Table of Contents

The PanDA Production and Distributed Analysis System

Introduction

Design and implementation

Principal design features

Architecture and workflow

Implementation

PanDA Components

Security

ATLAS Production System

Information for users

System Operations and Facilities

PanDA Project

PanDA development

PanDA beyond ATLAS

Mailing list

Documents, Talks

PanDA project logo

2013-2014

2010-2012

2007-2009

2004-2006

Meetings

Archive

Pre-2010 meetings, workshops, reviews

Previous generation systems and R&D

Misc presentations

Previous Contributors

2007-2009

2004-2006

Meetings

Archive

Pre-2010 meetings, workshops, reviews

Previous generation systems and R&D

Misc presentations

Previous Contributors
The PanDA Production and Distributed Analysis System

Introduction

The PanDA Production ANd Distributed Analysis system has been developed by ATLAS since summer 2005 to meet ATLAS requirements for a data-driven workload management system for production and distributed analysis processing capable of operating at LHC data processing scale. ATLAS processing and analysis places challenging requirements on throughput, scalability, robustness, efficient resource utilization, minimal operations manpower, and tight integration of data management with processing workflow. PanDA was initially developed for US based ATLAS production and analysis, and assumed that role in late 2005. In October 2007 PanDA was adopted by the ATLAS Collaboration as the sole system for distributed processing production across the Collaboration. It proved itself as a scalable and reliable system capable of handling very large workflow. In 2008 it was adopted by ATLAS for user analysis processing as well.

PanDA throughput has been rising continuously over the years. In 2009, a typical PanDA processing rate was 50k jobs/day and 14k CPU wall-time hours/day for production at 100 sites around the world, and 3-5k jobs/day for analysis. In 2014, PanDA processes about a million jobs per day, with about 150,000 jobs running at any given time. The PanDA analysis user community numbers over 1400.

PanDA was designed to have the flexibility to adapt to emerging computing technologies in processing, storage, networking and distributed computing middleware. This flexibility has also been successfully demonstrated through the past six years of evolving technologies adapted by computing centers in ATLAS which span many continents. This proven scalability and flexibility makes PanDA well suited for adoption by a wide variety of exabyte scale sciences. The interest in PanDA by other big data sciences brought the primary motivation to generalize PanDA, aka the BigPanDA project, providing location transparency of processing and data management, for High Energy Physics community and other data-intensive sciences, and a wider exascale community. The BigPanDA project has been active since September 2012.

Design and implementation

Principal design features

The principal features of PanDA's design are as follows.

- Support for both managed production and individual analysis users so as to benefit from a common WMS infrastructure and to allow analysis to leverage production operations support, thereby minimizing overall operations workload.
- A coherent, homogeneous processing system layered over diverse and heterogeneous processing resources. This helps insulate production operators and analysis users from the complexity of the underlying processing infrastructure. It also maximizes the amount of PanDA systems code that is independent of the underlying middleware and facilities actually used for processing in any given environment.
- Use of pilot jobs for acquisition of processing resources. Workload jobs are assigned to successfully activated and validated pilots based on PanDA-managed brokerage criteria. This 'late binding' of workload jobs to processing slots prevents latencies and failure modes in slot acquisition from impacting the jobs, and maximizes the flexibility of job allocation to resources based on the dynamic status of processing facilities and job priorities. The pilot is also a principal 'insulation layer' for PanDA, encapsulating the complex heterogeneous environments and interfaces of the grids and facilities on which PanDA operates.
- Extensive direct use of Condor (particularly CondorG), as a job submission infrastructure of proven

The PanDA Production and Distributed Analysis System 1
capability and reliability. While other job submission approaches are also supported such as local batch, CondorG is the backbone submission system of PanDA and has been very successful as such.

- Coherent and comprehensible system view afforded to users, and to PanDA's own job brokerage system, through a system-wide job database that records comprehensive static and dynamic information on all jobs in the system. To users and to PanDA itself, the job database appears essentially as a single attribute-rich queue feeding a worldwide processing resource.
- A comprehensive monitoring system supporting production and analysis operations; user analysis interfaces with personalized user level views; detailed drill-down into job, site and data management information for problem diagnostics; usage and quota accounting; and health and performance monitoring of PanDA subsystems and the computing facilities being utilized.
- System-wide site/queue information database recording static and dynamic information used throughout PanDA to configure and control system behavior from the regional level down to the individual queue level. The database is an information cache, gathering data from grid information systems, site experts, the data management system and other sources. It supports information access and limited remote control functions via an http interface. It is used by pilots to configure themselves appropriately for the queue they land on; by PanDA brokerage for decisions based on cloud and site attributes and status; and by the pilot scheduler to configure pilot job submission appropriately for the target queue.
- Integrated data management based on dataset (file collection) based organization of data files, with datasets consisting of input or output files for associated job sets. Movement of datasets as required for processing and archiving is integrated directly into PanDA's workflow. This design matches the data-driven, dataset-based nature of the overall ATLAS computing organization and workflow.
- Automated pre-staging of input data (either to a processing site or out of mass storage, or both) and immediate return of outputs, all asynchronously, minimizing data transport latencies and delivering (for analysis) the earliest possible first results. Pre-staging of input data prior to initiation of jobs is an essential efficiency and robustness measure, such that jobs themselves are not subject to data staging/transfer latencies and failure modes, improving efficiencies for job completion and resource utilization.
- Support for running arbitrary user code (job scripts), as in conventional batch submission. There is no ATLAS specificity or workload restrictions in the design.
- Easy integration of local resources. Minimum site requirements are a grid computing element or local batch queue to receive pilots, outbound http support, and remote data copy support using distributed data movement tools (e.g. xrootd).
- Simple client interface allows easy integration with diverse front ends for job submission to PanDA.
- Authentication and authorization is based on grid certificates, with the job submitter required to hold a grid proxy and VOMS role that is authorized for PanDA usage. User identity (DN) is recorded at the job level and used to track and control usage in PanDA's monitoring, accounting and brokerage systems. The user proxy itself can optionally be recorded in MyProxy for use by pilots processing the user job, when pilot identity switching/logging (via gLExec) is in use.
- Support for usage regulation at user and group levels based on quota allocations, job priorities, usage history, and user-level rights and restrictions.
- Security in PanDA employs standard grid security mechanisms -- see PandaSecurity.

### Architecture and workflow

Jobs are submitted to PanDA via a simple python client interface by which users define job sets, their associated datasets and the input/output files within them. Job specifications are transmitted to the PanDA server via secure http (authenticated via a grid certificate proxy), with submission information returned to the client. This client interface has been used to implement PanDA front ends for ATLAS production, distributed analysis (e.g. pathena), and other experiments. The PanDA server receives work from these front ends into a global job queue, upon which a brokerage module operates to prioritize and assign work on the basis of job type, priority, input data and its locality, available CPU resources and other brokerage criteria. Allocation of job sets to sites is followed by the dispatch of corresponding input datasets to those sites, handled by a data service interacting with the distributed data management system. Data pre-placement is a soft (formerly strict)
precondition for job execution: jobs are not released for processing until their input data is accessible from the processing site. When data dispatch completes, jobs are made available to a job dispatcher. Strict data colocation for analysis is being replaced by a softer requirement that data be accessible via an adequately performant network path, as determined by a networking site-to-site ‘cost matrix’.

An independent subsystem manages the delivery of pilot jobs to worker nodes, using a number of remote (or local) job submission methods as defined in its configuration. A pilot once launched on a worker node contacts the dispatcher and receives an available job appropriate to the site. If no appropriate job is available, the pilot may immediately exit or may pause and ask again later, depending on its configuration (standard behavior is for it to exit). An important attribute of this scheme for interactive analysis, where minimal latency from job submission to launch is important, is that the pilot dispatch mechanism bypasses any latencies in the scheduling system for submitting and launching the pilot itself. The pilot job mechanism isolates workload jobs from grid and batch system failure modes (a workload job is assigned only once the pilot successfully launches on a worker node). The pilot also isolates the PanDA system proper from grid heterogeneities, which are encapsulated in the pilot, so that at the PanDA level the grid(s) used by PanDA appears homogeneous. Pilots generally carry a generic ‘production’ grid proxy, with an additional VOMS attribute ‘pilot’ indicating a pilot job. Analysis pilots may use glexec to switch their identity on the worker node to that of the job submitter (see PandaSecurity).

The isolation of the pilot system in the architecture allows for many different pilot system implementations to be smoothly integrated with PanDA, making it possible to integrate widely diverse processing resources with different requirements for a pilot service. These include supercomputers, cloud resources, and self-contained computing environments like NorduGrid.

The PanDA architecture
Pilot workflow. For details see GenericPanDAPilot

Implementation

The implementation of PanDA (original but still largely accurate) is shown below. Supported front ends are
the ATLAS production system, pathena (a distributed analysis interface to the ATLAS offline software framework, Athena), prun (a more generic front end for ATLAS analysis), and a generic submission client user by other experiments. The PanDA server containing the central components of the system is implemented in python (as are all components of PanDA) and runs under Apache as a web service (in the REST sense; communication is based on HTTP GET/POST with the messaging contained in the URL and optionally a message payload of various formats). Oracle RDBMS is used to implement the job queue and all metadata and monitoring repositories. Condor-G is the distributed scheduler used by the pilot scheduling subsystem, the AutoPyFactory. A monitoring server works with the Oracle DBs, including a logging DB populated by system components recording incidents via a simple web service behind the standard python logging module, to provide web browser based monitoring and browsing of the system and its jobs and data. Since 2013, PanDA has also been (re)implemented against a MySQL back end as an alternative to Oracle.

Original Implementation

PanDA Components

- PandaServer - central PanDA hub composed of several components that make up the core of PanDA. Implemented as a stateless REST web service over Apache mod_python and with a MySQL back end
- PandaTaskBuffer - the PanDA job queue manager, keeps track of all active jobs in the system
- PandaBrokerage - matches job attributes with site and pilot attributes. Manages the dispatch of input data to processing sites, and implements PanDA’s data pre-placement requirement
- PandaJobDispatcher - receives requests for jobs from pilots and dispatches job payloads. Jobs are assigned which match the capabilities of the site and worker node (data availability, disk space, memory etc.) Manages heartbeat and other status information coming from pilots.

PanDA Components
**PandaDataService** - data management services required by the PanDA server for dataset management, data dispatch to and retrieval from sites, etc. Implemented with the ATLAS DDM system.

- **PandaDB** - Database system serving PanDA
- **PandaClient** - PanDA job submission and interaction client
- **AutoPyFactory** - Pilot submission, management and monitoring system. Supersedes first generation PandaJobScheduler, as well we the second generation system, the AutoPilot.
- **Harvester** - Harvester is a resource-facing service between the PanDA server and collection of pilots for resource provisioning and workload shaping. It is a lightweight service running on a VObox or an edge node of HPC centers to provide a uniform view for various resources. Central Harvester instances.
- **PandaPilot** - the execution environment (effectively a wrapper) for PanDA jobs. Pilots request and receive job payloads from the dispatcher, perform setup and cleanup work surrounding the job, and run the jobs themselves, regularly reporting status to PanDA during execution. Pilot development history is maintained in the pilot blog.
- **Panda Schedconfig** - Database table to configure resources and guide the submission of pilot jobs
- **PandaMonitor** - web based monitoring and browsing system that provides an interface to PanDA for users and operators.
- **PandaLogger** - logging system allowing PanDA components to log incidents in a database via the Python logging module and http
- **Bamboo** - interface between PanDA and the ATLAS production database. Superseded in the second generation ATLAS production system Prodsys2 by the JEDI extensions of PanDA.
- **PanDA extensions** - Extending PanDA for general use (Open Science Grid WMS program)
- **PD2P** - PanDA dynamic data placement model
- **JEDI** - An extension of core PanDA to add capability for task-level workload management
- **Event Service** - Event-level master/slave event processing in PanDA/JEDI
- **BigPanDAmonitoring**
- **BigPanDAinstanceInstallation**
- **BigPanDARPMbuildSystem**
- **BigPanDAforORNL**
- **Intelligent Networking**
- **PanDA share distribution and monitoring**
- **PanDA error table and retrial module**

---

**Security**

Security in PanDA employs security mechanisms that are standard in web and grid circles, augmented by built-in capabilities to monitor, track and control usage. See PandaSecurity for information. For operational PanDA security advice, see here.

**ATLAS Production System**

- Development of the ATLAS production system, now in its second generation Prodsys2, is closely allied with PanDA and utilizes the JEDI extensions of PanDA.

**Information for users**

- Distributed Analysis on PanDA
  - Client tools for PanDA analysis jobs
  - pathena: How to submit athena analysis jobs to PanDA
  - prun: How to submit ROOT/general jobs to PanDA
  - psequencer: How to perform sequential jobs/operations in PanDA based analysis
  - pbook: Bookkeeping for PanDA analysis jobs
puserinfo: Access Control on PanDA analysis jobs
• PandaClient interface to submit jobs to PanDA (for programmers)
• Panda job states (defined/assigned/waiting etc.)

System Operations and Facilities

• PanDA info and help page, problem reporting
• PanDA monitor
• CERN PanDA facilities
  ♦ Distributed computing machines at CERN
• BNL PanDA facilities
• PanDA at Tier3s
• Error codes
• Shift Guide
• How to send test jobs
• How to enable PanDA Production on a site without DQ2 site services

PanDA Project

• Project leads: KaushikDe, TorreWenaus
• Lead developer: TadashiMaeno

• Current Development Team:
  ♦ AlexeiKlimentov: general management, previously: DDM operations monitoring
  ♦ TorreWenaus: general management, PandaMonitor, previously: PandaLogger, AutoPilot, SchedConfig
  ♦ TadashiMaeno
    ◊ PanDA server (PandaTaskBuffer, PandaJobDispatcher, PandaDataService, PandaBrokerage)
    ◊ Bamboo production system interface
  ♦ PaulNilsson: PandaPilot
  ♦ MikhailBorodin, KaushikDe, DmitryGolubkov, AlexeiKlimentov
    ◊ Production System
  ♦ MaximPotekhin
    ◊ Previously: Pandjango (Monitor), PandaPilot, HBase, Mnemosyne
  ♦ Main.Siarhei.Padolski: PandaMonitor
  ♦ AldenStradling: SchedConfig DB, previously: AutoPilot

PanDA development

Source code repository: github.com/PanDAWMS

PanDA beyond ATLAS

• PanDA packaging: The Packaging Corner
• Big PanDA Project
  http://news.pandawms.org/lsst.html
• AMS: Taiwan PanDA and Rucio Project
• ORNL: PanDA on Titan, BigPanDAforORNL
• ALICE
• NRC-KI: BigPanDAforNRCKI
• ...
Mailing list

ATLAS PanDA development discussions take place on the CERN e-group atlas-adc-panda mailing list. A CERN e-groups forum is available for discussions on usage of PanDA for analysis.

Documents, Talks

PanDA project logo

- PanDA logo, 1800x1731 pixels:

- PanDA logo, 600x577 pixels:

- PanDA logo, 260x250 pixels:

- PanDA logo, 200x192 pixels:

2013-2014

- PanDA’s Role in ATLAS Computing Model Evolution, T.Maeno et al, ISGC 2014, Taipei, TW, Mar 2014

2010-2012

- Networking and Workload Management (PDF), Kaushik's talk in Dec'12 at the LHCONE Meeting
- Recent Improvements in the ATLAS PanDA Pilot (poster, paper), P. Nilsson, CHEP 2012, United States, May 2012
- Evolution of the ATLAS PanDA Production and Distributed Analysis System (slides, paper), T. Maeno, CHEP 2012, United States, May 2012
- The ATLAS PanDA Pilot in Operation (poster, paper), P. Nilsson, CHEP 2010, Taiwan, October 2010
- Overview of ATLAS PanDA Workload Management (slides, paper), T. Maeno, CHEP 2010, Taiwan, October 2010

2007-2009

- Distributed Data Analysis in ATLAS (slides, letter), P. Nilsson, ICCMSE 2009, Greece, September 2009
PanDA < PanDA < TWiki

- The PanDA System in the ATLAS Experiment (slides, paper), P. Nilsson, ACAT'08, Italy, November 2008
- PanDA talk, T. Wenaus, ISGC conference, Taiwan, April 2008
- PANDA : Distributed Production and Distributed Analysis System for ATLAS (slides,paper), T. Maeno, CHEP 07, September 2007
- Experience from a pilot based system for ATLAS (slides, paper), P. Nilsson, CHEP 07, September 2007

2004-2006

- The Production and Distributed Analysis system PANDA for OSG, T. Maeno, ATLAS software week, September 2006
- Update on distributed analysis with PanDA - pathena, T. Maeno, ATLAS software week, September 2006
- PanDA overview, T. Wenaus, OSG workflow management extensions meeting, September 2006
- Preliminary plans for PanDA extension in OSG, T. Wenaus et al, March 2006
- Panda:Production and Distributed Analysis System for the ATLAS Experiment, K. De et al, CHEP 06, February 2006
- panda-schematic-dec.pdf: PanDA schematic - Plan for December 2005
- panda-schematic-oct.pdf: PanDA schematic - Plan for October 2005
- panda-schematic-sep.pdf: PanDA schematic - Plan for September 2005
- prod-aug05-uta.pdf: worklist from UTA meetings 8/11-12
- panda-schematic-dec.pdf: PanDA schematic - Plan for December 2005

Meetings

PanDA project meetings currently take the form of BigPanDA project meetings. ATLAS PanDA issues are primarily discussed in ATLAS Distributed Computing development meetings.

Archive

Pre-2010 meetings, workshops, reviews

- Sep 2006 US ATLAS DDM operations workshop
- Mar 2006 US ATLAS review of PanDA
- Aug 10 2005 US prodsys development meeting
- Aug 8-9 2005 US ATLAS distributed analysis workshop
- Project planning and task list

Previous generation systems and R&D

- AutoPilot
- Components of Windmill Software
- Supplementary pages, including PanDA Extensions for Open Science Grid
- HBase noSQL solution for Panda Monitoring - testbed
- Mnemosyne - Cassandra (noSQL) based data storage for PanDA Monitoring
- Pandjango - Django-based Monitoring Application

Misc presentations

- DButils_upgrade_by_connection_pooling.ppt: DButils upgrade using Connection Pool
**Previous Contributors**

- KarthikArunachalam: First generation production system interface
- XinZhao: First generation pilot, job scheduler
- MarcoMambelli: First generation job scheduler
- WenshengDeng: DDM testing and integration, PanDA DQ2 data mover
- HorstSeverini: CVS repository, communications
- NurcanOzturk: QA testing and integration, pathena user support
- MarkSosebee: QA testing and integration, production operations
- JoseCaballero: PandaPilot, pilot security

**Major updates:**

-- TorreWenaus - 22-May-2014
-- MaximPotekhin - 13-Dec-2012
-- MaximPotekhin - 24-Oct-2012
-- MaximPotekhin - 14 Sep 2012
-- MaximPotekhin - 11 Oct 2010
-- AldenStradling - 31 Jul 2009
-- TorreWenaus - 25 Jun 2008
-- TorreWenaus - 05 Nov 2007
-- TorreWenaus - 06 Oct 2006
-- TorreWenaus - 06 Mar 2006
-- RobertGardner - 10 Aug 2005
-- KaushikDe - 01 Aug 2005

**Responsible:** TorreWenaus

This topic: PanDA > PanDA
Topic revision: r175 - 2018-05-09 - FernandoHaraldBarreiroMegino

Copyright &© 2008-2019 by the contributing authors. All material on this collaboration platform is the property of the contributing authors. Ideas, requests, problems regarding TWiki? Send feedback