PROCEEDINGS OF SCIENCE

The PanDA System in the ATLAS Experiment

Paul Nilsson[1a], Jose Caballero[b], Kaushik De[a], Tadashi Maeno[b], Maxim Potekhin[b], Torre Wenaus[b] on behalf of the ATLAS collaboration

[a] *University of Texas at Arlington, Science Hall, PO Box 19059, Arlington, TX 76019, United States*

[b] *Brookhaven National Laboratory, PO Box 5000, Upton, NY 11973, United States*

The PanDA system was developed by US ATLAS to meet the requirements for high throughput production and distributed analysis processing for the ATLAS Experiment at CERN. The system provides integrated service architecture with late binding of job, maximal automation through layered services, tight binding with the ATLAS Distributed Data Management system, advanced job recovery and error discovery functions, among other features.

This paper will present the components and performance of the PanDA system which has been in production in the US since early 2006 and ATLAS wide since the end of 2007.

XII Advanced Computing and Analysis Techniques in Physics Research
Erice, Italy, 3-7 November, 2008

[1] *Corresponding author, E-mail:* Paul.Nilsson@cern.ch

1. Introduction

PanDA is the *P*roduction *and D*istributed *A*nalysis system for the ATLAS experiment [1, 2]. It was designed to meet the ATLAS requirements for a data-driven workload management system capable of operating at the LHC data processing scale. PanDA has a single task queue and is using pilots to execute jobs. The pilots retrieve the jobs from an Apache based central server as soon as CPUs are available, leading to low latencies that is especially useful for user analysis which has a chaotic behavior.

The PanDA system is highly automatic and thus only needs low maintenance and operations manpower that is aided by an integrated monitoring system.

1.1 A brief history of PanDA

The development of PanDA started in the summer of 2005. Initially developed for US based production and analysis, the software infrastructure was deployed in late 2005. Since September 2006 PanDA has also been a principal component of the US Open Science Grid program in just-in-time (pilot based) workload management. In October 2007 it was adopted by the ATLAS collaboration as the sole system for distributed production across the collaboration. PanDA has successfully deployed at all WLCG sites, and is currently being integrated with NorduGrid.

2.Workflow

Jobs are submitted to PanDA via a simple Python client by which users define job sets and their associated datasets. The job specifications are sent to the PanDA server with secure https which is authenticated with a grid certificate proxy. The client interface has been used to implement the PanDA front ends for ATLAS production, distributed analysis and US regional production. The PanDA server receives work from these front ends and places it in a global job queue, upon which the brokerage module operates to assign and prioritize work on the basis of job type, priority, input data and its locality, available CPU resources and other criteria.

The allocation of job sets to sites is followed by the dispatch of the corresponding input datasets which is handled by a data service interacting with the ATLAS Distributed Data Management (DDM) system. Pre-placement of data is a strict precondition for job execution. For analysis jobs there is no dispatch of data – the data must pre-exist locally. Jobs are not released for processing until their input data has are available at the site. When the data dispatch has finished, jobs are made available to a job dispatcher from which the pilot jobs can download them. The pilots are delivered to the worker nodes by an independent subsystem via a number of different scheduling systems.

When a pilot has been launched on a worker node, it contacts the job dispatcher and asks for an available job that is appropriate to the site. If there is no job available at the moment, it waits for a minute and then asks again. If there is still no job available, the pilot will exit. For interactive analysis it is desirable to have minimal latency from

job submission to execution, so it is of particular importance that the pilot dispatch mechanism bypasses any latency in the scheduling system for submitting and launching the pilot.

Pilots carry a generic 'production' grid proxy with an additional 'pilot' VOMS attribute indicating a pilot job. Analysis pilots may use gLExec [3] to switch their identity to that of the job submitter on the worker node.

The PanDA system is illustrated in Fig. 1.

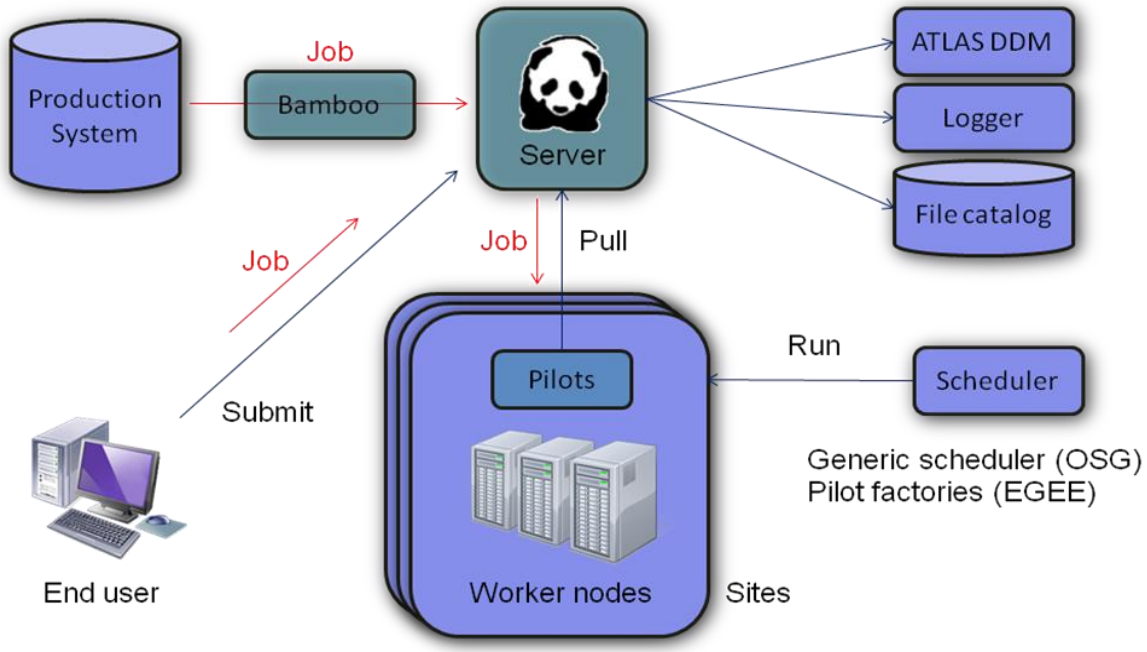

Fig. 1. The PanDA system.

3. PanDA components

3.1 Core components

The following constitute the core components of the PanDA system:

PanDA server. Central hub composed of several components that make up the core of PanDA (task buffer, job dispatcher, etc).
Panda DB. Oracle database for PanDA.
Panda Client. PanDA job submission and interaction client.
Pilot. Execution environment for PanDA jobs. Pilots request and receive job payloads from the dispatcher, perform setup and run the jobs themselves, while regularly reporting the job status to PanDA during execution.
AutoPilot. Pilot submission, management and monitoring system.
SchedConfig. Database table used to configure resources.
Monitor. Web based monitoring and browsing system that provides an interface to PanDA for operators and users.

Logger. Logging system allowing PanDA components to log incidents in a database via the standard Python logging module.
Bamboo. Interface between PanDA and the ATLAS production database.

Sections 3.2 – 3.4 will describe a selection of these components in more detail.

3.2 PanDA server

The PanDA server is implemented as a stateless multi-process REST web service with Apache `mod_python` and with an Oracle back-end. The interaction with the clients is done via http for passive read operations and https for active operations such as job submission and pilot interaction. Secure https is authenticated using grid certificates with Apache `mod_gridsite`. The server consists of the following components:

Task buffer. PanDA job queue manager which keeps track of all active jobs in the system.
Brokerage. Matching of job attributes with site and pilot attributes. It manages the dispatch of input data to processing sites, and implements PanDA's data pre-placement requirement.
Job dispatcher. Dispatcher receive requests for jobs from pilots and dispatches job payloads. Jobs are assigned depending on how they match the capabilities of the site and worker node, e.g. data availability, disk space, memory etc.
Data service. Data dispatch to and retrieval from sites.

3.3 AutoPilot

The AutoPilot is a simple and generic implementation of the PanDA pilot and pilot-scheduler for use in more varied environments than the ATLAS-specific production pilots and schedulers. It governs the submission of pilot wrapper scripts to target sites using Condor-G [4]. The AutoPilot is using a number of databases to manage information on queues, pilot wrappers, etc. One of these databases is the SchedConfig table that holds information about queue configurations. For example, pilot wrapper submission to a site that experiences a problem can easily be turned off by updating its status in this database. Doing so will effectively turn off the site from production. Many other site configurations can quickly be updated through this table as well.

The AutoPilot also provides a pilot implementation that contains no ATLAS specific content and is used by *CHARMM*, a protein structure application that uses OSG VO.

3.4 Pilot

The autopilot wrapper script performs a number of preliminary checks, downloads the site configuration from the SchedConfig DB before the main pilot code is downloaded from a cache and unpacked. When the PanDA pilot [5] is launched it collects information about the worker node and sends it to the job dispatcher. If a matched job does not exist, the pilot will end. If a job does exist, the pilot will fork the job wrapper and start monitoring it.

The job wrapper prepares the run time environment, performs stage-in, executes the payload, stages out the output files for a successful job, and then finishes by removing the input files (only) from the work directory. All information and actions performed by the pilot and the job wrapper are written to a pilot log file. For both successful and failed jobs, the pilot creates a job log (tar ball) of the remaining work directory and transfers it to the local storage element (SE). All output files and logs are available from the PanDA monitor [6], which allows for fast access to production and user jobs which is especially important for debugging problems. Since the job logs will not be available in the case of transfer problems during stage-out, debugging will rely on the pilot error reported to the dispatcher (error code and a relevant extract of the payload and pilot log files) and the batch system log which contains the pilot log as well as all stdout messages that were printed after the job log was created.

A payload that is not producing any output within a certain time, will be killed and failed by the pilot. The time limit can be set in the job definition. The pilot also monitors the size of the payload work directory. A job that is producing too much output is aborted by the pilot, which is better than having the batch system kill the job which in turn is much harder to debug since it leaves no hints on what went wrong.

3.4.1 Data transfers

Since the input data has already been transferred to the site by the DDM system before the pilot and payload start, the pilot only has to copy the files from the local storage to the worker node (and vice versa for the output files). The pilot will use a copy tool that is relevant for the site and is registered in the SchedConfig DB.

Special wrapper classes, known as site movers, have been written for a variety of copy tools (*cp, mv, dccp, rfcp, uberftp, gridftp, lcg-cp, dq2-get/put*, and locally defined copy tools). For dCache, Castor and Xrootd there is support for direct reading, i.e. the payload can read input files directly from the remote SE. Another practical feature is the time-out control for hanging transfers. In case this happens, the pilot will abort the copy after a given time. File transfers are verified by comparing either the adler32 or md5 checksums of the local and remote file.

3.4.2 Job recovery

When there are service problems (contacting the PanDA server for essential job updates) or site problems (with the local file or storage system), valid data from completed jobs can be stranded on the worker nodes or even be deleted by cleanup scripts running at the site. To handle these issues, the pilot has been equipped with a recovery mechanism that attempts to salvage these otherwise lost jobs, without a rerun of the original job being necessary. If a job is prevented from transferring its output to the SE, it leaves the output files, job log and a job state file on the worker node. A later pilot running on the same node checks for such stranded jobs, and retries the data movement if necessary. If it fails again, it will leave the remains for yet another pilot. Pilots will try a maximum of 15 times before failing the job permanently. The PanDA server can also fail the job if the recovery process has not finished within three days. A later pilot trying to recover an already failed job will be notified by the server to give up and remove the remains.

The job recovery mechanism, which is optional, has proved to be highly effective in waiting out service disruptions for later recovery. CPU cycles otherwise lost are thus recovered.

4. Security

PanDA services use the standard GSI grid security model of authentication and authorization based on X509 grid certificates, which is implemented via OpenSSL and secure https. Interactions with the PanDA server require secure https for anything other than passive read actions. The Virtual Organization Membership Service (VOMS) attributes of the proxy are checked to ensure that the user is a member of a virtual organization that is authorized for PanDA use. The authenticated users' distinguished name is part of the metadata of a PanDA job, so the user id is known and tracked throughout PanDA operations. Execution of production jobs and file management relies on production certificates, with the pilot carrying a special 'pilot' VOMS role to control its rights.

Analysis jobs also run under a production proxy unless gLExec is employed in identity switching mode. gLExec is a tool provided by the Enabling Grids for E-sciencE (EGEE) [7] project which runs the payload under the identity of the end-user. Optional gLExec based identity change on the worker node to submitter identity for user jobs is currently under testing. The end-user proxies are pre-staged in the MyProxy credential caching service [8] while the access information needed by the pilot is stored in the PanDA DB. The proper identity under which to run the job can then be extracted by gLExec from the user's proxy.

An optional pilot feature is the proxy hiding, where the proxy is removed from disk during the execution of the payload, and only restored after the payload has finished.

We have also defined a means of securing the payload from tampering in the PanDA DB by means of encryption of the payload using an RSA key pair, with decoding and validation in the pilot prior to execution. Furthermore, a token based system is currently being implemented in the system. The pilot will only be allowed to download a new job from the job dispatcher (and make job updates) if it presents a valid token that was passed on to the pilot from the PanDA server, where it was originally generated, through the batch system.

5. Performance

As of March 2009, some 18+ million jobs have been processed by the PanDA system, at a rate of up to 500k production jobs per week at about 100 sites around the world. During the same period, PanDA has also served almost 900 users, and about 5 million jobs during the last six months. In February 2009 PanDA reached more than 35k concurrent jobs in the system (Fig. 2).

As ATLAS data taking ramps up over the next few years, job counts are estimated to reach the order of 500k jobs per day, with the greatest increase coming from analysis.

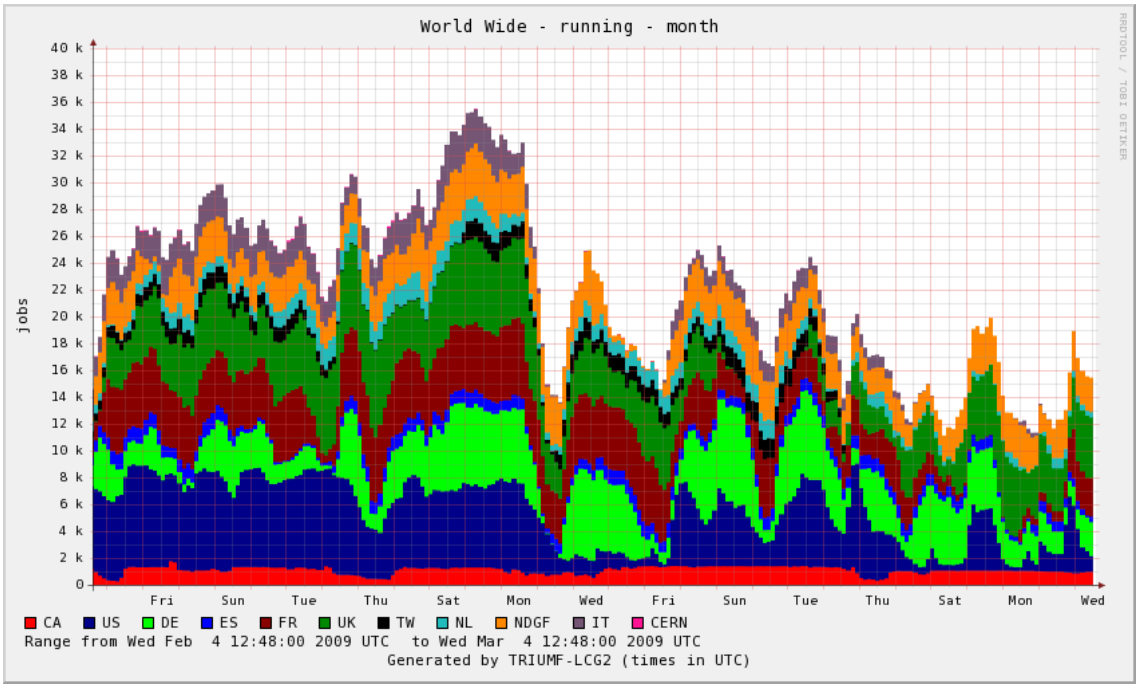

Fig. 2. Concurrent jobs running in the PanDA system, color coded by cloud.

6. Distributed analysis

Production and analysis jobs are running on the same infrastructure in PanDA, which means that they are both using the same software, monitoring and facilities. It also means that there is no need for any duplicated manpower for maintenance. The production and analysis jobs use different computing resources and thus do not have to compete with each other. The data transfer policies are also different. For production jobs, the input files are dispatched to the Tier-2's and the output files are aggregated to the Tier-1's via the DDM asynchronously, while analysis jobs do not trigger any data transfer. Analysis jobs only go to sites that hold the input files

The users can choose between two different tools to submit analysis jobs to PanDA, pAthena [9] and Ganga [10].

7. Conclusion

The PanDA system provides integrated service architecture with late binding of job, maximal automation through layered services, and tight binding with the ATLAS Distributed Data Management system. It has a single task queue and is using pilots to execute the jobs. The pilots are retrieving the payloads from an Apache based central server as soon as the CPUs are available, which leads to low latencies that is especially useful for the user analysis which has a chaotic behavior.

The PanDA system has been in production in the US since early 2006 and ATLAS wide since the end of 2007. The production across ATLAS is going very smoothly and PanDA stands ready to provide a stable and robust service for real data taking in the ATLAS experiment.

References

[1] ATLAS Technical Proposal, CERN/LHCC/94-43 (1994)
[2] T. Maeno. PanDA: Distributed production and distributed analysis system for ATLAS. J.Phys.Conf.Ser.119:062036, 2008.

[3] D. Groep, O. Koeroo, G. Venekamp, gLExec: gluing grid computing to the Unix world, J.Phys.:Conf.Series 119 (2008) 062032
[4] Condor Project, http://www.cs.wisc.edu/condor
[5] P. Nilsson. Experience from a pilot based system for ATLAS. J.Phys.Conf.Ser.119:062038,2008

[6] PanDA monitor, http://panda.cern.ch

[7] Enabling Grids for E-sciencE, http://www.eu-egee.org/

[8] J. Novotny, S. Tuecke, and V. Welch. An Online Credential Repository for the Grid: MyProxy. Proceedings of the Tenth International Symposium on High Performance Distributed Computing (HPDC-10), IEEE Press, August 2001, pages 104-111.

[9] pAthena, https://twiki.cern.ch/twiki/bin/view/Atlas/PandaAthena

[10] 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).