specific applications: less CPU and memory usage; VMs take up a lot of system resources [**?**].

The GFORCR cluster built in the IAPS LCD laboratory is designed around a two layers model. The first layer consists of physical computers that implement the computing structure, the network interconnections between the various elements and the storage system devoted to the end users home folder. The second layer is a logical layer composed by a set of Linux Container on top of physical hosts devoted to the computing queue management and jobs control.

Proxmox VE helped us to organize the first layer in a very effective mode. Proxmox VE is open source based on a Debian distribution and can be installed as standalone system; alternatively, advanced users can install it on top of an existing Linux system (detailed knowledge is necessary). A Proxmox container can be managed through VE web interface by connecting an internet browser to 8006 port. In this way new containers can be created using a set of templates. All the containers in GFORCR cluster access the same shared home folder. This is a very important constrain because when a parallel application is running every node must keep user data synchronized and in a consistent state; Proxmox VE makes it possible to make shared storage resources visible to a container, in our case based on the CEPH distributed file system.

This kind of architecture is useful also in solving the following kinds of problems:

- High Availability (failover and failback systems)
- Balancing (high performances systems)
- Quick reconfiguration (very flexyble systems)

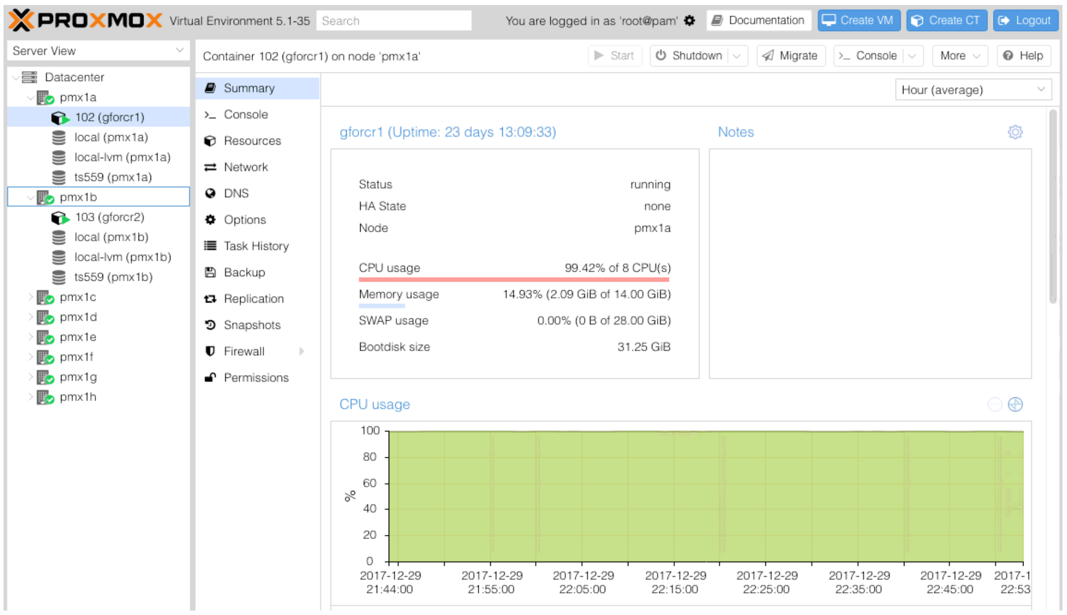
We are particularly interested in obtaining highly re-configurable systems so that we have several easily accessible computing environments.

Through the WEB console it is possible to control the start, stop and migration of containers between the hosts that compose the cluster as well as verify the occupation status of the resources such as CPU, central memory, number of cores etc. etc. Even the resources assigned can be redefined at any time without problems.

To run and control multiple instances of a program running on a machine, a resource manager (eg SLURM) or a queue manager (eq LSF) are good choices; queue managers are normally used in multi-cpu environments but are also effective in clustered environments. The second layer of GFORCR cluster is based on OpenLava. OpenLava, a 100% free, open-source, IBM platform LSF compatible workload scheduler [**?**].

6. Simulations environment

The version of GEANT toolkit installed on the gforcr containers is 4.10.02.p03 (the latest version available is 4.10.03 but is performed very slowly on LCD computers). OpenLava command



gforcr1 (Uptime: 23 days 13:09:33)

Status	running
HA State	none
Node	pmx1a
CPU usage	99.42% of 8 CPU(s)
Memory usage	14.93% (2.09 GiB of 14.00 GiB)
SWAP usage	0.00% (0 B of 28.00 GiB)
Bootdisk size	31.25 GiB

Figure 3: GFORCR status window

bjobs help to check the simulation jobs status

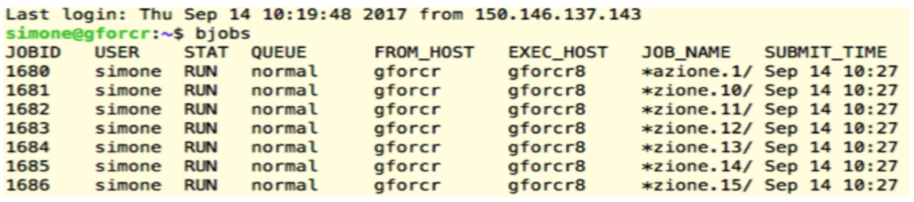

```
Last login: Thu Sep 14 10:19:48 2017 from 150.146.137.143
simone@gforcr:~$ bjobs
JOBID   USER    STAT  QUEUE      FROM_HOST   EXEC_HOST   JOB_NAME   SUBMIT_TIME
1680    simone  RUN   normal     gforcr      gforcr8     *azione.1/ Sep 14 10:27
1681    simone  RUN   normal     gforcr      gforcr8     *zione.10/ Sep 14 10:27
1682    simone  RUN   normal     gforcr      gforcr8     *zione.11/ Sep 14 10:27
1683    simone  RUN   normal     gforcr      gforcr8     *zione.12/ Sep 14 10:27
1684    simone  RUN   normal     gforcr      gforcr8     *zione.13/ Sep 14 10:27
1685    simone  RUN   normal     gforcr      gforcr8     *zione.14/ Sep 14 10:27
1686    simone  RUN   normal     gforcr      gforcr8     *zione.15/ Sep 14 10:27
```

Figure 4: BJOBS command output

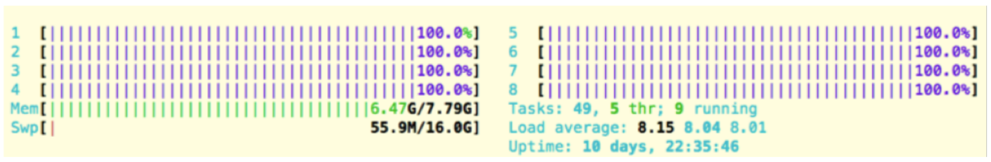
In order to make simpler by end users the use of our platform we wrote the following procedures:

- **G4Cprep** prepares environment by creating a folder structure each containing the macro file associated with a single simulation (number of events, random seed etc.) using bash commands
- **G4Crun** executes multiple instances of the simulation process (the maximum number depends on the user profile) using OpenLava *bsub* command

The bash *htop* command allows to check the CPUs load.

7. Multithread GEANT4 applications

A process is an executing (i.e., running) instance of a program. Processes are also frequently referred to as tasks. By running N instances of a program, every instance is expected to work on a



```
1  [||||||||||||||||||||||||||||||||||||||||100.0%]   5  [||||||||||||||||||||||||||||||||||||||||100.0%]
2  [||||||||||||||||||||||||||||||||||||||||100.0%]   6  [||||||||||||||||||||||||||||||||||||||||100.0%]
3  [||||||||||||||||||||||||||||||||||||||||100.0%]   7  [||||||||||||||||||||||||||||||||||||||||100.0%]
4  [||||||||||||||||||||||||||||||||||||||||100.0%]   8  [||||||||||||||||||||||||||||||||||||||||100.0%]
Mem[||||||||||||||||||||||||||||||||||||6.47G/7.79G]   Tasks: 49, 5 thr; 9 running
Swp[|                                  55.9M/16.0G]   Load average: 8.15 8.04 8.01
                                                      Uptime: 10 days, 22:35:46
```

Figure 5: HTOP command output

different core of the CPU (cores >= N; each instance is a process). Having N instances of a running program means to multiply the required computing resources for N:

- if a program requires 1.5 GB of RAM and is launched on 8 core processor it requires 12 GB
- the OS also requires RAM to work
- if the machine has only 8 GB will start to swap on the disk dramatically reducing performances

A thread is an entity within a process that can be scheduled for execution. Benefits of Threads usage are:

- resource sharing; threads within a certain process share its address space and can therefore use shared variables
- economy; threads are often referred to as light-weight processes
- scalability; for multi-threaded processes it is much easier to make use of parallel processing
- programming complexity reduction; since problems can be broken down into parallel tasks, rather than more complex state machines

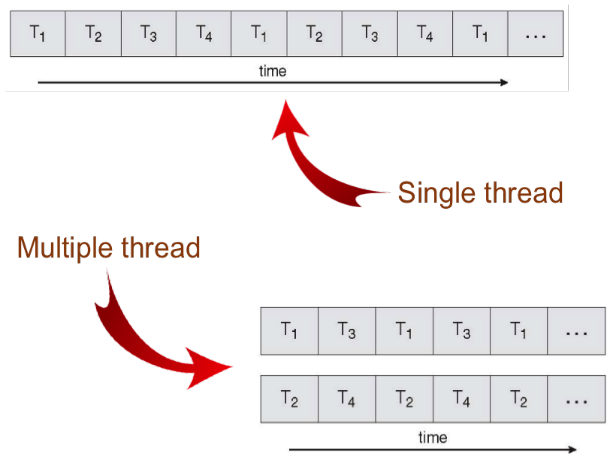


Figure 6: Single thread vs multiple thread

Given a computer with K cores, the design goal of multithreaded Geant4 was to replace K independent instances of a Geant4 process with a single, equivalent process with K threads using the many-core machine in a memory-efficient, scalable manner. The corresponding methodology involves transforming the code for thread safety and memory footprint reduction.

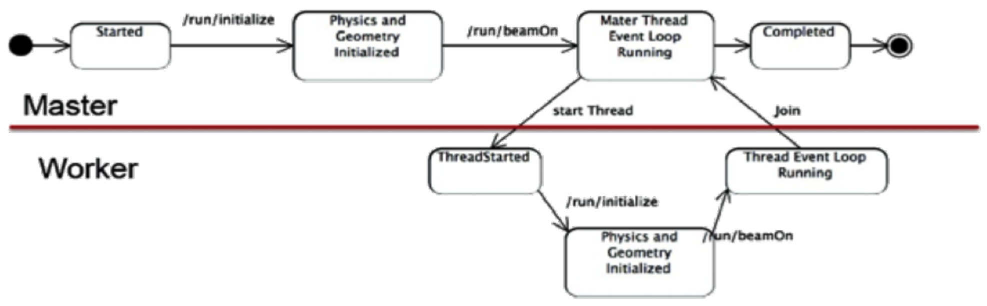
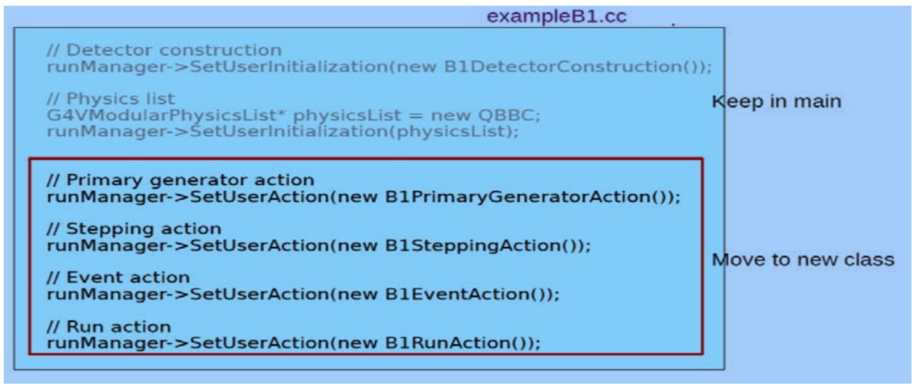

Started
/run/initialize
Physics and Geometry Initialized
/run/beamOn
Mater Thread Event Loop Running
Completed
Master
start Thread
Join
Worker
ThreadStarted
Thread Event Loop Running
/run/initialize
Physics and Geometry Initialized
/run/beamOn

Figure 7: GEANT4 multithread application flow

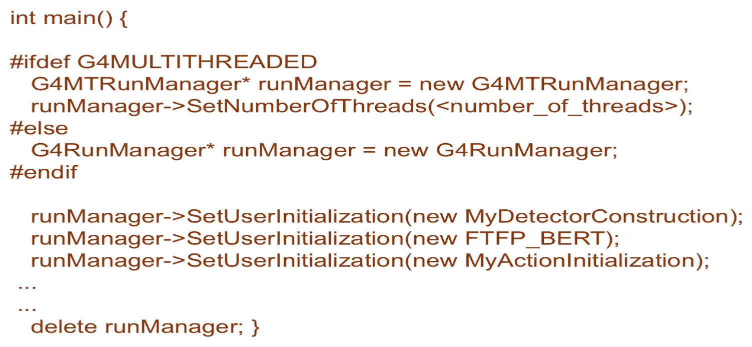
The multithreading option is based on a master-worker model in which one control sequence (the master) is responsible for initializing the geometry and physics, and for spawning and controlling worker threads. Workers are responsible for the simulation of one or more events. Master writes shared memory; when threads start they cannot change the shared memory. Threads do not exist before /run/beamOn. The migration strategy from a serial GEANT4 application to a multi-threaded one is to move user action class from *main ()* to a new **ActionInitialization** class:



```
// Detector construction
runManager->SetUserInitialization(new B1DetectorConstruction());

// Physics list
G4VModularPhysicsList* physicsList = new QBBC;
runManager->SetUserInitialization(physicsList);

// Primary generator action
runManager->SetUserAction(new B1PrimaryGeneratorAction());

// Stepping action
runManager->SetUserAction(new B1SteppingAction());

// Event action
runManager->SetUserAction(new B1EventAction());

// Run action
runManager->SetUserAction(new B1RunAction());
```

Figure 8: GEANT4 serial organization template

- **G4MTRunManager** is the replacement of G4RunManager for multi-threading mode. At the very end of Initialize() method, **G4MTRunManager** creates and starts worker threads.
- **G4VUserActionInitialization** is a newly introduced class for the user to instantiate user action classes. **G4VUserActionInitialization** has two virtual method to be implemented, one is Build() and the other is BuildForMaster(). **Build()** should be used for defining user action classes for worker threads as well as for the sequential mode, **BuildForMaster()** should be used for defining only the UserRunAction for the master thread.



```
int main() {

#ifdef G4MULTITHREADED
   G4MTRunManager* runManager = new G4MTRunManager;
   runManager->SetNumberOfThreads(<number_of_threads>);
#else
   G4RunManager* runManager = new G4RunManager;
#endif

   runManager->SetUserInitialization(new MyDetectorConstruction);
   runManager->SetUserInitialization(new FTFP_BERT);
   runManager->SetUserInitialization(new MyActionInitialization);
 ...
 ...
   delete runManager; }
```

Figure 9: GEANT4 MultiThread organization template

8. Conclusions

One of the biggest advantages of using the Beowulf **gforcr** cluster is its ease of use and quick reconfiguration where necessary due to the two-level model used. Another very important advantage is its scalability It is possible to increase the computing power simply by adding new calculation nodes and modifying the OpenLava configuration. The Ceph distributed file system in use aims primarily for completely distributed operation and fault-tolerant without a single point of failure. From a point of view closely related to GEANT4 simulations, gforcr allows, in addition to increasing the speed of calculation, to exceed the maximum number of events that can be managed. Furthermore, it is cheap and highly optimized.

9. Acknowledgement

We thank Franco Giovannelli, Piero Ubertini and the wonderful organizing committee. A very special thanks to our friend Memmo Federici, who has never really left us but it is always with us. Thank you Memmo!

References

[1] S. Lotti et al., *In-orbit background of X-ray microcalorimeters and its effects on observations*, *Astronomy & Astrophysics, A54*, [2014]

[2] D. M. Ceperley, *Microscopic simulations in physics*, *Reviews of Modern Physics, Vol. 71, No. 2*, [1999]

[3] K. Binder, D. Heermann, *Monte Carlo Simulation in Statistical Physics*, *Springer-Verlag*, [2010]

[4] R. Capuzzo Dolcetta et al., *High performance astrophysics computing*, *ASP Conference Proceedings, Vol. 453. p.379*, [2012]

[5] R. Gareth et al., *Evaluation of containers as a virtualisation alternative for HEP workloads*, *Journal of Physics: Conference Series 664* [2015]

[6] *LXC, linux conainers*, *URL http://linuxcontainers.org/.* [Accessed: 2017-06-02]

[7] A. Reuther et al., *Scheduler Technologies in Support of High Performances Data Analysis"*, *MIT* [2015]