Migration of ATLAS PanDA to CERN

Graeme Andrew Stewart[1], Alexei Klimentov[2], Birger Koblitz[3], Massimo Lamanna[3], Tadashi Maeno[2], Pavel Nevski[2], Marcin Nowak[2], Pedro Emanuel De Castro Faria Salgado[2], Torre Wenaus[2]

[1] Department of Physics and Astronomy, University of Glasgow, University Avenue, Glasgow, G12 8QQ, UK
[2] Physics Department, Brookhaven National Laboratory, Upton, NY 11973-5000, USA
[3] European Organization for Nuclear Research, CERN CH-1211, Genève 23, Switzerland

E-mail: g.stewart@physics.gla.ac.uk

Abstract. The ATLAS Production and Distributed Analysis System (PanDA) is a key component of the ATLAS distributed computing infrastructure. All ATLAS production jobs, and a substantial amount of user and group analysis jobs, pass through the PanDA system, which manages their execution on the grid. PanDA also plays a key role in production task definition and the data set replication request system. PanDA has recently been migrated from Brookhaven National Laboratory (BNL) to the European Organization for Nuclear Research (CERN), a process we describe here.

We discuss how the new infrastructure for PanDA, which relies heavily on services provided by CERN IT, was introduced in order to make the service as reliable as possible and to allow it to be scaled to ATLAS's increasing need for distributed computing.

The migration involved changing the backend database for PanDA from MySQL to Oracle, which impacted upon the database schemas. The process by which the client code was optimised for the new database backend is discussed. We describe the procedure by which the new database infrastructure was tested and commissioned for production use.

Operations during the migration had to be planned carefully to minimise disruption to ongoing ATLAS offline computing. All parts of the migration were fully tested before commissioning the new infrastructure and the gradual migration of computing resources to the new system allowed any problems of scaling to be addressed.

1. Introduction

The Production and Distributed Analysis system for ATLAS computing was developed from 2005 by US ATLAS[1] to address the need for 'petabyte' grid computing within the ATLAS Experiment[2] based at CERN. The success of PanDA led to its adoption at the end of 2007 as the unified production system for ATLAS across the OSG, EGEE and NDGF grids, a process recently completed by the final integration of NDGF resources into PanDA through the 'ARC Control Tower'.

The growing importance of PanDA to ATLAS offline computing led to the decision in autumn 2008 to migrate the system's infrastructure from Brookhaven National Laboratory to CERN. This migration took place over a long period, starting in November 2008 and finally completing in May 2009. The migration was able to proceed over this long time because the component based architecture of PanDA, which is described in §2, allowed an evolutionary approach to be taken. Firstly a PanDA infrastructure was put in place at CERN, which took advantage of

CERN's Fabric, Infrastructure and Operations' (FIO) services (§3). This allowed, firstly, the PanDA monitor to be setup at CERN. This was followed by the first database migration, the Task Request database (§4.2). This was also the first database which was migrated from MySQL to Oracle. Finally, the migration of the PanDA server began (§4.3), with at first just two clouds migrating to the new CERN Oracle setup. Then further clouds were added, gradually increasing the load on the new infrastructure until the migration was complete (§4.3.3).

2. PanDA Architecture

The architecture of PanDA is based around a central job queue, which holds information about all the jobs which the system wishes to be run at any point in time. For ATLAS production, PanDA receives these jobs from a higher component in the ATLAS production system, the Production Database, or ProdDB[3]. Once PanDA has information about jobs to be run it finds the input data for these jobs from the ATLAS Distributed Data Management system, DDM[4]. If the brokering algorithm in PanDA decides the job should be run at a site which does not currently hold the data, it uses DDM to move the data to the site before the job will be run. Likewise, when jobs have finished on Tier-2 sites, DDM is used to move the data back to the parent Tier-1 (following the ATLAS Computing Model[5]).

While the PanDA system decides which jobs to run at a site, the actual task of job execution is delegated to a pilot job[6]. Once the pilot starts executing on a site's computing infrastructure it contacts the server to ask for a job to execute. If the server has jobs waiting to run at that site it dispatches the current highest priority job for execution there. This allows very flexible late binding of resources to jobs[7]. If there are no jobs to be run at that time the pilot exits, however the fact that there are jobs slots available at a particular site is fed into the PanDA scheduler and will provide an added weight to broken jobs to this site.

Once the pilot has completed the job it signals the success or failure of this execution attempt back to the PanDA server, which updates the job's status and returns this back to ProdDB.

This architecture is sketched in Figure 1.

In addition to jobs from the ATLAS production system, PanDA is also able to execute user jobs, which are defined by means of a client API supported by both the ganga and pathena analysis tools[8].

The PanDA architecture is implemented by having a stateful database, with a number of stateless front ends, which provide load balancing and redundancy. Clients need not care which of the front ends they connect to, so they are hidden behind a DNS alias. For most purposes the pilots, which execute ATLAS jobs, connect to the *PanDA server*, which dispatches and manages job execution on the grid. In addition there is a *PanDA monitor*, which provides a human interface to the system. This allows jobs to be monitored, but also provides other functionality, such as the definition and monitoring of tasks themselves.

2.1. PanDA Scale

Since its adoption in 2007 as the unified ATLAS executor of production jobs, PanDA has proven its scalability. PanDA regularly manages more than 35 000 simultaneous running jobs and more than 120 000 jobs finishing per day. In addition, intelligent brokering of tasks to ATLAS clouds has been introduced, which reduces the level of manual intervention required to run the system.

A one year plot of PanDA jobs running across all ATLAS clouds demonstrates the growth in the scale of the system in Figure 2.

3. PanDA Infrastructure at CERN

3.1. Hardware

In consultation with the system administrators at BNL, it was decided to use three modern server class machines to host the PanDA service at CERN. These machines had dual quad core

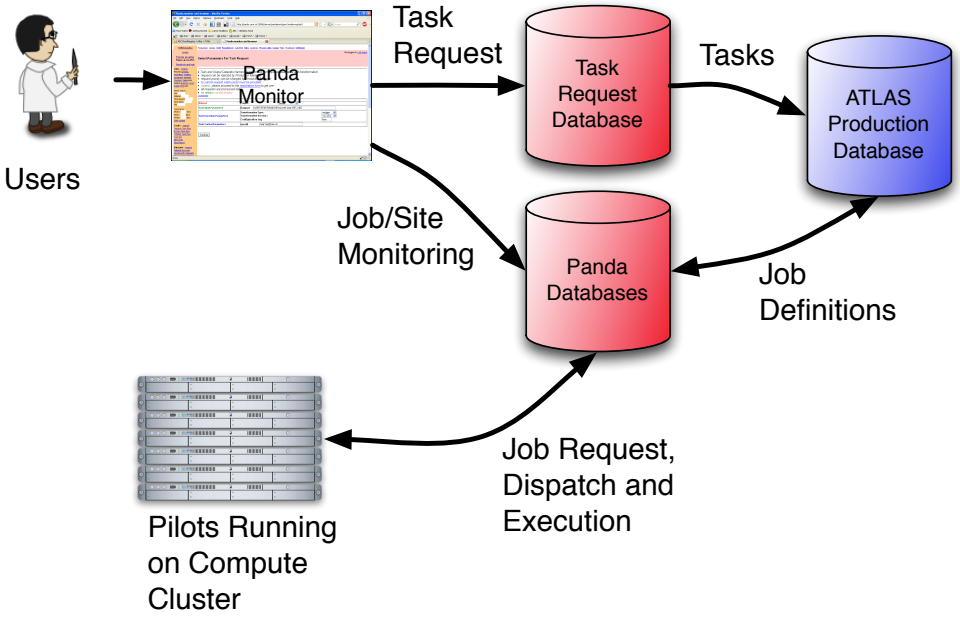

Figure 1. PanDA Architecture

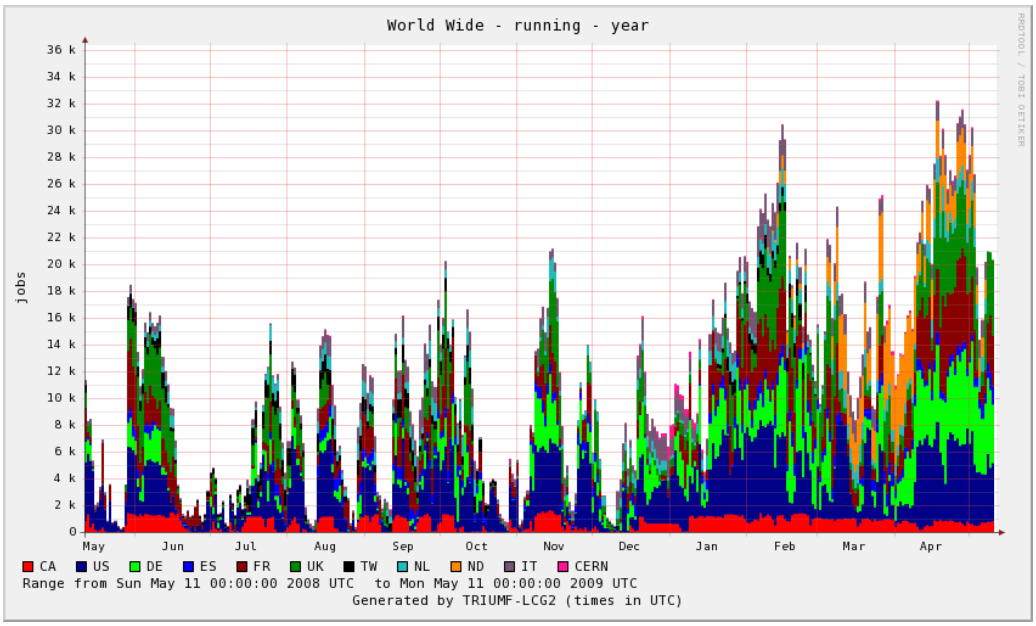

Figure 2. Running PanDA jobs over one year

Intel E5410 CPUs, 16GB of RAM and 500GB of hard disk space. The expected load on the machines was such that two servers would in fact be able to manage the load, with a third being present to provide redundancy.

At CERN, FIO manage these machines and provide hardware level intervention and maintenance as necessary.

The database service for the CERN setup was provided by the CERN database group. This service consists of an Oracle Real Application Cluster (RAC) setup with failover and redundancy

built in.

3.2. Setup
As CERN machines are managed using the Quattor[9] installation and maintenance system. This provides management of the operating system, basic packages, grid certificates and security updates. On top of this installation Quattor templates can be defined, which add additional package sets. For the ATLAS PanDA setup, templates used for the ATLAS Distributed Data Management system, including 64 bit Python 2.5, were applied, providing a similar setup to many other ATLAS central services.

3.3. Monitoring and Maintenance
Basic fabric monitoring of the machines is provided as part of the IT infrastructure managed by CERN FIO. Alarms for common failure conditions, such as excessive load, low disk space, etc. are handled automatically with a call to CERN's piquet service to take appropriate action. In addition, sensors for intelligently testing the health of the service are under development. These sensors will feed into the load balanced alias system[10] of which the three current production machines are a part. This ensures that only healthy machines behind the `panda.cern.ch` DNS alias are returned to clients.

4. Migration

4.1. Monitor
The PanDA architecture of stateless front ends to the system always allowed for multiple monitors to be setup. This was the first task of the migration, which was achieved in early December 2008. It involved verifying and validating the basic infrastructure to run the PanDA service.

As this was a parallel monitor setup there was no service interruption or downtime involved at this stage of the migration.

Once this new monitor setup was achieved, it became the primary PanDA interface for ATLAS users and shifters. However, at this point in the migration, the performance of the monitor actually degraded as a result of the long RTT between the new monitor infrastructure at CERN and the backend database, which at that time was MySQL at BNL. The monitor infrastructure at BNL was maintained throughout the migration process.

4.2. Task Request Database
As outlined in §2, the PanDA system also offers an web interface, through the PanDA monitor, which allows production coordinators to define new tasks in the ATLAS production system. This service requires a special *Task Request* database, which is separate from the rest of the PanDA job execution databases. This task request database, which is specifically designed to interact with task requestors via the web, then pushes the task into the ATLAS ProdDB, which is designed to interact with further automated systems. We note that ProdDB was always hosted on Oracle at CERN, while the jobs were executed though the BNL MySQL panda servers – this made ATLAS tasks a truly international process! (Figure 3.) As the task request database was decoupled from the rest of PanDA, it was felt to be a very suitable candidate for the first database to migrate from BNL MySQL to CERN.

Preparations were made by making an initial export of the data in the database (see §5 for more details), which then allowed the monitor database code, which is responsible for the generation of the SQL queries, to be debugged and optimised.

As large tasks themselves take many days to complete, it was acceptable to have a two day downtime in the task request system while the migration from BNL to CERN took place. On

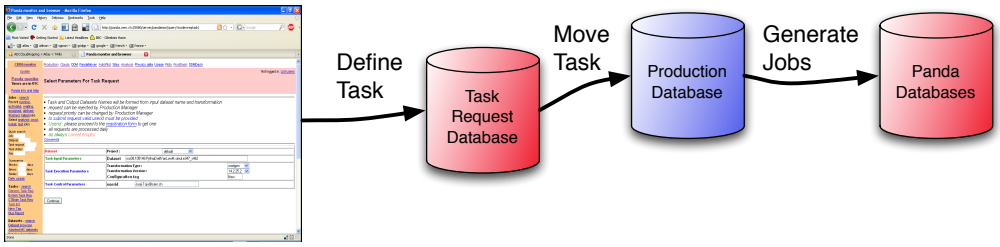

Figure 3. Task from PanDA monitor web interface to jobs defined for execution in PanDA databases

Database	Role
PandaDB	Fast buffer for jobs currently active in the system
LogDB	Pilot logfile extracts
MetaDB	Holds cloud, site and queue information
ArchiveDB	Archive of all jobs ever run by PanDA

Table 1. Panda databases

Monday the 8th of December the task request system was stopped. The data from the BNL MySQL database was exported to CERN and then imported into Oracle in about 2 hours. After this the remainder of the downtime was used to validate the task request interface and run trail task definitions through the system. On the afternoon of Tuesday the 9th of December the system was up and running for production coordinators to define tasks.

There was, however, one unexpected difficulty encountered after the migration: the python `cx_oracle` module used to connect the PanDA monitor to the Oracle databases proved to be unexpectedly hard to install into the BNL monitor infrastructure. This meant that after this phase of the migration the BNL monitors could no longer be used define tasks and the panda monitors were no longer identical.

4.3. Server

As the monitor and task request interfaces were being migrated, code development was underway to prepare the panda server for use with Oracle.

This would involve the migration of four databases from MySQL to Oracle. These databases are shown in Table 1.

Test versions of these databases schemas were established in the CERN database integration service (INTR) and some trial jobs were run successfully through the system to validate the code changes. However, the intention was to migrate PanDA into the production offline database service for ATLAS (ATLR).

4.3.1. First Cloud Migration Since early in 2008, when all of ATLAS production migrated to using the PanDA system, there had been a parallel PanDA setup at CERN, which supported two of the smaller ATLAS computing clouds, Italy and CERN itself (which manages access to some CERN resources through the grid). This setup was a mirror of the BNL one and used MySQL, which was not well supported. With the intention to move to Oracle at CERN it seemed natural to first move these two clouds into the new system.

This was prepared for 9th March, but the CERN Oracle Database Administrators (DBAs) considered that insufficient load testing had been done and advised strongly against a move to the ATLR database at this time (ATLR hosts other ATLAS offline databases and adverse load

from ill formed queries could cause problems for other services). This prompted some significant work over the weekend preceding the proposed migration date to run several thousand jobs through the integration system. Even so, the DBAs still feared that not all client code had been well enough tested and continued to advise further testing. This was problematic as extended testing could only be achieved by removing significant amounts of ATLAS computing resource from the collaboration, which was not acceptable.

Eventually a compromise was reached, where a partial migration would happen, but onto the INTR database. Regular backups of the system were put in place, which gave us sufficient confidence to run real ATLAS production jobs through this set up.

4.3.2. Further Load Initial indications, after the migration of these clouds to the new setup, were good. However, some optimisation of the server code and the database schema (§5.2) was done to improve query performance where required.

As load on the INTR database was low we added other clouds when there was opportunity to do so, e.g., after a significant downtime for the German cloud after a major storage intervention at their Tier-1.

Eventually a procedure was developed whereby clouds could be migrated live, without downtime. This involved a coordinated switch over for pilots to change the server they contacted to request jobs, together with a change in the PanDA configuration to ensure that new jobs were picked up by the new servers and not the old ones. Jobs in the old server which had started running were allowed to complete, but jobs which had not yet started were 'failed' in the old server, which pushed them quickly over to the new server.

Using this technique, eventually eight of the eleven ATLAS computing clouds were migrated to CERN, still running on the INTR database. This accounted for about 60% of ATLAS production capacity as the remaining clouds included two of the largest, France and the United States.

Monitor Issues In fact, the server code was relatively quick to validate as production ready. All queries made by the server are fast, well controlled and act, mainly, on smaller optimised tables. The panda monitor, on the other hand, has to support queries which are significantly more difficult, e.g., deep queries on the very large ArchiveDB table. This requires extensive customisation of the SQL and the use of Oracle specific features, to gain adequate performance. As the monitor continued to support searching of the archives at BNL this required significant amounts of work to finally produce fast well optimised queries.

4.3.3. Final Steps After these improvements in the monitor code were validated the final migration of the database from INTR to ATLR (§5.3) was done. Then the final ATLAS production clouds, FR and US were migrated into the CERN Oracle PanDA service at the beginning of May. This essentially completed the migration process, with only some bookkeeping work to produce a final merged version of ArchiveDB remaining.

5. Databases

The migration of the PanDA service from one database type to another might, at first glance, appear to be a simple matter. SQL is a well established standard and client queries against one database engine should work against any other.

In fact this turns out not to be the case for two main reasons:

(i) The SQL 'dialect' of each database engine may be non-standard in several ways and, unless development is done with portability specifically in mind, use of these non-standard features (which are often very convenient!) will appear in the code.

(ii) Each database engine will have quite specific features and implementation details which impact on how well queries run. In order to optimise performance, database queries must be written with these details in mind.

Much of the hard work of the migration involved reworking the SQL queries and optimising the underlying database schemas.

5.1. Schema Migration

Initial migration of the PanDA schemas was done using the Oracle SQLDeveloper Migration function. However, some improvements to this automated schema can be then made. e.g., the migration tool re-implements the MySQL `AUTOINCREMENT` function as an Oracle trigger 'on update', which means that the client code does not need to be changed. However, it is better if the client code itself implements the internal `NEXTVAL` explicitly in its query and after this is done the trigger can be removed.

5.1.1. Partitioned Tables The most significant change between the MySQL and Oracle schemas was in the ArchiveDB. For performance reasons this had been implemented in MySQL as a series of bi-monthly tables (`Jan_Feb_2007`, `Mar_Apr_2007`, ...). This meant that the client code had to be aware of this structure and generated appropriate chained MySQL queries, aggregating the results if necessary. In Oracle it was possible to re-merge all of these sub-tables into one large partitioned table, where a partition is used for each month's data. While this allowed for a simplification of the client code, it meant that great care had to be taken in structuring the SQL appropriately to avoid triggering a full table scan of a table containing millions of rows.

5.2. Optimisation

Optimising queries on the ArchiveDB in particular meant ensuring that two indexes were always used in the query, and that Oracle was delivered the correct 'hint' so that it chose the best execution plan for the query.

After this had been implemented then deep searches of the ArchiveDB, which probed hundreds of days of the archive and matched more than 100 000 rows could be done in less than 60s.

5.3. Oracle Migration

Once all queries had been optimised the PanDA databases were made ready to migrate from INTR to ATLR. Almost all of the historical data in the ArchiveDB could be migrated in advance, leaving only a small fraction to be moved on the day of migration. The smaller live tables were migrated using Oracle Data Pump while the PanDA server itself was shutdown. In order to avoid losing running jobs, as this migration was being done while the production system was live, the timeout on the pilot for the final update of job status for completed jobs, was increased from 12 minutes to 60 minutes. The migration was completed in just under 90 minutes, so its impact on the production system was very small.

6. Conclusions

The migration of the PanDA system from BNL and MySQL was accomplished in a little over five months. This was considerably longer than was first anticipated because of many small delays which held up the process, other pressures on the time of key people and the unexpectedly difficult work involved in managing the new merged ArchiveDB queries.

Now that the migration has been achieved, a highly scalable setup for this key component of the ATLAS offline computing system has been established. This should serve ATLAS well in the coming years of LHC data taking.

Acknowledgements
Support for this work was provided in part by GridPP, funded by the UK Science and Technology Facilities Council. BNL authors are employed by Brookhaven Science Associates, LLC under Contract No. DE-AC02-98CH10886 with the U.S. Department of Energy.

References

[1] Maeno T 2008 *Journal of Physics: Conference Series* **119** 062036 (4pp) URL `http://stacks.iop.org/1742-6596/119/062036`
[2] The ATLAS Collaboration 2008 *Journal of Instrumentation* **3** S08003 URL `http://stacks.iop.org/1748-0221/3/S08003`
[3] Nevski P, Kilmentov A and Tanaka J 2007 Steering of grid production in atlas experiment *Computing in High Evergy Physics 2007* URL `http://tinyurl.com/op6pnp`
[4] Branco M, Cameron D, Gaidioz B, Garonne V, Koblitz B, Lassnig M, Rocha R, Salgado P and Wenaus T 2008 *Journal of Physics: Conference Series* **119** 062017 (9pp) URL `http://stacks.iop.org/1742-6596/119/062017`
[5] Jones R and Barberis D 2008 *Journal of Physics: Conference Series* **119** 072020 (6pp) URL `http://stacks.iop.org/1742-6596/119/072020`
[6] Nilsson P 2008 *Journal of Physics: Conference Series* **119** 062038 (6pp) URL `http://stacks.iop.org/1742-6596/119/062038`
[7] Thain D and Livny M 2003 *Grid 2: Blueprint for a New Computing Infrastructure* (Elsevier) chap 19
[8] Harrison K, Jones R W L, Liko D and Tan C L 2006 Distributed analysis in the atlas experiment *Proceedings of the UK e-Science All Hands Meeting 2006* URL `http://www.allhands.org.uk/2006/proceedings/papers/720.pdf`
[9] Jouvin M 2008 *Journal of Physics: Conference Series* **119** 052021 (6pp) URL `http://stacks.iop.org/1742-6596/119/052021`
[10] Bahyl V and Garfield N 2006 Dns load balancing and failover mechanism at cern *Computing in High Evergy Physics 2006* URL `http://tinyurl.com/o7fma2`