GES 2007 German e-Science

Available online at http://www.ges2007.de

Distributed Analysis within the LHC computing Grid

J. Elmsheuser, G. Duckeck and D. Schaile

Ludwig-Maximilians-Universität München, Department für Physik,
Am Coulombwall 1, 85748 Garching, Germany
email: johannes.elmsheuser@physik.uni-muenchen.de
phone: +49 (0)89 289 14149, *fax:* +49 (0)89 289 14103

Abstract

The distributed data analysis using Grid resources is one of the fundamental applications in high energy physics to be addressed and realized before the start of LHC data taking. The needs to manage the resources are very high. In every experiment up to a thousand physicist will be submitting analysis jobs into the Grid. Appropriate user interfaces and helper applications have to be made available to assure that all users can use the Grid without too much expertise in Grid technology. These tools enlarge the number of Grid users from a few production administrators to potentially all participating physicists.

The GANGA job management system (http://cern.ch/ganga), developed as a common project between the ATLAS and LHCb experiments provides and integrates these kind of tools. GANGA provides a simple and consistent way of preparing, organizing and executing analysis tasks within the experiment analysis framework, implemented through a plug-in system. It allows trivial switching between running test jobs on a local batch system and running large-scale analyzes on the Grid, hiding Grid technicalities.

We will be reporting on the plug-ins and our experiences of distributed data analysis using GANGA within the ATLAS experiment and the EGEE/LCG infrastructure. The integration and interaction with the ATLAS data management system DQ2/DDM into GANGA is a key functionality. In combination with the job splitting mechanism large amounts of analysis jobs can be sent to the locations of data following the ATLAS computing model. GANGA supports tasks of user analysis with reconstructed data and small scale production of Monte Carlo data.

1 Introduction

The distributed data analysis using Grid resources is one of the fundamental applications in high energy physics to be addressed and realized in the near future [1]. An efficient analysis environment and the know how to use and enhance it are key goals for the community to achieve, if we are to profit from the high investments made into the accelerator and detectors at the LHC.

The needs to manage the resources are very high. In every experiment up to a thousand physicist will be submitting analysis jobs into the Grid, namely LCG [2], the Grid flavor developed especially for the large hadron collider LHC. Appropriate user interfaces and helper applications have to be made available to assure that all users can use the Grid without too much expertise in Grid technology. These tools enlarge the number of Grid users from a few production administrators to potentially all participating physicists.

2 Job Scheduler and Gap Analysis

Within the D-Grid High Energy Physics Computing Grid Working Package 3 [3] we are working on distributed and interactive data analysis on the Grid. In this context a gap analysis has been pursued to identify missing features and components of distributed analysis tools. All this was done with a closer look into the computing environment, Athena, of the ATLAS experiment [4].

An analysis job at the ATLAS experiment will typically consist of a Python or shell script that configures and runs a user algorithm in the Athena framework, reading and writing event files and/or filling histograms /n-tuples. More interactive analysis may be performed on large datasets stored as n-tuples. The distributed analysis system must be flexible enough to support all work models depending on the needs of a single user or an analysis team. A distributed analysis system should be robust and easy to use by all collaboration members. The look and feel of the system should be the same whether one sends a job to one's own machine, a local interactive cluster, the local batch system, or the Grid.

There are several scenarios relevant for a user analysis:

- analysis with fast response time and a high level of user interaction,
- analysis with intermediate response time and interaction,
- analysis with long response times and a low level of user interaction.

The first point is well matched by the parallel ROOT facility PROOF [5] for interactive usage and fast turn around times on a local computing cluster. For the second and third point an automatic job manager and scheduler in an distributed analysis environment is the key feature for a robust system.

The job and scheduling manager GANGA [6] proved to be a very good candidate and was closely examined. GANGA (Gaudi / Athena and Grid Alliance) is an interface to the Grid that is being developed jointly by the ATLAS and LHCb experiments. GANGA is a front end for job definition and management of analysis jobs to run in a distributed environment. It helps in the creation and configuration of user analysis jobs, submission of the jobs, monitors job status and helps in saving any output. In particular GANGA aims to help with setting up jobs that run the standard ATLAS and LHCb applications. It can be run on the command line, with Python scripts or via a graphical user interface. Many of the required features are already included in GANGA, but some are missing or some need further refinement. GANGA has been used in tests and real user interaction. It performs well in configuring, submitting, monitoring and output retrieval of a few hundred to thousand jobs.

An automatic job manager and scheduler should fulfill the following specifications or functionalities:

Interface for job configuration: There is a common interface to set the configuration of programs used, e.g. software version, configuration of program components, datasets, etc. On the one hand this interface has to be adapted to the specific needs of the user application, but must also prove to be flexible enough to interface to different applications.

Job submission interface for Grid and Batch systems: User analysis programs can be sent to different local and remote computing sites. One can choose both between different batch systems and Grid flavor or even experiment dependent production systems. Job splitting and parallel-/bulk submission of analysis jobs is supported.

Integration of data management: The analysis jobs use a data management system specific to the experiments. Based on this jobs are sent to the data location, to prevent the transfer of large data volumes. In addition datasets are divided and allocated to the sub jobs in a job-splitting mode.

Resource estimation: The resources required for a job are estimated, e.g. memory requirements or CPU time. In addition, available resource informations about the Grid are gathered.

Job monitoring: During execution of the user job there is a continuous monitoring of the job status. Information about the remote execution site, queue status and successful termination are collected.

Job error checking: It is verified that a job has been successfully terminated. Informations about possible errors are gathered and categorized. The job scheduler should offer an automatic job resubmission function for failed jobs.

Collecting and merging of the results: After a job finishes, logging and error messages should be automatically transfered to the submission site. It is possible to merge job results if there have been executed in parallel sub jobs.

Job archive: There exists a job archive, to offer informations about terminated user jobs and to provide templates for new user jobs. It would be desirable to connect this archive to a meta data storage system.

The basic functionality of the features mentioned above are already existing in several areas:

- Within GANGA the start parameters of generic programs or Athena programs can be configured interactively from the shell command line, by a graphical user interface or with Python scripts. Jobs can be sent to the following backends for execution: local desktop computer, LSF-, PBS- and SGE-batch systems and the LCG/gLite [7] Grid environment. Furthermore Grid backends for OSG [8] and NorduGrid [9] are under development.
- The data management was at the time of the gap analysis only implemented at a basic level. Data to be processed was either sent with the job or downloaded to the worker node during Grid execution. There was no option to split jobs into sub-jobs to analyze subsets of a large dataset in

parallel jobs.

- Main features of the job monitoring existed. More detailed status information of scheduled and running jobs are necessary and a web monitoring interface is desirable.
- Error checking only exists on the Grid- and batch-system level. A system to gather and categorize problems and errors during job execution was missing. There is no automatic resubmit mechanism for failed jobs.
- After job termination the job output and log-files are automatically retrieved and stored.
- A job archive exists in a simple matter.

Figure 1 shows the work-flow of a job scheduler. A user provides an application and its configuration before job execution. The dataset to be processed is queried for its location and contents in the DQ2/DDM data management database [10]. Depending on the size of the dataset the Grid job is divided into several sub-jobs that are executed in parallel and are only processing a subset of the input dataset. During execution jobs are monitored. During job completion the results and output files are stored as a output dataset with a reference in the DQ2/DDM data management system database. Afterwards the user can retrieve the output by simple matters.

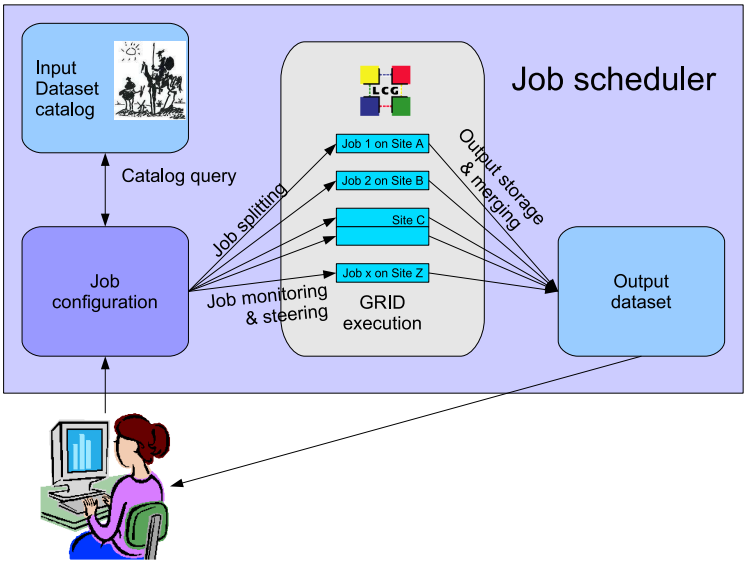

Figure 1: Outline of a job scheduler work-flow. A detailed explanation is given in the text.

3 Job Scheduler GANGA and Extensions

We have successfully extended the functionality of GANGA in numerous areas. The most important are the extension of job splitting for ATLAS jobs and the integration of the ATLAS data management system DQ2/DDM with direct access to the input data files via POSIX I/O [11]. These items are of great importance, since only with an intelligent job splitting and parallelization and a robust data management system a successful distributed analysis system can be built.

A more detailed description of these extentions is given in following paragraphs:

- The mechanism of job parallelizing and job splitting in ATLAS analysis jobs has been introduced to GANGA. The job parallelization functionality provides a splitting based on input files or input parameters. The number of input files of a dataset are evenly distributed among the parallel jobs and executed independently at the same time. Similarly jobs can be started in parallel with different parameter sets. Since high energy physics applications in general are easily dividable into independent tasks, the performance increases linearly with the number of jobs. This procedure greatly reduces the overall processing time of large datasets and is an integral part of the distributed analysis functionality.
- The access to the new ATLAS data management system DQ2/DDM has been integrated within GANGA and within the particular Grid jobs. Users only need to provide an input dataset name to process all corresponding data files on the Grid. The Grid jobs are sent to the site where the data is located. All additional input data staging and preparation is done automatically for the user.
- The domain detection for LCG Grid jobs is an important part of the ATLAS data management system DQ2/DDM integration. Four different detection mechanisms have been implemented.
- The direct access to the input dataset files on the Grid worker node via POSIX I/O protocols on dCache [12] or Castor [13] storage elements has been enabled with ROOT. Large datasets can be processed, without previous download to temporary areas. This reduces the load on the computing and storage elements on a Grid site.

There are several further GANGA extensions:

- A new plug-in has been created to configure and run scripts and transformations of the official ATLAS Monte Carlo production system. These can be used for a small scale MC production of a single user on the Grid.
- Grid jobs can be started with the Condor-G [14] interface. This offers the possibility to very efficiently submit jobs in large bulks in a short time. Generic programs and ATLAS Athena analysis jobs can be submitted not only to the LCG Grid but also other Grid flavors like OSG. The job submission time greatly decreases via the bulk submission mode. The submission time of one hundred jobs can be reduced from several minutes using the

default LCG resource broker to a few seconds using Condor-G. Using the new gLite resource broker also results in a similar gain in submission speed as the Condor-G submission mode.
- A further plug-in for input datasets on local files systems has been integrated. This enables the simple testing of program code on a local desktop computer before sending it to the Grid for a larger exercise.
- An improved mechanism to manage output datasets in the GANGA job repository has been introduced. Jobs are automatically tested for their integrity in the completing phase. Storage locations and Grid identifiers are saved in the DQ2/DDM database.
- A mechanism for output data merging of several sub-jobs has been introduced. ROOT files or text log files can be merged after they have been downloaded in background threads from the remote storage elements to the local desktop computer.
- A better mechanism to report errors of the Grid job execution has been implemented. The exit and error codes of all execution steps during a Grid job have been classified and are reported back to GANGA and saved in the job repository.
- Numerous tests of all these new functionalities in GANGA have been tested locally in Munich and CERN. Tests in the LCG Grid have been pursued especially at the Tier1 sites at Karlsruhe and Lyon. Several input datasets have been replicated to Karlsruhe and Lyon to test the DQ2/DDM integration of GANGA.

Most of the points mentioned in the job scheduler gap analysis of Section 2 have been improved. The following parts offer the necessary basic functionality, but have to improved in some details, or need to be better integrated or made more robust:
- Job configuration interface
- Job submission interface for Grid or batch systems with parallel jobs
- Integration of the DQ2/DDM data management system
- Collecting and merging of the results
- Job archive
- Job Monitoring

But firstly a new intensive testing phase with the a larger user community will happen. The following parts are partially available or need larger improvements or extensions within GANGA:
- Resource estimation
- Job progress monitoring
- Job Error Checking

4 Job scheduler usage

GANGA offers three different ways to configure a job:
- on the shell command line by specifying the job configuration as command line input parameters

- in an interactive IPython shell [15] of GANGA by configuring the parameters of the GANGA plug-in objects
- in a graphical user interface (GUI) which similarly configures the parameters of the objects.

Figure 2 shows an example job configuration of an Athena job for command line submission to the Grid. The user specifies the input dataset, the output dataset file name, the number of sub-jobs for parallel processing, the backend and computing element the job should be sent to and the job executable configuration file. The name of the computing element can also be omitted. If it is omitted the job follows the ATLAS computing model and is automatically sent to the location of the data.

```
ganga Athena
  --inDS csc11.005320.PythiaH170wwll.recon.AOD.v11004107
  --outputdata AnalysisSkeleton.aan.root
  --split 3
  --lcg
  --ce ce-fzk.Gridka.de:2119/jobmanager-pbspro-atlasS
  AnalysisSkeleton_topOptions.py
```

Figure 2: Example Athena job configuration of a shell command line job submission with GANGA.

The monitoring of running jobs is happening in the interactive IPython shell or GUI. A monitoring loop is querying the status of all running jobs and controls the output retrieval of finished jobs. Similarly to the shell command line jobs can be configured with the IPython shell or GUI. Figure 3 shows an screen shot of the GANGA GUI. On the left side of the window a panel with the job repository with several finished and running jobs is show. A detailed job configuration of the GANGA plug-in objects for a single jobs is shown on the right panel.

A typical example for a user task is a small scale Monte Carlo production of a few ten thousand events. This was exercised using GANGA and the ATLAS experiment components for generation, simulation and reconstruction. This scenario mimics the analysis patterns mentioned before of intermediate to low user interaction and intermediate to long response times.

The Monte Carlo event production was done in several steps: event generation, simulation, digitization and reconstruction. For each step jobs were sent to 3 different German Tier1/2 sites in separate Grid jobs with input and output files and results stored on the Grid storage element at Gridka in Karlsruhe. Every job consisted of a start and wrapper script that configured the different Athena settings and input and output datasets. The processing of the datasets, that consisted of a few thousand events, had all been parallelized into sub jobs of 50 events each. One production of 10000 Monte Carlo events was done with 603 jobs and only 2 failed jobs because of worker node failure at one Grid site.

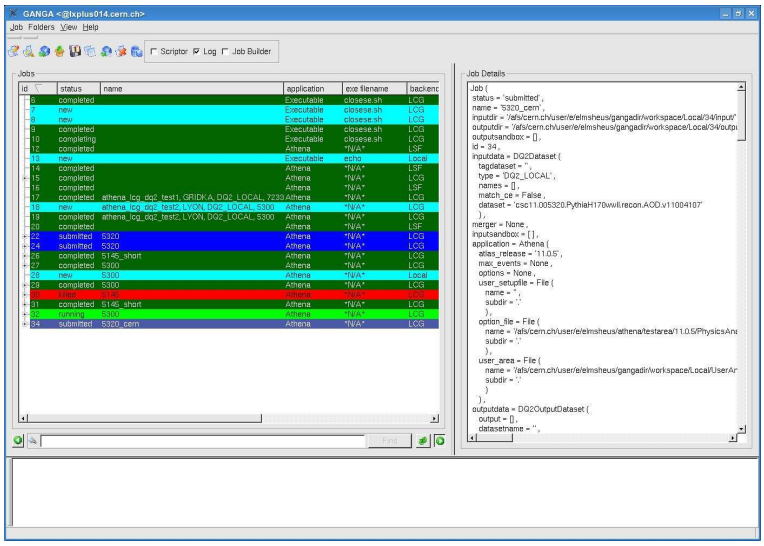

Figure 3: Screen shot of the GANGA GUI. On the left side the job repository panel with several finished and running jobs is show. A detailed job configuration of the GANGA plug-in objects for a single jobs is shown on the right panel.

GANGA performs well in this test of configuring, submitting, monitoring and output retrieving of these few hundred jobs. The submission time of a single job to the LCG is about 10-20 seconds, ie. submission of a few hundred jobs need a bulk submission feature like in gLite. Furthermore the error handling and recovery of failed jobs in the user analysis code needs to be improved by an automatic resubmission or error parsing. This could be assisted by a bookkeeping mechanism of the processed datasets.

5 Conclusions

It has been discussed that the distributed data analysis using Grid resources is one of the fundamental applications in high energy physics that is being used in the upcoming phase of LHC experiment data taking. Several different user analysis scenarios require different response times and levels of user influence. User analysis with intermediate to long response time and low influence need a robust and easy to use interface and job scheduler to make use of all available resources. The job scheduler GANGA is a very good candidate for these type of jobs. We have extended GANGA in various area to improve its core functionality. Numerous tests using GANGA have been carried out at different Grid sites like the Tier 1 computing centers at Karlsruhe and Lyon. All these tests show a good performance and stable behavior of the GANGA components.

Acknowledgements

This work was supported by the BMBF, Germany.

References

1. D. Baberis *et. al*, Common Use Cases for a HEP Common Application Layer for Analysis, LHC-SC2-2003-032
2. LCG project web page: `http://lcg.web.cern.ch/LCG/`
3. D-Grid HEPCG project web page: `https://www.d-Grid.de/`
4. ATLAS Computing TDR, CERN-LHCC-2005-022
5. ROOT and PROOF project web page: `http://root.cern.ch/`
6. GANGA project web page: `http://ganga.web.cern.ch/ganga/`
7. gLite project web page: `http://glite.web.cern.ch/glite/`
8. OSG project web page: `http://www.opensciencegrid.org/`
9. Nordugrid project web page: `http://www.nordugrid.org/`
10. ATLAS DQ2/DDM project wiki page: `https://twiki.cern.ch/twiki/bin/view/Atlas/DistributedDataManagement`
11. Portable Application Standards Committee project web page: `http://www.pasc.org/plato/`
12. dCache project web page: `http://www.dcache.org/`
13. Castor project web page: `http://castor.web.cern.ch/castor/`
14. Condor project web page: `http://www.cs.wisc.edu/condor/`
15. IPython project web page: `http://ipython.scipy.org`