PRODUCTION OF SIMULATED EVENTS FOR THE BABAR EXPERIMENT BY USING LCG

D. Andreotti, E. Antonioli, C. Bozzi#, E. Luppi, P. Veronesi, INFN Sezione di Ferrara and Università degli Studi di Ferrara, I-44100 Ferrara, Italy
M. Melani, Stanford Linear Accelerator Center, Stanford, CA 94309, USA

for the BaBar Computing Group

Abstract

The BaBar experiment has been taking data since 1999. In 2001 the computing group started to evaluate the possibility to evolve toward a distributed computing model in a Grid environment. In 2003, a new computing model, described in other talks, was implemented, and ROOT I/O is now being used as the Event Store. We implemented a system, based onthe LHC Computing Grid (LCG) tools, to submit full-scale MonteCarlo simulation jobs in this new BaBar computing model framework. More specifically, the resources of the LCG implementation in Italy, INFN-Grid, are used as computing elements (CE) and Worker Nodes (WN). A Resource Broker (RB) specific for the Babar computing needs was installed. Other BaBar requirements, such as the installation and usage of an object-oriented (Objectivity) Database to read detector conditions and calibration constants, were accomodated by using non-gridified hardware in a subset of INFN-Grid sites. The BaBar simulation software was packed and installation on Grid elements was centrally managed with LCG tools. Sites were geographically mapped to Objectivity databases, and conditions were read by the WN either locally or remotely. An LCG User Interface (UI) has been used to submit simulation tests by using standard JDL commands. The ROOT I/O output files were retrieved from the WN and stored in the closest Storage Element (SE). Standard BaBar simulation production tools were then installed on the UI and configured such that the resulting simulated events can be merged and shipped to SLAC, like in the standard BaBar simulation production setup. Final validation of the system is being completed. This gridified approach results in the production of simulated events on geographically distributed resources with a large throughput and minimal, centralized system maintenance.

INTRODUCTION

The production of simulated events for the BaBar experiment [1] at the SLAC PEP-II e+-e- Asymmetric B-Factory is a distributed task involving several computing farms spread around the world. The amount of events to be simulated is at least three times the hadronic cross-section at the Y(4S) peak, which is about 4.5nb. The BaBar experiment has accumulated to date about 250 fb-1 integrated luminosity, which translates to a production in excess of 1 billion simulated events in the timescale of one year. This has been accomplished in the past by devising and deploying a system [2] which allows the production of Monte-Carlo events to be distributed over 25 sites, for a total of about 1000 CPUs. Events are transferred to SLAC and to other major computing centers, where they are made accessible to the final users. Each remote farm is managed by a local production manager. On each farm the whole BaBar software release installation is required. An object-oriented database (Objectivity) [3] is used to read detector conditions and calibration constants, while ROOT I/O [4] is used to mix real background triggers with the simulated events, and as Event Store [5].

It is natural for such a system to evolve into a simpler and more efficient one by using an approach based on the tools and middleware provided by Grid systems such as the LHC Computing Grid Project, LCG [6]. Ideally, a Grid-based Monte-Carlo production system would be based on a single production manager who submits jobs to a grid of remote sites, retrieves the outputs and updates the bookkeeping. In practice, since BaBar is a running experiments with a huge and continuous demand for simulated events, it was decided as a first step to implement a system which is capable of using grid resources, but which is seen by the BaBar simulation production infrastructure as yet an additional production farm, like the others. The schema used in this work is based on INFN-GRID [7], an Italian project implemented on the LCG middleware. Participating institutes in the INFN-GRID project whose resources have been used in this work is shown in Fig. 1.

SYSTEM IMPLEMENTATION

The software needed to produce Monte-Carlo events has been packaged, including the executable and additional libraries and data files, has been packaged in a simple tar archive of about 37 MBtyes (130 Mbytes uncompressed). This avoids the import of a full BaBar

Figure 1: Map of Italy showing the sites participating in this work. The locations of the core services (conditions DB, background triggers and Resource Broker) are also shown.

software release, which sums to about 1.5 GBytes. The package is then distributed on each Grid Worker Node (WN), by using procedures based on the LCG middleware to install/uninstall new software and publish new tags. These procedures can be used by a site manager, mapped as a special user, by simply submitting specific jobs on the grid, after choosing which sites must be involved.

As already mentioned, the production of simulated events implies to read the detector conditions and calibration constants from an object-oriented database, which consists of about 30GBytes data, frequently updated [8]. The typical amount of conditions data read by a single simulation job is about 1/50 if the total. Since it is not practical to install and maintain such a database in every grid site, it was decided to use only three sites (Ferrara, Padova, Naples), where Babar resources are present, both in terms of manpower and dedicated hardware. Detector conditions are therefore read generally through the Wide-Area Network (WAN), by using an Advanced Multithreaded Server (AMS) [9]. Non-gridified resources were used to host the conditions database, given the present complexity in maintaining such a system, and the system load to which it is subject.

The BaBar simulation jobs involve the mixing of genuine random trigger data with Monte-Carlo generated events in order to have a realistic simulation of the detector backgrounds. These background triggers are stored as ROOT I/O files (about 2 GBytes each) in the same three sites which host the conditions database, and accessed by using the xrootd server [10]. Background collections are produced on a monthly basis, and consist of about 100k random triggers, 2000 of which are read by a typical production job.

Standard BaBar software tools for simulation production are installed on the UI and properly configured. These tools are needed to merge several collections of homogeneous events in a single collection, transfer events to SLAC, and to update a bookkeeping database at SLAC [11]. This procedure is the same as in any other standard BaBar simulation production farm.

JOB CONFIGURATION AND SUBMISSION

In order to submit production runs, a JDL [12] script is prepared. This script sets up all data needed and specifies the parameters needed to simulate events. The latter include: type of events, run number, background triggers to use, number of events, detector conditions. Jobs are then submitted through an LCG User Interface (UI) to a Resource Broker (RB) specifically installed for BaBar, both located in Ferrara. The RB is able to manage Italian and European resources directly involved in BaBar. The RB performs a matchmaking between resources available on each site of the Grid, published by Computer Elements (CE), and jobs requirements (memory, specific software release, type of the batch queue). Jobs are sent to CEs which satisfies all requirements, and distributed on WNs, where execution starts.

The generated Monte-Carlo events are saved as ROOT I/O files and transferred from WNs to a Storage Element (SE) located in Ferrara. The final steps, described in the previous section (i.e. merging, transferring to SLAC and updating the bookkeeping database) are then performed.

A conceptual schema of the flow described above is shown in Fig. 2.

SYSTEM PERFORMANCE

Several pre-production tests, for a total of several hundreds runs were submitted and the output was retrieved as explained in the previous section. It was soon discovered that the number of clients which can access the conditions database simultaneously was limited to 20 at most. If that limit was exceeded, the latency increased significantly and massive job crashes were observed. A customized configuration of the AMS server was therefore required to increase this number, which led to a maximum of 90 concurrent clients without any efficiency loss.

The inefficiency in reading the detector conditions and the background triggers by using the WAN depends widely on the bandwidth available to each computing farm, and on the traffic over the network. During the tests, the inefficiency due to the WAN varied from a few percent to 50% at most.

After tuning the AMS server properly, about 96% of jobs were completed and the outputs were successfully

retrieved on the SE. The main reasons due to failures were due mainly to transient instabilities of the network and of the grid components.

The final validation of the system is now being performed, and it is foreseen that such a grid-based farm will significantly contribute to the production of simulated events in BaBar.

REFERENCES

[1] B. Aubert et al. (BaBar Collaboration), Nucl.Instrum.Methods A479, pp. 1-116,2002

[2] D. Smith, F. Blanc and C. Bozzi, "A Millennium, Work in Under a Year – BaBar Simulation Production", presented at this Conference.

[3] http://www.objectivity.com

[4] R. Brun and F. Rademaker, "ROOT – An Object Oriented Data Analysis Framework", Proceedings AIHENP'96 Workshop, Lausanne, Sep. 1996, Nucl. Instrum. Methods A389, pp. 81-86, 1997. See also http://root.cern.ch.

[5] M. Steinke, "How to Build and Event Store – The New Kanga Event Store for BaBar", presented at this Conference.

[6] L. Robertson, "The LCG Project – Preparing for Startup", presented at this Conference.

[7] http://grid.infn.it

[8] I. Gaponenko, "CDB – Distributed Conditions Database of BaBar Experiment", presented at this Conference.

[9] AMS reference.

[10] Xrootd reference

[11] D. Smith, "BaBar Book Keeping Project – a Distributed Meta-Data Catalog of the BaBar Event Store", presented at this Conference.

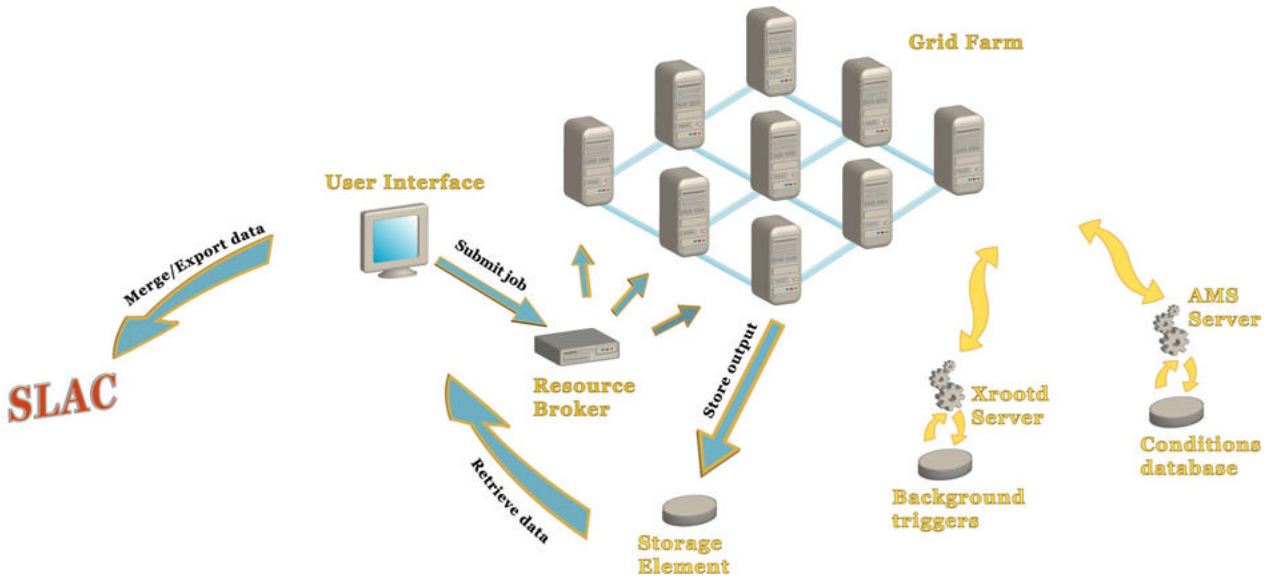

Figure 2: Schematic flow diagram of the production of simulated events performed in this work.