Functional and large-scale testing of the ATLAS distributed analysis facilities with Ganga

To cite this article: D C Vanderster et al 2010 J. Phys.: Conf. Ser. 219 072021

View the article online for updates and enhancements.

Related content
- Distributed analysis functional testing using GangaRobot in the ATLAS experiment
  Federica Legger and the ATLAS collaboration
- Ganga: User-friendly Grid job submission and management tool for LHC and beyond
  D C Vanderster, F Brochu, G Cowan et al.
- HammerCloud: A Stress Testing System for Distributed Analysis
  Daniel C van der Ster, Johannes Elmsheuser, Mario Úbeda García et al.

Recent citations
- Analysing I/O bottlenecks in LHC data analysis on grid storage resources
  W Bhimji et al
- Tuning grid storage resources for LHC data analysis
  W Bhimji et al
- Establishing Applicability of SSDs to LHC Tier-2 Hardware Configuration
  Samuel C Skipsey et al
Functional and Large-Scale Testing of the ATLAS Distributed Analysis Facilities with Ganga

D C Vanderster¹, J Elmsheuser², M Biglietti³, F Galeazzi⁴, C Serfon² and M Slater⁵

¹ European Organization for Nuclear Research, CERN CH-1211, Genève 23, Switzerland
² Ludwig-Maximilians-Universität München, Geschwister-Scholl-Platz 1, 80539 Munich, Germany
³ The University of Birmingham, Edgbaston, Birmingham, B15 2TT, United Kingdom
⁴ Istituto Nazionale di Fisica Nucleare, Sezione di Napoli, Italy
⁵ Istituto Nazionale di Fisica Nucleare, Sezione di Roma Tre, Italy

E-mail: daniel.colin.vanderster@cern.ch

Abstract. Effective distributed user analysis requires a system which meets the demands of running arbitrary user applications on sites with varied configurations and availabilities. The challenge of tracking such a system requires a tool to monitor not only the functional statuses of each grid site, but also to perform large-scale analysis challenges on the ATLAS grids. This work presents one such tool, the ATLAS GangaRobot, and the results of its use in tests and challenges. For functional testing, the GangaRobot performs daily tests of all sites; specifically, a set of exemplary applications are submitted to all sites and then monitored for success and failure conditions. These results are fed back into Ganga to improve job placements by avoiding currently problematic sites. For analysis challenges, a cloud is first prepared by replicating a number of desired DQ2 datasets across all the sites. Next, the GangaRobot is used to submit and manage a large number of jobs targeting these datasets. The high-loads resulting from multiple parallel instances of the GangaRobot exposes shortcomings in storage and network configurations. The results from a series of cloud-by-cloud analysis challenges starting in fall 2008 are presented.

1. Introduction

ATLAS is one of four experiments at the LHC at CERN in Geneva, Switzerland. Scheduled to start taking data in fall 2009, the ATLAS detector will produce a vast amount of data resulting from particle collisions within the detector. This data will be distributed around the world to the facilities which comprise the Worldwide LHC Computing Grid (WLCG) [1]. With the data being distributed in this way, physicists will need to perform distributed analysis in order to process the data for interesting physics events.

The topic of this paper is related to the operations of the distributed analysis infrastructure for ATLAS. First, we describe the tools which validate the functionality of the distributed facilities, and then we described methods for stress testing of the infrastructure. The first activity, functional testing, is performed using daily short tests which verify the various distributed analysis workflows. The second activity, large-scale stress testing, is achieved using infrequent (~weekly) large-scale tests.
which are tailored in order to stress specific components of the infrastructure. Using a combination of both activities, the readiness of the ATLAS distributed analysis facilities can be measured and thereby improvements can be deployed.

In this work, both the functional and stress testing tools are built around the Ganga distributed analysis end-user tool [2]. Ganga is a python program developed by LHCb and ATLAS to meet the needs of the analysis user communities. It is a flexible tool that allows users to run arbitrary applications on a number of backends; these backends allow users to run locally, on a batch system, or on the Grid. Ganga is also a convenient programming tool, effectively providing an application programming interface (API) to the ATLAS distributed computing infrastructure. As such, Ganga allowed for rapid development of the functional and stress testing tools described in this work.

For the remainder of this paper, we first give an overview of the distributed analysis situation in ATLAS, then present the tools for functional and stress testing and some of the results of their operation, and finally we conclude.

2. Overview of Distributed Analysis in ATLAS
ATLAS has established a distributed infrastructure to meet its computing needs. By taking part in the WLCG, ATLAS works together with the other LHC experiments to develop and maintain this infrastructure. Until recently, the highest priorities in ATLAS Distributed Computing (ADC) were to solve the problems of data distribution and distributed production: the former is important to ensure that the data produced by the detector can be safely stored at sites around the world, and the latter covers the experiment’s needs to produce simulated Monte-Carlo (MC) data, and to reprocess this MC and real data. Having had a few years to understand these problems, they are now well tested and ATLAS is confident that they will successfully meet the needs of data taking.

Now, the priority is moving to distributed analysis. Even though ATLAS has developed end-user tools, and these tools reuse the same facilities as the distributed production tasks, distributed analysis poses unique challenges to the infrastructure, and as such it is important to validate its readiness.

2.1. Analysis Workflows in ATLAS
When end-users first approach the Grid for their computing needs, they have the expectation that the distributed facilities should be able to automatically execute the analysis code that they have developed; in effect, from a user’s point of view, whatever runs on his or her laptop computer should run on the Grid. The variety of workflows in ATLAS computing makes this requirement a challenge, though the distributed analysis tools attempt to make the Grid as transparent as possible.

By inspecting the specific workflows that users follow, the requirements of the distribute analysis facilities can be understood. First, the classic analyses in ATLAS are built around the Athena data analysis tool. Users are interested in using Athena, and its related application AthenaROOTAccess, to process a variety of data, including MC data, cosmics, and reprocessed data. The ATLAS computing model describes the various formats and sizes of the ATLAS data; notably, the users’ primary need is to process AOD objects, though DPDs and ESDs are also in demand. Finally, TAG-based analyses will be used to directly access data using the help of a metadata index.

In addition, users undertaking calibration and alignment studies may require RAW data access (normally on tape rather than disk) and remote access to the conditions databases. Other users will produce personal MC samples: in effect, this use-case is a small scale replica of the ATLAS MC production. Finally, some users may ultimately want to run arbitrary applications on the Grid, be they ROOT analyses or any type of generic executable.

A more detailed review of the distributed analysis workflows and tools in ATLAS is presented in [3].
2.2. Distributed Analysis Resources in ATLAS

Figure 1 depicts the current architecture of distributed analysis in ATLAS. Users are presented with two frontends, namely the Panda Client (formerly known as pathena) [4], and Ganga, a tool co-developed by LHCb and ATLAS. These end-user tools (which share a common library for mapping ATLAS analyses to the grid [5]) submit analysis jobs to three workload management systems (referred to as backends): the Production and Distributed Analysis system (PanDA) [6], the gLite Workload Management System (gLite WMS) [7], and the Advance Resource Connector (ARC) [8]. These three workload management systems forward the analysis jobs to a variety of computing grids, namely the Open Science Grid (OSG), the EGEE grid and NorduGrid.

The complexity of these grids is seen in the services of which they are comprised. Examples of these services include Don Quixote 2 (DQ2) [9] system for distributed data management, the variety of storage technologies in use (e.g. DPM, Dcache, Lustre, GPFS, CASTOR), and the Oracle conditions databases. In addition to the variety of service types in use across the three grids, the deployments are also highly heterogeneous in their deployment strategies and configurations.

The distributed computing architecture in use by ATLAS presents a number of challenges for validating its readiness for analysis with real data. The remainder of this work presents the methods employed to validate the use-case functionalities and the behaviour of the facilities under load.

3. Functional Testing with GangaRobot

The aforementioned Ganga end-user tool has been used to develop a functional testing system for ATLAS distributed analysis. Specifically, the system makes use of GangaRobot, a component of Ganga which allows for the rapid definition and execution of test jobs. By providing API hooks for pre- and post-processing of the test jobs, GangaRobot can be easily exploited to develop a testing system.

3.1. The ATLAS GangaRobot Testing System

The ATLAS GangaRobot is one such system. As an ATLAS service, it performs variety of daily tests of the distributed computing facilities in order to validate the various workflows in ATLAS analysis. The workflow of the ATLAS GangaRobot is depicted in Figure 2 (a). First, the test jobs are defined by an operator; the jobs are composed of a specific version of the Athena software, and bundled package of analysis code, a list of valid input datasets, and a list of sites to test. These test jobs are usually short
in their execution time, as their purpose is simply to validate the service availabilities, software installations, and data access.

Having the test jobs defined, the GangaRobot then submits the jobs to the WLCG using Ganga. The test jobs are then periodically monitored until they have completed successfully or failed; the results are recorded to a local file. The GangaRobot then publishes the results to three systems. First the raw results are presented in a web interface to be used by grid operators and shifters. Next the results are sent to the Service Availability Monitoring (SAM) \[10\] system; this allows the GangaRobot test results to be presented along with the full suite of SAM tests run at the sites. Finally, the results of the GangaRobot tests are used to generate a runtime information system; this is used by Ganga in order to avoid problematic sites (i.e. sites failing GangaRobot tests are “blacklisted” temporarily until the problem is resolved).

Figure 2 (b) presents the monthly and daily summary pages of the GangaRobot web interface. For failed tests, operators and shifters can troubleshoot failures by viewing error messages and raw outputs of the test jobs. In general, the results of the GangaRobot tests need to be used by the end-user tools to avoid problematic sites. Though the results for EGEE sites are used in a Ganga “blacklist”, the OSG and NG sites are not yet managed in this way; for these grids, failing sites need to set offline manually by a shifter. In the future, a central ATLAS information system (the AGIS project), will need to be used to disseminate the test results.

One problem with the GangaRobot web interface is that, though it has all of the information about test failures, it can cause information overload for non-experts. The SAM interface is more user-friendly and presents the information in a more organized way. Additionally, the GangaRobot test results are used in a runtime information system to avoid problematic sites, and these results are presented to grid operators and shifters through the web interface. The SAM interface is used to present the results along with the full suite of SAM tests run at the sites. Finally, the results of the GangaRobot tests are used to generate a runtime information system, which is used by Ganga to avoid problematic sites.
friendly and better integrated with the grid monitoring systems, and as such, GangaRobot also publishes results to this system. An example of the SAM interface is presented in Figure 3.

3.2. Results from the GangaRobot Functional Testing

A summary of the overall results is presented in Figure 3. These plots were generated by the SAM dashboard at http://dashb-atlas-sam.cern.ch/, and present the daily and percent availability of EGEE sites during spring 2009. It is important to note that failures in these plots do not necessarily indicate responsibility – a failure at a site could result from fault anywhere in the Grid, is not necessarily the result of a problem at the site.

In general, the results show that there are a number of sites with greater than 90% availability over the test period. However, less than half of the EGEE sites pass the tests with 80% efficiency. In figure 3 (a) we see that most errors are transient in nature (i.e. most problems are solved within a few days), and only a few sites have permanent failures.

In summary, the ATLAS GangaRobot is an effective system for validating the distributed analysis facilities. The results show that there is room for improvement for the sites to pass the tests more regularly.

Figure 3. (a) The daily availability of EGEE sites as measured by the ATLAS GangaRobot between December 2008 and March 2009, and (b) the percent availability of the sites over the same period.
4. Stress Testing with HammerCloud

Distributed Analysis presents unique challenges to the grid facilities. For example, consider the requirements of a basic Athena AOD analysis. When running on a typical laptop computer, such an analysis proceeds at roughly 20Hz (i.e. 20 events are processed per second). An AOD event is ~200 kilobytes in size, so this implies a data input rate of 4MBps (assuming each event is read exactly once). It is clear that at this rate, the CPU, and not the local disk, is the bottleneck. However, if this processing rate were to apply to a typical Tier2 analysis facility, with 200 CPUs, the storage and network would be stressed at 800MBps. This is a rather large requirement, and implies 10Gbps network with a powerful storage service. On the other hand, using the same analysis at a 200 CPU site with a 1Gbps network would result in 3Hz, nearly an order of magnitude slower than running on a laptop.

The first stress testing of the distributed analysis facilities was performed in Italy. In this series of tests, five users coordinated to simultaneously submit ~200 jobs to each of the Italian sites. The tests easily saturated the 1Gbps at the sites, and verified the 3Hz number calculated above. Being highly manual and user intensive, it was clear that to scale the testing up to all clouds, an automated stress testing system was needed. Thus, we developed HammerCloud.

4.1. Overview of HammerCloud

HammerCloud is a service developed around Ganga to perform automated stress testing of distributed analysis facilities. The workflow of HammerCloud proceeds through five key stages: test definition, generation, submission, monitoring, and cleanup. To define the tests, an operator first specifies a Ganga job template along with a bundle of the analysis code and a list of appropriate input datasets. For a specific test, the operator decides on a list of sites to test and how many jobs to send to each site. Other test parameters, such as the data access method at the sites (e.g. posix, copy-and-process, or FileStager), can also be specified.

In general, each HammerCloud test job runs an Athena analysis over an entire input dataset. Given a list of datasets, HammerCloud attempts to use the same inputs at all sites, though the data distribution does not provide 100% overlap everywhere. At the specified start time, HammerCloud queries DQ2 to get the up-to-date locations of the requested input datasets, constructs a set of test jobs, and finally submits them with Ganga (using the LCG, NG, or Panda backends, depending on the sites under test). While the jobs are running, HammerCloud periodically polls the jobs and writes their progress to a local MySQL database. In a parallel thread, a script periodically reads the database and produces a web page summarizing the progress of each test. At the predefined test completion time, all of the leftover jobs are killed and the final results are recorded.

To be representative of real conditions, the test jobs used by HammerCloud need to be real analyses from physicists. In particular, most of the testing thus far has been based on an AOD muon analysis; this analysis has been modified to accept any AOD dataset as input to ensure it can run in sufficient numbers at all sites. This analysis is similar to the aforementioned example analysis, and as such is representative of all AOD analyses where reading the input data is the bottleneck. Other analyses, such as a DPD analysis that requires remote access to the conditions database at the local Tier1, are also used to test the sites.

The elaborate metrics recorded by HammerCloud are the main strength of the system. Unlike the pre-existing systems, which record only exit status and log files of test jobs, HammerCloud also records the performance of the test jobs. Specifically, the system records the CPU utilization (i.e. CPU/Wallclock ratio), the number of events read per second, a variety of job timings (including queuing delay, middleware overheads, input data preparation times, job execution times, and the output data storage times). In addition, HammerCloud validates the analyses by comparing the number of input files that were requested to be processed with the number that were actually processed.

At present, the main target of HammerCloud tests has been to stress the storage systems at the sites. HammerCloud evaluates three different data access methods. First, it tests posix access to the files using local protocols; this is used to tune the protocol (e.g. rfio, dcap, gsidcap, and file), including...
testing with various read-ahead buffer sizes. The second method is a copy-and-process strategy; this strategy relies on a large scratch area on the worker nodes. The third method relies on the FileStager module in Athena; this module uses a background thread to just-in-time copy the input files from storage. In this way, FileStager requires a scratch area of twice the nominal file size (once for the current file, and once for the next file).

4.2. HammerCloud Testing Results

The HammerCloud test results are presented in a web interface at http://gangarobot.cern.ch/st/. The results are intended to be kept over a long period of time, so that test results over time can be measured.

A sample output of a HammerCloud test is presented in Figure 4. In this particular test, 12 sites were tested with 600 jobs; the test processed ~20000 files containing 50 million events in under 24 hours. On the left of figure 4, we see the test efficiencies; overall, 85% of jobs were successful, but, for example, some sites completed all of the jobs successfully, while at one site all the jobs were still running after the test had completed. On the right of figure 4, we see how the performance metrics relate. Well performing sites typically have a CPU usage figure approaching 100%; in the example a figure of ~75% leads to an event rate of 15Hz. In contrast, a site that does not keep the CPUs busy (e.g. less that 20% CPU utilization in the example) leads to an event rate under 5Hz. This example demonstrates the importance of the local storage in the distributed analysis architecture.

Overall, HammerCloud has been used to process 74 sites in more than 200 large scale tests. The system has processed ~3 billion events in more than 10 million files using more than 50000 jobs running for ~2 hours on average. In the tests we have observed a success rate similar to the GangaRobot (9 sites >90%, 29 sites >80%). We have also observed that roughly a quarter of the tested sites have an event processing rate greater than 10Hz, and only 7 sites can process data faster than 15Hz on average. Using the FileStager module improves this figure, with 20 sites able to process data faster than 15Hz. The continuously updated summary statistics are available online at http://gangarobot.cern.ch/st/summary.html.

In terms of subjective results, we have found a number of benefits from running the HammerCloud tests. First, we have found that most sites are not optimized for user analysis; HammerCloud is able to identify the typical storage and network I/O weaknesses. Next, many site administrators now rely on HammerCloud tests as a benchmark for their sites; they can make changes to local configurations then request a test to see how the change affects the performance. Finally, we have found that there is no
single data access method that will work optimally at all sites; maintaining a database of the optimal method at each site will be necessary to keep user analysis jobs running at peak capacity.

5. Conclusions

In summary, this paper has presented two systems that ATLAS is employing to validate the distributed analysis infrastructure. The ATLAS GangaRobot service is used to run daily short functional tests; the test results are used to “blacklist” or otherwise mark sites offline when problems arise. The HammerCloud system is a stress testing system which is used to validate the analysis facilities under load; it is used continuously to uncover storage and network limitations at the sites. In future, we plan to continue both of these activities, scaling up both the variety and number of tests performed.

References
[1] Worldwide LHC Compute Grid: http://lcg.web.cern.ch/LCG/
[2] F Brochu et al. Ganga: a tool for computational-task management and easy access to Grid resources. Submitted to Computer Physics Communications. arXiv:0902.2685v1.
[3] J Elmsheuser et al. Distributed Analysis in ATLAS using GANGA. These proceedings.
[4] PanDA Client: https://twiki.cern.ch/twiki/bin/view/Atlas/DAonPanda
[5] D C Vanderster et al. A PanDA Backend for the Ganga Analysis Interface. These proceedings.
[6] T Maeno. PanDA: distributed production and distributed analysis system for ATLAS. 2008 J. Phys.: Conf. Ser. 119 062036 (4pp) doi: 10.1088/1742-6596/119/6/062036
[7] E Laure et al. Programming the Grid with gLite. Computational Methods in Science and Technology. Computational Methods in Science and Technology, 12(1):33-45, 2006.
[8] NorduGrid: http://www.nordugrid.org/
[9] M Branco et al. [ATLAS Collaboration], J. Phys. Conf. Ser. 119 (2008) 062017.
[10] W Ollivier et al. Portal for VO-specific SAM tests and VO-customised site availability. These proceedings.