Eur. Phys. J. C 53, 467–471 (2008)
DOI 10.1140/epjc/s10052-007-0499-9

THE EUROPEAN PHYSICAL JOURNAL C

Special Article – Scientific Note

First experience and adaptation of existing tools to ATLAS distributed analysis

S.G. De La Hoz[1,a], L.M. Ruiz[1], D. Liko[2]

[1] IFIC – Instituto de Física Corpuscular, Centro Mixto Universitat de València – CSIC, Valencia, Apartado de Correos 22085, 46071, Spain
[2] CERN – European Organization for Nuclear Research, 1211 Geneva, Switzerland

Received: 16 October 2007 / Revised version: 28 November 2007 /
Published online: 18 December 2007 –

Abstract. The ATLAS production system has been successfully used to run production of simulation data at an unprecedented scale in ATLAS. Up to 10 000 jobs were processed on about 100 sites in one day. The experiences obtained operating the system on several grid flavours was essential to perform a user analysis using grid resources. First tests of the distributed analysis system were then performed. In the preparation phase data was registered in the LHC file catalog (LFC) and replicated in external sites. For the main test, few resources were used. All these tests are only a first step towards the validation of the computing model. The ATLAS management computing board decided to integrate the collaboration efforts in distributed analysis in only one project, GANGA. The goal is to test the reconstruction and analysis software in a large scale Data production using grid flavors in several sites. GANGA allows trivial switching between running test jobs on a local batch system and running large-scale analyses on the grid; it provides job splitting and merging, and includes automated job monitoring and output retrieval.

1 Introduction

The primary goal of the distributed analysis is to bring computation power to individual ATLAS physicists. This is achieved by providing easy access to the computing resources of the various grids, in a way that hides most of the complexities of grid environments.

The distributed analysis model is included on the ATLAS computing model [1] and stipulates that data is distributed in various computing facilities. User jobs are in turn routed depending on the availability of relevant data. A typical analysis job consists of a Python [2] script that configures and executes a user defined algorithm in Athena (ATLAS software framework). The script specifies the input data and produces one or more files containing plots and histograms. The expected volume of data recorded for offline reconstruction and analysis will be of the order of 1 PB (10^{15} bytes) per year. Due to the size of this expected data volume it is necessary to use distributed resources all over the world to perform reconstruction and analysis of the data.

The ATLAS computing model covers all aspect of this operation. It includes organized production of simulated data, and also user analysis. In this paper we describe our experience running ATLAS distributed analysis tools.

The ATLAS production system has been developed to perform the simulation data production of the experiment using grid resources. It provides a robust framework to execute a large number of jobs in the grid infrastructures.

This experience will allow us to compare the execution of analysis task using such a system versus using direct submission to the infrastructure. This activity was part of the data challenges (DC's) that were organized to validate the computing model to ensure the correctness of the technical choices.

The collaboration decided to perform these DC's in the context of the LHC computing grid project, LCG [3], which contains the majority of ATLAS resources, but also to use both the middleware and the resources of two other grid projects, OSG [4] and NorduGrid [5]. The aim is to prepare the computing infrastructure for the simulation, processing and analysis of the LHC data.

1.1 Atlas production system

In order to handle the task of DC's an automated production system was designed Fig. 1. The ATLAS experiment requires a large amount of simulated data. The ATLAS production system (ProdSys) [6] allows to manage the production of a large amount of Monte Carlo data using grid resources in an automatic way, with minimal human intervention.

This system is implemented in a modular way to enable ATLAS to use resources out of these three infrastructures. All jobs are defined in a specific schema and

[a] e-mail: santiago.gonzalez@ific.uv.es

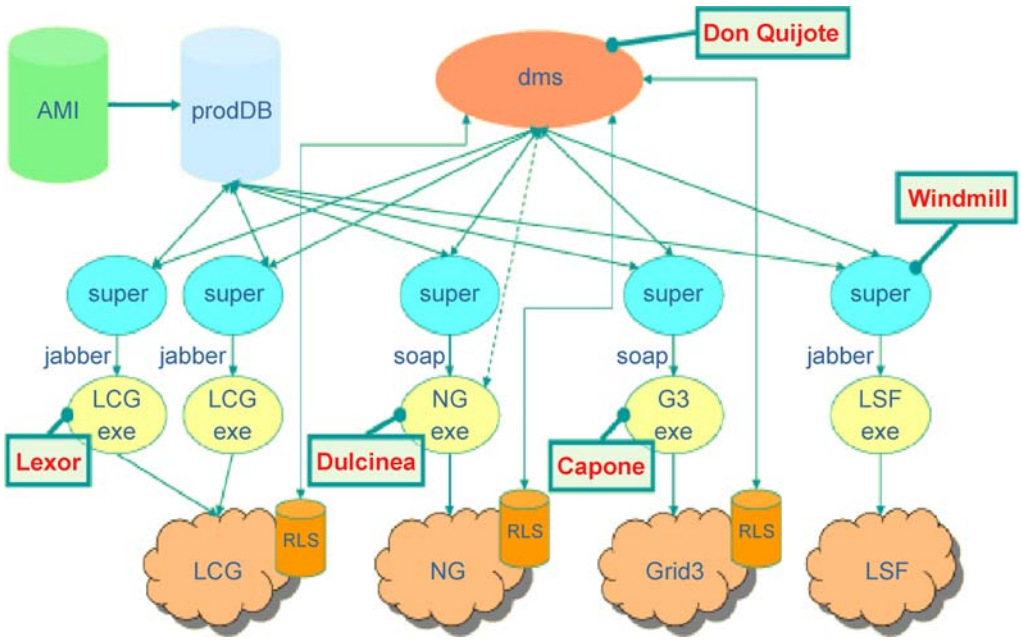

Fig. 1. Atlas production system

stored in a central database. A supervisor agent picks them up, and sends their definition as an XML message to the various executors. Executors are specialized agents, able to convert the ATLAS specific XML job description into a grid specific language. Three executors were developed, for LCG (Lexor and CondorG), NorduGrid (Dulcinea) and OSG (Capone). All the data management operations are performed using a central service, Don Quijote (DQ) [7]. DQ moves files from their temporary output locations to their final destination on some Storage Element and registers this location in the Replica Location Service of the corresponding grid flavor. Thus all the copy and registration operations are performed through an abstraction layer provided by DQ. This allows operating the different replica catalogues of the three grid flavors in a similar way.

The ATLAS ProdSys has been used since the ATLAS Data Challenge 2 (DC2) and is currently being used for the ATLAS Data Challenge 3 (DC3, also called Computing System Commissioning, CSC).

The ATLAS ProdSys distinguishes between two levels of abstraction: task and job. A task transforms input datasets into output datasets by applying a task transformation. Datasets are usually quite large and consist of many logical files. In this case, a job transforms input logical files into output logical files by applying a job transformation. In this way, one could say that a task is split into several jobs and these jobs are managed by ATLAS ProdSys on the different grid flavour resources, so the overall production system relies on the performance of the individual grid systems.

According to the ATLAS full simulation chain, one can classify these jobs as: event generation (evgen), simulation (simul), digitalization (digit) and reconstruction (recon) jobs. The ATLAS full simulation requires a chain of different programs with different characteristics in term of memory usage and CPU time consumption. Typically, a simulation (long) job runs for 24 h, whereas a digitization or reconstruction (short) job runs for 3 to 4 h.

The ATLAS production system was successfully used in DC2 to run production jobs at an unprecedented scale for a system deployed on about 100 sites around the world. On successful days there were more then 10 000 jobs processed. In the ATLAS DC2 exercise a total of 10 million events were processed in $\sim$ 260 thousand jobs, consuming $\sim$ 200 kSI2k years of CPU and producing $\sim$ 60 TB of data.

During DC2 period, which took 6 moths, the automatic production system submitted about 235 000 jobs, reaching approximately an average of 2500–3500 jobs per day, distributed over the three grid flavours. Overall, they consumed $\sim$ 1.5 million SI2k-months of CPU ($\sim$ 5000 CPU months on that average present day CPU) and produced more than 30 TB of physics data. About 6 TB of these data were moved using DQ servers.

2 Distributed analysis strategy

The ATLAS distributed analysis system is in evolvement and several approaches are being studied and evaluated. In this way, ATLAS has adopted a multi-pronged approach to distributed analysis by exploiting its existing grid infrastructure via the various supported grid flavours and indirectly via the ATLAS production system (see Fig. 2).

Figure 2 shows various front-end clients enabling distributed analysis on the existing grid infrastructures. These front-end clients (PanDA [8]/Pathena, GANGA [9] and ATCOM [10]) are intended to perform distributed analysis according to Fig. 2.

PanDA (production and distributed analysis) is a job management system associated with OSG designed specifically for both distributed production and analysis. PanDA

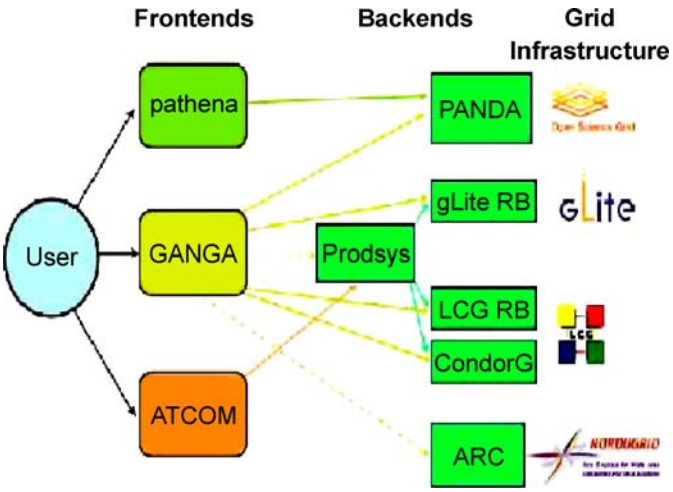

Fig. 2. Various front-end clients are intendet to perform distributed analysis on the existing grid infrastructure. These front-end are PanDA/Pathena, ATCOM and GANGA

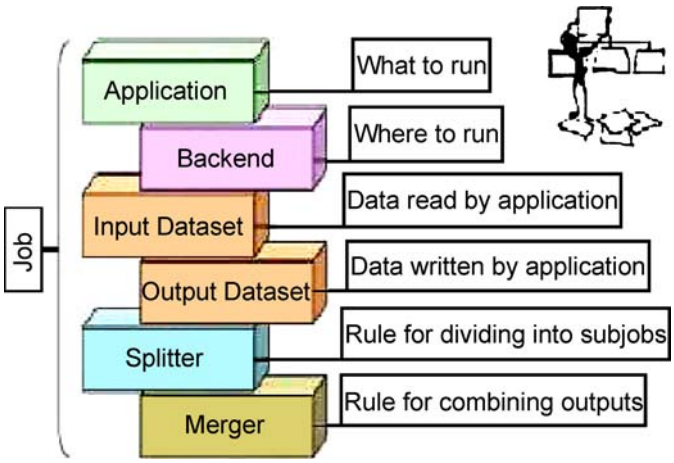

Fig. 3. Building blocks for constructing a GANGA job

has native support for the ATLAS distributed data management (DDM) system allowing accepting datasets as input (pre-staging it whenever required) and producing datasets as output (retrievable using DDM tools). PanDA offers users a comprehensive system view presenting heterogeneous distributed resources as a single uniform resource, accessible via a standard interface. It also has extensive web-based job monitoring and browsing capabilities. Panda does not have a graphical user interface (GUI) but looks to GANGA [9] in order to provide a graphical job definition and submission interface. PanDA has achieved this by exposing a useful set of client API.

Pathena is a python script designed to enable access to OSG resources via the PanDA job management system. It is in the process of becoming a drop-in replacement for the executable used in the ATLAS software framework. Users are able to exploit distributed resources for their analysis activities with minimal inconvenience. Pathena makes the submission of analysis jobs to the PanDA system a painless two-stage process involving an optional build step (where user code can be compiled) followed by an execution step (with built-in job splitting capabilities). A further merge step is in development, allowing the resulting output datasets from split jobs to be consolidated.

ATCOM (the ATLAS commander) [10] was the dedicated graphical user interface front-end to the production system, designed to be used by a few expert users involved in large-scale organized production of data. It had the potential to be used for distributed analysis purposes as well. The ATLAS management computing board decided to integrate the collaboration efforts in distributed analysis in only one project, GANGA.

GANGA (Gaudi/Athena and grid alliance) is a powerful user friendly front-end tool for job definition and management, jointly developed by the ATLAS and LHCb experiments. GANGA provides distributed analysis users easy access to the whole grid infrastructure supported by ATLAS. This is achieved by interfacing to an array of submission back-end mechanisms.

It currently provides two user interface clients: a command line interface (CLI) and a graphical user interface. In addition, it can also be embedded in scripts for non-interactive/repetitive use. GANGA, intended to satisfy both ATLAS and LHCb requirements (unlike Pathena and ATCOM that are designed for specific ATLAS tasks), has been designed from the onset to be a highly extensible generic tool with a component plug.-in architecture. This pluggable framework makes the addition of new applications and backends an easy task.

GANGA allows switching between testing on a local batch system and large-scale processing on the Grid, and helps to keep track or results. A job in GANGA is constructed from a set of building blocks (Fig. 3). All jobs have to specify the software to be run (application) and the processing system (back-end) to be used. Many jobs specify an input dataset to be read and/or an output dataset to be produced. Optionally, a job may also define functions (splitters and mergers) for dividing a job into subjobs that can be processed in parallel, and for combining the various outputs. In this case, after splitting, the job becomes a master job and provides a single pint of access for all subjobs.

GANGA is currently in active development with frequent software releases and has an increasing pool of active developers.

2.1 Distributed analysis using the production system and using GANGA

Distribution of data on several sites and local access to the data is a very important issue to minimize failures. In total 155 GB of merged datasets were used for distributed analysis. The dataset were registered at CERN in Logical File Catalog (LFC) and were replicated in the sites shown in Table 1.

The algorithm of choice has been a $Z_H \to t\bar{t}$, a heavy Z decaying into tops in the little Higgs model. This dataset was made in the official production for exotics working

group using the ATLAS full chain. A total of 400 analysis object data (AOD's) were produced using the Athena full simulation chain, each one containing 50 events (20 000 events in total). The analysis has been performed using the production system and GANGA.

Despite the possibility to run analysis jobs via the production system, not all functionalities to support distributed analysis were currently available. In the following, the technical issues that had to be addressed are discussed in turn.

A dedicated database was setup for analysis jobs to separate private work from the ongoing production. A generic analysis transformation was created, that compiles user code or the user package on the worker node, processes AOD input files and produces histograms as output. ATCOM was used to define jobs. It was also used to monitor the status.

To perform an analysis the user has to define a task and associated jobs according to the conventions of the ATLAS production system. The task contains summary information about the jobs to be run (input/output datasets, transformation parameters, resource and environmental requirements), while individual jobs are defined by their specific parameters needed for execution. Task and jobs were created in the ATLAS dedicated analysis database using ATCOM, a pilot tool to provide a GUI for the production system. This tool supported only creating, editing and submitting jobs. A task and four jobs were defined. Each job had as input 100 AOD's (5000 events in total).

We used a production system instance to pick up the jobs defined in the database and submit them to LCG resources. Moreover, ATCOM was used to monitor the progress status of tasks and jobs. Some performance test were made running these analysis jobs at sites shown in Table 1, because jobs should be submitted where data are located. The time consumed by copy of input data (stage in) from the storage element (usually data on tapes) to the worker node was around 11 min per job, algorithm CPU processing time around 4 min and copy of output data to local storage element (stage out) around 1 min.

Output data were stored using the IFIC/CERN CASTOR storage system. These output ROOT files containing histograms were merged using ROOT.

We defined the same jobs using GANGA. It provides a set of ATLAS-specific features such as application configuration based on the Athena framework and input data location based on DDM. It can be run either on the command line, with Python scripts or through a GUI. Users need to enter just a few commands to set application properties and submit jobs to run the application on selected back-ends (grid flavours or local batch system).

In this case the time for each job in the stage in was around 12 min, algorithm CPU processing time around 5 min and stage out around 1 minute.

In both cases, with the production system and with GANGA, each job produced three output files (ntuple, histogram and log) stored on Castor based storage elements (SE). All the jobs were running at several sites (see Table 1) and were instructed to save output at one single storage elements close to the user. The ROOT package was used to merge the histogram output files and to analyze the results. Finally, after merging, Fig. 4 shows the analysis result.

The same AOD's were copied in a local desktop in order to have local disk access and they were analyzed without using the production system or GANGA, therefore it means without using the grid. In this case the time for the algorithm CPU processing was around 100 min, which was worse than we got with GANGA or with the production system using grid resources.

Our experience in the ATLAS Distributed system is only limited to the LCG grid flavour, so it means that we have not used Pathena yet. Comparing GANGA and ProdSys prototypes:

- GANGA provides two user interface clients: A command line interface and a GUI. Prodsys provides a GUI (ATCOM) also to define and monitor tasks and jobs, but id doesn't hide all the grid environment complexities for users as GANGA. In this sense, GANGA is more user-friendly than ProdSys. Users only need to fulfil some fields using GANGA GUI to submit jobs.
- ProdSys has been used by "expert users" who have some background about the system. However, GANGA is used by general users who don't need to have any background about the back-end systems.
- GANGA supports different back-ends to run the jobs (see Fig. 2), including ProdSys as one of these back-ends. However, ProdSys only supports itself.
- ProdSys is easy to use, but it needs a ProdSys instance running, which makes the system more difficult to run. In addition, it is unclear who should maintain this ProdSys instance for analysis purposes: expert users, ProdSys tem or any other group.

Table 1. Sites where datasets were replicated

Site	Storage element	Computing element
IFIC	castorgrid.ific.uv.es	lcg2ce.ific.uv.es
Sinica	Lcg00123.grid.sinica.edu.tw	lcg00126.grid.sinica.edu.tw
Cnaf	grid007g.cnaf.infn.it	gridit-ce-001.cnaf.infn.it
PIC	castorgrid.pic.es	ce01.pic.es
MI	t2-se-01.mi.infn.it	t2-ce-01.mi.infn.it
Cern	castorgrid.cern.ch	lxgate13.cern.ch
RO	T2-se-01.roma1.infn.it	t2-ce-01.roma1.infn.it

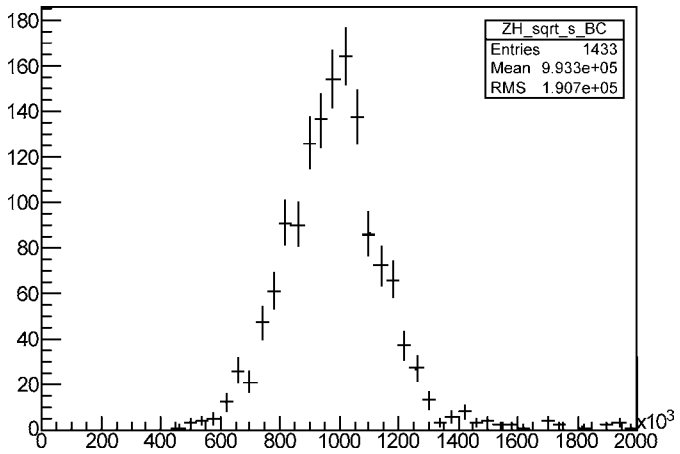

Fig. 4. $Z_H \to tt$ invariant mass distribution after merging the output files (GeV units in the y-axis)

All these tests are only a first step towards the validation of the computing model and distributed analysis. More realistic tests have to involve many physicists working in concurrent mode. This requires not only progress in the application, but also progress in the grid middleware and the site configurations. An example is the ongoing discussion on job priorities in LCG which should allow the coexistence of production and user analysis activities.

3 Conclusions

The ATLAS production system and GANGA have been used to submit physics analysis jobs to LCG grid flavor. Using the production framework for analysis has the advantage to profit from the experience of the large scale production. Only limited additional resources were necessary to perform the required modifications to support analysis.

For the main test few resources were used in nine sites. The production system and GANGA were able to process jobs with 20 k events in about 10 min. In these jobs data were already in the worker node and the stage-in was avoided. It is fair to say that it was difficult to achieve this performance due to the instability of the major components of the software that were still in a development phase. Nevertheless we consider this first test as encouraging and promising. With the startup of the LHC we expect much more data and resources for analysis will be only available using the grid.

Comparing the production system with GANGA, we could observe a more robust execution and we were also profiting from the advanced monitoring capabilities of the production system. A drawback is that such a system represents an additional infrastructure element, which has to be operated by the experiment.

Distributed analysis is still work in progress. With the startup of LHC in the next year we expect a dramatic increase of the data volume. This will require the general ATLAS user to use resources on the grid to perform his analysis.

Acknowledgements. This work is supported by the Spanish National Research Council (CSIC). Work of D. Liko partially supported by Marie Curie grant MERG-CT-2006-44258 of the European Union.

References

1. ATLAS Computing Technical Design Report, CERN-LHCC-2005-022
2. G. Van Rossum, F.L. Drake Jr. (eds.), Python Reference Manual 2.4.3
3. M. Lamana et al., The LHC computing grid project. NIM A **534**, 1 (2004)
4. R. Gardner, Grid3, in: Proc. CHEP04 **2**, 1318
5. M. Ellert et al., The NorduGrid. Project, NIM A **502**, 407 (2003)
6. L. Goossens, Production System in ATLAS DC2, in: Proc. CHEP04, 501
7. M. Branco, Don Quijote, CERN Yellow Report 2005-002, p. 661
8. PanDA (http://twiki.cern.ch/twiki/bin/view/Atlas/Panda)
9. GANGA (http://ganga.web.cern.ch/ganga)
10. ATCOM (http://uimon.cern.ch/twiki.cern.ch/twiki/bin/view/Atlas/AtCom)