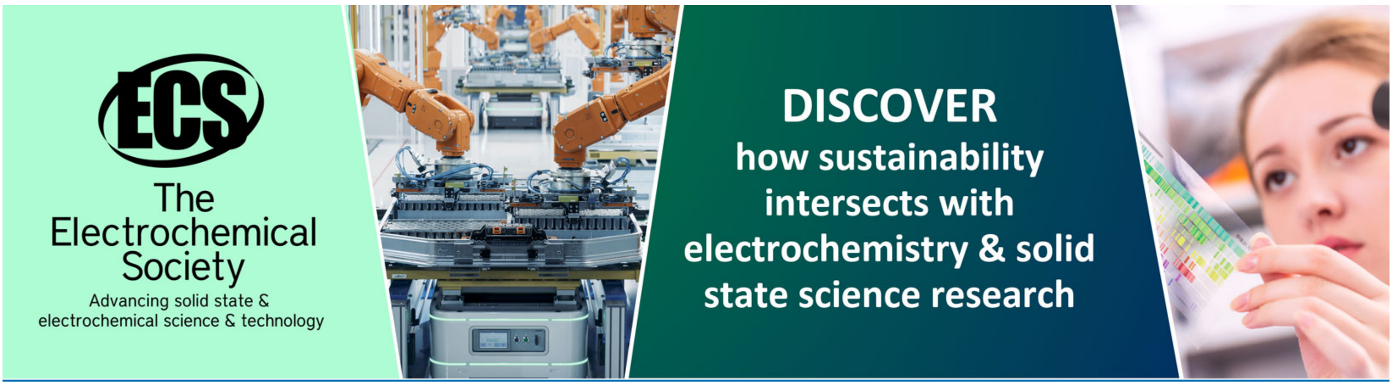
Processing of the WLCG monitoring data using NoSQL


ECS
The Electrochemical Society
Advancing solid state & electrochemical science & technology
DISCOVER how sustainability intersects with electrochemistry & solid state science research

Processing of the WLCG monitoring data using NoSQL

J Andreeva[1], A Beche[1], S Belov[2], I Dzhunov[1], I Kadochnikov[2], E Karavakis[1], P Saiz[1], J Schovancova[3] and D Tuckett[1]

[1]CERN, European Organization for Nuclear Research, Switzerland
[2]Joint Inst. for Nuclear Research, Russia
[3]Brookhaven National Laboratory, USA

Email: Edward.Karavakis@cern.ch

Abstract. The Worldwide LHC Computing Grid (WLCG) today includes more than 150 computing centres where more than 2 million jobs are being executed daily and petabytes of data are transferred between sites. Monitoring the computing activities of the LHC experiments, over such a huge heterogeneous infrastructure, is extremely demanding in terms of computation, performance and reliability. Furthermore, the generated monitoring flow is constantly increasing, which represents another challenge for the monitoring systems. While existing solutions are traditionally based on Oracle for data storage and processing, recent developments evaluate NoSQL for processing large-scale monitoring datasets. NoSQL databases are getting increasingly popular for processing datasets at the terabyte and petabyte scale using commodity hardware. In this contribution, the integration of NoSQL data processing in the Experiment Dashboard framework is described along with first experiences of using this technology for monitoring the LHC computing activities.

1. Introduction

The Large Hadron Collider (LHC) [1] experiments have been collecting data for three years. The data are distributed, processed and analysed at more than 150 grid and cloud sites in nearly 40 countries distributed around the world and contained within the Worldwide LHC Computing Grid (WLCG) collaboration [2]. The total throughput of data transfer is more than 6 GB/s and, at any given time, there are more than 250000 concurrent jobs running and in excess of two million jobs submitted on a daily basis.

This highly distributed infrastructure is also heterogeneous with multiple middleware flavours, job submission and execution frameworks, and several methods of transporting and accessing the data. The large scale activity and heterogeneity of the WLCG increases the level of complexity of the system and, thus, increases the probability of failures or inefficiencies arising. Effective monitoring is essential for providing a comprehensive way to identify and address any issues within such a highly distributed and heterogeneous infrastructure. It is also a key factor in the effective utilisation of the system.

Significant effort has been invested within the Experiment Dashboard [3][4] project to ensure effective and flexible monitoring. The Experiment Dashboard system is a Python-based framework that provides the building blocks for the development of grid monitoring applications. It provides generic solutions that cover the full range of experiments' computing activities, such as data distribution and processing over a large number of sites. These solutions are extensively used by different categories of High Energy Physics (HEP) users, ranging from daily operations to resource management, as evidenced by more than 2500 unique visitors per month. This amounts to a total of 7 TB of monitoring data since 2010.

The Experiment Dashboard must be able to handle the ever-increasing amount of monitoring data that it receives. This paper evaluates NoSQL technologies as the data storage technology for the Experiment Dashboard. Two main use-cases were evaluated:

- Applications that require grouping of data by multiple fields.
- Applications that group data by a single field.

The current work is not intended as a technology benchmark, it is purely a comparison between the performance of the existing Oracle cluster provided by CERN and potential alternative solutions. Only use-cases that are relevant to the Experiment Dashboard applications were evaluated.

2. Evaluation of NoSQL solutions within the Experiment Dashboard

The evaluation of NoSQL solutions within the Experiment Dashboard was aided by the fact that the Web User Interfaces (UIs) are decoupled from the data storage implementation, thus, it is irrelevant whether the data back-end is implemented in Oracle, MySQL, any NoSQL solution or even a simple file on the user's hard disk, provided the information that is being passed to the Web User Interface is in the predefined format.

There are many different technologies to consider as an alternative to Oracle, depending on the data schema and the use-case of the application. For data with a lot of dependencies between data objects, an alternative open source Relational Database Management System (RDBMS), such as MySQL and PostgreSQL, would be the perfect candidate. For simpler schemas with few or no dependencies between data objects, employing a NoSQL solution would be a reasonable approach. There are a wide variety of options in the NoSQL world but it was decided to evaluate two solutions; Hadoop / HBase [5] and Elasticsearch [6]. This decision was driven largely by a desire to conform to the common strategy of the CERN IT department [7][8], and to avoid relying on experimental solutions which lack official long-term support within the department.

The cluster specifications are illustrated in figure 1. Oracle runs on a high-performance physical setup and it is a shared service between all of the Experiment Dashboard applications and other IT applications, whereas Elasticsearch and Hadoop run on virtual machines. Oracle tests were performed under considerable load in a multi-user environment while Elasticsearch and Hadoop had few users. It is worth noting that the 'parallel' execution hint in Oracle was not used during the benchmark as it is not recommended by the CERN Database Administration team for use on a shared service.

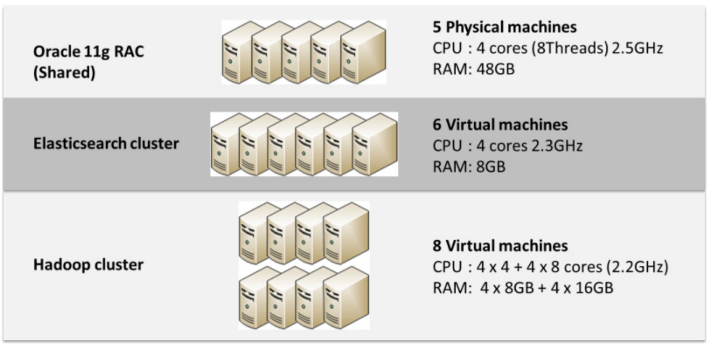


Figure 1. Cluster specifications.

2.1. Job Accounting

Job processing data are visualised in the job accounting dashboard [9]. The application is used by a wide variety of users within the ATLAS and CMS collaborations [10] to mine long-term job statistics. It offers time-series data plus filtering and grouping by multiple fields.

2.1.1. HBase. 8 million rows of data were imported into HBase (equivalent to a ~2.4 GB CSV file). The HBase key uniquely identifies a record and comprises 10 columns from the original Oracle table concatenated into a string and separated by the underscore sign (‘_’). An HBase record also consists of 52 columns that belong to one HBase column family.

The HBase key is prefixed with a date timestamp in the format of ‘*YYYYMMDD*’ because the queries are always performed on the time range and data need to be accessed in a time-ordered way. Importing and querying time series data in HBase can be quite problematic, firstly because HBase is optimised for random read access and secondly because the inserted data result in monotonically increasing row-keys, preventing the full leverage of parallelism. Many alternatives were considered in order to overcome this issue. Initially, a shard identifier was prefixed at the beginning of the key (date timestamp modulo number of region servers) but this approach becomes problematic in the case of performing queries over long time-ranges, as prefixing and un-prefixing the values is required with the results being joined in memory. Another alternative would be to replicate data proportionally to the number of columns in the HBase key but this approach was considered too expensive as data would have to be replicated 10 times.

Since the Experiment Dashboard framework is written in Python, a decision was made to use HappyBase, a high-level Python library for interacting with HBase, in preference to the native Java client. The THRIFT [11] interface was used instead of the Stargate Representational state transfer (REST) [12] interface as REST is considerably slower than THRIFT because it carries over the schema definition along with the data either in XML or JSON format. THRIFT uses a different approach; it returns a chunk of bytes that can then only be deserialised against a generated entity (based on the THRIFT IDL definition). Below the surface, HappyBase uses the Python THRIFT library to connect to HBase via the THRIFT gateway, which is included in the latest HBase releases.

The initial scanning results using the default HBase cluster configuration parameters were surprisingly slow and thus, the following optimisations were performed:

- Increased the number of Remote Procedure Call (RPC) Server instances spun up on RegionServers from 1 to 100.
- Increased the number of rows to fetch when calling next on a scanner if it is not served from memory from 1 to 1000.
- Increased the size of the memstore that will be flushed to disk if the size of the memstore exceeds this value from 128 MB to 256 MB.
- Decreased the size of the maximum HStoreFile size from 1 GB to 256 MB.
- Increased the percentage of the maximum heap (-Xmx setting) to allocate to block cache used by HFile/StoreFile from 25% to 30%.
- Increased the number of RPC Server instances spun up on the HBase Master from 25 to 100.
- Forced HBase to write the checksum into the datablock and avoid having to do the checksum seek whenever a read is performed.

A comparison between Oracle and HBase is shown in table 1. Scans over many time ranges were performed, with and without filters. The column with the grouped Oracle benchmark corresponds to the query performed on the Dashboard production server when Oracle returns back the grouped results that are then exposed in the visualisation / user-interface layer. During the course of the tests HBase Coprocessors were not used to aggregate the results, and a fair comparison between Oracle and HBase was required, the same Oracle scans were executed but this time without performing any grouping. The performance benchmark shows that for the application’s use-case, Oracle is many times faster than HBase with time-series data.

2.1.2. Elasticsearch. Elasticsearch was suggested as an alternative by the CERN Agile Infrastructure (AI) Monitoring team. It is a flexible and powerful open source, distributed real-time search and analytics engine built on top of Apache Lucene designed to provide high availability. Tests with Elasticsearch were conducted using the same amount of data as for HBase. Table 2 shows that over many different scan time ranges, with or without any filters, Elasticsearch outperforms Oracle. However, there are some limitations on Elasticsearch's ability to group data that are described in section 2.2.

Table 1. Job Accounting: Oracle VS HBase.

Scan type		**Oracle 1st hit[a] (grouping)**		**Oracle 1st hit[a] (no grouping)**		**HBase (no grouping)**	
Period	Filter	Time in secs.	Avg. rows	Time in secs.	Avg. Rows	Time in secs.	Avg. rows
1 day	0	0.031	116	0.61	10K	2.13	10K
1 week	0	0.2	807	4.54	70K	13.49	70K
1 month	0	0.956	3.6K	59.03	337K	88.26	337K
1 day	1	0.013	13	0.019	144	0.206	144
1 week	1	0.018	98	0.074	1K	0.977	1K
1 month	1	0.101	431	0.473	5.4K	2.25	5.4K
1 day	2	0.010	5	0.010	28	0.20	28
1 week	2	0.013	28	0.021	178	0.681	178
1 month	2	0.055	123	0.122	925	1.692	925

[a] "1st hit" implies scanning without using the Oracle cache.

Table 2. Job Accounting: Oracle VS Elasticsearch.

Scan type			**Oracle 1st hit[a]**	**Elasticsearch**
Period	Filter	Avg. rows	Time in secs.	Time in secs.
1 day	0	116	0.031	0.017
1 week	0	807	0.2	0.118
1 month	0	3.6K	0.956	0.138
2 months	0	7K	2.27	0.160
1 day	1	13	0.013	0.016
1 week	1	98	0.018	0.021
1 month	1	431	0.101	0.056
2 months	1	864	0.16	0.062
1 day	2	5	0.010	0.003
1 week	2	28	0.013	0.004
1 month	2	123	0.055	0.031
2 months	2	259	0.101	0.097

[a] "1st hit" implies scanning without using the Oracle cache.

2.2. WLCG Transfers

The WLCG Transfers application [13] is extensively used to monitor the data transfers occurring within the WLCG collaboration and it mainly offers two possibilities:

- Matrix statistics that allow filtering and grouping by multiple fields.
- Plot statistics that are time series data and allow filtering and grouping by multiple fields.

A small benchmarking test was performed directly on the Hadoop cluster, thus eliminating any delay of transferring data from the cluster to the server where the client was running, the results of which can be seen in table 3. It shows that querying HBase directly using a native Java client is many times faster than going through the THRIFT interface with Python.

Table 3. HBase scan comparison between native Java and THIFT.

# records	**Native Java Client**	**THRIFT Client**
69K	0.629 secs	11.04 secs

Given the poor performance on HBase with time series data, it was decided to evaluate the WLCG Transfers application's performance on Elasticsearch. 1 month's worth of data statistics (July 2013) were imported in 10 minute bins from the WLCG Transfers application, comprising 12.8 million rows and approximately 2.9 GB of CSV file.

The current version of Elasticsearch (0.90.5) does not support grouping by multiple fields for statistical aggregations, but this will be supported in version 1.0. Since the WLCG Transfers application offers grouping by multiple fields, many workarounds were investigated, resulting in the following options:

- Oracle Grouping (OG). Query using 'group by' for user selected grouping fields.
- Elasticsearch No Grouping (ENG). Query for all data and then perform the grouping in the web action from the Python side.
- Elasticsearch Index Grouping (EIG). Add a single field in index with all possible grouping fields concatenated as strings.
- Elasticsearch Query Grouping (EQG). Query to list number of distinct combinations of selected grouping fields and then query that many times, filtering by distinct combinations.

In figure 2, the comparison between Oracle 1st hit (un-cached result) and Elasticsearch for the plot and the matrix loading times can be seen. ENG is much faster than Oracle for small row counts but it won't scale with bigger row counts as large volumes of data have to be transferred from the Elasticsearch cluster to the client over the network. EIG is faster than Oracle in all cases but is inflexible. For example, if grouping by a new field is required in the future, all of the data would need to be removed and re-imported with the new grouping column in the index along with all possible grouping fields. EQG is much faster for a few distinct grouping values but won't scale if there are many distinct combinations and thus, distinct queries.

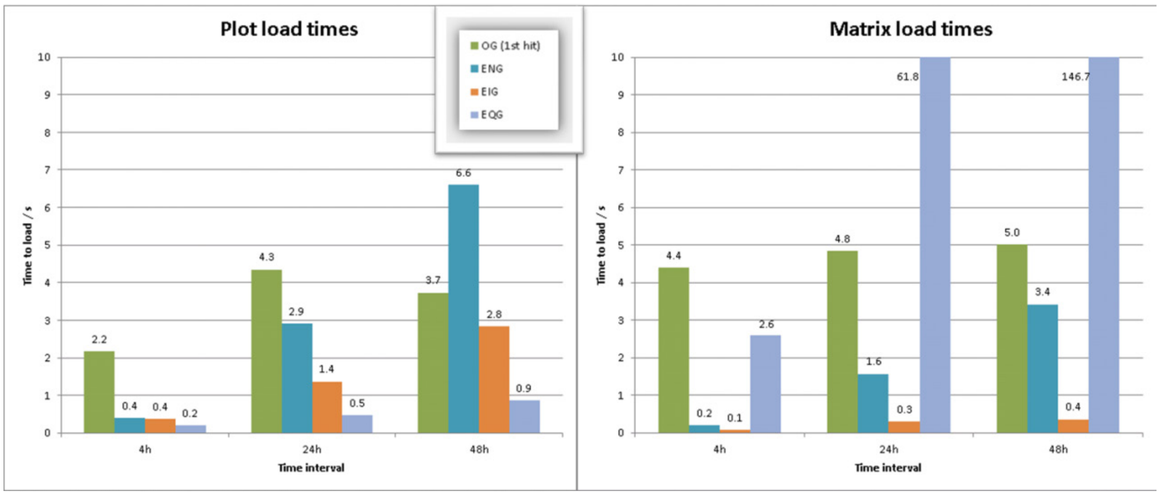


Figure 2. Comparison between Oracle 1st hit (un-cached result) and multiple grouping methods of Elasticsearch.

2.3. Site Status Board

Site Status Board [14] is a core system for monitoring the health of sites and services within the WLCG collaboration. It is widely used for the distributed computing shifts, site commissioning and testing activities. The application provides two possibilities to the users:

- Check the current status of a site or a service that supports filtering by multiple fields.
- Present historical time series data and support filtering by multiple fields and grouping by a single field.

For this application, one metric with 3 years' worth of data was imported in Elasticsearch, comprising approximately 4 million rows. The comparison between Oracle and Elasticsearch can be seen in table 4. As shown in the table, Elasticsearch outperforms Oracle by at least a factor of 3.

Table 4. Site Status Board: Oracle VS Elasticsearch.

Scan type	Avg. rows	Oracle 1st hit [a] (in secs.)	Elasticsearch (in secs.)
1 day all sites	3K	5.6	0.2
1 week all sites	29K	7.76	0.8
1 month all sites	130K	29	4
3 months all sites	440K	53	16
1 month multiple times	22K	3.3	0.6

[a] "1st hit" implies scanning without using the Oracle cache.

3. Future work

This evaluation is still very much a work in progress. For HBase, Coprocessors will be used to aggregate the data and then perform benchmarking once again, this time by querying the aggregated data directly. Additionally, Jython will be used instead of HappyBase to investigate whether scan times will be reduced by interacting with Hadoop directly through Java. Elasticsearch will be re-evaluated when version 1.0 becomes available, which will support grouping by multiple fields for statistical aggregation. Its performance on a shared physical cluster in a multi-user environment will also be evaluated.

4. Conclusions
Considering the first results of this evaluation, the conclusion was reached that there is no single solution for every possible use-case in the WLCG monitoring domain. Current experience with HBase showed poor performance with sorted time series data. Better performance was achieved with Elasticsearch than with the current Oracle cluster (comparing performance with non-cached query results). At the same time, integrating Elasticsearch with an application is straightforward only for use-cases requiring at most a single field grouping. For applications that require multi-field grouping, diverse workarounds are required and these use-cases will have to be re-evaluated using the next release of Elasticsearch.

In general, these early results are quite positive. Experiment Dashboard's design principles provide for the decoupling of the data storage implementation from the User Interface which greatly simplifies the evaluation process. For some WLCG monitoring applications, appropriate solutions were already identified. For others, more investigation is required.

Acknowledgments
The authors would like to thank the members of the CERN Agile Infrastructure Monitoring team for the fruitful collaboration and exchange of ideas and knowledge.

References
[1] Evans L and Bryant P, "*LHC Machine*", 2008, *JINST* **3** S08001 doi:10.1088/1748-0221/3/08/S08001
[2] Bird I, "*Computing for the Large Hadron Collider*", 2011, *Annual Review of Nuclear and Particle Science*, **61**:99–118 doi:10.1146/annurev-nucl-102010-130059
[3] Andreeva J et al, "*Experiment Dashboard - a generic, scalable solution for monitoring of the LHC computing activities, distributed sites and services*", 2012 *J. Phys.: Conf. Ser.* **396** 032093 doi:10.1088/1742-6596/396/3/032093
[4] Andreeva J et al, "*Experiment Dashboard for monitoring computing activities of the LHC virtual organizations*", 2010, J. Grid Comput. **8** 323-339 doi:10.1007/s10723-010-9148-x
[5] Apache Hadoop, http://hadoop.apache.org retrieved 2013-10-21
[6] Elasticsearch, http://www.elasticsearch.org retrieved 2013-10-21
[7] Andrade P et al, "*Agile Infrastructure Monitoring*", to appear in Proceedings of the 20th International Conference on Computing in High Energy and Nuclear Physics 2013
[8] Baranowski Z et al, "*Sequential Data access with Oracle and Hadoop: a performance comparison*", to appear in Proceedings of the 20th International Conference on Computing in High Energy and Nuclear Physics 2013
[9] Karavakis E et al, "*Common accounting system for monitoring the ATLAS Distributed Computing resources*", to appear in Proceedings of the 20th International Conference on Computing in High Energy and Nuclear Physics 2013
[10] Karavakis E et al, "*User-centric monitoring of the analysis and production activities within the ATLAS and CMS Virtual Organisations using the Experiment Dashboard system*", in proceedings of EGI Community Forum 2012 / EMI Second Technical Conference, 2012, PoS(EGICF12-EMITC2)110
[11] HBase THRIFT API, http://wiki.apache.org/hadoop/Hbase/ThriftApi retrieved 2014-01-08
[12] HBase REST API, http://wiki.apache.org/hadoop/Hbase/Stargate retrieved 2014-01-08
[13] Andreeva J et al, "*Designing and developing portable large-scale JavaScript web applications within the Experiment Dashboard framework*", 2012 *J. Phys.: Conf. Ser.* **396** 052069 doi:10.1088/1742-6596/396/5/052069
[14] Dzhunov I et al, "*Towards a centralized Grid Speedometer*", to appear in Proceedings of the 20th International Conference on Computing in High Energy and Nuclear Physics 2013