Large-scale ATLAS simulated production on EGEE

X. Espinal
PIC (Port d informacio Cientifica)
Universitat Autonoma de Barcelona
Campus UAB, Edifici D 08193 Bellaterra, Barcelona (SPAIN)
espinal@pic.es

S. Campana, D. Barberis, L. Goosens and G. Poulard
CERN (European Laboratory for Particle Physics)
Rue de Geneve 23 CH 1211 Geneva, Switzerland

L. Perini, S. Resconi, D. Rebatto, G. Negri, A. de Salvo
INFN (Instituto Nazionale di Fisica Nucleare)
Via Celoria 16, 20133 Milano, Italy

R. Walker
TRIUMF (Tri - University Meson Facility)
4004 Wesbrook Mall Vancouver (BC), Canada

S. Padhi
University of Wisconsin-Madison
150 University Avenue, Madison, WI 53706 (United States)

J. Kennedy
Ludwig-Maximilians-Universitat (LMU)
Theresienstrasse 39, 80333 Munich, Germany

K. Bos
NIKHEF (The National Institute for Nuclear Physics and High Energy Physics)
Kruislaan 409 PO Box 41882, 1009 DB Amsterdam (The Netherlands)

Abstract

In preparation for first data at the LHC, a series of Data Challenges, of increasing scale and complexity, have been performed. Large quantities of simulated data have been produced on three different Grids, integrated into the ATLAS production system. During 2006, the emphasis moved towards providing stable continuous production, as is required in the immediate run-up to first data, and thereafter. Here, we discuss the experience of the production done on EGEE resources, using submission based on the gLite WMS, CondorG and a system using Condor Glide-ins. The overall walltime efficiency of around 90% is largely independent of the submission method, and the dominant source of wasted cpu comes from data handling issues. The efficiency of grid job submission is significantly worse than this, and the glide-in method benefits greatly from factorising this out.

1. Introduction

The Large Hadron Collider (LHC) will begin to take data in Early 2008 at the European Laboratory for Particle Physics (CERN) in Geneva, Switzerland.

ATLAS (A Toroidal LHC ApparatuS) is one of the four big experiments being prepared for the Large Hadron Collider (LHC), a particle accelerator ring installed in a 50-150 metre underground tunnel 27 kilometres in

circumference astride the border between Switzerland and France, ATLAS is designed to explore the fundamental nature of matter and the basic forces that shape our universe.

Protons accelerated by the LHC in two counter-rotating beams will be kept circulating for hours, guided by thousands of powerful superconducting magnets operating at 300 degrees below room temperature, before colliding in the heart of the ATLAS detector at almost the speed of light (pp @ $\sqrt{s}$=14Tev) . The resulting energy of colliding protons will transform fleetingly into particle debris to be examined for signs of extremely rare events, such as the creation of the much-sought Higgs boson.

With a rate of 800 million collisions per second, the LHC will be worlds largest and most powerful particle accelerator when it commences operations in 2008, and ATLAS will be the largest collaborative effort ever attempted in the physical sciences with 1,800 physicists (including 400 students) participating from more than 150 universities and laboratories in 34 countries, all of whom eager to see what new discoveries will be revealed.

With such a rate of collisions, even including the reduced rate due to the online trigger processing farms, the expected volume of data recorded for offline reconstruction and analysis will of a few order of Petabytes ($10^{15} bytes$). This will be analyzed by physicists all over the world [1].

Simulated events are also a key feature for the LHC experiments, commonly named Monte Carlo (MC) production. These events are used to compare theory with the real data.

MC production is performed all over the world, 10 T1s are associated with the EGEE grid infrastructure. Its resources are used to generate huge amount of simulated data, around 120 million events were produced within the period between October 2006 and June 2007.

2 The Atlas production system

The design and construction of an experiment like ATLAS requires a large amount of simulated data in order to optimize the detector design, estimate physics performance, and test the software and computing infrastructure. These samples consists of a large number of simulated events, representing collisions between protons. The full simulation requires the following steps:

• **Event generation**: Hadronic final states using the proton-proton collisions are generated using programs relying on theoretical calculations, phenomenological models and experimental inputs.

• **Detector Simulation**: Interaction of the generated particles inside the ATLAS detector is simulated. Taking into account the real geometry, distribution of material, etc.(CPU time per event = 800 kSI2k.seconds, event size = 2 MB).

• **Digitization**: The detector response is derived from the particle interactions and it is written in a format compatible with the real output of the detector. In addition, because of the high rate of collisions in the LHC, digitized signals from several simulated events can be piled-up to create samples with a realistic experimental background. The digitized events (with or without pile-up) can now be used to test the software suite that will be used on the real LHC data. (CPU time per event= 25 kSI2k.seconds, Event size = 2 MB)

• **Reconstruction**: particle trajectories and energies from the detector are reconstructed. Usually final samples to be used by the physicists.(CPU time per event =15 kSI2k.seconds, Event size = 1.2 MB)

This chain requires to run different programs with different characteristics in terms of memory usage and CPU consumption. Typically a simulation job run for about 24 hours, while a digitization or reconstruction jobs runs for 3 or 4 hours.

The production system distinguishes between two levels of abstraction. On the higher level, input datasets are transformed into output datasets by applying a task transformation. The process of doing this is called a task. Datasets are usually quite large and consist of many logical files. At a lower level of abstraction, input logical files are transformed into output logical files by applying a job transformation. This process is called a job [4].

The ATLAS production system provides a common framework in which any grid flavor may be integrated, is formed from several individual elements which when plugged together provide the required functionality for the submission, tracking, recovery and validation of jobs. The individual elements of the production system are the following (fig. 1):

• Common database for the production jobs (ProdDB).

• Data management system to data transfer and file cataloging (DDM).

• Common Supervisor (Eowyn).

• Executors developed by middleware experts (CondorG, Lexor and Cronus in EGEE).

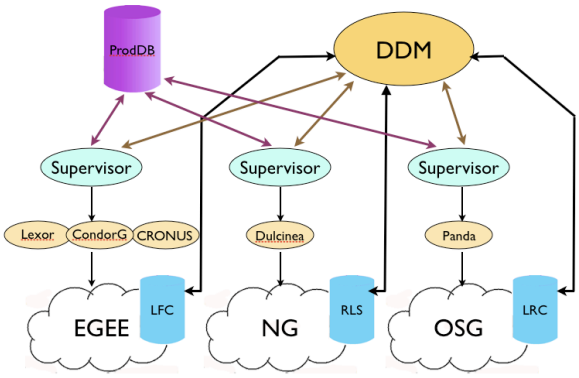

Figure 1. ATLAS Production system schema

The core of the ATLAS production system is formed by the coupling of the Executor with the Supervisor. The Supervisor provides an interface to the job definition data and metadata associated to the computing resources, and retrieves job specific information (Needed software release, data I/O, number of events, etc.) The Supervisor-Executor system allows these jobs to be passed to one of the grid flavors, and then jobs land at the batch system of one of the ATLAS sites around the globe.
Supervisor continues to monitor the state of the submitted job and finally retrieve detailed information about the job once ended. All job info is sent to an ORACLE backend Database which is used to monitor the production system.

2.1 Production elements:

• **Job Transformations**:
The job transformations are the scripts that set up the runtime environment, allow possible compilation of patches to the software in a release, run the Athena executable, parse the log file for known warning and error messages, and tidy everything up at the end. The transformations used up to release 11 of the ATLAS software were implemented as shell scripts, while since release 12 (end of 2006) they are implemented as python script. They can include any data file or patched shared library that may be missing from the release or that may be needed for a particular job. The transformations are production-oriented, but the KitValidation testing suite, used to test the software distribution kits, is also using them. This is done by encapsulating the Job Transformations in a KVT (KitValidation Transformation) test-oriented object. Building the transformations for production jobs is at present a manual operation; the transformations are then put into a Pacman cache and loaded by Grid jobs at start-up. Work is in progress to provide a generic transformation for all usages, including non-production jobs. The transformations will also check the integrity of the expected output file at the end of the job. (This needs the knowledge of the number of output events as counted inside the Athena job.)

• **Production Database**:
There is only a single logical production database. This database holds tables with records for: job transformations; job definitions; job executions and logical files. A job-transformation record describes a particular combination of executable and release. The description includes the signature of the transformation, listing each formal parameter together with its type (restricting the possible values) and its meta-type (indicating how the values should be passed to the executable). Each job-definition record points to its associated job transformation. Other fields allow one to keep track of the current attempt at executing this job (lastAttempt), which supervisor component is handling this job (supervisor), what is the relative priority of this job (priority), etc. The bulk of the job definition is, however, stored as an XML tree in the field jobXML. It lists the actual values to be assigned to the formal parameters of the transformation and additional information about logical input files and logical output files.
For each job definition there can be zero, one, or more job-execution records, corresponding to each attempt at executing the job. Each attempt has a unique number which is appended to the names (both logical and physical) of all files produced, ensuring interference-free operation even in the case of lost and/or zombie jobs. The execution record also records information like start- and end-time of the job, resources consumed, where the outputs were stored, etc.
Last field (logicalFile), the production system stores all metadata about logical files. Most of the information is redundant with respect to the information stored in the respective metadata catalogues of the Grids (size, guid, md5sum, logicalCollection), but at the time the production system was developed these metadata catalogues did not support schema evolution and ATLAS did not know a priori what metadata was needed. Consequently, it was decided to deploy temporarily our own catalogue in addition to filling and using the existing ones.
The production database used in 2004-2005 for Data Challenge 2 and subsequent productions was implemented as an Oracle database hosted at CERN. A MySQL version of the production database is also available for small-scale productions.

• **Supervisor**:

Jobs are retrieved by the Supervisor (Eowyn) from the production database and submitted to the executors and then the jobs are sent over the Grid. The Supervisor maintain a database of the jobs and modify the production database on every change of state. It also manages failed jobs, releasing to be retried (in case failures happened), or abort them in case of persistent failures. And manages the post-processing by filling output files into DDM datasets.

• **Executors**:
There are three executors running in EGEE (Lexor, CondorG and CRONUS), and two more for other grid flavors: PANDA (OSG) and Dulcinea (NG). Executor creates the wrapper files and submit the jobs to the Grid (taking into account the free slots in the sites, etc.)
Executors are able to interpret the job related errors and grid specific problems and retrieve the job after execution. The three type of executors used in EGEE production are presented in more detail in section 2.2.

• **Computing Element (CE**:
gatekeeper/job-manager that submit the jobs to the worker nodes by means of a batch system. Two different types are being used in the EGEE production (LCG-CE and glite-WMS), section 2.3.

• **Worker Nodes**:
Processing farm members that matches the requirements of the production jobs.

2.2 Executors in EGEE

• **Lexor** Lexor is not much more than a translator of prodsys-to-wms requests. It converts the python objects passed by the supervisor into the User Interface API specific python objects, and vice versa. The main ideas leading Lexor implementation were not to duplicate existing middleware functionalities, and to have a thin, stateless layer (states are already stored in the production database and in the grid middleware).
Some manipulation is anyway required, as the mapping between middleware and prodsys objects is not always that trivial. For example, Lexor needs to aggregate jobs in order to take profit of the "bulk submission" feature of the WMS, thus introducing a jobs' collection concept which is extraneous to the production system. A similar bulk operation for retrieving the status of the jobs is available in the middleware, and will soon integrated in Lexor.
In its original implementation, Lexor also included the runtime wrapper (i.e. the script around the actual transformation, responsible in particular of the whole data transfer from and to the grid). This is now part of the Common Executor - the code shared among the three LCG executors - and evolved a lot since its first implementation. It was rewritten in Python and better integrated with both the transformation itself (which is now in Python too) and the DDM layer.

• **CondorG** CondorG is standard Grid middleware for remote job submission to CEs, and indeed it forms part of the LCG RB. In this case the RB chooses the destination CE, and CondorG submits to the named site. However, when given information about the resources, CondorG can also do the resource brokerage. This information is taken from the BDII and converted into the Condor ClassAd format. The fundamental difference, compared with the original Lexor executor, is that the resource brokerage and the submission are done by separate components. The Negotiator and one or more Schedulers run on different machines, and scalability is achieved by increasing the number of Schedulers only. Furthermore, the scheduler is sufficiently lightweight to run much closer to the UI, perhaps on the same machine. The interaction with the local Scheduler is therefore much faster. Similarly the status and getOutput requests are instantaneous as the response is like that of a local batch system.
There are, however, two perceived deficiencies with this approach. First, if the UI machine hosts the Scheduler then it cannot be turned off, which is inconvenient if the UI machine is, for example, a laptop. A second concern was the lack of central logging and book-keeping (L&B) when using CondorG. We should stress "central" because there is in fact a local record of the stages in the job's life, and a mechanism exists to extract this to a MySQL database. The LCG central L&B has been identified as a potential cause of the poor performance, so not having this architecture is an advantage of CondorG. This does not prevent the L&B information being migrated to a central place, asynchronous to job submission. Lexor was used as the basis for the Lexor-CG executor because its modular design allowed the easy exchange of the LCG submission with the CondorG submission. Everything else, including the run scripts, stage-in, stage-out, validation, etc. remained the same and was re-used. During production operation, improvements to Lexor were also applied to Lexor-CG.

•**CRONUS** Production jobs goes to a scheduler that interacts with the Cronus-Central Manager (CCM). From then the Codor-G glide-ins are submitted to the CE's and finally ending in to the WN's, once activated they preserve the Master-Worker relationships, with the worker pulling the production jobs sequentially until the expiry of their lifetimes. The communication between the WN and the

CCM is performed via ClassAds and if the glide-in find that the WN requirement are correct, jobs are submitted. This comunication with the CCM allows to have a full control and monitoring of the jobs running across the grid (fig. 2).

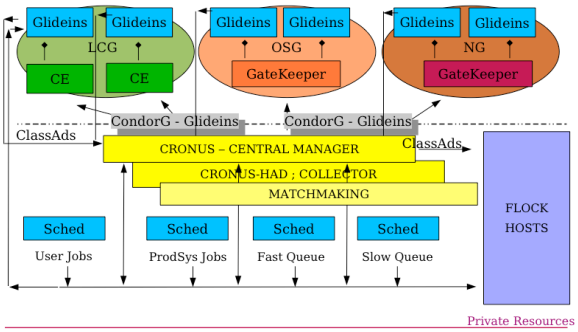

Figure 2. Cronus glide-in job submission method

2.3 Grid middleware dependencies

Th executors contact the CE's in order to deliver the jobs, two types has been used: the LCG-CE and the glite-WMS.

•$LCG-CE$ The Computing Element (CE) contains two logical parts: The gatekeeper/job-manager and the worker nodes. Jobs are distributed to the worker nodes by means of a batch system, such as the Portable Batch System (PBS). Technically, the gatekeeper/job-manager and the batch system server run on one machine, usually called the CE node, to which a number of separate worker nodes (WN) is connected, preferably in a private subnet. The CE provides its local computing resources, such as batch queues, number of processors, and access rights, by way a Monitoring and Directory Service (MDS) which is based on LDAP. This is the so-called Grid Resource information Service (GRIS). On request those data are replied to a central Information Service (IS), such as the Berkely Database Information Index (BDII), or the more recent Grid Information Index (GIIS). The BDII is contacted by a resource broker to match resources, namely an appropriate CE, to a submitted job.
The globus-gatekeeper receives the job from the RB's Job Submission Server (JSS) and calls the globus-job-manager to submit job to the PBS queue.

•$glite-WMS$ The Workload Management System (WMS) comprises a set of grid middleware components responsible for the distribution and management of tasks across grid resources, in such a way that applications are conveniently, efficiently and effectively executed.
The core component of the Workload Management System is the Workload Manager (WM), whose purpose is to accept and satisfy requests for job management coming from its clients. For a computation job there are two main types of request: submission and cancellation.
In particular the meaning of the submission request is to pass the responsibility of the job to the WM. The WM will then pass the job to an appropriate Computing Element for execution, taking into account the requirements and the preferences expressed in the job description. The decision of which resource should be used is the outcome of a matchmaking process between submission requests and available resources.

3 Experience and scope

One of the targets of the production system is to prove the operability of high level distributed computing, since computing demands of the LHC has no precedents. Simulated production was planned with a continuous ramp-up in the number of simulated events.
For that reason, since November 2006, the ATLAS simulated production in EGEE is supervised by the EGEE production team, a group of people following a shift system to perform the job and data babysitting for all the production jobs running on EGEE. This period covers the Service Challenge 4, where was successfully a ramp-up challenge for the simulated production (fig. 3).
The target was to finish 20 M events during November and December 2006, and 40M events during the next quarter (January, February and March 2007). This ramp up ended earlier as the disk resources of almost all the Tier-1 centers were quickly filled. Clearly the production infrastructure was proved and the milestone accomplished:

Period	# events	Comb.job eff.	Comb. WCT eff.
4Q2006	20M	42 %	76 %
1Q2007	63M	59 %	82 %
2Q2007	38M	57 %	90 %

Table 1. Simulated production in EGEE, quarter report. Number of events is extrapolated assuming a mean value of 50 events simulated per job. The combined efficiencies for job and Wall Clock Time (WCT) are the mean value for the three executors actually running on EGEE (Lexor, CondorG and CRONUS).

Executor	Fin. Jobs	Job eff.	WCT eff.	Weight
Lexor	816029	53 %	85 %	34%
CondorG	1039646	51 %	86 %	43%
CRONUS	577680	61 %	86 %	23%

Table 2. Job and WCT efficiencies for the three executors: Lexor, CondorG and CRONUS. The numbers are the average since Novembre 2006 ("begin" of 4Q2006) until June 2007 (end of 2Q2007). Last column (weight) show the percentage of produced events by each one of the executors.

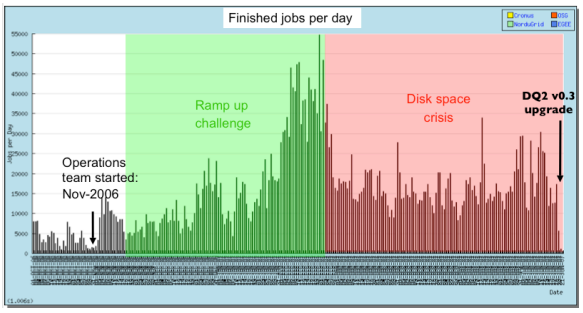

Figure 3. Finished jobs distribution from October 2006 to January 2007 (three grids). Two main zones are clearly seen: ramp-up period (green) and steady state production (red). Finished jobs peaked at 55000 jobs finished in a day.

Job and WCT efficiency has been almost continuously improving (figs. 4, 5), keeping in mind that simulated production is a vivid body as new releases/patches appears regularly provoking temporal periods of inefficiency (validation periods). Since the starting of the joint operations in November 2006 more than 2.5M jobs finished, yielding an amount of 125M simulated events (fig. 6)

3.1 Workload management and Data management

EGEE production operations team look after the jobs running in the grid. Basically there are two main parts to control: Workload and Data management (I/O), hence this two different duties are taken by two different persons during the shift period.

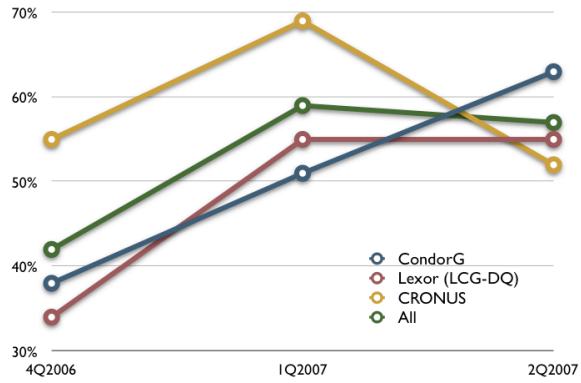

Figure 4. Job efficiency

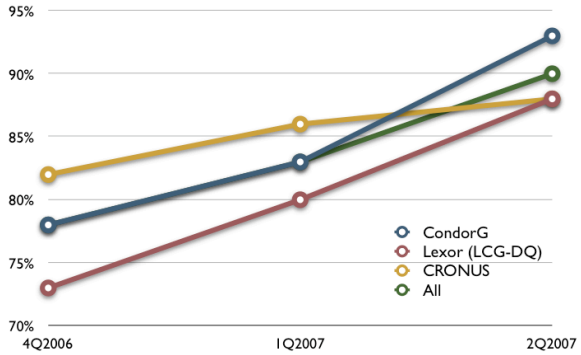

Figure 5. WCT efficiency

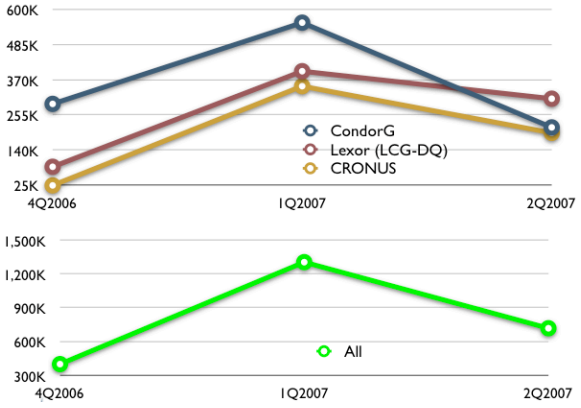

Figure 6. Finished jobs

Workload management is referred to the intrinsic job related problems, a huge variety can yield errors when starting the jobs. **Software** problems at the site: filesystem, local grid configuration, etc. (usually these errors are not easily found and need special investigation at the sites

batch system). **Grid-related** problems: proxy failures, etc. Or **ATLAS** specific software errors: Transformation, task definition,etc.

Data management is related to the stage-in and stage-out failures. This usually have two different sources, one that is related to the site: SE outages, LFC time-outs, BDii errors, etc. And a different one that is a problem of the global Data Management: missing files in the grid needed by the jobs, corrupted files that yield error after the size/checksum exploration,etc.

Errors can be joined in three main groups that are the dominant ones: Executor related errors, Data handling errors and Software related errors (tab. 3

Error group	Job	WCT	Ineff*WCT
Data I/O	35 %	65 %	11%
Software	9 %	32 %	5.4%
Executor	41 %	3 %	0.5%

Table 3. In the second column is shown the percentage of job errors (over the total jobs failed), in the third column Wall Clock Time (WCT) loss is counted for each of the cases (over the total WCT loss produced by the job failures). The last column is the most relevant, showing the real CPU loss by each one of the processes, this has been counted multiplying the WCT "mean inefficiency" during the studied period (Ineff=17%) with the error weight.

Executor errors: Quickly spotted as the job consumed almost no CPU. This is clearly seen in (tab. 3) where the percentage of job error related to the group Executor is 41% but the impact in the lost WCT is very litttle (3%). This errors are mainly related to site-specific problems and are particularly difficult to debug as for a large fraction of failures no output is produced. This requires to log into a site and chase the problem at the WN level. Some examples of this type of errors at WMS/CE level are: proxy expired, hit retry count or job cancelled by the batch system.

Data handling errors: are the most worrying ones, and can be separated in two main groups: Stage-in and Stage-out failures. Both consumes CPU and block the WN while is trying to get or send data. Stage-in process consist of querying the Local File Catalogues of the Tier-1 centers, then the most proximal one is chosen, and the files are copied back to the WN using underlying lcg-utils (lcg-cp) directly form the source Storage Element (SE). This may cause some problems in case the LFC are not responding or the SE is not reachable, as the command for staging-in keeps trying until times-out (Stage-in failures lcg-cp: 27% , errors in size/md5sum: 4%, No replicas found: 1%).

Stage-out process is the more critical one as the file cannot be copied back after the successful finishing of the jobs, wasting all the CPU consumed. Fallback solutions has been implemented since some time, at first step the job try to stores the file at the local SE, in case of failure the job try another SE from the same cloud [1]. If those two failed CERN (Tier-0) is taken as a final stage and the job try to store the file there. Even with this three layer of fallback, errors in the stage-out (lcg-cr) weights 32.5% of the total loss in the WCT.

Software errors: this are commonly due to a problem in the task definition or in the transformation, provoking the job to abort unexpectedly, the impact on the WCT has been around the 20% within the considered period, this is because the transformation is executed some time after the job start. Also site software related problems has been counted in this field: error in lfc-mkdir (7.6%) or downloading Pacman Job Transform (2.3%), software missing at the site (1.5%).

In (fig. 7) is shown the percentage of the job failures since 1st. November 2006 until 30th June 2007:

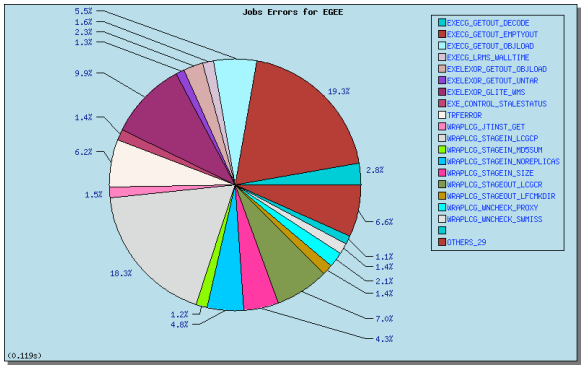

Figure 7. Job error pie, since 1st November 2006 until 30th June 2007

In (fig. 8) is shown the percentage of the WCT failures since 1st. November 2006 until 30th June 2007:

[1] Atlas is organized in clouds. For a cloud is understood the Tier-1 and its associated Tier-2s

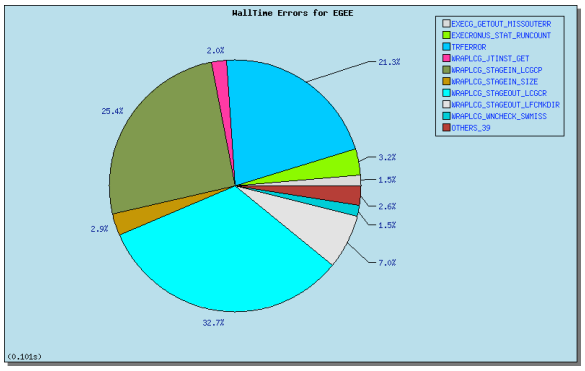

Figure 8. WCT error pie, since 1st November 2006 until 30th June 2007

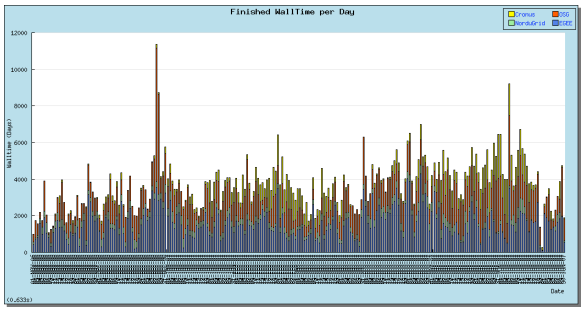

Figure 10. Finished Walltime per day during 1st November 2006 until 30th June 2007, the three Grid flavors are shown together with the CRONUS instance.

3.2 Resources

Resources for ATLAS simulated production are spread around more than 50 sites (10 Tier-1s and approximately 40 Tier-2s), yielding a total power of 26000 kSI2k.month and more than 100TB of disk. The merged walltime days from all sites has been increasing, reaching an average of around 3500 days/day in the last months in (fig. 9) is shown the finished vs. failed walltime and in (fig. 10) the finished walltime is shown for the three different Grid flavors.

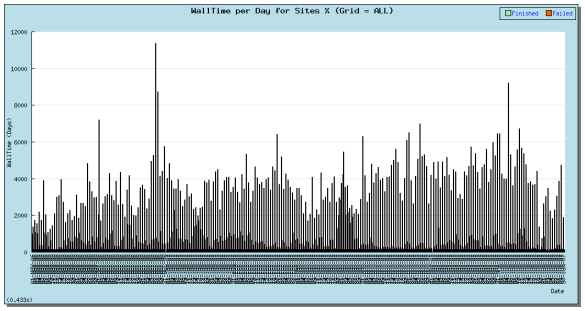

Figure 9. Walltime per day during 1st November 2006 until 30th June 2007, darker bars show the failed CPU time.

4 Summary and Conclusions

The EGEE production system clearly showed to cope with the requirements from ATLAS experiment. The CPU Walltime has been improved obtaining an efficiency of 90% during the past three months. There is still room to improve in the data managing, specially with the control of stage-out failures, which are the most worrying ones.
As the simulated production is expected to grow in the following months, while facing the start of the LHC, some automation for the job babysitting is envisaged.
Also EGEE production system is expecting new advances in middleware as the new SRM (Storage Resource Manager), FTS (File Transfer Service), etc. This would yield a more robust infrastructure and a higher performance system.

References

[1] G. Poulard *Experience on large scale production on the grid*, CHEP 2006.

[2] J. Kennedy et al. *The ATLAS production system*

[3] A. De Salvo (INFN Roma), G. Negri (CNAF Bologna), D. Rebatto, L. Vaccarossa (INFN Milano) *LEXOR, the LCG-2 Executor for the ATLAS DC2 Production System* CHEP 2004.

[4] Computing TDR: http://atlas-proj-computing-tdr.web.cern.ch/atlas-proj-computing-tdr/Html/Computing-TDR-50.htm

[5] http://grid.desy.de/testbed/EDG/CE.html

[6] S. Padhi, Production using CRONUS, ATLAS SW March 2007, presentation &confId=5060

[7] All statistics has been taken from the ATLAS monitoring web: http://atlas.web.cern.ch/Atlas/GROUPS/SOFTWARE/OO/php/DbAdmin/Ora/php-4.3.4/proddb/monitor/Home.php