THE BIGPANDA SELF-MONITORING ALARM SYSTEM FOR ATLAS

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

[1] *Tomsk Polytechnic University*

[2] *Brookhaven National Laboratory*

E-mail: [a] aleksandr.alekseev@cern.ch

The BigPanDA monitoring system is a Web application created to deliver the real-time analytics, covering many aspects of the ATLAS experiment's distributed computing. The system serves about 35000 requests daily and provides critical information used as input for various decisions: from distribution of the payload among available resources to issue tracking related to any of 350 000 jobs running simultaneously. It evolves intensively; in particular, in 2017, the system received 933 commits, delivering new features and expanding the scope of the presented data. The experience of operating BigPanDA in 24/7 mode led to development of a multilevel self-monitoring alarm system. This ELK-stack based solution covers all critical components of the BigPanDA: from user authentication to management of the number of connections to its database backend. The developed solution provides an intelligent error analysis, delivering to the operators only those notifications that need human intervention. We describe the architecture, principal features, and operation experience of self-monitoring, as well as its adaptation possibilities.

Keywords: BigPanDA monitoring system, self-monitoring alarm system, ELK-stack

1. Introduction

The BigPanDA monitoring is a multicomponent Web application developed for the ATLAS experiment [1] at the Large Hadron Collider. It provides a comprehensive and coherent view of the jobs executed by the PanDA workload management system [2], from high level summaries to detailed drill-down job diagnostics [3]. The BigPanDA monitoring processes over 40000 requests daily, including about 25000 API calls from external consumers, such as Hammercloud service [4] or AES monitoring [5]. The core of the application is deployed on 9 backend server nodes behind a load-balancer and operates in the 24/7 mode. There are 5 principal components that define the system architecture: Web-server (Apache), load-balancer (Nginx), DB backend (Oracle), distributed cache storage (Redis) and external authentication providers (CERN, Google, GitHub). A failure of any of them will lead to the whole BigPanDA unavailability and, consequently, loss of one of the most important sources of information about massive calculations performed by ATLAS on GRID and another computing resources [6]. Therefore it was crucial to create an advanced self-monitoring alarm system for the BigPanDA monitoring, which would analyze its state in the real time mode and immediately notify developers when an error happens. The self-alarm system design should reflect the following:

- Execution time of complex data aggregation algorithms must be an observable metric.
- The self-alarm system should have a capability to monitor dependencies, such as libraries, modules and external components.

To satisfy the described needs, an ElasticSearch, Logstash and Kibana (ELK) [7] based solution was proposed and implemented. This paper presents a state of the art overview, the architecture and results of implementation of the self-monitoring alarm system which currently serves as a driver for the process of improving the BigPanDA reliability.

2. Classification of the BigPanDA system errors

The BigPanDA monitoring related errors could be grouped into three main categories depending on the layer where these errors occur and degree of their impact on a performance and the system stability:

- **Internal BigPanDA system errors**. These errors occur at the Django application layer and include the following types:
  - View errors. This is a critical type of errors originating in the Python code when unit tests failed to cover a particular case, and they lead to unavailability of some views to users. An unhandled exception of a wrong variable type or none values, in particular, leads to errors in this category.
  - Request errors. This is a non-critical type of errors coming from a combination of user’s query parameters and system state due to a lack of its support in the processing algorithms. A malformed URL request is an example.
- **External systems errors.** Errors of this category are caused by external components and system modules failures. They include the following types:
  - Database errors. These are a critical type of errors which happen when the system unsuccessfully attempts to retrieve data from the PanDA database. Such errors could be caused, for example, by exceeding the number of simultaneous sessions to the Oracle backend.
  - Dependency libraries errors are non-critical and raised in Python modules which are used by the BigPanDA monitoring system. One of the most frequently observed example is a “missing user session state” error in the social-auth library.
  - Cache errors is a critical type of errors which are related to the Redis distributed cache storage.

- **Superfluous requests.** This category of events that also needs to be handled is associated with users requests and their frequency when they create a negative impact on the system stability. The DoS-attack or irresponsible user behavior are examples of such events.

3. State of the art approaches for self-monitoring

The self-monitoring is a process of collecting, processing, aggregating, and displaying information about the current state of server nodes, components and modules in the real-time mode. There are two main approaches that can be used both separately and jointly to provide the self-monitoring functionality. The Simple Network Management Protocol (SNMP) is widely used for collecting information from network devices and servers. Solutions built on SNMP consist of four main components: an agent which collects various metrics values, SNMP-managed devices and resources, SNMP-manager and the management information base (MIB) [8]. The following self-monitoring software implements this approach:

- **SolarWinds Server and Application Monitor (SAM)**. It provides easy installation and setup, great visualization capabilities, but there are some limits implied by commercial licencing and this software to the best of our knowledge, supports only Microsoft Windows platform [9].
- **Zabbix** is a very powerful open-source software which provides rich functionality for Django applications monitoring but requires installation of special agents and customization efforts [10].
- **Nagios** is an open-source software which provides rich opportunities for servers and software health monitoring network infrastructure but needs a special configuration for the BigPanDA what may require considerable time for customization [11].

Another approach is based on the event logs analysis. Aggregation of the logs content allows to detect hidden system errors, track user activity, collect other system statistics [12]. The following solutions can be used for analyzing and processing logs:

- **Graylog** is an open-source log monitoring solution which provides rich opportunities for data processing and visualization. However, it requires a special infrastructure to be deployed, including ElasticSearch as a logs storage and MongoDB as a configuration data storage [13].
- **ELK-stack** is a collection of the open-source products which could be used for logs processing at data center scales. In this stack, the ElasticSearch, Logstash and Kibana are responsible for storing, processing and visualizing logs messages correspondingly. When the project has started, the ATLAS experiment had the ELK infrastructure already deployed, but a component dedicated to logs collecting was missing [13]. Additionally, it was required to develop a custom log messages processing scheme.

As it was pointed out, both SNMP and Event logs based analysis approaches require specific deployment procedures. Since ATLAS had an already existing ELK based infrastructure for the PanDA system logs processing [14] in operation and centrally supported, it was decided to develop the self-monitoring alarm system for the BigPanDA using this approach.

4. The BigPanDA self-monitoring alarm system

Filebeat [15] was installed on all BigPanDA server nodes in order to collect asynchronously all logs generated by the system. That data is streamed to the Logstash servers in the real time mode. Logstash aggregates all data flows from the Filebeat nodes, parses event logs messages and forwards the processed data to ElasticSearch. When an issue happens, the self-monitoring alarm system sends a notification message containing an error description to the developers. Architecture of the BigPanDA self-monitoring alarm system is shown in Figure 1.

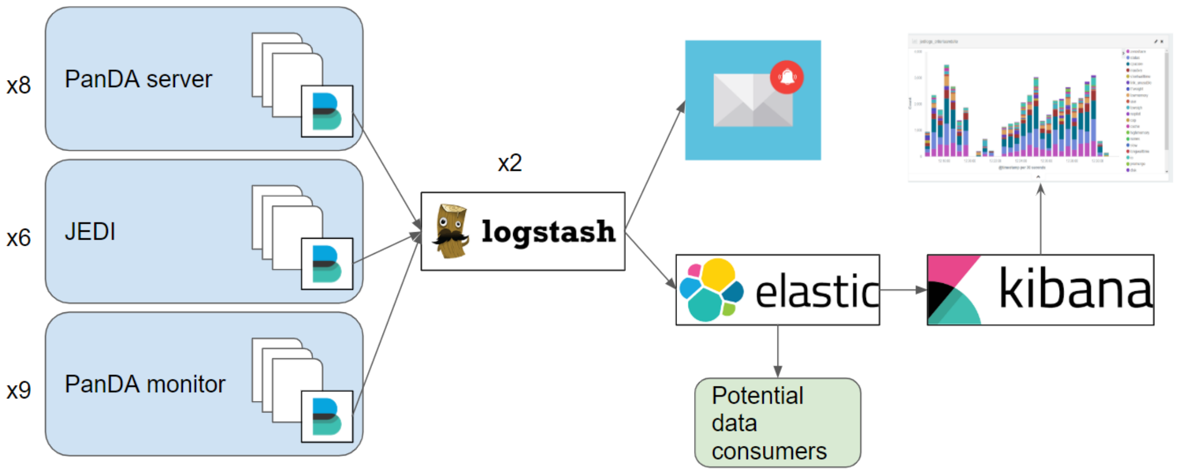

Figure 1. Architecture of the BigPanDA self-monitoring alarm system

The BigPanDA monitoring system produces two types of log files. The first one is a Django application log containing comprehensive information about the system activity. The second one is a Web-server log where all user requests are collected. An error description consists of information from both log types merged into a single message. The critical errors filtered by “Internal Server Error” condition can be caused by issues described in section 2 of this paper. The non-critical errors related to the infrastructure components are catched from Apache logs and filtered by the messages signature. This approach of BigPanDA logs processing allows to get only information from a large array of data which require an urgent developer intervention. The system sends error notifications both to BigPanDA developers and to a ADC Central Services operations member managing the BigPanDA infrastructure.

5. Results

The self-monitoring alarm system for the BigPanDA monitoring is in production since May 2017 and processes around 1 million log messages daily. The ELK-stack is used as core of the system for message filtering and sending error notifications to BigPanDA developers. Generally about 4000 notification candidates are generated daily by BigPanDA monitoring system and 99.5% of them are from broken client connections that can occur when a human user or script interrupts a session before the system deliver results. The self-monitoring system delivers only 0.5% of errors which explicitly require BigPanDA developers attention. Rest of them does not provide impact on the user experience and not needs to be fixed. Based on the information received from the developed solution, patches for Nginx load balancer, WSGI garbage collector and DDoS protection mechanism were developed and implemented. Development and implementation of the self-monitoring alarm system allowed to increase stability of the BigPanDA monitoring system and reduce the total number of errors.

Acknowledgments

This work was funded by the Russian Science Foundation under contract No 16-11-10280.

References

[1] The ATLAS Collaboration, 2008 The ATLAS experiment at the CERN Large Hadron Collider Journal of Instrumentation vol 3 S08003

[2] Maeno T. et al 2008 PanDA: distributed production and distributed analysis system for atlas Journal of Physics: Conference Series vol 119 no 6 P 062036

[3] Schovancova J., De K., Klimentov A., Love P., Potekhin M., Wenaus T. 2014 The next generation of ATLAS PanDA Monitoring. The International Symposium on Grids and Clouds (ISGC) 2014 March 23-28, 2014 Academia Sinica, Taipei, Taiwan

[4] Daniel C van der Ster, Johannes Elmsheuser, Mario Úbeda García1 and Massimo Paladin 2011 HammerCloud: A Stress Testing System for Distributed Analysis J. Phys.: Conf. Ser. 331 072036

[5] AES Monitoring. Available at: https://atlante.cern.ch/dashboard/db/event-service-sites?orgId=1 (accessed on 15.10.2018)

[6] De K., Klimentov A., Maeno T., Nilsson P., Oleynik D., Panitkin S., Petrosyan A., Schovancova J., Vaniachine A., Wenaus T. on behalf of the ATLAS Collaboration 2015 The future of PanDA in ATLAS distributed computing J. Phys.: Conf. Ser. 664 062035

[7] Welcome to the ELK Stack: Elasticsearch, Logstash, and KibanaPosted by John Vanderzyden July 17, 2015 https://qbox.io/blog/welcome-to-the-elk-stack-elasticsearch-logstash-kibana (accessed on 15.10.2018)

[8]SimpleNetworkManagement Protocol (SNMP) Documentation. Available at: https://searchnetworking.techtarget.com/definition/SNMP (accessed on 15.10.2018)

[9] Server & Application Monitor Guide. Available at: https://www.solarwinds.com/-/media/solarwinds/swdcv2/licensed-products/server-application-monitor/resources/product-guides/sam_evaluators_guide.ashx?la=es (accessed on 15.10.2018)

[10] Zabbix Documentation 4.0. Available at: https://www.zabbix.com/documentation/4.0/manual (accessed on 15.10.2018)

[11] Nagios Core Documentation. Available at: https://assets.nagios.com/downloads/nagioscore/docs/nagioscore/4/en/toc.html (accessed on 15.10.2018)

[12] Techopedia. Log Analysis. Available at: https://www.techopedia.com/definition/31756/log-analysis (accessed on 15.10.2018)

[13] Log Monitoring and Analysis: Comparing ELK, Splunk and Graylog. Available at: https://devops.com/log-monitoring-and-analysis-comparing-elk-splunk-and-graylog/ (accessed on 15.10.2018)

[14] Saiz P., Schwickerath U. 2017 Centralising elasticsearch *Technical report* (Geneva, CERN)

[15] Filebeat overview Available at: https://www.elastic.co/guide/en/beats/filebeat/current/filebeat-overview.html (accessed on 29.10.2018)