2006

Geant4 simulation in a distributed computing environment

Susanna Guatelli
University of Wollongong, susanna@uow.edu.au

Alfonso Mantero
INFN Sezione di Genova

Patricia Mendez Lorenzo
CERN

Jakub Moscicki
CERN

Maria Grazia Pia
INFN Italy

Publication Details

Guatelli, S., Mantero, A., Mendez Lorenzo, P., Moscicki, J. & Pia, M. Grazia. (2006). Geant4 simulation in a distributed computing environment. In B. Phlips (Eds.), IEEE Nuclear Science Symposium Conference Record (pp. 110-113). New Jersey, United States: IEEE.

Geant4 simulation in a distributed computing environment

Abstract
A general system to perform a Geant4-based simulation in a distributed computing environment is presented. The architecture developed makes the system transparent to sequential or parallel execution and to the environment where the simulation is run: a single machine, a local cluster or a geographically distributed grid.

Keywords
computing, environment, distributed, geant4, simulation

Disciplines
Engineering | Science and Technology Studies

Publication Details
Guatelli, S., Mantero, A., Mendez Lorenzo, P., Moscicki, J. & Pia, M. Grazia. (2006). Geant4 simulation in a distributed computing environment. In B. Phlips (Eds.), IEEE Nuclear Science Symposium Conference Record (pp. 110-113). New Jersey, United States: IEEE.

Geant4 Simulation in a Distributed Computing Environment

Susanna Guatelli, Alfonso Mantero, Patricia Mendez Lorenzo, Jakub Moscicki, Maria Grazia Pia

Abstract— **A general system to perform a Geant4-based simulation in a distributed computing environment is presented. The architecture developed makes the system transparent to sequential or parallel execution, and to the environment where the simulation is run: a single machine, a local cluster or a geographically distributed grid.**

Index Terms— **Parallel computing, Grid, Geant4, Monte Carlo.**

I. Introduction

Monte Carlo simulation is widely used in many experimental physics domains to study the effects of radiation. Simulation plays a fundamental role in detector design, in the evaluation of the physics reach of experiments, in the development and optimization of reconstruction and data analysis algorithms. Monte Carlo simulations are also developed for sensitive applications requiring high precision, like oncological radiotherapy and nuclear medicine, or mission critical studies in space science.

Monte Carlo codes often allow evaluating radiation effects with great accuracy; however, the precision of the simulation is usually achieved at the price of a long processing time. The lengthy execution time makes Monte Carlo simulation unsuitable to some experimental applications that would greatly profit of its precision, but would require a quick calculation response; in other cases precise simulation would require prohibitively long execution time to achieve adequate statistical significance. Various solutions have been developed in the past decades to speed up Monte Carlo simulations, such as variance reduction techniques, inverse Monte Carlo methods, approximations through analytical calculations and parameterization of detector response to particle exposure (also known in high energy physics experiments as *fast simulation*). These methods allow reducing the simulation execution time; nevertheless they also affect the precision of the simulation itself, by introducing approximations in processing the physics interactions of particles in the experimental set-up.

Manuscript received November 24, 2006.

S. Guatelli, A. Mantero and M. G. Pia are with INFN Sezione di Genova, Genova 16146, Italy (e-mail: Susanna.Guatelli@ge.infn.it, Alfonso.Mantero@ge.infn.it, MariaGrazia.Pia@ge.infn.it).

P. Mendez Lorenzo and J. Moscicki are with CERN, CH-1211 Geneva, Switzerland (e-mail:Patricia.Mendez.Lorenzo@cern.ch, Jakub.Moscicki@cern.ch).

Parallel execution has also been widely explored as a suitable technique to reduce the time for obtaining the results of a Monte Carlo simulation. This solution does not perturb the accuracy of the simulation results; nevertheless, it opens other kinds of problems, like the availability of extensive computing resources, and usually the need of manipulations of the simulation code for its execution in parallel mode.

Recent evolutions in software architecture concepts, such as component-based architecture, the wide availability of computing resources at affordable price, and new emerging concepts in parallel computing, like the grid technology [1], suggest to revisit the problem of parallel Monte Carlo simulation under a novel perspective.

This paper presents a system design and quantitative studies for applications of the Geant4 [2]-[3] simulation toolkit in a distributed computing environment. The present paper addresses the problem in a general way, by providing a design solution suitable to any Geant4 application and different distributed environments; both the execution in a conventional local computer farm and in a geographically spread grid environment are documented. Quantitative results from a systematic study in a realistic use case are presented.

II. Requirements

The execution of a simulation in a distributed computing environment is characterized by some technical requirements of general nature, irrespective of the specific simulation application. The high level user requirements are discussed in the two following subsections.

A. Simulation requirements

The distributed execution of a Monte Carlo simulation may be envisaged both in a local computing cluster and in a geographically spread grid. Ideally, the same software system should be able to handle both kinds of distributed computing environment.

The execution of a simulation in a distributed mode introduces some technical complexity with respect to sequential processing. The main technical requirements involve the management of random number generation and of external files needed for the simulation execution (such as, for instance, cross section files used by physics models in the Monte Carlo system). The management of random numbers also affects the reproducibility of the simulation.

Other requirements concern a reliable error recovery, to avoid the loss of partially executed simulations, monitoring the progress of simulation execution, and the traceability of failed tasks for debugging purposes.

The simulation processing must be split into small chunks for distributed execution. Monte Carlo simulation exhibits a natural subdivision into smaller independent units at the level of event processing. Parallel processing can be envisaged also at the level of a single track; however, this kind of parallelization is more complex, and parallelization at event level is sufficient to address the user requirements of execution speed in most typical use cases.

It would be desirable that the execution in parallel mode minimize the modifications to the application code with respect to its sequential execution. Ideally, the application code for sequential and parallel execution should be the same, and the execution mode, sequentially on a single processor, in parallel on a local computing or on a grid, should be completely transparent to the user.

B. *Analysis requirements*

A simulation application is usually complemented by data analysis to collect, manipulate and evaluate the simulation results. Typically, a simulation application produces analysis objects, like histograms or tuples. These objects require a persistent storage to be available to the user for visualization and further elaboration. Usually analysis objects are created at the beginning of a simulation run; histograms and tuples are filled at the appropriate level of the simulation. At the end of the simulation analysis objects contain the results of the whole simulation.

The distributed execution requires that analysis objects produced in parallel streams are combined: in practice, this means summing partial histograms and merging tuples, and making the final analysis results available to the user. This process must be automated and transparent to the user.

III. SOFTWARE TOOLS

The simulation application uses three software systems: the Geant4 toolkit, an AIDA [4] compliant analysis system, and the DIANE [5] distributed analysis environment.

These systems are all characterized by the same component-based architectural approach. Component-based architecture has emerged recently as a software technique facilitating the flexibility, extensibility and maintainability of software systems; the architectural model adopted by the three software systems also enables a general design of applications based on them for execution in a distributed computing environment, irrespective of the intrinsic application details.

A. *The Geant4 Toolkit and its application*

Geant4 is an object oriented toolkit for the simulation of particle interactions with matter. It provides advanced functionality for all the domains typical of the simulation of radiation effects: geometry and material modelling, physics processes, particle tracking, run and event control, interactive capabilities and visualization. Geant4 is widely used by a huge experimental community in many application domains: high energy and nuclear physics, astrophysics and space science, medical physics, radiation protection etc.

Geant4 architecture is based on a rigorous domain decomposition, which minimizes the dependencies among the various packages of the toolkit, each one responsible for a well defined functionality. This modular architecture facilitates the execution of Geant4-based applications in parallel mode: independent units suitable for parallel execution of the simulation are easily identified.

Geant4 architecture defines a protocol for applications to interact with Geant4 kernel. Base classes are defined for user initializations and actions; user applications must provide concrete implementations of these classes, which are handled transparently by Geant4 kernel. This architectural design allows the user a wide flexibility to model his/her experimental set-up and to configure the simulation environment; by defining a general protocol for interaction with the kernel, it also facilitates a general design to configure Geant4-based applications for parallel execution, independent from the specific application itself.

B. *Analysis systems*

The AIDA (Abstract Interface for Data Analysis) system encompasses a set of interfaces, which define protocols for data analysis. The application code handles analysis objects through abstract interfaces; concrete analysis systems implementing the AIDA interfaces are selected by the user by loading the corresponding libraries. Like in the case of Geant4, the design of the application code in terms of AIDA abstract interfaces facilitates a generic configuration for parallel execution, by handling the objects' interactions through well defined protocols.

C. *The DIANE Distributed Analysis Environment*

The DIANE distributed Analysis Environment is a software system acting as an intermediate layer between an application and the middleware for distributed computing. It hides the complexity of the underlying distributed technology from the user application, and it provides specific functionality to facilitate the application execution in a distributed environment.

DIANE is based on a Master-Worker architectural pattern. It provides several application-oriented facilities, such as error recovery policies, automatic bookkeeping of jobs, and monitoring and control tools. It allows the transparent use of various middleware implementations, such as load balancing service and security service. Because of its architectural design, DIANE is completely independent from applications: that is, it is not aware of the specific functionality of the application which uses it. It can also handle the access to different types of distributed computing environments transparently, such as a local cluster or the grid.

Its architecture, its specific functionality, and its blindness to application details make DIANE a suitable system to develop a general design for Geant4 applications in a distributed computing environment.

IV. Geant4-Based Applications in the DIANE Environment

The architecture of the system is illustrated in Fig. 1 through a UML (Unified Modelling Language) [6] diagram.

A generic Geant4 application design has been conceived, which is suitable to the transparent execution of the simulation either in sequential mode or in distributed mode via DIANE.

An abstract interface *IG4Simulation* class is responsible of steering the simulation execution in the DIANE framework. Its member functions *initialize*, *executeMacro*, and *finish* encapsulate the essential actions needed to execute a simulation. The *setSeed* member function allows the simulation user to define the initial seed for the generation of random numbers throughout the simulation.

A concrete class derived from *IG4Simulation* must be provided by the simulation developer. Its *initialize* implementation instantiates the classes required to interact with the Geant4 kernel: the Geant4 *G4RunManager* and all the user initialisation and action classes derived from the corresponding Geant4 base classes. The implementation of *finish* destroys the *G4RunManager* instance. User configuration actions are encapsulated in a macro file, which is supplied to the job through the *executeMacro* member function. A macro file may contain directions for the configuration of the Geant4 simulation and of its execution via DIANE. The interaction with DIANE requires the specification of the total number of simulation events to be generated, the number of tasks into which the job is to be divided, and the initial seed for random number generation. The generation of random numbers in the tasks is handled transparently to the user; the specification of the initial seed is sufficient to ensure the reproducibility of a simulation.

This application design is transparent to the mode of the simulation execution: the same code implementation can be used for sequential or parallel execution.

Analysis objects, such as histograms or tuples, may be generated in the course of a simulation execution; they must be available to the user in a persistent storage resulting from the execution of the whole simulation. In a parallel execution mode each task would produce the analysis objects corresponding to its own portion of the simulation; the final analysis objects are assembled by the Master in the final stage of the execution, resulting from the sum of the partial ones. The user can retrieve both the final and the partial analysis objects.

V. Benchmarking

The performance of the distributed computing system described in the previous sections was benchmarked against a real-life Geant4 application, the *brachytherapy* Geant4 Advanced Example.

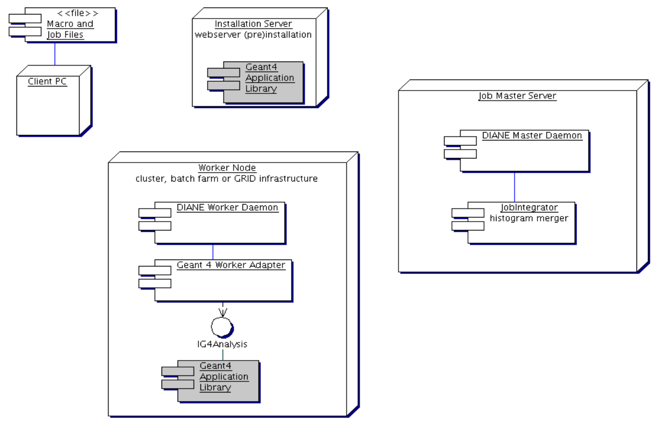

Fig. 1 Architecture of the system for Geant4 distributed simulation.

The overhead introduced at the initialization stage by running a Geant4 application through DIANE was estimated by comparing the execution of one event on a single machine in sequential mode as a standalone application and via DIANE.

The time elapsed in the execution includes the time spent in the initialization phase and the time to process the single event; since this is negligible (< 1 ms) in comparison to the time needed for initialization, the measurement can be ascribed to the initialization phase entirely with a safe approximation. In the case of execution through DIANE, the effects of managing the simulation execution and of integrating the analysis results at the end of the simulation were evaluated. These measurements were performed by running the application in identical conditions several times. The results are summarized in Table I.

The efficiency of the simulation execution via the DIANE system was evaluated in a cluster of 30 Pentium IV, 3 GHz bi-processors. A study of the load balancing was performed, by computing the efficiency of the system in various configurations of workers and task distributions. The results are summarized in

Fig. 2, which shows how a larger number of tasks optimize the execution efficiency.

Preliminary studies for the execution in the LCG grid [7] demonstrate the capability of running the simulation transparently in this environment. However, grid computing is not a mature technology yet, and one observes large fluctuations in the actual availability of nominally accessible grid resources and of the grid load, as it is shown in Fig. 3 and Fig. 4. Because of these large fluctuations, a calculation of the efficiency of the simulation efficiency would not be meaningful in such highly variable conditions.

VI. Conclusion

A general system for the execution of Geant4 simulation applications in a distributed computing environment has been developed. Thanks to the system architecture, a Geant4 simulation can be run transparently in sequential or parallel mode, either on a single computer or in a local cluster or in a geographically distributed grid.

TABLE I
OVERHEAD INTRODUCED BY DIANE

Standalone application	4.6 ± 0.2 s
Application via DIANE, simulation only	8.8 ± 0.8 s
Application via DIANE, with analysis integration	9.5 ± 0.5 s

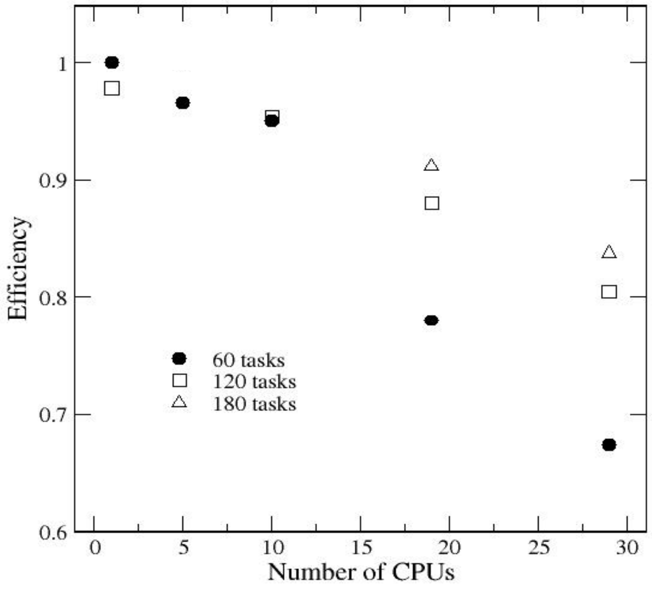

Fig. 2 Efficiency of the simulation execution in a PC cluster for various configurations of workers and tasks.

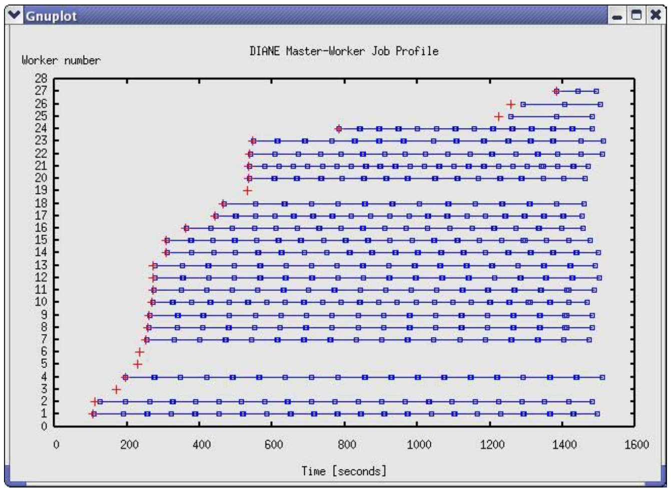

Fig. 3 Graph of a case of simulation execution in the LCG.

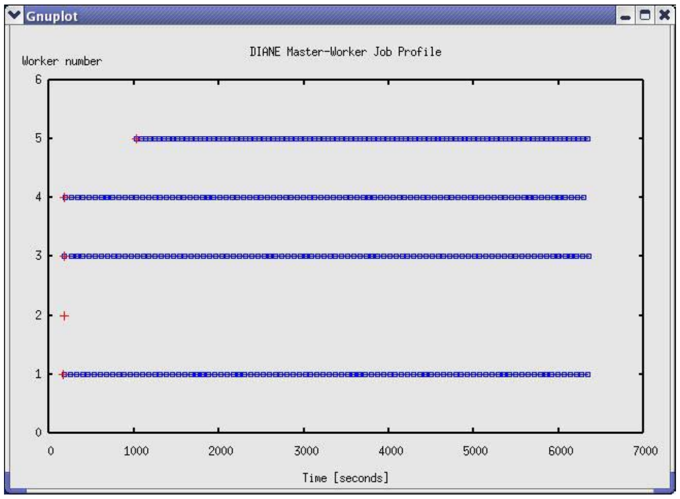

Fig. 4 Graph of a case of simulation execution in the LCG; the same initial conditions as in the previous figure were specified.

REFERENCES

[1] I. Foster and C. Kesselman, The Grid, Morgan Kaufmann, 2003.
[2] S. Agostinelli, J. Allison, K. Amako, J. Apostolakis, H. Araujo, P. Arce et al., Geant4: a Simulation Toolkit, *Nucl. Instrum. Methods A,* vol. 506 pp. 250-303, Jul. 2003.
[3] Allison, J.; Amako, K.; Apostolakis, J.; Araujo, H.; Dubois, P.A.; Asai, M. et al., Geant4 Developments and Applications, *IEEE Trans. Nucl. Sci.,* vol. 53, no. 1, pp. 270- 278, Feb. 2006.
[4] G. Barrand, P. Binko, M. Donszelmann, A. Johnson, A. Pfeiffer, "Abstract interfaces for data analysis: component architecture for data analysis tools", Proc. of CHEP 2001 Int. Conf. on Computing in High Energy and Nuclear Physics, pp. 215-218, Science Press, Beijing, 2001.
[5] J. T. Moscicki, Distributed analysis environment for HEP and interdisciplinary applications, Nuclear Instruments and Methods in Physics Research A, vol. 502, no. 2-3, pp 426-429, Apr. 2003.
[6] G. Booch, J. Rumbaugh, and I. Jacobson, "The Unified Modeling Language User Guide", Addison-Wesley, 1999.
[7] LHC Computing Grid, [Online]. Available: http://lcg.web.cern.ch/LCG/