The next generation of ATLAS PanDA Monitoring

Jaroslava Schovancová
Brookhaven National Laboratory, Physics Department, Bldg. 510A, Upton, NY 11973, USA
E-mail: `jschovan@bnl.gov`

Kaushik De
University of Texas in Arlington, Department of Physics, Arlington TX, United States of America

Alexei Klimentov
Brookhaven National Laboratory, Physics Department, Bldg. 510A, Upton, NY 11973, USA

Peter Love
Lancaster University, Physics Department, Lancaster University, Lancaster, UK

Maxim Potekhin
Brookhaven National Laboratory, Physics Department, Bldg. 510A, Upton, NY 11973, USA

Torre Wenaus
Brookhaven National Laboratory, Physics Department, Bldg. 510A, Upton, NY 11973, USA

on behalf of the ATLAS Collaboration

For many years the PanDA Workload Management System has been the basis for distributed production and analysis for the ATLAS experiment at the LHC. Since the start of data taking PanDA usage has ramped up steadily, with up to 1M completed jobs/day in 2013. The associated monitoring data volume has been rising as well, to levels that present a new set of challenges in the areas of database scalability and monitoring system performance and efficiency. Outside of ATLAS, the PanDA system is also being used in projects like AMS, LSST and a few others. It is currently undergoing a significant redesign, both of the core server components responsible for workload management, brokerage and data access, and of the monitoring part, which is critically important for efficient execution of the workflow in a way that's transparent to the user and also provides an effective set of tools for operational support. The next generation of the PanDA Monitoring System is designed based on a proven, scalable, industry-standard Web Framework – Django. This allows us to achieve significant versatility and possibilities of customization, which is important to cover the needs of the growing community of PanDA users in a variety of science and technology areas. We describe the design principles of the core Web application, the UI layout of the presentation layer, and the challenges that must be met in order to continue the necessary support of the ATLAS experiment while expanding the scope of applications handled by PanDA.

The International Symposium on Grids and Clouds (ISGC) 2014
March 23-28, 2014
Academia Sinica, Taipei, Taiwan

1. Introduction

The PanDA (Production and Distributed Analysis) Workload Management System [1] plays a key role in the infrastructure of the ATLAS [2] distributed computing [3]. ATLAS PanDA processes Monte Carlo simulations and data reprocessing jobs, as well as physics analysis carried out by physics groups or individual physicists. As of March 2014 ATLAS PanDA manages up to 1.5 M jobs per day with 1400 distinct users submitting their analysis jobs through PanDA. PanDA has performed very well during ATLAS data taking period in the LHC Run I [4]. During the ongoing LHC Long Shutdown I period various parts of PanDA are being enhanced in order to ensure smooth operation during future data taking periods, and to address needs of users in a variety of science fields of the growing community of the PanDA ecosystem. Design enhancements of the PanDA Monitoring are described in this paper.

2. The current ATLAS PanDA monitoring

The current ATLAS PanDA monitoring has evolved over many years. It provides a real-time and short-term-history monitoring tool for PanDA system.

The primary goals of PanDA monitoring are the rapid identification of failures, and monitoring of progress of a distributed physics analysis from the submission of the set of jobs to finalization of the latest running job. Ease in spotting failures allows users to focus on the most serious failures first, collect as much information as possible, and raise the issue with an expert or a responsible person, so that the issue can be resolved.

The PanDA monitor provides very useful summary views with emphasis to roles: a physicist, who runs distributed analysis jobs on the ATLAS distributed computing resources; production manager, who manages large-scale Monte Carlo simulation or data processing campaign on behalf of a physics group or the whole experiment; site administrator, who is monitoring performance of the experiment running jobs on their site; distributed computing operations personnel, who is monitoring health of the overall ATLAS distributed computing resources and chasing failures in a timely manner.

Members of each of the role groups can drill down from the distilled summary information through information about sets of jobs, to very detailed description of a single PanDA job, access log files of that job, or navigate to other ATLAS monitoring tools to follow up various other aspects of distributed computing and monitor all other parts of the distributed computing infrastructure.

3. The next generation of ATLAS PanDA monitoring

The next generation of PanDA monitoring is developed as a part of the project "Next Generation Workload Management and Analysis System for BigData". The project gives us the opportunity to factorize and generalize PanDA monitoring. The aim of the effort is to design generic components and APIs of a monitoring service, which can be customized to address needs of the various experiments of the PanDA ecosystem. The generic components and APIs of the next generation of PanDA monitoring, the BigPanDA monitoring (BigPanDAmon), are described in the following sub-sections.

3.1 BigPanDA monitoring at a glance

BigPanDAmon is a modular monitoring package that clearly separates data access layer and visualization of the data. The monitoring is built around common key objects of the PanDA system, such as PanDA job or PanDA resource. For each of the objects a set of views is available. The view visualizes data in form of tables, plots, and lists. Data access layer provides pre-filtered information describing a set of objects, tailored to the corresponding view. Data access layer benefits from a group of REST API resources.

3.1.1 BigPanDA monitoring backend

BigPanDAmon is a web application built in the Django Web Framework [5]. On the backend it uses Django Web Framework, and several Django plugin libraries, such as Django REST framework [6], and Django DataTables view [7].

The monitoring application data is stored in a relational database. The two main database backends supported in BigPanDAmon core are MySQL and Oracle. As Django Web Framework supports even more DB backends (SQLite, PostgreSQL in addition to MySQL and Oracle), the choice of DB backend rests with the community of the experiment/project, which can fully leverage available resources in a flexible way.

Basic object selection is done with Django Querysets. In special case of the model of PanDA job the Queryset has been replaced by its generalization, Querychain. DB queries are optimized using the raw SQL code.

3.1.2 BigPanDA monitoring modules

BigPanDAmon instance for any of the experiments/projects is composed of two parts: the core part with functionality around the common PanDA objects, and the experiment/project-specific part which extends the common core. Schema of the BigPanDAmon site is depicted in Fig. 1.

The common core consists of several Modules which encapsulate functionality to present objects; of UI elements which directly represent either elements of the monitoring user interfaces, or provide support for the visual part of the monitoring; and of the Common configuration part not only with the Django application settings, but with templates, static files, and common user-uploaded content.

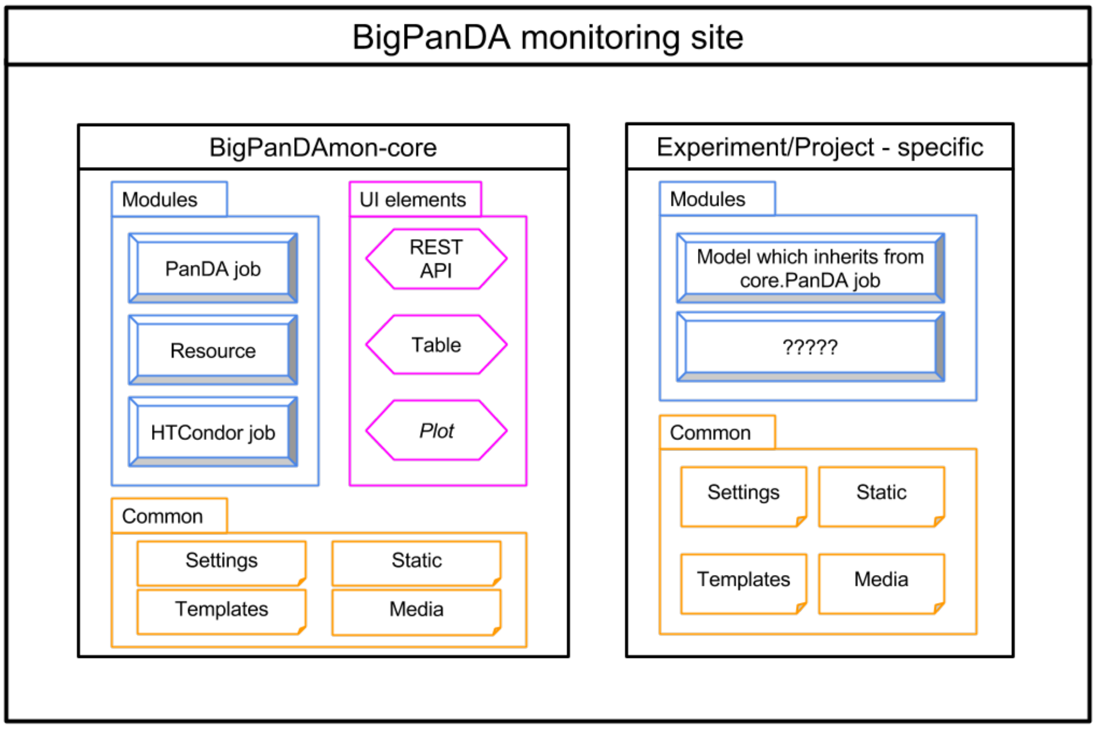

Figure 1. Module structure of BigPanDA monitoring package for an experiment/project.

3.1.3 BigPanDA monitoring customization

The BigPanDAmon frontend consists of a set of views which engage javascript library jQuery [8] and its plugins, such as DataTables [9] and Highcharts [10].

The core feature of design of monitoring package with clear separation of data layer and visualization lies in the flexibility and ease of customization of the user interfaces of the monitoring. Each experiment/project can extend the provided BigPanDAmon frontend, resulting in a different look-and-feel for each of the BigPanDAmon sites.

Modularity of the BigPanDAmon core allows for experiment/project-specific extension of any of the parts of the core: from feature enhancement of the views around a particular Model, through introduction of a completely new Model central to the PanDA instance of that particular experiment, to modifications or re-organizations of the monitoring views.

The UI elements, such as tables, plots, or APIs can be generalized to visualize information about a completely new Model. The BigPanDA monitoring package can serve as a base framework for both PanDA and external-to-PanDA monitoring, as discussed later in case of the HTCondor monitoring.

3.1.4 BigPanDA monitoring configuration

BigPanDAmon application configuration extends the usual Django application configuration. It separates quasi-static general project configuration, configuration of machine-specific environment with sensible default values, and settings to override machine-specific environment. The core configuration can be extended or overridden in the experiment/project-specific configuration.

3.2 REST APIs

BigPanDAmon REST API follows API design recommendation and best practices of Apigee.com [11]. The frontend views are supplied with data from the RESTful APIs. In combination with X509 authentication framework common in the Grid computing world we can benefit from a REST API to not only list available resources, but to create new ones, modify them, or delete them as well.

3.2.1 HTCondor monitoring

HTCondor [12] monitoring is an excellent example of the PanDA monitoring generalization in the BigPanDAmon package. The evolving PanDA-HTCondor REST API supplies data to the HTCondor monitoring interface, which is one of the BigPanDAmon family members.

There are several views of the HTCondor monitoring. Each of the views exposes instances of the HTCondor job object. The HTCondor REST API supplies data with information about the HTCondor jobs to those views. The views are equipped with a filter that enables a user to limit selection of HTCondor jobs desired to monitor. The filter selection reflects in the URL of the view, endorsing multi-user collaborative environment which leverages the BigPanDAmon package. The HTCondor API resources are described in Fig. 2.

Resource	**`/v2/api-auth/htcondor/jobs/`**	
HTTP verb	Purpose	Description
`POST`	create	Bulk create new HTCondor job.
`GET`	read	Bulk list HTCondor jobs.
`PUT`	update	Bulk update HTCondor jobs.
`DELETE`	delete	Bulk delete HTCondor jobs.

Resource	**`/v2/api-auth/htcondor/jobs/<ID>`**	
HTTP verb	Purpose	Description
`POST`	create	*Error*
`GET`	read	Show HTCondor job <ID>.
`PUT`	update	If exists, update HTCondor job <ID>. If does not exist, *error*.
`DELETE`	delete	Delete HTCondor job <ID>.

Figure 2. REST API resources of the HTCondor monitoring with BigPanDAmon package.

4. Summary

The PanDA Workload management system successfully manages up to 1.5 M grid or cloud jobs per day. PanDA monitoring is an essential component of the PanDA system. The BigPanDA monitoring package constitutes a great potential and opportunity to generalize and evolve PanDA monitoring, and to leverage PanDA monitoring for external applications.

Acknowledgements

This manuscript has been authored by employees of Brookhaven Science Associates, LLC under Contract No. DE-AC02-98CH10886 with the U.S. Department of Energy. The publisher by accepting the manuscript for publication acknowledges that the United States Government retains a non-exclusive, paid-up, irrevocable, world-wide license to publish or reproduce the published form of this manuscript, or allow others to do so, for United States Government purposes.

Authors of this paper gratefully acknowledge support from the US DOE ASCR and US DOE HEP Computing.

References

[1] T. Maeno for the ATLAS Collaboration, *PanDA: Distributed production and distributed analysis system for ATLAS,* 2008 *J. Phys.: Conf. Series* **119** 062036

[2] ATLAS Collaboration, *The ATLAS Experiment at the CERN Large Hadron Collider,* 2008 *JINST* **3** S08003

[3] S. Campana for the ATLAS Collaboration, *Evolution of the ATLAS Distributed Computing during the LHC long shutdown, to appear in proceedings of 20th International Conference on Computing in High Energy and Nuclear Physics 2013*

[4] T. Maeno for the ATLAS Collaboration, *Evolution of the ATLAS PanDA Workload Management System for Exascale Computational Science,* to appear in proceedings of *20th International Conference on Computing in High Energy and Nuclear Physics 2013*

[5] Django – high-level Python Web framework that encourages rapid development and clean, pragmatic design. https://www.djangoproject.com

[6] Django REST framework – powerful and flexible toolkit that makes it easy to build Web APIs. http://www.django-rest-framework.org/

[7] Django DataTables View – a base view for handling server side processing for the awesome DataTables. https://pypi.python.org/pypi/django-datatables-view

[8] jQuery: The Write Less, Do More, JavaScript Library. http://jquery.com/

[9] DataTables – highly flexible tool adding advanced interaction controls to any HTML table. http://datatables.net/

[10] Highcharts JS – Interactive JavaScript charts library. http://www.highcharts.com/

[11] Apigee API best practices http://apigee.com/about/api-best-practices/api-design/ebook

[12] D. Thain, T. Tannenbaum, M. Livny, *Distributed Computing in Practice: The Condor Experience* in *Concurrency and Computation: Practice and Experience*, Vol. 17, No. 2-4, pages 323-356, February-April, 2005