Building analytical platform with Big Data solutions for log files of PanDA infrastructure

A A Alekseev[1], F G Barreiro Megino[2], A A Klimentov[3], T A Korchuganova[1], T Maendo[3], S V Padolski[3]

[1] Tomsk Polytechnic University, 30, Lenina Ave., Tomsk, 634050, Russia
[2] University of Texas at Arlington, 701 South Nedderman Drive, Arlington, TX 76019, USA
[3] Brookhaven National Laboratory, P.O. Box 5000, Upton, NY 11973-5000, USA

E-mail: frt@tpu.ru

Abstract. The paper describes the implementation of a high-performance system for the processing and analysis of log files for the PanDA infrastructure of the ATLAS experiment at the Large Hadron Collider (LHC), responsible for the workload management of order of 2M daily jobs across the Worldwide LHC Computing Grid. The solution is based on the ELK technology stack, which includes several components: Filebeat, Logstash, ElasticSearch (ES), and Kibana. Filebeat is used to collect data from logs. Logstash processes data and export to Elasticsearch. ES are responsible for centralized data storage. Accumulated data in ES can be viewed using a special software Kibana. These components were integrated with the PanDA infrastructure and replaced previous log processing systems for increased scalability and usability. The authors will describe all the components and their configuration tuning for the current tasks, the scale of the actual system and give several real-life examples of how this centralized log processing and storage service is used to showcase the advantages for daily operations.

1. Introduction

Nowadays, there are many large-scale, distributed systems that generate a lot of meta information during their business operation. One example could be the workflow management system PanDA (Production and Distributed Analysis) [1] of the ATLAS [2] experiment at the Large Hadron Collider (LHC). PanDA is responsible for generating, brokering and monitoring up to 2 million tasks a day across 150 computer centers in the Worldwide LHC Computing Grid (WLCG) [3], associated HPC centers and Cloud resources. PanDA consists of several components, including the JEDI backend (Job Execution and Definition Interface) [4], the PanDA server frontend and the PanDA monitoring [5] web interface. These 3 components are deployed on a total of 20 servers and its agents write their actions to around 60 log files with a total volume of 400GB per day. It requires a high level of expertise to find the relevant log events for a particular action. Since the services are running on multiple machines, it is necessary to ssh to all servers and grep for the correct pattern in the correct files. For this reason, during the history of PanDA, different solutions have been implemented in order to collect and show centrally important log fragments to the operations team:

1. **HTTP server and Oracle based log collection**. PanDA and JEDI loggers were instrumented with a HTTP handler, which would send selected messages to a dedicated HTTP server. The server received these messages and stored them in a special table of the PanDA database (implemented in Oracle). However, this way of processing and storing logs had several disadvantages:
   1.1. **Coupling**. PanDA components were coupled to the shipping of log messages. Delays in the sending of log messages would also slow down the PanDA agents. If messages hit the time-out, they would be discarded to avoid affecting the business logic.
   1.2. **Scalability**. Messages were sent one at a time and would require an intermediate layer to set up bulk processing. The messages to be sent to the logging service had to be chosen carefully.
   1.3. **Accessing the data**. Apart from a few optimizations such as an index on the job ID and time partitioning, the messages were stored as plain text and required scanning the table in order to search for particular messages.
   1.4. **Representation of information**. Information was presented in dedicated tables with not too many options.
2. **Flume server and ElasticSearch based log collection**. This solution consisted in replacing the HTTP server by load-balanced Flume servers, which shipped the messages to a ElasticSearch ATLAS cluster. Shortcomings 1.1 and 1.2 remained. The access of the data improved slightly, but the authors were not splitting the messages and making full usage of the ElasticSearch schema-free storage capabilities. The representation of information was also limited by the latter point.

To solve these shortcomings, the authors decided to upgrade to a Filebeat, Logstash and ElasticSearch set up, which will be explained in detail during the next sections.

2. The Filebeat, Logstash and ElasticSearch setup

Filebeat and Logstash are part of the ELK technology stack, which is responsible for collecting (Filebeat), processing (Logstash), storing (ElasticSearch) and visualizing (Kibana) the logs.

Filebeat is a daemon installed on all PanDA machines that allows one to process asynchronously all possible logs. Filebeat processes tail the desired log files and track the position in each file. Data are exported to Logstash servers in real time. Each node is configured using the special configuration file, which is based on the YAML markup language. The configuration file is used to tune the Filebeat daemon, specify the number of processes, processing rules and bulk sizes.

Logstash servers aggregate all data from the Filebeat daemons, parse messages using special filters and forward the processed data to ElasticSearch. For redundancy reasons, the Logstash setup consists of two virtual machines (4 cores, 8 GB RAM) running in a load balanced mode (figure 1).

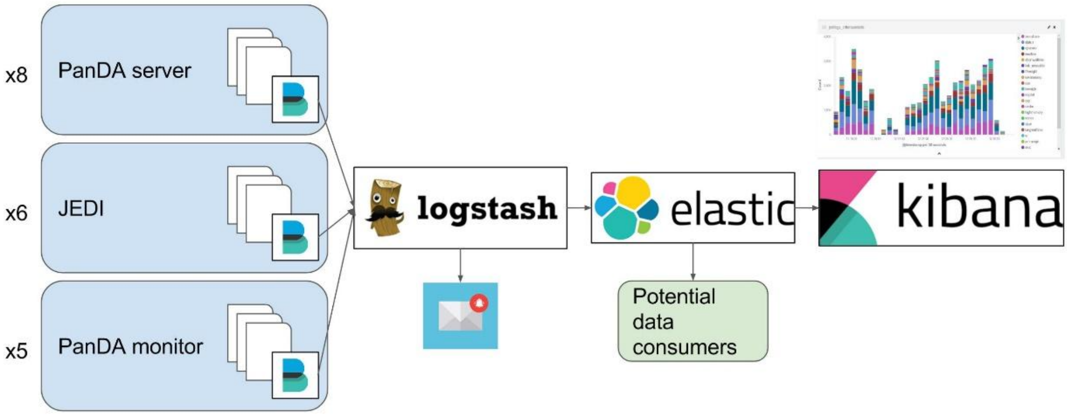
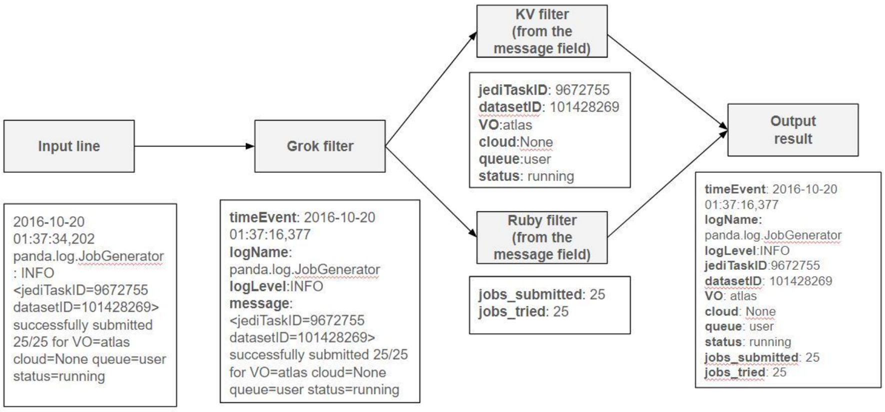


Figure 1. Data flow in the ELK stack for PanDA infrastructure.

For the processing of log messages, the authors have defined around 60 filters (one per PanDA log file), which specify how to split each text line and which fields want to be indexed. Indexed fields can later be searched, filtered and aggregated. It is important to mention that many of the PanDA components are up to 12 years old and the hundreds of different log messages were not following any patterns or meant to be processed programmatically at the time. The implementation of filters was an arduous task and we are also defining certain rules for the structure of new messages to simplify this task. Each filter can consist of several plugins, an example of a log file processing result using plugins is shown in figure 2, which are responsible for processing the input string:

1. **The Grok plugin.** This plugin uses a regular expression to split strings into separate fields.
2. **The KV plugin.** This plugin extracts all values that match the pattern of key-value.
3. **The Ruby plugin.** This plugin is used for processing strings and getting necessary values using special conditions.


Input line
Grok filter
KV filter (from the message field)
Ruby filter (from the message field)
Output result

2016-10-20 01:37:34,202 panda.log.JobGenerator : INFO <jediTaskID=9672755 datasetID=101428269> successfully submitted 25/25 for VO=atlas cloud=None queue=user status=running

timeEvent: 2016-10-20 01:37:16,377
logName: panda.log.JobGenerator
logLevel:INFO
message: <jediTaskID=9672755 datasetID=101428269> successfully submitted 25/25 for VO=atlas cloud=None queue=user status=running

jediTaskID: 9672755
datasetID: 101428269
VO:atlas
cloud:None
queue:user
status: running

jobs_submitted: 25
jobs_tried: 25

timeEvent: 2016-10-20 01:37:16,377
logName: panda.log.JobGenerator
logLevel:INFO
jediTaskID:9672755
datasetID: 101428269
VO: atlas
cloud: None
queue: user
status: running
jobs_submitted: 25
jobs_tried: 25

Figure 2. Stages of log processing by the example of a «JobGenerator» message.

After processing, the data are exported to ElasticSearch (ES). ES is based on the Lucene search engine and provides a non-relational, distributed data store, where the data is stored as indices. The current setup is based on a ES cluster provided centrally by CERN IT [6] and shared with other services of the ATLAS Distributed Computing infrastructure. The ES data can be viewed using the Kibana web interface, which allows inspecting the raw data, making graphs or tables and combining them into logical groups – dashboards.

3. Results and discussion

On a daily basis, ES receives about 20GB (30M-40M messages) of data from JEDI and about 40GB (60M-80M messages) of data from PanDA servers (figure 3). The data volume from the PanDA monitor is more modest at the moment. This data volume is a trade off between overflowing the ElasticSearch cluster with data and having the necessary information. The authors also had to tune other parameters (e.g. to limit the memory consumption of Filebeat) to guarantee a stable service. Since the ES cluster is shared and sized to ~10TB, the authors have also set up a Curator service. Curator purges old data, limiting its lifetime to 20 days.

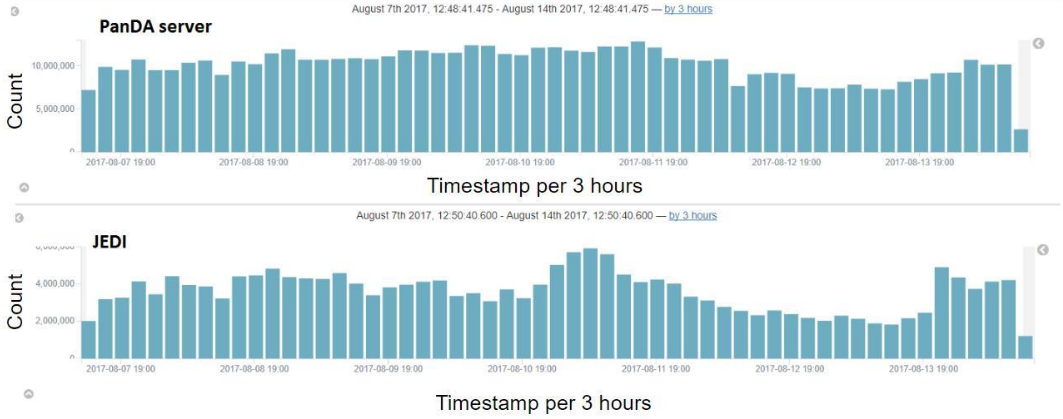
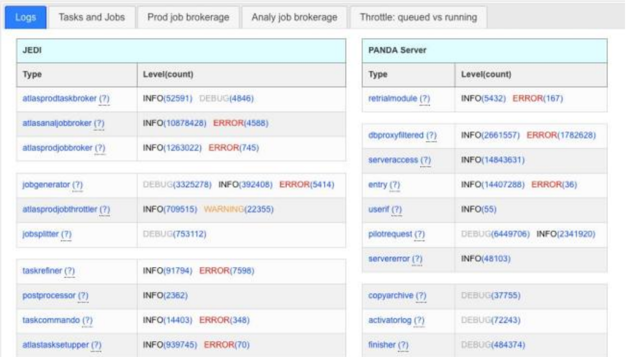


Figure 3. Graphics taken from Kibana with the number of entries received from Logstash, in the period from August 7 to August 14, with 3 hours interval.

Given the data volume and different types of logs, we needed to simplify the access to the data. For this reason, we have implemented an index page (figure 4) in the PanDA monitor, which is used on a daily basis by the operations team, shifters and end users to monitor jobs and troubleshoot potential issues. Two tables (left one for JEDI and right one for PanDA server) link to the different types of logs in ElasticSearch. The order was agreed with the operations team and organizes the logs by order of interest, i.e. the most interesting log types for users are at the top of the page. In addition, tooltips have been added to explain the information contained each log file.



JEDI	
Type	Level(count)
atlasprodtaskbroker (?)	INFO(52591) DEBUG(4846)
atlasanaljobbroker (?)	INFO(10878428) ERROR(4588)
atlasprodjobbroker (?)	INFO(1263022) ERROR(745)
jobgenerator (?)	DEBUG(3325278) INFO(392408) ERROR(5414)
atlasprodjobthrottler (?)	INFO(709515) WARNING(22355)
jobsplitter (?)	DEBUG(753112)
taskrefiner (?)	INFO(91794) ERROR(7598)
postprocessor (?)	INFO(2362)
taskcommando (?)	INFO(14403) ERROR(348)
atlastasksetupper (?)	INFO(939745) ERROR(70)

PANDA Server	
Type	Level(count)
retrialmodule (?)	INFO(5432) ERROR(167)
dbproxyfiltered (?)	INFO(2661557) ERROR(1782628)
serveraccess (?)	INFO(14843631)
entry (?)	INFO(14407288) ERROR(36)
userif (?)	INFO(55)
pilotrequest (?)	DEBUG(6449706) INFO(2341920)
servererror (?)	INFO(48103)
copyarchive (?)	DEBUG(37755)
activatorlog (?)	DEBUG(72243)
finisher (?)	DEBUG(484374)

Figure 4. Page for PanDA and JEDI logs in BigPanDA monitoring system.

Links to the raw log data are still too difficult to manage for the end user. Therefore, the authors have implemented dashboards to address some of the most frequently asked questions. Some examples are:

1. **Brokerage.** Brokerage assigns jobs to the WLCG computing centers based on different criteria (availability of data, availability of the software releases, memory, space, etc.). There are frequently questions why a site is not getting any jobs or why a particular task was assigned to site X. The brokerage dashboard allows filtering brokerage logs by site or task to find the answer to these questions. Figure 5 shows a real life example, where a site administrator wanted to know why his site was not getting any jobs. The dashboard revealed that the top reasons for not getting any jobs were that the site was configured to run only certain types of jobs and did not have the necessary software releases installed (cache). Once the software releases were installed on the site, jobs began to be assigned.

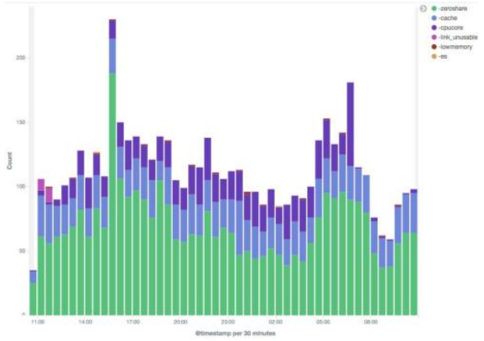
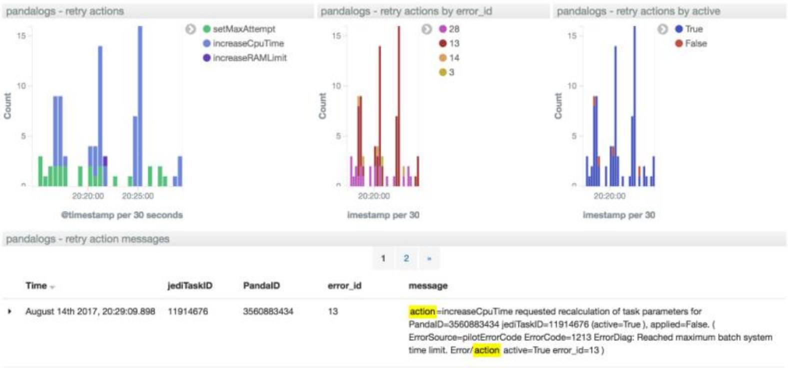


Figure 5. Exclusion criteria for a particular site.

2. **Job throttler.** The throttler decides if a particular workqueue (processing pipe for a certain activity) should be throttled or more jobs need to be generated. This dashboard can plot the queued and running jobs for each workqueue.
3. **Retry module actions.** The retry module tunes the memory and CPU time of failed jobs for their retry or blocks jobs with fatal errors from being retried, based on the error code/message and rules

defined by the operations team. The operations team frequently wants to know if an action was applied to a particular job/task and if new rules are triggering (figure 6).


pandalogs - retry actions
setMaxAttempt
increaseCpuTime
increaseRAMLimit
15
10
5
Count
20:20:00
20:25:00
@timestamp per 30 seconds
pandalogs - retry actions by error_id
28
13
14
3
imestamp per 30
pandalogs - retry actions by active
True
False
imestamp per 30
pandalogs - retry action messages
1
2
»

Time	jediTaskID	PandaID	error_id	message
August 14th 2017, 20:29:09.898	11914676	3560883434	13	action=increaseCpuTime requested recalculation of task parameters for PandaID=3560883434 jediTaskID=11914676 (active=True), applied=False. (ErrorSource=pilotErrorCode ErrorCode=1213 ErrorDiag: Reached maximum batch system time limit. Error/action active=True error_id=13)

Figure 6. A snapshot of the retry module dashboard. From left to right, the plots show the incident breakdown by action, rule and whether the rule is active/passive. Messages/plots can be filtered for a given task or job ID for easy debugging.

4. **Server metrics.** Helps to monitor PanDA server metrics by indexing the httpd access logs (figure 7). Available information is number of requests per second, execution times and return codes. This information is useful for scaling the system, observing whether the load balancing across the 8 servers is functioning properly and if there are any health issues. We have used this information to verify high load alarms for one of the servers and to identify a client using HTTP methods not supported by PanDA server.

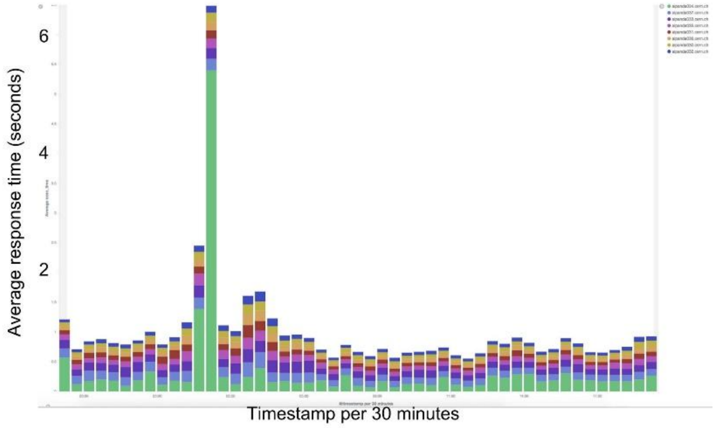
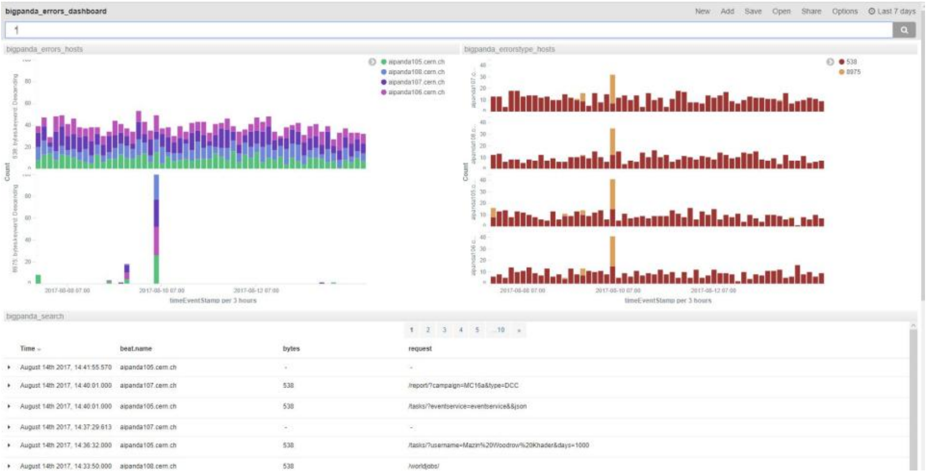


Figure 7. An example of average response times by PanDA server. The bottom server (green) is showing worse execution times than the rest and needed to be restarted.

More recently, PanDA monitor nodes have also been instrumented with Filebeat. At the moment the authors have implemented dashboards to check monitor errors (figure 8) and an automated error report using the logstash emailing service. The error reports are used for proactive fixing of broken views and to increase user satisfaction.



Time	beat.name	bytes	request
August 14th 2017, 14:41:55.570	aipanda105.cern.ch	-	-
August 14th 2017, 14:40:01.000	aipanda107.cern.ch	538	/report/?campaign=MC16a&type=DCC
August 14th 2017, 14:40:01.000	aipanda105.cern.ch	538	/tasks/?eventservice=eventservice&&json
August 14th 2017, 14:37:29.613	aipanda107.cern.ch	-	-
August 14th 2017, 14:36:32.000	aipanda105.cern.ch	538	/tasks/?username=Mazin%20Woodrow%20Khader&days=1000
August 14th 2017, 14:33:50.000	aipanda108.cern.ch	538	/worldjobs/

Figure 8. A example of average response times by PanDA server. The bottom server (green) is showing worse execution times than the rest and needed to be restarted.

4. Conclusion

The authors have implemented a solution for centralized log analysis using up-to-day, big data solutions. The current setup works close to real time and processes 60GB of logs per day without signs of hitting scaling issues. The authors have now a more scalable, useful and completely decoupled system, which has overcome all shortcomings of the previous logging systems. The authors have also shown the current integration with the monitoring system and several dashboards designed to answer day to day answers from the operations team and users.

5. Acknowledgments

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

References

[1] Maeno T et al 2008 Panda: distributed production and distributed analysis system for atlas *Journal of Physics: Conference Series* **119** 6 P 062036

[2] The Atlas Collaboration, Aad G, Abat E, et al 2008 The atlas experiment at the cern large hadron collider *Journal of Instrumentation* **3** 08 P S08003

[3] LCG Collaboration 2005 Lhc computing grid: Technical design report *document lcg-tdr-001* cern-lhcc-2005-024

[4] De K, Golubkov D, Klimentov A, et al 2014 Task management in the new atlas production system *Journal of Physics: Conference Series* **513** 3 P 032078

[5] Schovancova J, De K, Klimentov F, Love P, Potekhin M, Wenaus T 2014 The next generation of ATLAS PanDA Monitoring *The International Symphosium on Grids and Clouds (ISGC)*

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