Evaluation of NoSQL databases for DIRAC monitoring and beyond

Evaluation of NoSQL databases for DIRAC monitoring and beyond

Z Mathe,[1] A Casajus Ramo,[2] F.Stagni[1] and L. Tomassetti [3]

[1] CERN, European Organization for Nuclear Research, Switzerland
[2] University of Barcelona, Barcelona, Spain
[3] University of Ferrara, Ferrara, Italy

E-mail: `zoltan.mathe@cern.ch`

Abstract. Nowadays, many database systems are available but they may not be optimized for storing time series data. Monitoring DIRAC jobs would be better done using a database optimised for storing time series data. So far it was done using a MySQL database, which is not well suited for such an application. Therefore alternatives have been investigated. Choosing an appropriate database for storing huge amounts of time series data is not trivial as one must take into account different aspects such as manageability, scalability and extensibility. We compared the performance of Elasticsearch, OpenTSDB (based on HBase) and InfluxDB NoSQL databases, using the same set of machines and the same data. We also evaluated the effort required for maintaining them. Using the LHCb Workload Management System (WMS), based on DIRAC as a use case we set up a new monitoring system, in parallel with the current MySQL system, and we stored the same data into the databases under test. We evaluated Grafana (for OpenTSDB) and Kibana (for ElasticSearch) metrics and graph editors for creating dashboards, in order to have a clear picture on the usability of each candidate. In this paper we present the results of this study and the performance of the selected technology. We also give an outlook of other potential applications of NoSQL databases within the DIRAC project.

1. Introduction

DIRAC interware (Distributed Infrastructure with Remote Agent Control)[1, 2] is a software framework, which delivers a complete solution for managing distributed heterogeneous computing resource, such as Grid, Cloud. It allows users to efficiently use different computing resources managed by DIRAC. Within DIRAC various activities can be performed such as data analysis, data processing, data replication, etc. The accounting and monitoring system are crucial in order to monitor each activity, and to keep track of the usage of the distributed computing resources as well as monitoring the DIRAC system. The DIRAC Accounting System based on MySQL [3] is developed to fulfil these criteria. The number of computing tasks managed by DIRAC is growing by the size of the datasets, which needs to be processed and by the size of the computing resources. The current Accounting system can not handle efficiently the huge amount of monitoring information. The tasks, which required monitoring, are executed in a certain time, which needs to be associated to the task. Consequently, the information, which needs to be stored are time series data. Various database systems are available but they are not optimised for storing huge amount of time series data.

The LHCbDIRAC Workload Management System (WMS) [4, 5], provides a jobs scheduling mechanism. WMS History is an accounting type of the Accounting system, which keeps the

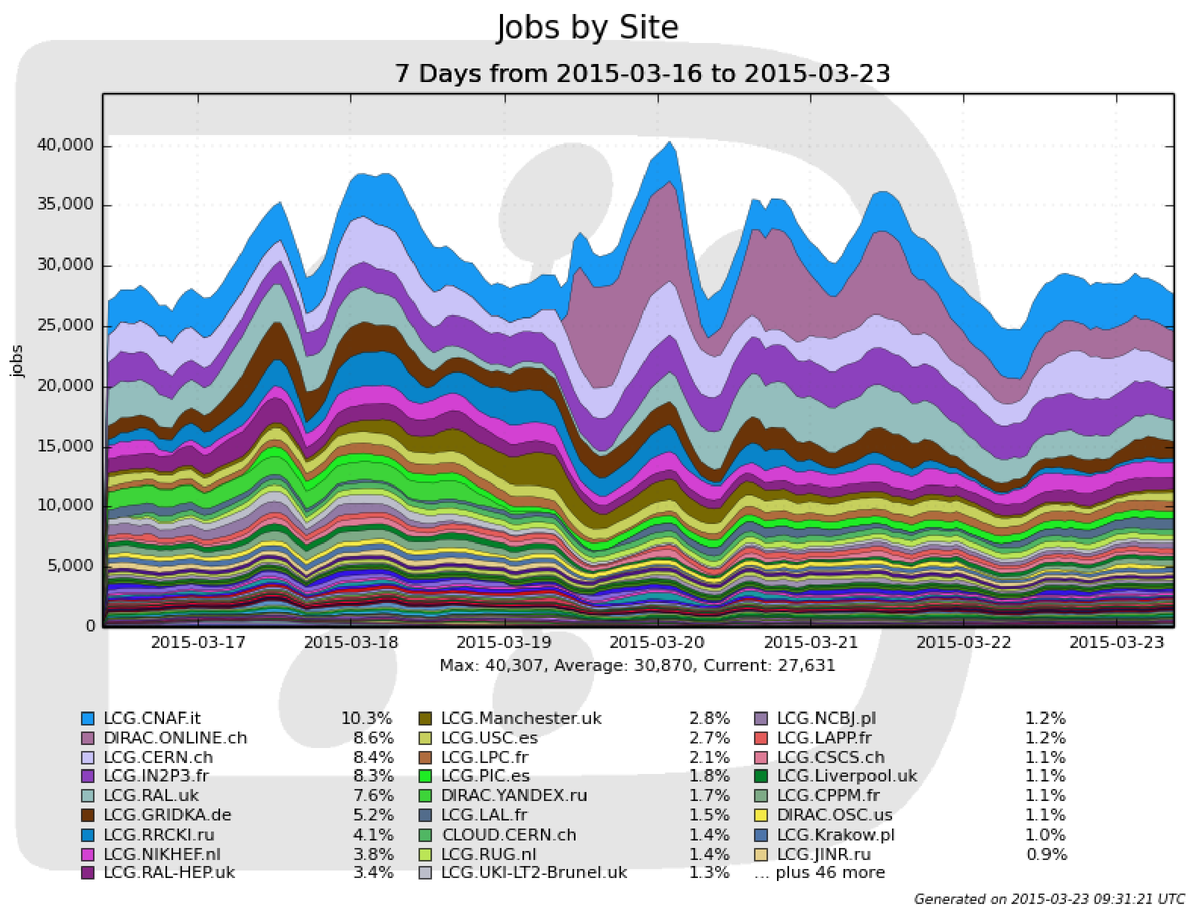


Figure 1. Running jobs grouped by site in DIRAC during 1 month

history of jobs treated by the WMS. In order to assess the performance of the databases under test, we set up in parallel with the current system a new monitoring system to which we publish the same data that we publish on the old production system.

In this paper we present the new monitoring system, which is based on NoSQL and designed to store time series data. In order to choose an appropriate database for storing time series data we studied the performance of OpenTSDB [6], Elasticsearch [7] and InfluxDB [8] time series databases. The result of the study is presented in this paper. In section 2, we present the disadvantages of the current system. We present 3 distributed time series NoSQL databases and data visualization tools in section 3. In section 4, we present the new monitoring system. The performance of the new monitoring system is presented in section 5.

2. Disadvantages of the current Accounting system

The current Accounting system is implemented in MySQL and is not optimised for analysis of time series data. Since MySQL is a RDBMS, every time when a new accounting type is defined the corresponding relational schema and the appropriate table indexes have to be created. The current accounting system is not designed for real time monitoring, because each record (*raw* data) is inserted to an intermediate table. Different parameters from the *raw* records are accumulated in time buckets and classified according to suitable parameters [3]. These buckets are used to generate different plots. The current system has limited scalability. For example it requires approximately 400 second to generate Figure 1 using DIRAC WMS History data, which contains more than 500 million rows.

3. Time series NoSQL database and data visualization tools

We decided to use 3 of the most popular NoSQL systems, which are used for time series analysis:

(i) OpenTSDB is a time series database, which stores and serves massive amounts of time series data without losing granularity. It is based on HDFS [9] and HBase [10] and provides a set of command line utilities used to manage the database. HBase schema is optimised for fast aggregations and storage usage.

(ii) InfluxDB is a distributed time series, metrics and analytics database. It supports SQL like query language.

(iii) Elasticsearch is a document store distributed real time search and analytics database based on Apache Lucene[11]. It is designed to use when the data have been written once to the database and have been read many times as the previous and other NoSQL databases. Since the database operations always perform on indexes, most of the time when the data format is changed the indexes have to be recreated.

We also evaluated the following visualization tools:

(i) Grafana [12] is an open source, rich metrics dashboard and graph editor for Graphite, InfluxDB and OpenTSDB. It is a powerful tool to create and share very complex dashboards. It provides pluggable panels, which can be easily extended.

(ii) Kibana [13] is an open source; browser based flexible analytic and visualization platform for Elasticsearch. It allows creating very complex dashboards, which can be shared easily as well the dashboards can be embedded.

4. Overview of the new monitoring system

The monitoring system has been running on 3 clusters with 4 nodes each provided by the CERN OpenStack cloud, using SLC6 operating system. Each node has 4 cores Intel Core i7 9xx (Nehalem Class Core i7) 2266.746 MHz CPUs, 8GB memory and 80GB disk.

A high level overview of the monitoring system is shown in Figure 2. The **RabbitMQ** [14] message broker is used to exchange messages between different components of the system. The Advanced Message Queuing Protocol (AMQP) is used to publish the data to the message broker, as well as to consume the data. We developed the DIRAC *StatesMonitoring* agent [15], which every 5 minutes retrieves the history of the jobs from the WMS and send to a RabbitMQ. We use a RabbitMQ exchange to insert the messages into different queues: OpenTSDB, InfluxDB and Elasticsearch. A RabbitMQ consumer is running on each database nodes to consume the messages from its queue and insert to the database. Python API is used to communicate with the broker as well as to insert the data to the databases. We set up **Grafana**, which is the front end of InfluxDB and OpenTSDB and **Kibana** for Elasticsearch.

5. Performance comparison

In our tests we used 600 million records which have been recorded in 1.5 month. We compared only the performance of OpenTSDB and Elasticsearch because InfluxDB almost immediately proved not to be a good solution for us, because it did not scaled as the other ones. We defined five different queries shown in figure 3 and four randomly generated query intervals: 1, 2, 7 and 30 days. We generated high workload using 10, 50 and 100 clients (python threads), which had been simultaneously running in parallel during 7200 second. We developed a simple python script, which is executed by the python threads. Using the REST APIs, this simple python script had been simulating the users behaviour by generating and executing the database queries. In our tests we measured the response time and the throughput of the front end.

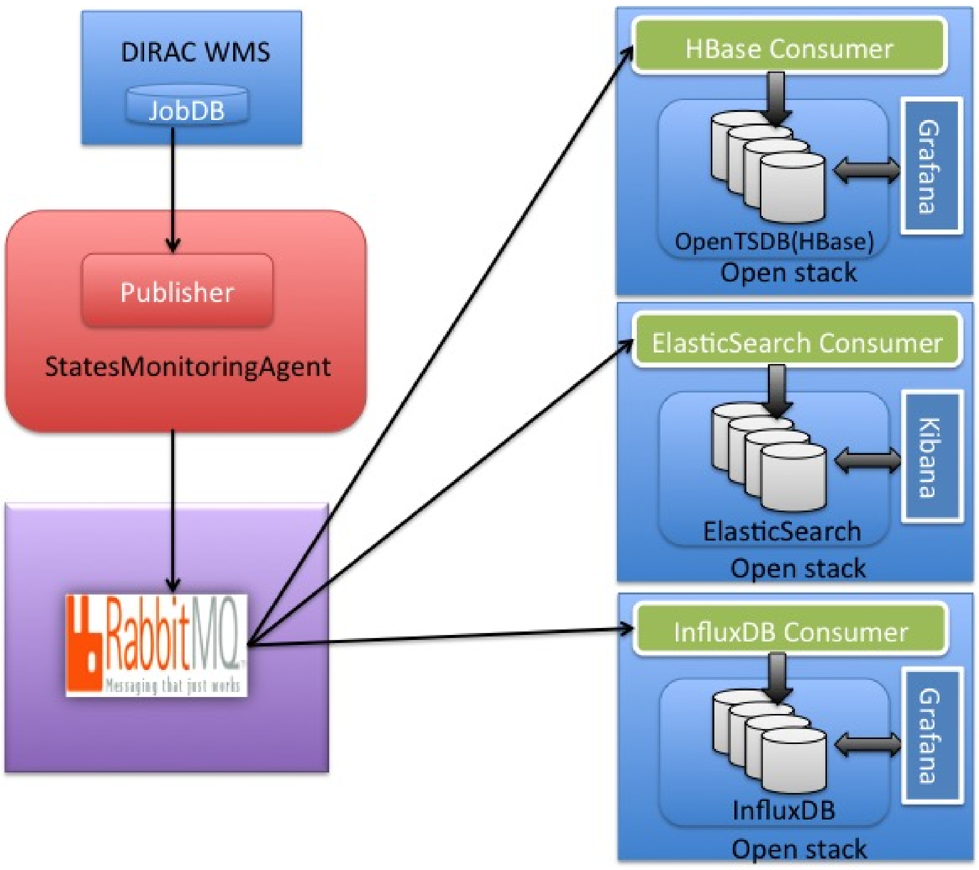
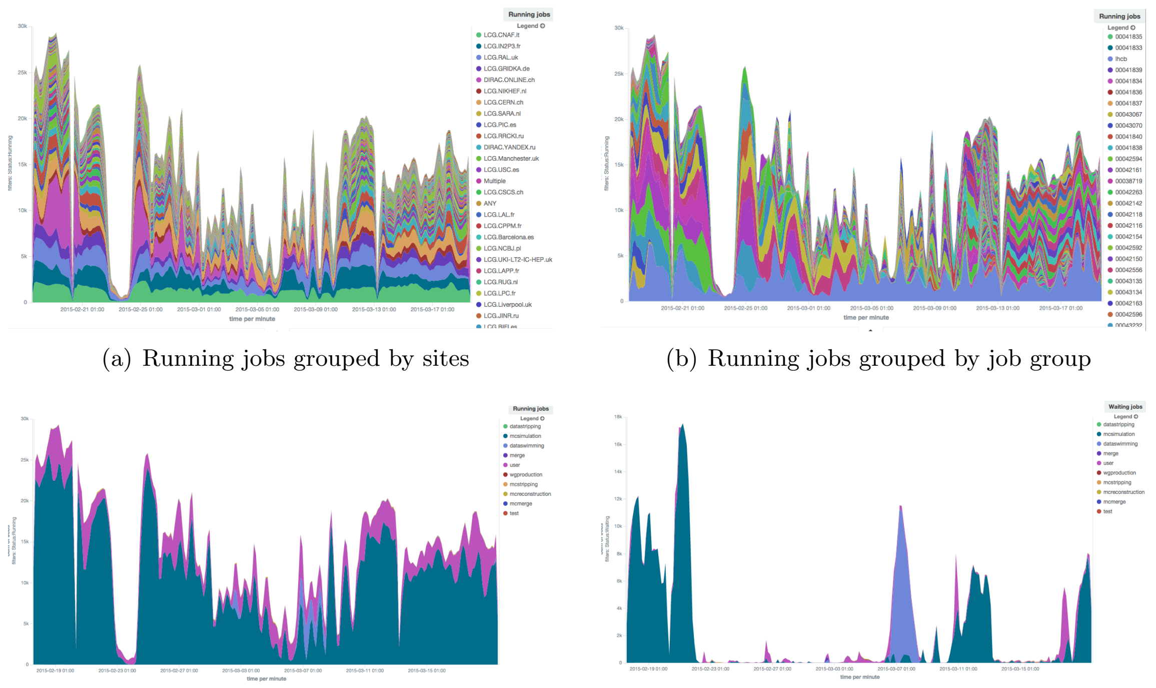


Figure 2. New monitoring system

10 clients
First we only used 10 "virtual" users in order to not overload the databases. The response time of the databases was almost the same when the query interval was 1 day. Increasing the query intervals to 7 days increased the response time of OpenTSDB, as well as the number of executed queries (throughput) are decreased by almost a factor two. The response time of the Elasticsearch is increased by a factor 4 by using 30 days time interval compared to the average response time of the 7 day plot. The response time of OpenTSDB increased by a factor 4. Elasticsearch was almost 2 times faster than OpenTSDB comparing the average response time of the 30 days plots. Taking into account the results of this experiment we can conclude that the Elasticsearch cluster is faster than the OpenTSDB cluster, and the response time is increasing by increasing the query interval in both cases.

50 clients
In order to load the system, we increased the number of clients (python threads) to 50. The results of the test are found in figure 5. Increasing the number of clients in both cases increases the average response time and the number of executed queries. However, the response time of OpenTSDB is 5 times higher than the previous tests. The number of executed queries does not changed too much. In this test Elasticsearch was scaling better than OpenTSDB.



(a) Running jobs grouped by sites (b) Running jobs grouped by job group

(c) Running jobs grouped by activities (d) Waiting jobs grouped by activities

Figure 3. Four different plots used to test the system

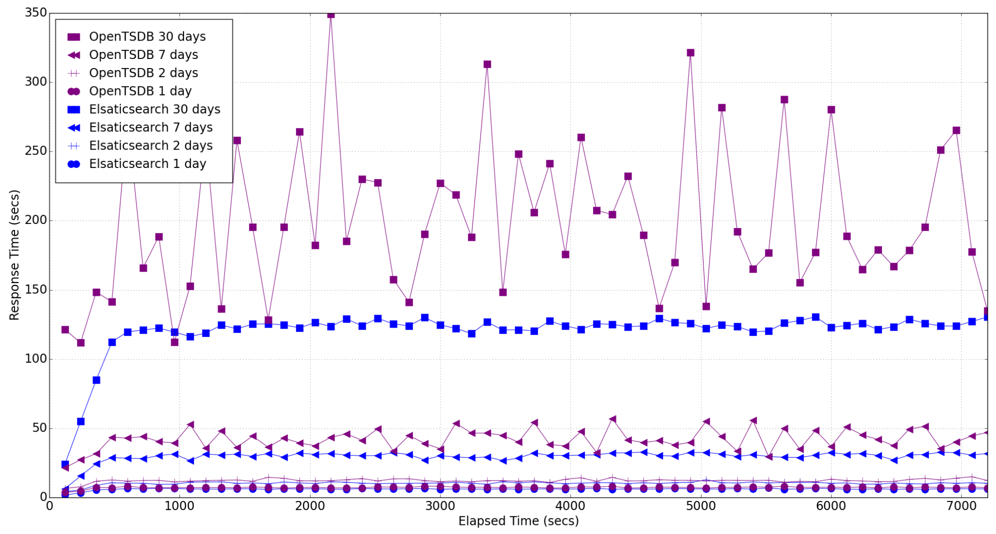


Figure 4. Average response time of Elasticsearch and OpenTSDB using 10 parallel clients.

100 clients
In an attempt to see how the databases behave under very high load we increased the number of clients to 100. Figure 6 shows the result of the tests. The response time of Elasticsearch has not changed significantly compared to the previous tests. The number of executed queries is increased which shows that Elasticsearch is able to scope under very high load. The response

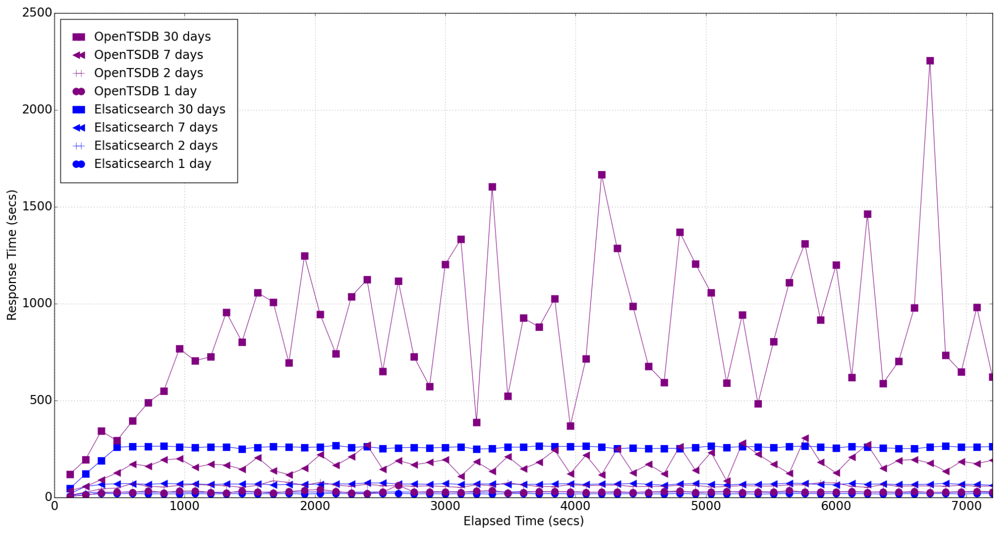


Figure 5. Average response time of Elasticsearch and OpenTSDB using 50 parallel clients.

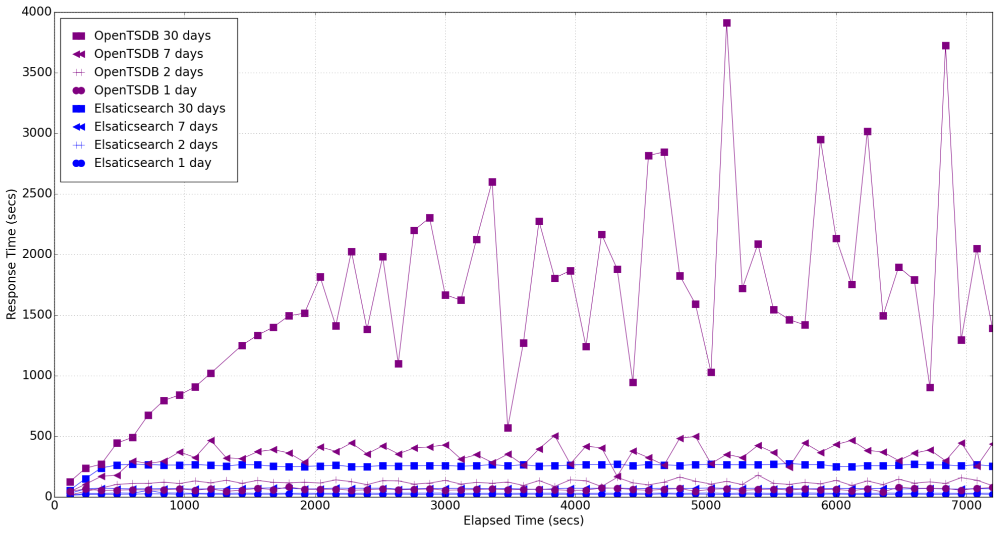


Figure 6. Average response time of Elasticsearch and OpenTSDB using 100 parallel clients.

time of the OpenTSDB is increased by a factor two compared to the previous test (50 clients), while the number of executed queries is almost the same.

Throughput
We measured the throughput, which is the number of transactions during 7200. Figure 7 shows the throughput of the OpenTSDB and Elasticsearch. According to Fig. 7 Elasticsearch is able to serve more transactions than OpenTSDB. When the query interval was more than 1 day the executed queries per second is not changed significantly. The result of the tests show

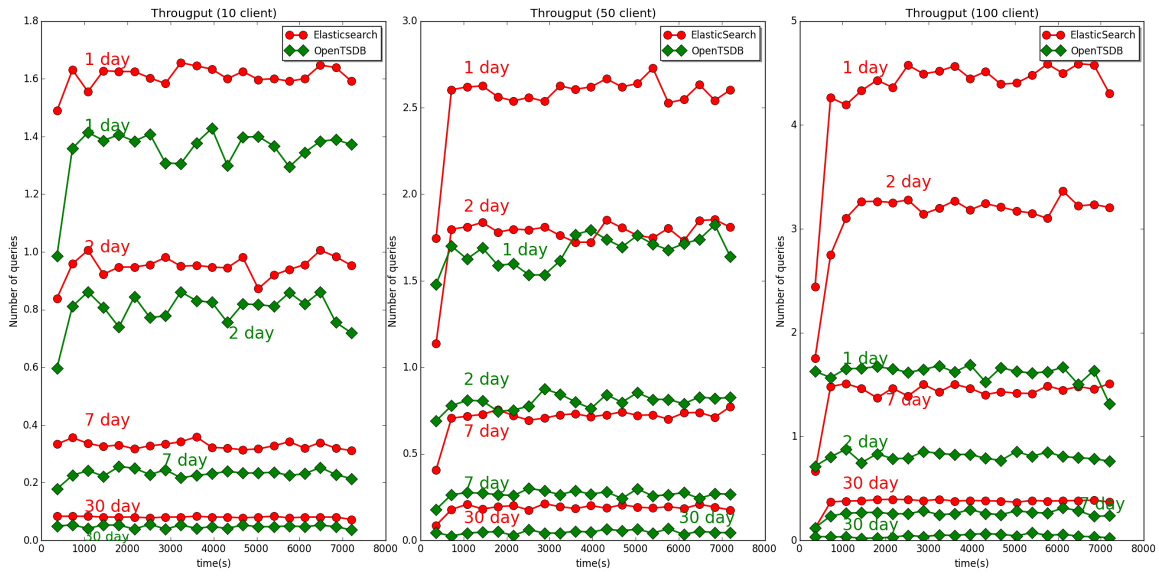


Figure 7. Average response time of Elasticsearch and OpenTSDB using 100 parallel clients.

that Elasticsearch can handle approximately four transactions per second while OpenTSDB can handle almost two transactions per second.

Taking into account the manageability and performance of the databases presented above we have decided to use Elasticsearch. OpenTSDB may scale better when the size of the cluster is bigger than what we used in our performance comparison.

6. Possible usage of NoSQL within DIRAC

NoSQL technologies have recently introduced within DIRAC. We studied the possible usage of NoSQL technologies in order to make DIRAC more reliable and robust. NoSQL can be used efficiently for storing non structured data and when the data have to be written once and have to be read many times. Elasticsearch can also be used to monitor the DIRAC components like services, agents and executors.

Currently, the logging information of the DIRAC components is not analysed. Logstash can be used to analyse the logging information of the DIRAC components and insert to Elasticsearch. As well as Hadoop [16] can be used to store the log files of the DIRAC components in the HDFS file system.

Another use case is to store statistical information about the LHCb Monte Carlo productions.

7. Future work

The evaluation of NoSQL databases is still in progress in order to find use cases in DIRAC to improve its performance. We have started to replace the current MySQL database used for WMS monitoring by Elasticsearch. In addition to Kibana, we would like to extend the existing DIRAC Accounting web application used to visualise the data, which is stored in Elasticsearch.

8. Conclusion

Elasticsearch is fulfilling our needs and can be used for monitoring purpose. Kibana is a very good tool to create complex dashboards, but it is not possible to create very good quality plots. According to the results Elasticsearch was faster than OpenTSDB, but it required optimisations for using in production. InfluxDB is a new time series database, which is rapidly changing and the number of developers who are contributing to the code are significantly increasing. RabbitMQ message broker can be used for exchanging information between the loosely coupled components.

References

[1] Casajus A, Ciba K, Fernandez V, Graciani R, Hamar V, Mendez V, Poss S, Sapunov M, Stagni F, Tsaregorodtsev A and Ubeda M 2014 *Journal of Physics: Conference Series 396 032107* URL `http://stacks.iop.org/1742-6596/396/i=3/a=032107`
[2] Tsaregorodtsev A 2014 *Journal of Physics: Conference Series 513 032096* URL `http://stacks.iop.org/1742-6596/513/i=3/a=032096`
[3] Casajus A, Diaz R G, Puig A and Vazquez R 2011 *Journal of Physics: Conference Series 331 072059* URL `http://iopscience.iop.org/1742-6596/331/7/072059/pdf/1742-6596_331_7_072059.pdf`
[4] Paterson S and Closier J 2010 *Journal of Physics: Conference Series 219 072015* URL `http://iopscience.iop.org/1742-6596/219/7/072015/pdf/1742-6596_219_7_072015.pdf`
[5] Casajus A, Graciani R and Tsaregorodtsev S P A 2010 *Journal of Physics: Conference Series 219 062049* URL `http://iopscience.iop.org/1742-6596/219/6/062049/pdf/1742-6596_219_6_062049.pdf`
[6] Opentsdb URL `http://opentsdb.net/`
[7] Elasticsearch URL `https://www.elastic.co/`
[8] Influxdb URL `http://influxdb.com/`
[9] Hdfs URL `http://hadoop.apache.org/docs/r1.2.1/hdfs_design.html`
[10] Hbase URL `http://hbase.apache.org/`
[11] Apache lucene URL `https://lucene.apache.org/core/`
[12] Grafana URL `http://grafana.org/`
[13] Kibana URL `https://www.elastic.co/products/kibana`
[14] Rabbitmq URL `https://www.rabbitmq.com/`
[15] Tsaregorodtsev A, Bergiotti M, Brook N, Ramo A C, Castellani G, Charpentier P, Cioffi C, Closier J, Diaz R G, Kuznetsov G, Li Y Y, Nandakumar R, Parerson S, Santinelli R, Smith A C, Miguelez M S and Jumenez S G 2008 *Journal of Physics: Conference Series 119 062048* URL `http://iopscience.iop.org/1742-6596/119/6/062048`
[16] Hadoop URL `http://hadoop.apache.org/`