The LCG based mass production framework of the H1 Experiment

Ch. Wissing*, Universität Dortmund, Germany
M. Karbach, Universität Dortmund, Germany
M. Vorobiev, ITEP Moscow, Russia

Abstract

The H1 Experiment at HERA records electron-proton collisions provided by beam crossings of a frequency of 10 MHz. The increased event rates after the luminosity upgrade of the HERA accelerator at DESY led to a more demanding usage of computing and storage resources. The analysis of those events also requires an increased amount of Monte Carlo events. In order to exploit the new necessary resources, which are becoming available via the Grid, the H1 collaboration has therefore started to install a mass production system based on LCG.

The H1 mass production system utilizes Perl and Python scripts on top of the LCG tools to steer and monitor the productions. Jobs and their status are recorded in a MySQL database. During autonomous production a daemon lunches appropriate scripts while a web interface can be used for manual intervention. Additional effort has been put into the sandbox environment in which the executable runs on the worker node to ensure a stable unattended processing.

The system has proved to be able to track several hundred jobs allowing for production rates of up to 20 million events per week. Beginning of 2006 H1 has access to nearly 3000 CPUs contributed by 15 sites from seven countries.

INTRODUCTION

The H1 Experiment is running at the electron proton collider HERAII at DESY with beam energies of 27.5 GeV and 920 GeV, respectively. The collider was upgraded in the years 2001/2002 in order to provide an increased luminosity. The integrated luminosity delivered in the year 2005 has achieved the same amount as the complete running period from HERAI (1994 to 2000). The luminosity to be provided in the remaining operation time of HERA is expected to exceed the delivered one.

The H1 Experiment [1],[2] is a general purpose detector with about half a million readout channels to observe the electron proton scatterings. The basic parameters for the multi-level trigger system and the data acquisition system are driven by the beam crossing frequency of 10 MHz, the typical rate of beam induced background of about 10 kHz and the actual rate of physics events, which is about 1 kHz. The data logging facility of the online reconstruction farm [3] writes data with a frequency of up to 25 Hz and typical event sizes of 100 kB. While these are stored on tapes a compressed subset of the event containing important quantities for analysis is kept permanently on disk to allow for fast access.

In addition to the raising event rates due to the improved performance of the HERAII collider the amount of Monte Carlo (MC) events is increasing continuously. As the understanding of the detector has become better over the years and more advanced algorithms are being used in the event reconstruction the absolute processing time for simulated events has grown although faster CPUs became available.

The MC production has been performed on few selected sites with sizable computing resources, including RAL and other British H1-institutes, Dortmund and the central H1-farm at DESY. These farms are running conventional batch system like PBS and are operated individually either locally or remote by a production manager.

In order to allocate more computing resources for H1 using the Grid for mass productions is the natural choice. As all institutes involved in H1 are also participating in an LHC experiment they contribute computing resources via LCG.

H1 GRID PRODUCTION SYSTEM

The H1 Grid based production system is organized in several independent modules. The modules communicate with each other via a MySQL database, which has a special state field implemented. This approach allows an independent development of any module. There is no need to use a particular programming language as long as the communication to the database can be established. Presently the modules are written in Perl and Python.

The database is organized as follows. In addition to a few global settings of the system entries are add on bases of production requests, which are usually MC productions. Since a production gets distributed over sometimes up to thousands of jobs, that need to be tracked, all enter the database with a link to the request they belong to. With each job several quantities are stored, e.g. files belonging to it, job description files (JDL file) and the important field for the state mentioned before.

Since a job might need a resubmission before it succeeds another level of jobs, so called LCG-jobs, are recorded in the database. These entries are created during the actual submission and have additional information attached like time of submission and the site the job was processed.

* wissing@physik.uni-dortmund.de

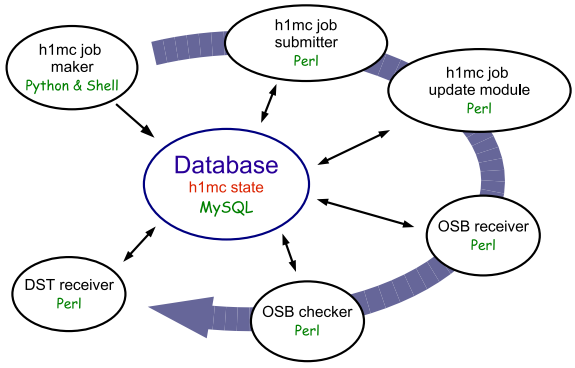

Figure 1: Modules of the H1 Grid Batch System.

Modules of the System

The first modules that runs for a production request is the *JobMaker*. It prepares all necessary files needed later on. For a typical MC request files, which contain the generated events to be put through the detector simulation, get split into junks of 10,000 to 20,000 events. Jdl files and configuration files are created for all jobs. All these jobs are marked with the state *new*.

Next module is the *JobSubitter* which acts only on jobs with state *new* or *failed* in order to submit or resubmit them. If there are resources available at one or more sites an LCG-job is created and added to the database while the state field changes to *running*.

An *JobUpdate* module continuously asks for the status the LCG-jobs which should go from " scheduled" to "running" to "done". The database is always updated with the most recent status.

To collect the output sandboxes (OSB) the *OSBreceiver* module gets started for all jobs that reached the status "done". All files found in the OSB are stored in a particular place in the directory structure and are made known to the database. The state field in the database changes to *received*.

To determine the success of a job the *OSBchecker* modules parses certain logfiles. Depending on the result the state field is changed to *succeeded* or *failed*. In case of to many submission attempts or failures that are likely to happen also in a further resubmission the state can be put to *broken* indication that extra effort has to made.

For the present MC production an additional *DSTreceiver* modules puts the data to same mass storage area where the results of all other MC productions are stored. Since the DESY d-Cache mass storage system has been equipped with ab SRM interface [5] the final step of the production is basicaly a file replication. For further analysis the Grid produced events can be accessed as before.

For an unattended running a daemon calls all modules in steerable time intervals. Via a command line interface the daemon can be influenced e.g. to process additional requests or exclude certain modules from the calling sequence.

Another important part of the production system is the environment in which the job runs on the remote processing node. Since there are usually no possibilities to login failures have to be recovered autonomously as efficient as possible. Effort was put into a reliable scheme that ensures a correct transfer of all data by implementing checksums. These checksums are stored as attributes in the file catalogue.

Experiences from past MC productions have shown that due to bugs in the application the MC program sometimes gets stuck in endless loops or crashes completely. Nevertheless it turned out that events that were processed up to one that caused the trouble could be used. If the production is relaunched one event before the critical one in most cases the production can continue. In order to detect the described malfunctioning the production executable runs parallel to a watch-dog process, which check certain logfiles being updated. In cases where the program crashed or was hanging the production is reconfigured to start again from point of failure. Finally it is made sure that the partly processed events are merged together properly.

Web Interface

The production framework is equipped with a web interface that offers various possibilities for manual interference, inspection of job quantities. It also includes a monitoring.

For manual interference via the web interface the same modules are called as by the daemon. The only difference might be some parameters that ensure to act on one or more particular jobs selected by the user.

The monitoring part generates pages that display all jobs of a request including their states. More detailed information on particular jobs are easily accessed by Hyperlinks. A system based on the widely used RRD tool [6] collects several parameters and completes the monitoring by creating performance plots out of them.

Implementation

The general implementation concept introduces an interface layer between the application and the actual grid middleware such that the application can stay unchanged even if the middleware gets modified.

The approach follows very much the one used for the ZEUS-Grid-toolkit [4]. The actual middleware commands for file transmission and job handling a placed into wrapper PERL classes, which have a defined interface and usually an enhanced functionality over the pure middleware command line tools. These enhancements are forced timeouts, a configurable number of retries and additional crosschecks for the success of a command.

On the application level generic objects are created and used. The interface layer determines the used middleware and creates specific objects accordingly. For new versions of the middleware additional code has to be provided only for the interface layer.

PRODUCTION EXPERIENCES

The described system is installed on a special UserInterface machine running at DESY giving access to H1 mass storage system by conventional methods. All core services like management for virtual organization hone, file catalogue and the resource broker are provided by the DESY Grid infrastructure [7].

The basic functionality of the system is available since mid of 2005. Quite some experiences could be gained already during production leading to further improvements over the months.

Observed Difficulties

Most critical part of the production turned out to be the data transfer to the worker nodes. For the H1 MC production quite some files in addition to the generated events have to be copied. The executable with some suppling libraries is available on each Storage Element (SE) of the sites. The biggest amount of data are the so called noise files for various subdetectors, which sum up to two Gigabytes. Furthermore a files with a size of about 100MB contains calibration constants and the detector geometry. Each job creates about 400MB of output data. Table 1 summarizes the files and their sizes.

Table 1: Typical sizes of files involved in a MC job.

File	Size
Executable+Libraries	20 MB
Input events	50 MB
Noise files	2 GB
Constants files	100 MB
Processed events	400 MB

Figure 2 shows the efficiency of single data transfers as it has been observed over the last months. A noticeable fraction of sometimes 80% and more of the transfer actions fail because the transfer command run into a timeout or a mismatch of checksums was detected. Since the framework has included retry mechanisms most failures are recovered automatically and the job comes to a successful end.

In the more recent past the situation has improved. Most relevant for that are the improved capabilities of the lcg-commands in LCG release 2_6_0, which have also some retry mechanism included. Furthermore it is taken care that the big noise files are available at each site.

In some situations jobs end without success due to errors during data transfer. These situations typically occur at certain points in time. Often it could be correlated with problems in the central services, e.g. the file catalogue.

Figure 3 shows how often a job had to submitted before it succeed. About 25% of all jobs get submitted more than once but finally terminate correctly. The failed jobs are observed to be victims of mis-configured sites or problems

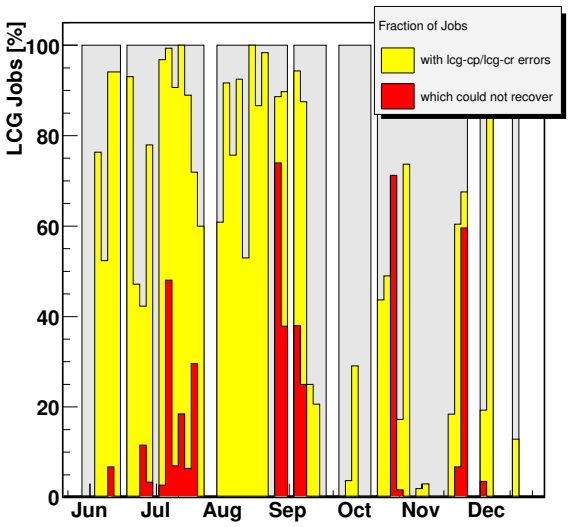

Figure 2: Data transfer efficiencies using lcg-commands.

of individual compute nodes. A fraction of jobs of a few per mill finally needs some manual interference.

Overall Performance

Since mid of 2005 H1 MCs can be produced in the Grid in addition to the existing approaches. Over this time period several LCG sites enabled the support for the H1 experiment. Among them are two big TIER-1 centers in the UK and in France, rather big TIER-2 centers mainly in the UK but also several medium and small sites, that also provide a sizable contribution. The total number of CPUs adds up to about 3.000.

Figure 4 shows the integrated event production since middle of 2005. Since then about 120 million MC events have been processed on the Grid. End of 2005 a production

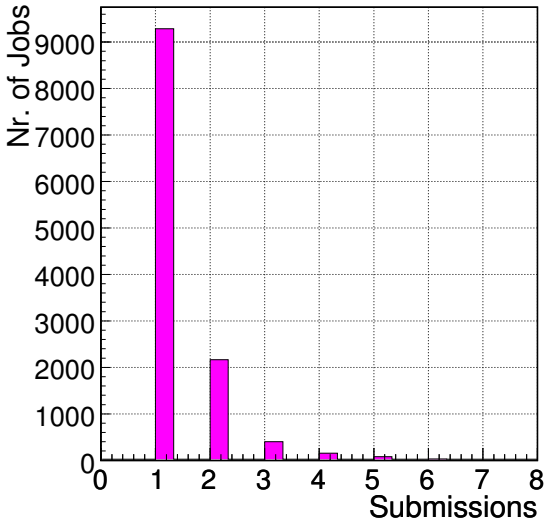

Figure 3: Number of submission until a job succeeds.

rate of about 20 million events per week were achieved during a stress test. This numbers have to be compared with total MC production in 2005 using the classical approach of about 450 million events. It is expected that the fraction of Grid produced MC events will raise with time since institutes are going to install further resources within LCG. This has the further advantage that the central computing farm of H1, that sometimes is heavyly loaded by MC production , has more capabilities for user applications, which in present analysis farm works depend on this environment.

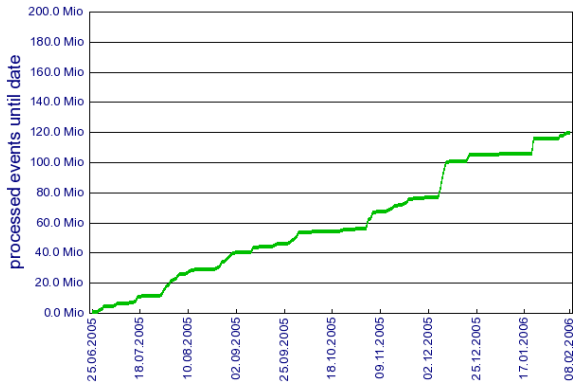

Figure 4: Overall production of H1 MonteCarlo events using the Grid.

BEYOND MC PRODUCTION

As the production framework was developed to serve the needs of any mass production use cases beyond MC simulation are under study. One frequently used application is the creation of analysis trees within the new object oriented analysis environment H1OO [8] based on ROOT. During a conversion process event properties from the DSTs, which are stored in an H1 specific format, get filled in to class objects. This kind of conversion is done for every major release of the H1OO software, typically twice a year. Since all available real data plus requested MC sets get processed the resource requirements are sizable. Thus a distributed production over the grid could be an option.

The use pattern of this application is different from MC simulation, where mainly CPU usage counts, while here a fast and reliable network connection to send the DSTs and the trees is crucial. Nevertheless first test indicate that sites with a fast link to DESY can be used.

In contrast to MC production software for the analysis tree creation depends on a complete installation of the H1 software including H1OO and almost all FORTRAN packages. Since there is no batch enabled installation method available the work presently focuses on providing these features in order to allow a deployment in the VO directories at the LGC sites.

ACKNOWLEDGMENTS

The authors want to thank the experienced MC and software experts of the H1 collaboration for providing the necessary insides and support to build this framework. In particular we want to thank Alan Campbell and Cristi Diaconu for their special interest in this project and their help to prepare this contribution. We thank the DESY-IT stuff that is continuously developing the Grid infrastructure and services at DESY always with the aim to serve our needs in the best possible manner. The project is funded by the German Bundesministerium für Bildung und Forschung.

REFERENCES

[1] I. Abt *et al.*, "The H1 detector at HERA," *Nucl. Instrum. Meth.*, vol. A386, pp. 310–337, 1997.

[2] ——, "The tracking, calorimeter and muon detectors of the H1 experiment at HERA," *Nucl. Instrum. Meth.*, vol. A386, pp. 348–396, 1997.

[3] A.Campbell, S. Levonian, M. Vorobiev, "The High Level Filter of the H1 Experiment at HERA ," Proceedings of the CHEP04 conference, Interlaken, Switzerland.

[4] K. Wrona *et al.* "ZEUS Grid Toolkit", Proceedings of the CHEP 06 conference, Mumbai, India

[5] T. Perelmutov, "Enabling Grid features in dCache", Proceedings of the CHEP 06 conference, Mumbai, India

[6] T. Oetiker, "RDDtool", http://people.ee.ethz.ch/õetiker/webtools/rrdtool/

[7] A. Gellrich *et al.* "DESY Grid infrastructure", Proceedings of the CHEP 06 conference, Mumbai, India

[8] J. Katzy *et al.* "H1OO - an analysis framework for H1,", Proceedings of the CHEP04 conference, Interlaken, Switzerland.