PROCEEDINGS OF SCIENCE

SGI (Silicon Graphics International Corp) UV2000: the porting of GFORC system

Bruno Luigi Martino* [1] **and Memmo Federici** [2]

[1] *CNR-IASI: Istituto di Analisi dei Sistemi ed Informatica*
Via dei Taurini 19, 00185 Roma, Italy
bruno.martino@iasi.cnr.it

[2] *IAPS-INAF: Istituto di Astrofisica e Planetologia Spaziali*
Via Fosso del Cavaliere 100, 00133 Roma, Italy
memmo.federici@iaps.inaf.it

This paper describes the migration of the system devoted to Monte Carlo simulations based on the toolbox GEANT4 in use at the IAPS in Rome from the current platform based on a PC cluster to a new configuration focused on a architecture BLADE SGI UX2000. Are taken into account aspects related to the parallelization, the user interface and to the storage system.

The Golden Age of Cataclysmic Variables and Related Objects - III, Golden2015
7-12 September 2015
Palermo, Italy

*Speaker.

PoS(Golden2015)076

1. Introduction

In designing a system of revelation, from nuclear physics to medical physics, as well as to space applications, it is necessary to simulate the interaction of particles with the materials. According to Emilio Segre was Enrico Fermi who invented the Monte Carlo method (without using that name), while he was studying in Rome in the early 30s the motion of neutrons.

The Monte Carlo code GEANT4 provides a set of software tools (libraries) for the description of the geometry of the experimental apparatus and materials used, the transport of particles and their interaction with matter. It is an open source platform, developed in the programming language object-oriented C ++ and is the result of an international collaboration.

As the scale and complexity of High Energy Physics experiments increase, simulation studies require more and more care and become essential to design and optimise the detectors, develop and test the reconstruction and analysis programs, and interpret the experimental data. GEANT4 [1] is a system of detector description and simulation tools that help physicists in such studies.

2. GEANT4

The GEANT4 code is designed for high-energy physics, in particular for the LHC project. During code development it has followed a generalization philosophy and abstraction of the specific parts so as to permit the improvement of the code over time [2] . The Geant4 collaboration is divided into several groups around the world for the development in parallel of the different parts of the code. Each group is responsible for the implementation and validation of models and algorithms Monte Carlo introduced in the different versions.

The main difference between the previous version (GEANT3) and the current (GEANT4) consists of techniques were used in the fact that previously the procedural programming (Fortran) while the current has been developed using widely Object Oriented programming techniques (C++). In procedural languages the basic element of the program is the routine or procedure; a program is broken into routine to efficiently process the informations.

When the structure of the data to be processed becomes very complex, it is more convenient to tie in a single structure and procedures; now the basic element of the program is no longer the routine, but the class. A class is essentially the combination of a data structure and routine to process the same data, so that the different parts of the code become "black boxes" and modify any part of the code is much simpler. All the basic physical processes of GEANT4 implemented in C++ language are made in the form of classes and are divided into 7 categories:

- electromagnetic;
- hadronic;
- decay;
- photolepton-hadron;
- optical;
- parameterization;

- transportation.

Each class of a particular physical process (elastic scattering, annihilation, etc.) contains the methods and procedures to invoke the different models based on the energy of the reaction and the type of particles involved.

2.1 Sequential programming vs parallel programming

Traditionally, programs are written for a sequential computation model (Von Neumann) so to be executed on a computer with a single CPU. A problem is broken into sequences (discrete) of instructions that are executed in sequence (one after the other); at a given instant of time a single instruction is running.

There are several reasons that limit the construction of faster and faster sequential computers as the data transfer speed (9cm / nS in copper), the limits of miniaturization and economic issues. By the term parallel computing means the use of more computing units (CPUs or cores). One problem is decomposed into parts that can be solved concurrently (in parallel): each part is broken into sequences of instructions to be executed on a single CPU, and the instructions of each of the parts are performed simultaneously on different CPUs [3].

2.2 Clusters and grids

The calculation model parallel Multiple Instruction, Multiple Data (MIMD) is the most popular model. Most parallel computers fall into this category. Multiple Instruction: every processor executes a different instruction stream; Multiple Data: every processor works on a different data stream. The real difficulty is to develop parallel applications because parallel architectures reach high performance only if programmed appropriately.

In the world of high-performance computing research focuses largely on cluster architecture, cluster of workstations (COW) and PC cluster, namely static architectures characterized by the aggregation of computing nodes completely dedicated to infrastructure and interconnected via an interface standard-based network. The next step is marked by a flowering of projects aimed at defining systems metacomputing. The metacomputing systems have many features in common with the cluster system in fact, as the latter, they are groups of standard computational nodes interconnected using standard network interfaces. However, they possess in themselves the seminal characteristics of the concept of grid computing, which define the dynamic systems formed by the aggregation of computational nodes not involved in metacomputer, also interconnected through the Internet and belonging to security domains and management do not homogenous

3. GforC

By now the environment used in the IAPS institute to perform Monte Carlo simulations is based on a Geant4 software installation on a cluster computing platform (GForC)

The originality of GForC is the capability to execute GEANT4 toolkit runs (written for serial platforms) in a pseudo parallel mode to achieve performance similar to native parallel system. GforC highlight are:

Figure 1: GforC cluster

- Debian Operating system: high reliability and stability
- File system OCFS2: high availability, high performance, concurrent access to resources, no limits on the total number of files and directories
- SLURM multicore resources manager: allocation of resources, arbitration requests and a graphic framework for management purposes

GFORC system is able to split the simulation program in a set of instances running simultaneously on the allocated nodes cores; thanks to this mechanism, the speed obtained is comparable with that obtained on parallel platforms. This modus operandi reproduces, in principle, the MapReduce computing paradigm (a processing technique and a program model for distributed computing based on java).

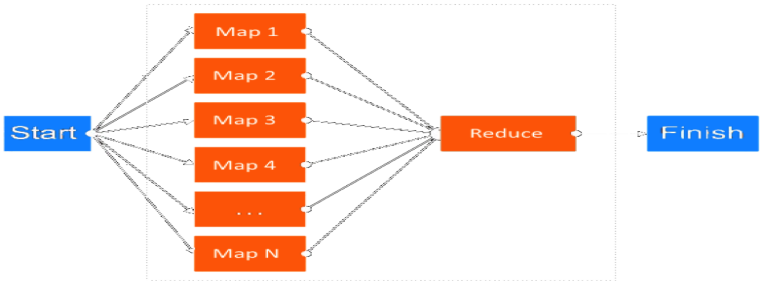

Figure 2: Map Reduce paradigm

The MapReduce algorithm contains two important tasks, namely Map and Reduce. Map takes a set of data and converts it into another set of data, where individual elements are broken down into subsets. Secondly, reduce task, which takes the output from a map as an input and combines those data sets. As the sequence of the name MapReduce implies, the reduce task is always performed after the map job.

4. GforB

The system GforC, although very powerful and still working, showed some weaknesses related

to long term reliability GForC consists of outdated hardware and for this reason the replacement of individual faulty nodes has become increasingly complex (in some cases impossible) In the following, a list of system's issues:

- The HW was built using PC and its RAM cannot be expanded beyond a fixed ceiling - Communications between nodes composing the cluster are made through the local Ethernet network whose speed is limited to 1 GB max
- Currently GForC is not provided by checkpoint mechanisms, this makes it vulnerable to power supply breaks and machine failures
- The storage subsystem is connected to the computer unit via LAN causing low data exchange speed problems

To overcome the limitations of the current architecture, it was necessary to recourse to totally different hardware and software platform: the BLADE technology.

5. Blades

A blade server is a server chassis housing multiple thin, modular electronic circuit boards, known as server blades. Each blade is a server in its own right, often dedicated to a single application [4]. Blades are literally servers on a card, containing processors, memory, integrated network controllers, and optional Fiber Channel host bus adaptors (HBA) and other input/output (IO) ports. Summarizing, a Blade server:

- reduces energy consumption
- improves power management
- simplifies cabling requirements and wiring
- allows more processing power in less space
- consolidates resources (such as storage and networking)

The above considerations make this system a platform able to effectively support GEANT4 simulations and become the starting point for future computing applications made at the distributed computing laboratory (LCD), such as the study of the background of the detectors on board of the mission ATHENA.

5.1 SGI UV2000

The SGI UV 2000 series is a family of multiprocessor distributed, shared memory (DSM) computer systems that can scale Intel's processor cores as a cache-coherent single system image (SSI). The UV family provide the ability to support "big memory" applications, whose datasets can stretch into the multiple-terabyte realm. Since the architecture supports large amounts of global

shared memory, applications don't have to slice their data into chunks to be distributed and processed across multiple server nodes, as would be the case for compute clusters. Thanks to the SGI's NUMAlink interconnect, UV is able to glue together hundreds of CPUs and make them behave as a single manycore system with gigabytes of memory. Essentially, you can treat the machine as an ultra-scale Linux machine. Currently our system consists of one single module 10U with 1TB RAM and 160 Cores. The maximum expansion is made of 4 x 4 (16) modules 10 U

6. Software components

The more computing power made available by UV2000 led to revise some of the design lines followed during the development of GforC. In any case, and beyond these considerations, the main steps for porting simulation software are as follows:

- Porting the pseudo parallelization launcher for GEANT4
- Porting of the graphical user interface
- Optimization of computing resources (Cgroups)
- Implementation of a faster RAM disk

We will now analyze in detail every single point.

6.1 Launcher porting

With the GForC we developed a system through specific programs able to parallelize serial analysis instances to GEANT4; heavy modifications of such procedures has also allowed their use on GForB. The engine used in the old environment is SLURM (Simple Linux Utility for Resource Management). SLURM has three key functions: first, it allocates access to resources (compute nodes) to users for some duration of time so they can perform work. Second, it provides a framework for starting, executing, and monitoring work (normally a parallel job) on the set of allocated nodes. Finally, it arbitrates contention for resources by managing a queue of pending work. Unlike Cforc, GForB is equipped by the OPEN LAVA queue manager able to allocate to user's jobs the needed system resources in a very granular mode (RAM, Cores number, CPU Time etc.); checkpointing is natively supported. Platform Computing has had one of the most successful and arguably robust commercially supported resource managers, named LSF, for many years. A while ago, Platform created a cluster management tool that is now called IBM Platform Cluster Manager (PCM) [Note: Platform was acquired by IBM]. PCM is an open source package, and Platform wanted to integrate an open source resource manager with the tool. In 2007 Platform took an older version of LSF, version 4.2, and created an open-source resource manager, which they named Platform Lava or just "Lava." Not too long ago, some developers based a new resource manager on Lava, calling it openlava[5].

6.2 UI porting

The graphical interface is a crucial resource for users who need to use a computing system. As in the previous system (GForC) users are driven to the insertion of the analysis parameters without having to type long, complex strings using the command line The GUI uses a free and cross-platform program which allows the use of GTK + dialog boxes inside bash scripts (Zenity); this program takes advantage of simple programming and a reduced consumption of CPU resources

6.3 Resources optimization

The cluster devoted to Monte Carlo simulations GforC was composed of independent nodes that were allocated to a user for the entire time of computation. The minimum allocation unit was the single node, and was manufactured using a specific command of Slurm. A machine like UX2000 lets you do view the user processes and applications as a machine with a single CPU and a number of cores equal to the amount of the available cores. To allocate to users the Necessary resources we used to not so well known component of Linux kernel: Cgroups. Cgroups (stands for Control groups) is a Feature That Allows the limiting and insulation usage of resources (CPU, memory, disk I / O, network, etc.) of a collection of processes. Resources are assigned to each user for carrying out its analysis, in particular:

- CPU cores number
- quantity of RAM memory
- maximum lifetime of a job

Because of the feature of the multi-user and multitasking system, it is necessary to ensure that none of the authorized users may saturate the resources of the machine Some programs developed by us ensure that through:

- The control of the number of processes launched by the users in every session
- The maintaining of a database containing the state of resources
- The updating of the quantities of resources allocated to users, bringing them back within the limits assigned using a timed daemon (shepherd daemon)

Cgroups is now the heart of modern containerization systems (e.g.LXC) [6].

7. Ramdisk

GForB is equipped with a large amount of fast RAM (1TB). We decided to use some of it as a virtual disk at high speed and the analysis time has been reduced considerably A temporary file system (TMPFS) using local memory is faster than magnetic disks improving system performance of reading and writing files The RAM disk has obtained performance even higher than a SSD (Solid State Disk, about a factor of 10) We planned to make some tests using a new unit NVM Express, capable to reach performances close to RAM disk but able to hold data in non volatile mode.

8. Storage system

GforB storage system is composed by two units Qsan 48 TB each These units are currently shared by the two computing systems AVES and GForB: AVES is connected by Ethernet at 1 GB speed, GForB is connected by Fibre Channel at 8 GB speed

Figure 3: Qsan F600Q-D316 main controller

The data update and his consistency are guaranteed by a dedicated machine (spone) via a task started by the cron system job provides to connect with ISDC when needed. This type of activity is usually performed using the rsync utility but unfortunately the ISDC firewall blocks connections using this protocol and allows only FTP connections. To work around this problem, the remote repository is attached to the local machine using the curlftpfs utility; in this way the entire remote data repository is seen as replicated on the local machine within the file system[7]. It therefore becomes possible to use the rsync protocol to synchronize data ISDC (now regarded as local) with an appropriate folder containing the historical record of all the mission data.

9. Conclusions

The new system GforB Allows to a considerable increase in the speed of processing Geant4 simulations. Since this is not of a simple calculation engine (as the old GforC) but a general purpose system it was possible to install the necessary software for the subsequent phases of analysis of the results obtained, in particular:

- IDL
- FTOOLS
- MATLAB
- etc.

Moreover, thanks to the increased computing capabilities and taking into account the fact that it is a machine in which the resource management software has no delay time introduced by the communications established through the network, is able to support in an extremely effective way a heavy Multi-user environment.

Acknowledgments

Pietro Ubertini (IAPS Director), Franco Giovannelli and LOC

References

[1] Copyright CERN, *GEANT Detector Description and Simulation Tool*, *Geneva* [1993]

[2] E. Billi et al., *Simulazioni di background del rivelatore BaF2 n TOF del CERN*, *Journal of Nuclear Physics* [2007]

[3] B. Barney, *Introduction to Parallel Computing, LLNL*, *URL http://computing.llnl.gov/tutorials/parallel_comp/.* [Accessed: 2015-05-20]

[4] Q. Haiping, et al., *Algorithms and Architectures for Parallel Processing*, *Springer* [2010]

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

[6] *LXC, linux conainers*, *URL http://linuxcontainers.org/.* [Accessed: 2015-05-20]

[7] B. L. Martino, M. Federici, *An high availability data storage subsystem for the INTEGRAL data analysis*, *Mem. S.A.It. Vol. 75, 282* [2008]