THE ATLAS BIGPANDA MONITORING SYSTEM ARCHITECTURE

T. Korchuganova [1, a], S. Padolski [2], T. Wenaus [2], A. Klimentov [2], A. Alekseev [1] on behalf of ATLAS Collaboration

[1] *Tomsk Polytechnic University*

[2] *Brookhaven National Laboratory*

E-mail: [a] tatiana.korchuganova@cern.ch

Currently-running large-scale scientific projects involve unprecedented amounts of data and computing power. For example, the ATLAS experiment at the Large Hadron Collider (LHC) has collected 140 PB of data over the course of Run 1 and this value kept increasing at the rate of ~800MB/s during Run 2. Processing and analysis of such amounts of data requires development of complex operational workflow and payload management systems along with building top edge computing facilities. In the ATLAS experiment a key element of the payload management is the Production and Distributed Analysis system (PanDA). It consists of several components and one of them is the BigPanDA monitoring component. It is responsible for providing a comprehensive and coherent view of the tasks and jobs executed by the system, from high level summaries to detailed drill-down job diagnostics. The BigPanDA monitoring has been in production since the mid-2014 and it continuously evolves to satisfy increasing demands in functionality and growing payload scales. Today it effectively keeps track of more than 2 million jobs per day distributed over 170 computing centers worldwide in the largest instance of the BigPanDA monitoring: the ATLAS experiment. In this paper we describe the monitoring architecture and its principal features.

Keywords: monitoring, PanDA, data aggregation, Django application

1. Introduction

The BigPanDA monitoring system is one of the PanDA workload management system [1] components. It provides many services, including the system state overview characterized by its object parameters, tracking operational progress and serving as a source of detailed data for troubleshooting. Input information for this representation analysis are scattered among real-time data and historical archives. There are various levels of detalization and groupings required to satisfy the needs of four types of groups. They are physicists who do their own analysis; managers who operate simulation and data processing campaigns on behalf of a physics group or the whole experiment; shifters who monitor the health of the overall distributed computing resources, chasing failures in a timely manner; and developers using the monitor as a window into the PanDA system. Due to the continuously growing computational and functional needs of the system, it should be developed as a scalable system with extensible functionalities. This paper describes requirements to the system as well as its architecture and structure.

2. Input

Originally the monitoring system was developed for the ATLAS experiment [2] at LHC. The system described here is its latest generation, developed to address the continuously increasing demands [3]. For this reason the system is capable of aggregating massive volumes of data in a close to the real time mode. One of the most important and the most demanding task of the monitoring system is to aggregate and expose information on every job being handled by PanDA. Properties of a job entity are assembled from a number of PanDA database tables including a jobs table which stores primary properties and tables with related objects, such as events, files, datasets, tasks, and computing sites where jobs are being processed.

An event refers to a distinct particle collision event recorded by the detector or simulated by a Monte Carlo software. A job is a payload which is supposed to process number of input events or produce them using initial random generator seeds conditions. A task is a collection of jobs united by the same data sample split along them. A request can contain a set or a chain of tasks as well as a single task. A campaign is a set of requests which are united by a physics objective. This hierarchy of objects has the following statistics on average: a job contains hundreds of events, a task contains hundreds of thousands of events, a request contains more than a billion of events and finally a campaign has a trillion of events. Similarly, the average time to process an event varies from seconds to several minutes, a job lasts from few hours to a couple of days, a task can take up to several weeks, a request is processed for more than a month and a campaign generally lasts for a year. These estimates show that the architecture of the monitoring system should be adequate to analyse and represent an extremely wide scale of data.

3. Architecture

The BigPanDA monitor is built as a web application. The architecture of the system is shown in Figure 1. The application backend is based on the Django model-view-controller framework [4] which is a powerful open source package written in Python. The monitor is hosted by Apache [5] web servers through the Web Server Gateway Interface (WSGI) [6]. The system supports various relational database (DB) backends using abstraction layers provided by Django. However, relational DBs are not the only data source for the monitoring system. It also acquires data from non-relational sources like ElasticSearch (ES) [7] and Redis cache instances [8]. In addition, ATLAS's Rucio data management system [9] provides logs, and the Dashboard service provides historical histograms [10].

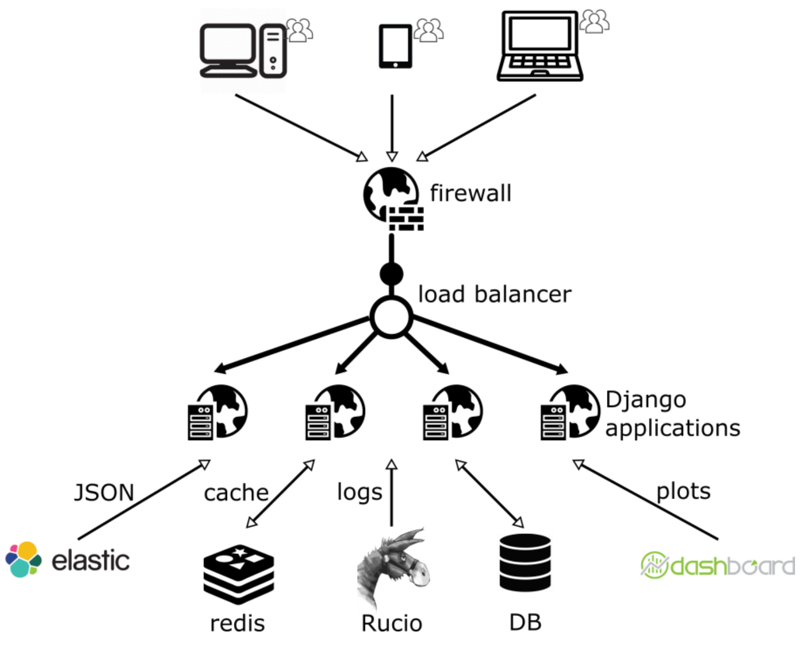

Figure 1. Architecture scheme of the BigPanDA monitor instance

Data-flow for the system is shown in Figure 2. The raw data from PanDA system is stored in the DB. Taking into account the fact that user requests may involve millions of DB rows to process, data aggregation algorithms are split between DB and Web server backends, which allows to reap benefits from both engines, reduce data transfers and increase the performance. In addition, the monitoring system has an advanced caching system. A data prepared for displaying is divided into a common and a user specific parts. In the common part data is proactively cached, whereas user specific data is processed for each user request individually. Pre-aggregated data from DB or cache storage is loaded through the standard Django interface as well as indexed data from ES. The user specific data can contain either settings for a page or a list of references to the most relevant pages determined by analysis of BigPanDA browsing history. The user specific data is protected by several policies, in particular, SSO authentication and HTTPS protocol.

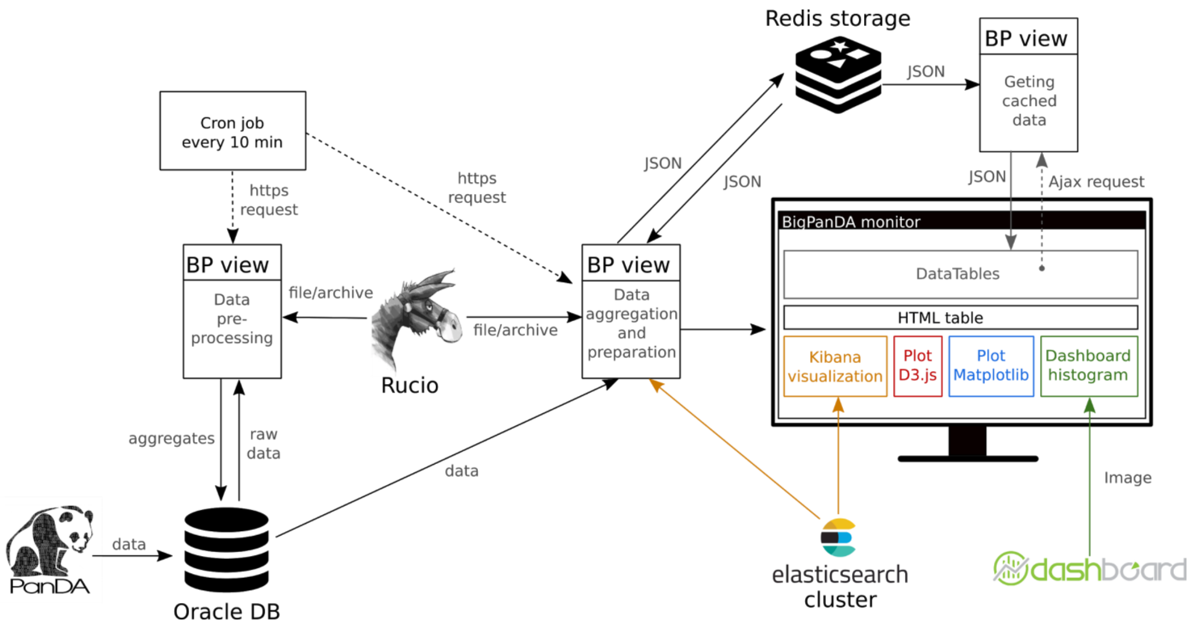

Figure 2. Data-flow diagram of the BigPanDA monitor

Data is displayed using Foundation CSS [11] containers or dynamic sortable DataTables [12]. Data for DataTables is delivered asynchronously using jQuery [12]. The containers can include static tables, responsive menus, visualisations generated either on the client side using D3.js or on the server side using the matplotlib library [14-16]. Besides self-generated visualizations, Kibana dashboards and histograms provided by Dashboard service can be embedded into pages directly. The monitoring system also provides aggregated data in JSON format to allow the system to serve as a programmatic source of information. We are also planning to integrate monitoring and Data Knowledge Base of the experiment in the future [17]

The principal views of the monitoring system displays key objects of data processing and analysis, such as jobs, tasks, files, and its aggregates. These views are unified into a single module defining the system's core (see Figure 3). More specific and accounting views are implemented as specific plugins. This approach enables system customization by plugging in and out existing components or implementing new ones. For example, the basic BigPanDA monitor core is installed on Amazon Elastic Compute Cloud (EC2) [18] for serving different experiments which use PanDA for workload management. For the COMPASS experiment at SPS [19] the extra module was developed and deployed in addition to the core ones. The largest instance for the ATLAS experiment includes core and 9 more specific modules, such as the ATLAS Release Tester (ART) monitor showing tests results of nightly software builds, and Reports providing a wide overview of a campaign computation.

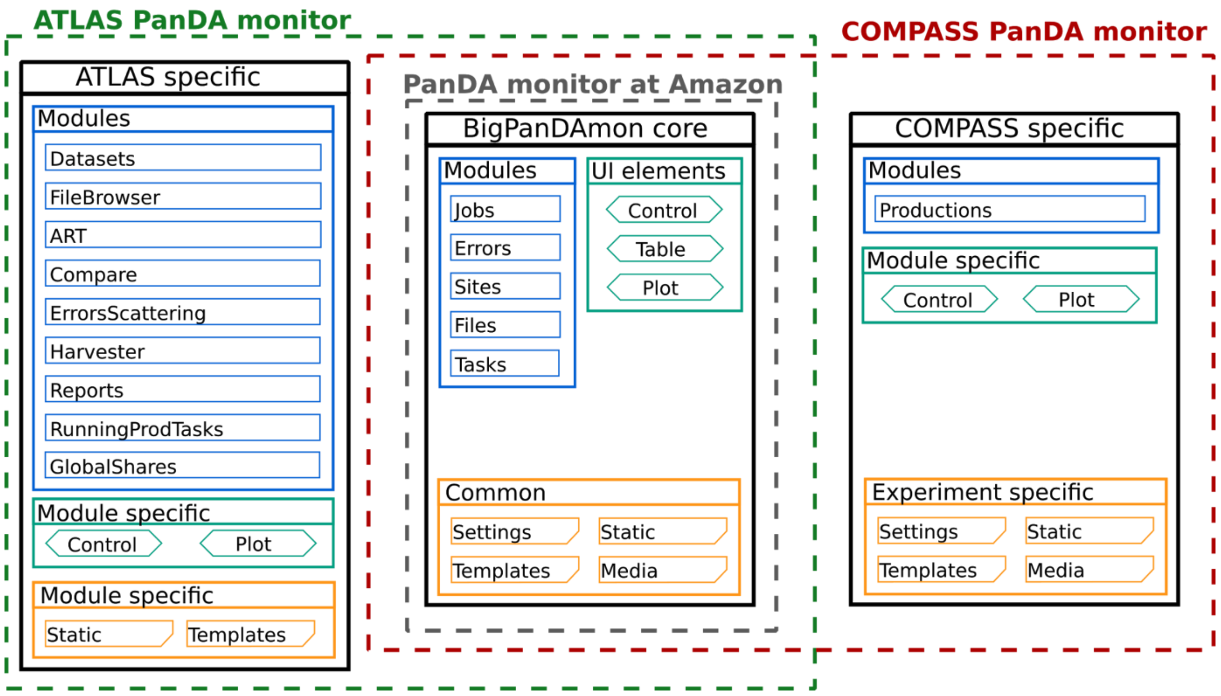

Figure 3. Architecture of the BigPanDA monitor project

4. Summary

The BigPanDA monitoring system is in production since the middle of 2014 and thanks to the flexible architecture the functionality of the system is continuously evolving without interrupting the service. In September 2018 the BigPanDA monitoring system in ATLAS handled more than 35 thousand requests in a day, where 77% of them are key views, in particular, jobs, tasks, sites, and files. The successful ATLAS experience made this product also of interest to other experiments, in particular at the moment of writing 3 instances of BigPanDA monitor serve payload monitoring for ATLAS, COMPASS and other experiments beyond High Energy Physics.

Acknowledgements

TPU team work is supported by the Russian Science Foundation grant under contract №16-11-10280.

References

[1] Maeno T. et al., 2017, PanDA for ATLAS distributed computing in the next decade, J. Phys. Conf. Ser. 898 052002

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

[3] Schovancova J. et al, 2014, The new Generation of the ATLAS PanDA Monitoring System, 035.10.22323/1.210.0035.

[4] Django Documentation. Available at: https://docs.djangoproject.com/en/1.11/ (accessed on 5.07.2018)

[5] Apache HTTP Server Version 2.4 Documentation. Available at: https://httpd.apache.org/docs/2.4/ (accessed on 10.08.2018)

[6] WSGI. Available at: https://wsgi.readthedocs.io/en/latest/ (accessed on 9.08.2018)

[7] Elastic Stack and Product Documentation. Available at: https://www.elastic.co/guide/index.html (accessed on 13.07.2018)

[8] Nelson J., 2016, Mastering Redis, Birmingham:Packt Publishing, 340 p.

[9] Lassnig M. et al., 2015, Monitoring and Controlling ATLAS data management: The Rucio web user interface, J. Phys.Conf. Ser. 664

[10] Andreeva J., Campana S., Karavakis E. et al., 2012, ATLAS job monitoring in the Dashboard Framework, J. Phys.Conf. Ser. 396

[11] ZURB Foundation. Available at: http://foundation.zurb.com/sites/docs/ (accessed on 3.07.2018)

[12] DataTables. Available at: http://datatables.net/ (accessed on 25.07.2018

[13] jQuery - Asynchronous JavaScript Library. Available at: http://jquery.com/ (accessed on 24.05.2018)

[14] Data-driven documents Available at: https://d3js.org/ (accessed on 17.06.2018)

[15] Matplotlib Overview Available at: https://matplotlib.org/contents.html (accessed on 17.06.2018)

[16] Padolski S., Korchuganova T., Wenaus T., Grigorieva M., Alexeev A., Titov M., Klimentov A., 2018, Data visualization and representation in ATLAS BigPanDA monitoring, Scientific Visualization №10, p. 69-76.

[17] Grigorieva M., Aulov V., Gubin M., Klimentov A., 2016, Data knowledge base for scientific experiment, Open Systems, DBMS, Vol. 4, p.42-44

[18] Amazon EC2. Available at: https://aws.amazon.com/ec2/ (accessed on 17.09.2018)

[19] Abbon P. et al., 2007, The COMPASS experiment at CERN, Nucl. Instrum. Meth., Vol. A577, p.455-518