A parallel implementation of Geant4 application using OOMPI

Luís Augusto Perles, Adelaide de Almeida
Departamento de Física e Matemática
Faculdade de Filosofia Ciências e Letras de Ribeirão Preto
USP - Brazil

Almost of all high energy Monte Carlo simulations require a considerably high CPU time to produce good results. Sometimes we could spent several months in such simulations to achieve low statistical uncertainties. In Medical Physics applications such as treatment planning system for clinical use we need to get a three dimensional (3D) dose distribution from a composition of several fields in few minutes. Standard libraries for high energy Monte Carlo simulations as Geant4, MCNP4, EGS4 can produce good results in few minutes when they run in parallel mode in computer clusters. In this work we present a full object oriented way to parallelize a Geant4 application used for simulate a 3D dose distribution in a simple phantom[1].

Parallel Geant4 applications have been developed with TOP-C library which makes use of MPI to control the execution of parallel tasks. The main problem of TOP-C is its structured design, that can lead to a break of the object oriented feature of all serialized Geant4 applications. OOMPI (Object Oriented MPI) is a C++ class library that implements MPI interface. With this library we can send and receive any object inherited from a special class called OOMPI_Datatype.

We have developed an application to run in one CPU to simulate the 3D dose distribution in a phantom made by a homogeneous box where its material and dimensions can be set by a macro file. The beam particle type and its energy can be selected in the same way as the phantom characteristics. The simulated absorbed dose has been stored in a ROOT 3D histogram class in double precision.

To parallelize that application we needed to add 2 classes, change other 2 classes and the `main()` function. As suggested in Geant4 examples for use with TOP-C we have added a class to handle the state of the random number generator for master and slave nodes. We also added a RunMananger class called ParRunManager responsible to manage the distribution of the jobs and the reception of the results. We have changed the phantom sensitive detector class in order to store data in a temporary histogram object and at the end of a job this object is sent by ParRunManager to the master node. The last one modified class was the analysis class in which we have added a method to merge node simulated data to a main histogram object. In the `main()` function the modifications were about initializing MPI properly.

The application can be started in batch mode as usual. The master node will send only a small number of histories for each node to be processed, in this case we chose 100,000 per node. After sending the number of history for a node the ParRunManager class will choose the seed for the random number engine and send it to that slave node. This process is repeated until all started slave nodes get their jobs. The next step is wait for a result that can be sent from any node in any order. When the master node receives a result it properly stores this result and send a new job to the same slave node. This cycle is repeated until all histories are processed.

References

[1] S. Agostinelli, J. Allison, and et al. Geant4–a simulation toolkit. *Nucl. Instrum. Methods A*, 506:250--303, 2003.