DØ data reprocessing within EDG/LCG

Torsten Harenberg*, Peter Mättig†, Bergische Universität Wuppertal, 42097 Wuppertal, Germany
Kors Bos‡, David Groep§, Willem van Leeuwen¶, Jeff Templon‖,
NIKHEF, P.O.Box 41882, 1009 DB Amsterdam, The Netherlands
Rob Byrom**, Steve Fisher††, Rutherford Appleton Laboratory, U.K.

Abstract

The DØ experiment at the Tevatron is collecting some 100 Terabytes of data each year and has a very high need of computing resources for the various parts of the physics program. DØ meets these demands by establishing a world wide distributed computing infrastructure - increasingly based on GRID technologies.

Distributed resources are used for DØ MC production and data reprocessing of 1 billion events, requiring 250 TB to be transported over WANs. While in 2003 most of this computing at remote sites was distributed manually, some data reprocessing was performed with the EDG. In 2004 GRID tools are increasingly and successfully employed.

We will report on performing data reprocessing using EDG's Application Testbed. We will explain how the DØ computing environment was linked to this GRID platform, and will discuss some lessons learned (for both Grid computing and preparing applications for distributed operation) from the DØ reprocessing on EDG, subjecting a generic Grid infrastructure to real data for the first time.

An outlook on plans for applying LCG within DØ is given.

INTRODUCTION

In September 2003, the DØ experiment at Tevatron has launched a reprocessing effort of part of its Run II data, which were taken since March 2001. The effort was partly done at remote sites (around 20% was calculated outside of Fermilab's own resources) [2]. In total, 519,212,822 events were reprocessed during this efford. Out of these events, 97,619,114 at remote sites [1]. Besides the more manually driven operation, the EDG Application Testbed was used for the first time to take part in an efford like this, where data from a running experiment is processed.

A high-level overview of this reconstruction process we established is as follows:

1. the cluster team receives a reconstruction assignment from the production manager. The assignment comes as a list of "projects", each project containing on the order of seventy data files that require processing. Each data file contains on average 700 MB of information. Processing a file takes about twelve hours on a one-gigahertz Pentium machine.

2. The cluster team contacts the DØ Data Management System (called SAM [5], for Sequential data Access via Metadata) and retrieves a project, meaning all the files are transferred to the cluster site.

3. Jobs are submitted to process the files belonging to a project.

4. The cluster team uses SAM to store the files produced by the reconstruction program.

INTEGRATING DØ SOFTWARE

DØ specific infrastructure

DØ is using its own data management system called SAM to access the data to be re-processed. Furthermore, the re-processed data have to be stored back into that system for collaborative wide use. In order to cope this condition, we set up the following process:

1. making the input data files available on the EDG Data Management System (DMS) [6];

2. adaptation of the DØ reconstruction software and environment to work on EDG resources, and its distribution to worker nodes running the reconstruction jobs;

3. monitoring progress of the jobs and bookkeeping of the results;

4. declare the reprocessed results back into the SAM system.

Processes 1 and 4 above both require some sort of channel between the Data Management Systems of EDG and DØ. We chose to approach this by arranging for a certain storage area, physically present on a back-end server machine, to be visible both from a SAM-enabled machine ("SAM Station") and from EDG machines at the same site (see figure 1).

This "gateway" was established at NIKHEF and works, allthough it requires much manual work.

* harenberg@physik.uni-wuppertal.de
† maettig@physik.uni-wuppertal.de
‡ bosk@nikhef.nl
§ davidg@nikhef.nl
¶ a03@nikhef.nl
‖ templon@nikhef.nl
** r.byrom@rl.ac.uk
†† sm.fisher@rl.ac.uk

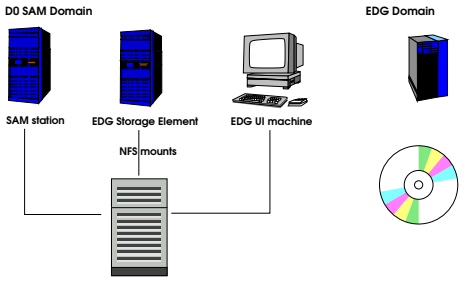

Figure 1: The gateway between SAM and EDG

Adaption of DØ software

The DØ software for running reconstruction consists of four packages:

- The DØ reconstruction executables and libraries, along with the necessary run-time environment such as configuration files;
- The `mc_runjob` package which is a script system for steering the execution and loading the proper configuration files;
- The `pyxml` package, required by `mc_runjob`, used in the DØ monitoring system. This system was not used by the EDG runs, but its removal would have required major surgery in the `mc_runjob` system, and `mc_runjob` complains if the package is not installed. However it was possible to turn off the actual invocation of the monitoring;
- Python version 2.1, required by `mc_runjob`. The EDG testbed has both Python versions 1.5.2 and 2.2.2 but the DØ software required version 2.1.

The DØ software required little adaptation to run on EDG nodes. We benefited from the fact that there are several other groups in DØ that are attempting to run on generic computing facilities, such as GridKa in Germany and WestGrid in Canada. The main problems we encountered were discovering which options to "turn off" in `mc_runjob`, these options corresponding to things that only work in a DØ-specific setup, as well as deleting certain copies of system libraries shipped with the DØ distribution (e.g. libc.so) that caused interaction problems with native libraries.

The DØ distribution was still converging at the start of the evaluation, with one new release per week. Given this relatively frequent update, we did not want to use the usual mechanism from EDG where RPMs were made, pushed to remote sites as part of a new EDG release, and installed on worker nodes during the upgrade. We chose to make compressed tar archives of each of the four packages and register them as Grid files, meaning they were stored on an SE and registered in the Replica Location Service (RLS) [7]. Jobs were programmed to retrieve the four packages (using the EDG replica manager client tools [8]) and install them before actually starting any DØ work.

One side effect of this setup is that we ran the reconstruction in a subdirectory of /tmp, instead of in the pooled account directory. The reason is that when one appends the rather long DØ pathname to the even longer GASS working directory for the job, the resulting directory pathname exceeds 100 characters, which is a hard limit somewhere within the DØ executables. The job fails to find several important files located in this long directory, and then aborts. The temporary space we created had names like /tmp/@1423.d0reco which fall well within the 100-character limit. The disadvantage is that the automatic cleanup mechanisms of GASS do not apply, so that a job encountering an unrecoverable error may leave a couple gigabytes of junk in /tmp on the worker node.

We have reported this bug to the DØ software team, but on the timescale of this validation project it was not possible to get a proper fix to the problem.

Monitoring and Bookkeeping

As in every distributed environment, job monitoring and bookkeeping is very important. While EDG provides basic infrastructure to monitor a job, bookkeeping is not implemented in EDG at all. Furthermore, monitoring is limited to job states, e.g. "running", "waiting", "scheduled", "done" or "aborted".

However the R-GMA[3] system of EDG allows users and their programs to publish information into the general information system of the testbed. This information can be inspected by users at other sites, or archived in a database for later analysis. We used this functionality to generate a monitoring system for the DØ reprocessing.

R-GMA uses an SQL model, so that all information is published to some SQL table. We constructed four tables for DØ monitoring:

- a "submission table" that records the submission of jobs to the resource broker. This records the jobID returned, the submission time, and the user interface machine from which the job was submitted.
- a "job start table" holding information published when a job starts to execute on a worker node. This table includes a unique hash for the particular instance of the job, the jobID, the job start time, the worker node on which the job is running, the input LFN for the job, the SAM process ID (see below), information on the worker-node CPU (cpu type, cpu frequency, real memory and swap space). The unique hash is constructed from concatenating the jobID and the start time. It is needed because one jobID may correspond to multiple table entries, if a job aborts for some reason and is restarted by the EDG workload management system.
- a "job end table" for which information is published immediately before the job stops executing on the worker node. This table includes the same unique job instance hash as the start table, the job execution

ending time, the processor and wall times taken by the job, the number of DØ events processed, the output physical file name (including the SE host name), and the GUID of this file, which is the unique key by which the EDG Replica Location Service can locate the file or one of its replicas.

- a "command table" that is used for job failure diagnostics. Critical commands are run by a wrapper script – if the command returns normally, no special actions are taken. If the command fails, a failure record is published in this table. This record contains the unique job instance hash, a further hash formed from the command name plus the job instance hash (this serves as the primary SQL key in the table, but otherwise serves no function), the time at which the command started execution, the command that failed (including arguments provided), the exit status returned, and the last 255 characters of the command's output. Records in this table allow us to recognize which command was responsible for a failed job. Linking the information in this table with that from other tables provides further information, for example the name of the worker node on which the command failed. This provides a powerful facility for debugging the system – a poorly configured worker node may generate what appears to be an intermittent failure of the DØ job, but analysis of these logs will show that all failed jobs failed on the same machine.

THE DØ JOB SCRIPT

A Python script was written to manage the job setup, publishing of monitoring information into R-GMA tables, running the DØ reconstruction, and recording the results. The script takes the following actions (commands run under the logging command wrapper are marked with an asterisk):

1. records the start time of the job
2. creates a temporary working directory (TWD) in /tmp in which the software and data will be unpacked
3. copies some files from the input sandbox into the TWD
4. instantiates the job start table object, fills the data, and publishes the start record into R-GMA
5. for each of the four software packages needed:
   (a) retrieve a copy* of the package from the DMS.
   (b) unpack the package* (untar it) into the TWD.
6. Some links are made (needed by the DØ machinery) within the TWD
7. run the m4 preprocessor* to fill a template macro (input for the `mc_runjob` program) with specific values for the data file being reprocessed.
8. invoke* `mc_runjob` with the resulting macro, which runs the reconstruction program
9. create a tar archive* containing all the important output files
10. place the resulting archive in the EDG DMS* using edg-rm copyAndRegisterFile
11. interrogate the DMS* (using the edg-lrc command) to find the physical name of the registered copy
12. instantiate the job-end table, fill the values, and publish the job-end record to R-GMA
13. remove the TWD
14. exit.

RESULTS

Comments on the usability

The system provided the neccessary functionality for production work. Jobs submitted from NIKHEF were distributed over the available resources inside EDG. The data management and the information system allowed us to program the job script in a generic fashion.

However, several components or subsystems were lacking functionality or stability. In particular:

- The WP5 Storage Element was not working most of the period in which EDG/DØ software was being developed. As this was critical to us, we installed a so-called "SE Classic" at NIKHEF.

- We had to build our own versions of some of the EDG replica manager client commands. This was necessary since the EDG versions check for existence of a requested file before trying to transfer it. In general this is a good thing, but in this specific case the existence check is made using Globus tools that fail if invoked from machines lacking inbound IP connectivity. The largest fraction of the available worker nodes indeed lacks this connectivity, so these commands were unusable for us.

- The R-GMA system was unstable and has being restarted as often as four times per day, with a total down time per restart of approximately one hour. During downtimes, no new jobs could be submitted. Furthermore, R-GMA did not check version compatibility between clients and servers.

- Job throughput is quite slow, mostly because of direct queries made by the WM machine to local GRIS (MDS information) servers on each participating CE machine.

- The `getBestFile` and `copyFile` commands of EDG's Data Management system both a Globus routine for file-existence checking that tries to start an "active" FTP connection. Such connections require the originating computer (in this case a worker node) to allow inbound IP connections. Most large computer facilities forbid such connections for their worker nodes, as do the largest EDG farms (at RAL and NIKHEF during this evaluation).

- Replication of files is asynchronous. If ten jobs show up at a site and all try to access the same file via `getBestFile`, and the file hasn't previously been registered at the local SE, one can potentially wind up with ten independent replicas of the file on the local SE.

More details can be found in [10].

Conclusions

The EDG production testbed provided the necessary functionality to do production high energy physics work for the first time in a full-scale generic grid infrastructure. The stability of the information system turned out to be the limiting factor. This limited also our contribution to the DØ reprocessing efford. But we proofed that generic grid infrastructure is able to provide a reasonable amount of computing resources even to the allready running high energy physics experiments.

The problems and instabilities we saw have been reported to the developers of the grid middleware and most of them have been fixed. This prooves that a test with the requirements of the running experiment is essential for a generic infrastructure to proove its functionality.

Outlook

After the European DataGrid project has been ended at the end of March 2004, we continued our work in the context of the LHC Computing GRID "LCG" [4], which used the EDG software as its basis. The R-GMA information system, which turned out to be the critical part of EDG, had been replaced with MDS [9]. Meanwhile, R-GMA has been reintegrated into LCG since release LCG-2_2_0, but MDS is still the basis for the information system.

We have continued to work on the integration of SAM into our projects to establish a more sophisticated interaction between these two systems. This could replace the interface described earlier and would save the optimization SAM does on the disk and tape access for the DØ jobs inside the LCG infrastructure. We allready ran successfully Monto-Carlo production.

For the next reprocessing efford, which is planned for the end of November 2004, we're planning to contribute significantly.

REFERENCES

[1] http://www-d0.fnal.gov/computing/reprocessing/p14/

[2] Daniel Wicke, Michael Diesburg: Data reprocessing on worldwide distributed systems; Proceedings of the 2004 Conference on Computing in High Energy Physics (CHEP 2004), Interlaken, Switzerland.

[3] http://www.r-gma.org

[4] http://lcg.web.cern.ch/LCG

[5] http://d0db.fnal.gov/sam/

[6] http://edg-wp2.web.cern.ch/edg-wp2/

[7] http://edg-wp2.web.cern.ch/edg-wp2/replication/replica-location-service/index.html

[8] DataGrid: User Guide for EDG Replica Manager v1.5.4. DataGrid-02-ERM-USER-GUIDE

[9] Grid Information Services for Distributed Resource Sharing. K. Czajkowski, S. Fitzgerald, I. Foster, C. Kesselman. Proceedings of the Tenth IEEE International Symposium on High-Performance Distributed Computing (HPDC-10), IEEE Press, August 2001.

[10] Report on the results of Run #2, Final application report. DataGrid-08-D8.4-0126-1-2.