Distributed Analysis Experiences within D-GRID

J. Elmsheuser, Ludwig-Maximilians-Universität München, Germany

Abstract

The German LHC computing resources are built on the Tier 1 center at Gridka in Karlsruhe and several planned Tier 2 centers. These facilities provide us with a testbed on which we can evaluate current distributed analysis tools. Various aspects of the analysis of simulated data using LCG middleware and local batch systems have been tested and evaluated. Here we present our experiences with the deployment, maintenance and operation of the tools.

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. 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.

D-GRID PROJECT

Since September 2005 five community projects and the D-Grid [2] integration project (DGI) started within the D-Grid consortium to build a sustainable Grid infrastructure in Germany. This infrastructure will help to establish methods of e-science in the German scientific community. The community projects will develop together with the integration project a general and sustainable Grid-infrastructure, that will be available for all German scientists.

In the context of the German D-Grid project, different aspects of Grid tools have been assessed. Existing Grid middleware and tools for distributed and interactive analysis are reviewed. A gap analysis is pursued to identify missing features and components. All this is done with a closer look into the computing environment, Athena, of the ATLAS experiment [3].

DISTRIBUTED ANALYSIS

The need for distributed analysis follows from the distribution of the data in various computing facilities according to the ATLAS Computing Model and other LHC experiments [3], and by the availability of the CPU resources required to perform the actual analysis on large datasets. Table 1 summaries the size and time budget and the number of events expected for the ATLAS experiment and current experiences from the DØ experiment. The analysis at DØ is based on centrally produced sub skims of the whole data set based on reconstructed particles. These skims are usually further reduced in a second skimming step by analysis specific selection criteria to about 100 to 100000 events in n-tuple format. The processing is done in a few hundred parallel jobs on local batch system farms and reduces the size of the sub skims from a few Terabytes to a few Gigabytes. The need for more computing resources in an distributed environment in the first steps of the analysis chain becomes clear by extrapolating these numbers to the LHC experiments, which will have larger event sizes and number of total events,

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. Similarly DIANE [6] can be used on a local computing cluster or Grid environment for fast response and parallel processing but with

lower user level interaction. 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. Here the distributed analysis environment GANGA [4] is planned to serve the need for a common user interface for user analysis job configuration, scheduling and submission to different Grid flavors.

Table 1: The table shows the event data sizes, the corresponding processing times and related operational parameters for the ATLAS and DØ experiments.

Item	Unit	Value
ATLAS Raw Data Size	MB/evt	1.6
DØ Raw Data Size	MB/evt	0.25
DØ Rec Data Size	KB/evt	∼10-50
ATLAS Events	evt/year	2×10^9
DØ Total Events '02-'04 ($\mathcal{L}$= 380 pb^{-1})	evt	1×10^9
DØ 2μ-Skim Size	evt	55.2×10^6
DØ 2μ-Skim Size	TB	1.1
DØ 2μ Presel. n-tuple	evt	2×10^5
DØ Analysis on TMB	evt/s	10-30

JOB SCHEDULING

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

Interface for job configuration: There is a common interface to set the configuration of programs used, e.g. software version, configuration of program components, data sets, 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 data sets 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.

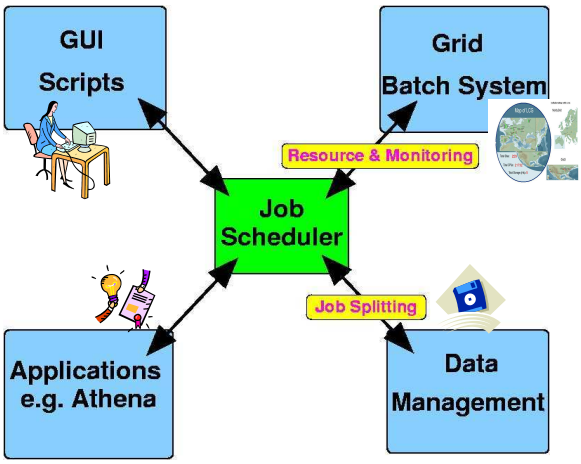

Figure 1: The figure shows a sketch of job scheduler interactions. The job scheduler has a direct user interaction and needs to be flexible to interact with the different user application, job execution systems and data handling systems.

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.

Figure 1 shows the outline and interaction of a job scheduler with the different components. The job scheduler has direct user interaction and needs to be flexible to interact with the different user application, job execution systems and data handling systems.

JOB SCHEDULER EXAMPLES

As an example of a job and scheduling manager GANGA was closer examined and used. 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 an 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 main ATLAS and LHCb applications. It can be run on the command line, with Python scripts or via a graphical user interface. Many of the features mentioned in the previous section are already included in GANGA:

- In GANGA one can setup generic user programs or e.g. Athena, interactively, with Python scripts or with a graphical user interface. The jobs can be sent for execution to the following systems: local computing systems, LSF- and PBS batch systems, LCG and gLite and more. It is planned to also include the ATLAS production system and further grid flavors like OSG and NorduGrid.

- The data management is currently implemented at a basic level and will be extended in the near future. Data is currently sent with the job to a remote site or it can be accessed directly at the remote site. Based on the functionality of the LCG middleware user jobs are sent to the data. Job splitting and parallel execution are currently being developed.

- Job monitoring is included. The level of information being fed back into GANGA depends on the user analysis job and the amount of information available through the Grid middleware. An extension to web based monitoring might be useful especially when a larger amount of jobs are being executed.

- Job error checking exists on the level of Grid- and Batch system middleware. Error handling in program execution is not implemented.

- After job termination program logging and error messages are automatically stored in a job archive together with the job configuration. There is not yet a mechanism to merge results.

GANGA has been successfully extended to incorporate different functionalities: a plugin was written to execute generic and experiment specific user programs on PBS batch systems. This extension was developed and intensively tested at the German Tier1 computing site Gridka in Karlsruhe. There also exists a prototype for the parallel execution of several ATLAS framework jobs. This plugin is used to split Athena jobs into several sub jobs based on the data set. The sub jobs can be executed on a local or remote computing systems.

JOB SCHEDULER USAGE

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. A second and third production of 1000 events each was done with a total of 106 jobs each and about 50% jobs failing because of file catalog problems and database changes in the ATLAS system.

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 data sets.

CONCLUSIONS

It has been shown 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.

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] D-Grid project webpage: `http://www.d-grid.de/`

[3] ATLAS Computing TDR, CERN-LHCC-2005-022

[4] GANGA project webpage: `http://ganga.web.cern.ch/ganga/`

[5] ROOT and PROOF webpage: `http://root.cern.ch/`

[6] DIANE project webpage: `http://cern.ch/diane/`