Distributed Computing in ATLAS

Guido Negri[*a], J. Shank[b], D. Barberis[c], K. Bos[d], A. Klimentov[e] and M. Lamanna[a]
[a]*CERN – Switzerland*
[b]*Boston University*
[c]*Università & INFN – Genova*
[d]*NIKHEF – Amsterdam*
[e]*BNL – Brookhaven National Laboratories*

E-mail: guido.negri@cern.ch, shank@bu.edu, dario.barberis@cern.ch, kors.bos@cern.ch, alexei.klimentov@cern.ch, massimo.lamanna@cern.ch

The LHC machine has just started operations. Very soon, Petabytes of data from the ATLAS detector will need to be processed, distributed worldwide, re-processed and finally analyzed. This data-intensive physics analysis chain relies on a fabric of computer centers on three different sub-grids: the Open Science Grid, the LHC Computing Grid and the Nordugrid Data Facility--all part of the Worldwide LHC Computing Grid (wLCG). This fabric is arranged in a hierarchy of computing centers form Tier0 to Tier3. The role of the Tier-0 center is to perform prompt reconstruction of the raw data coming from the on-line data acquisition system, and to distribute raw and reconstructed data to the associated Tier-1 centers. The Tier1 centers mainly do raw data reprocessing after updated software releases and calibration constants are ready. The Tier2 centers have two major roles: simulation and physics analysis. Tier-3s, finally, are smaller analysis facilities supporting specific groups studies. This talk will describe the software components of the ATLAS data chain and the flow of data from the Tier0 center at CERN to the distributed Tier1, Tier2 and Tier3 centers. There are five major components which will be discussed. The ATLAS Distributed Data Management system, that is responsible for all data movement and registration in ATLAS. The Storage Resource Management system for dealing with heterogeneous local storage systems. The PanDA pilot based system used to run managed production for both simulated data and real data re-processing. The detailed monitoring system (ARDA dashboard monitoring system) which allows us to debug problems. Finally, the systems which allow distributed physics analysis called GANGA and pAthena..

XII Advanced Computing and Analysis Techniques in Physics Research
Erice, Italy

[*] Speaker

3-7 November, 2008

1. Introduction

The ATLAS[1] experiment is meant to study proton-proton collisions at the LHC accelerator, registering 200 events per second for approximately 50 thousands seconds per day, producing on the average around 16TB of bytestream data per day that will then be analyzed and made available to a very large community of physicists around the world.

In order to cope with such an amount of data and such a wide community, ATLAS has decided to adopt a hierarchical "grid" model: data will be first reconstructed centrally at CERN, in the so called Tier-0, to be translated in a handier object-oriented representation, and then shipped to 10 major regional centers (Tier-1s), distributed all around the world, and then to regional Tier-2s (about 50, in total), smaller analysis facilities situated in the same region of their respective Tier-1, to be further analysed and studied. Finally, the model foresees the presence of Tier-3s, smaller analysis facilities, possibly fractions of a facility otherwise regarded as a Tier-2 or even a Tier-1, where the resources are devoted to local users communities for personal analysis.

ATLAS users will then have to access and make use of data and resources spread over different regions and different Grid flavours. The aim of the distributed analysis projects is to supply the physicist community with a common, simple and intuitive interface to access all the different Grid resources without having to care of what lies beneath it. Applications have been developed to provide such interfaces to the collaboration, to allow users to submit jobs on all Grid flavours, to store and retrieve data from distributed storage systems and to monitor all of these operations.

Focus will be made on the applications that let the users and the collaboration groups to submit jobs to the Grid, efficiently exploiting the available CPUs and assuring the lowest possible latency time in execution.

2. ATLAS analysis

Analysis actions sequence in ATLAS foresees that the user first selects, through a metadata database (AMI [2], in ATLAS), the data he is interested in (based on physics trigger signatures, time range, detector status etc.).

The next step is to create the jobs that have to analyze such data using ATLAS specific software and submit them to the most appropriate resource to run them. The output produced by the analysis can be either stored on distributed storage facilities or retrieved locally on the machine of the user, depending on how such data will be used.

ATLAS jobs can be generated and managed either using GANGA[3] or the combination pAthena[4]/PanDA[5]: both of the two solutions provide an application that can run locally on the user's machine and that can submit jobs to and interact with several Grid flavours.

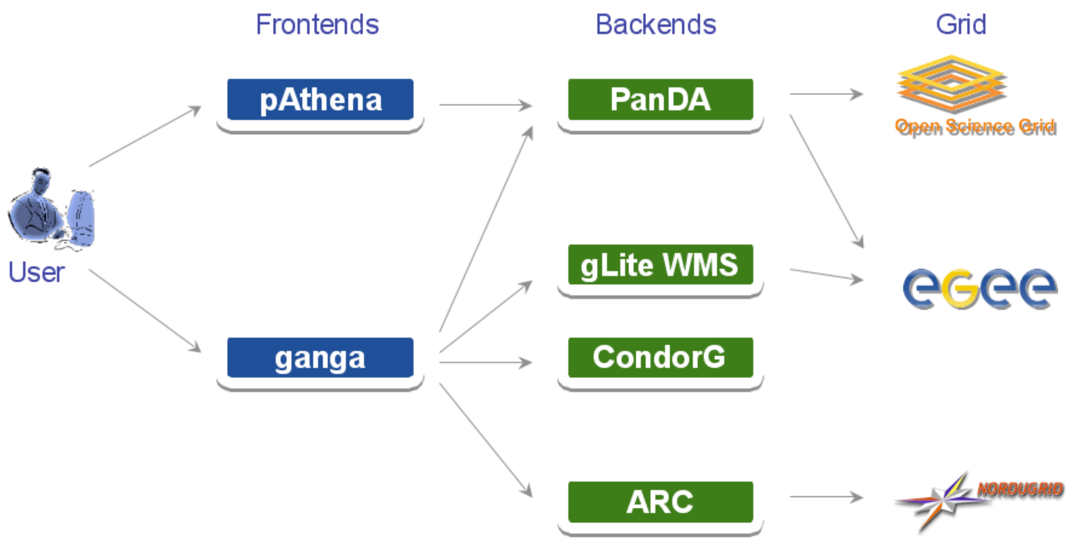

Fig. 1 – ATLAS distributed analysis architecture: different Grid flavours can be accessed through different production systems

2.1 pAthena and PanDA

PanDA was conceived as a massive job submission manager, capable of ensuring an efficient exploitation of the available resources and a minimization of latency and think time, but supporting, at the same time, individual user analysis.

The main requirements that have driven the development of PanDA are throughput, scalability, robustness, efficient resource utilization, minimal operations menpower and tight integration of data management with processing workflow. The result is a modular system that performs a “pilot”-driven exploitation of the computing resources: the PanDA Scheduler sends pilots (job wrappers which set up the environment as required by the actual ATLAS jobs) to the computing machines (Worker Nodes, in the Grid vocabulary). Then the pilots poll the PanDA server, which collects all ATLAS jobs, to receive jobs to be run. The Scheduler is the component that directly interacts with the Grid: the PanDA team has so far developed several schedulers, using different submission mechanisms. A generic scheduler, known as the “AutoPilot”, has been now developed, supporting submission of pilots using Condor-G[6], local batch schedulers (PBS[7] and Condor[6] are supported) and the gLite[8] Workload Management System.

The PanDA server is the main component of the system and it provides a task queue managing all job information centrally. The PanDA server receives jobs through a client interface into the task queue, ordered by a brokerage module which prioritizes and assigns work on the basis of job type, priority, input data and its locality, availability of CPUs, ...

The PanDA server then receives requests for new jobs from the pilots: jobs at the top of the task queue are sent to the Worker Node where the pilot is running.

All communications to and from the PanDA server use a simple Python/HTTP interface. The Apache module mod_gridsite[9] is loaded to provide access control with X.509 certificates.

pAthena is a glue script to submit ATLAS user-defined jobs to PanDA. It provides a simple and straightforward command line interface that has almost the same syntax of the ATLAS analysis software, Athena[10], and it basically wraps it up adding some more information in order to prepare the job for a Grid runtime environment.

pAthena automatically defines ATLAS jobs starting from very simple input informations: the name of the input files to be analyzed, the file containing the jobOptions (the setup used to run the ATLAS software), the name of the output files. pAthena can cope with all production steps (event generation, simulation, pileup, digitization, analysis, ...). It is available under AFS and it installs with CMT[11], Configuration Management Tool, that is the same tool used for installing and setting up the ATLAS software.

Once the jobs have been created by pAthena, they are sent to the PanDA server and they will end up in the task queue and will be eventually picked up as soon as a pilot will ask for new jobs.

Jobs submitted with pAthena can be monitored either using the command line interface provided by pAthena itself (with the so called pAthena_util, installed together with pAthena) or through the PanDA monitoring system: jobs in PanDA have a unique identifier which can be used to track their status from the PanDA monitoring web page. PanDA monitoring mainly relies on a Nagios[12]-based system. But PanDA is now also well integrated with the ARDA Dashboard[13] project and its jobs information can be monitored through the dashboard web pages.

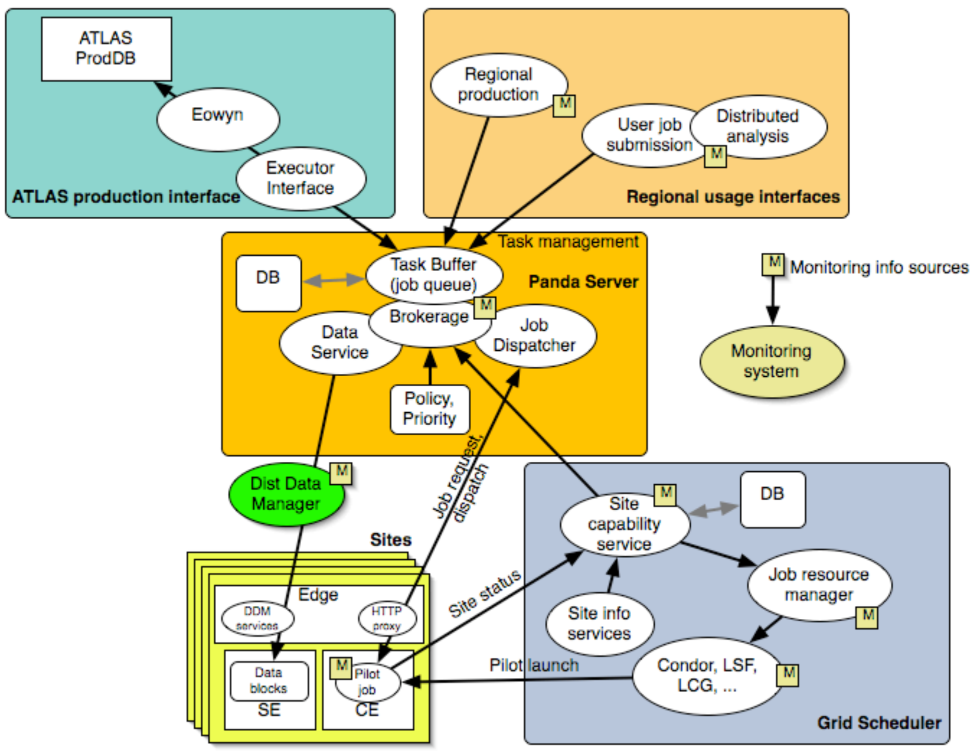

Fig. 2 – PanDA architecture

2.2 GANGA

GANGA started as a LHCb[14]/ATLAS project. The aim was to build an application able to perform, almost automatically, the complete lifecycle of a job, starting from the building of the jobOptions and of the analysis scripts up to submission, monitoring and finally displaying the resulting data.

GANGA supports submission to the local machine (interactive or in background), to local batch systems (LSF[15], PBS, SGE[16] and Condor), Grid systems (LCG[17], gLite and NorduGrid[18]) and production systems (PanDA and Dirac[19]).

GANGA uses pluggable modules to interact with external tools for operations such as querying metadata catalogues, job configuration and job submission. It is written in Python and exposes to the user a public API called the GANGA Public Interface (GPI) which can be accessed via either a Graphical User Interface (GUI), a Command Line Interface (CLI) or by writing scripts and using GANGA as the script interpreter. The CLI offers a shell, called IPython, consisting of an extended Python shell, endowed with all Python modules plus a subset of GANGA modules and packages.

The interaction with underlying Grids is achieved through plugins which hide the complexity of different and new architectures from the end user, who simply has to define his jobs and decide which Grid to use (that is, which plugin to load) for submission.

As it happens with pAthena, also GANGA simply needs to know the name of the input files needed by the job, the job type and its setup and the name of the output: the system automatically defines the job, splits it in a suitable number of sub-jobs, submits them and stores the output on Grid resources. All the processes are perfectly integrated with the ATLAS Distributed Data Management[20] system.

All the jobs information is stored in a loca directory. Jobs status can be monitored through the GUI or the CLI, but also the ARDA Dashboard is automatically updated, through python API, offering the users a very useful monitoring web page.

2.Conclusions

ATLAS activities over different Grid flavours are continuously increasing and a peak is foreseen when there will be real data. Automatic analysis frameworks allow users to access all the available data regardless of where they are stored. Moreover, compared to direct submission to the infrastructure, these frameworks assure the users a more robust execution and profit from the advanced ARDA monitoring architecture they are tightly integrated with and from the thoroughly tested and exploited ATLAS Distributed Data Management system.

Up to now, pAthena/PanDA and GANGA are both suitable solutions for a end users and they are extensively used by the physicist community, offering a simple, common interface for a large variety of tasks. Even though they began as separate projects trying to accomplish different tasks, now they are developing and they are close to become competitive applications, just using different approaches.

References

[1] http://www.cern.ch/ATLAS

[2] http://ami.in2p3.fr/

[3] http://cern.ch/ganga

[4] https://twiki.cern.ch/twiki/bin/view/Atlas/DaonPanda

[5] https://twiki.cern.ch/twiki/bin/view/Atlas/PanDA

[6] Condor, D. Thain, T. Tannenbaum, and M. Livny, "Distributed Computing in Practice: The Condor Experience" Concurrency and Computation: Practice and Experience", 17, 2-4 (2005), 323, http://www.cs.wisc.edu/condor/

[7] Portable Batch System, http://www.openpbs.org

[8] http://glite.web.cern.ch/

[9] McNab A., The GridSite security architecture, Journal of Physics: Conference Series

[10] https://twiki.cern.ch/twiki/bin/view/Atlas/AthenaFramework

[11] http://www.cmtsite.org/

[12] http://www.nagios.org/

[13] http://dashboard.cern.ch/

[14] Large Hadron Collider beauty experiment, http://lhcb.cern.ch/

[15] http://www.platform.com/Products/platform-lsf

[16] Sun Grid Engine, http://gridengine.sunsource.net

[17] LHC Computing Grid, www.cern.ch/lcg

[18] http://www.nordugrid.org/

[19] http://dirac.cern.ch

[20] https://twiki.cern.ch/twiki/bin/view/Atlas/DistributedDataManagement