Monitoring of services with non-relational databases and map-reduce framework

M. Babik, F. Souto
European Organization for Nuclear Research (CERN), 1211 Geneva 23, Switzerland

Abstract. Service Availability Monitoring (SAM) is a well-established monitoring framework that performs regular measurements of the core site services and reports the corresponding availability and reliability of the Worldwide LHC Computing Grid (WLCG) infrastructure. One of the existing extensions of SAM is Site Wide Area Testing (SWAT), which gathers monitoring information from the worker nodes via instrumented jobs. This generates quite a lot of monitoring data to process, as there are several data points for every job and several million jobs are executed every day. The recent uptake of non-relational databases opens a new paradigm in the large-scale storage and distributed processing of systems with heavy read-write workloads. For SAM this brings new possibilities to improve its model, from performing aggregation of measurements to storing raw data and subsequent re-processing. Both SAM and SWAT are currently tuned to run at top performance, reaching some of the limits in storage and processing power of their existing Oracle relational database. We investigated the usability and performance of non-relational storage together with its distributed data processing capabilities. For this, several popular systems have been compared.

In this contribution we describe our investigation of the existing non-relational databases suited for monitoring systems covering Cassandra, HBase and MongoDB. Further, we present our experiences in data modeling and prototyping map-reduce algorithms focusing on the extension of the already existing availability and reliability computations. Finally, possible future directions in this area are discussed, analyzing the current deficiencies of the existing Grid monitoring systems and proposing solutions to leverage the benefits of the non-relational databases to get more scalable and flexible frameworks.

1. Introduction

Availability monitoring is one of the core activities in the operations of the WLCG infrastructure. It ensures feedback on the quality of services delivered by the sites and identifies and helps to mitigate outages caused by middleware failures. For WLCG and EGI, this is performed by the Service Availability Monitoring framework, a well established monitoring platform that computes the overall availability and reliability of services [4].

One of the existing extensions of SAM is Site Wide Area Testing (SWAT), which gathers information about worker nodes for all sites running gLite middleware[1]. The motivation for SWAT was a need to better understand the evolution of the software versions as well as the patterns of the job workflows, i.e. sequences of service instances participating in the job execution that can include resource brokers, VOMS servers, computing elements, queues and worker nodes. SWAT collects this information by running jobs that are *instrumented* with simple scripts at the beginning and the end of their execution. The information collected by those scripts is then

[1] http://glite.cern.ch/

sent via message bus to a central server [5]. The central server samples this information and computes statistics that are then presented in the SWAT web interface.

SWAT has introduced several important aspects to the subject of the service monitoring. Unlike current SAM collection mechanism which can target only specific worker nodes depending on what job queue is configured for operational jobs, SWAT is able to gather information from all existing worker nodes at a site and thus provide a better sample of the overall site usability. This is an important aspect as it's no longer possible to dedicate specific worker nodes to represent operational availability and reliability of the entire site. SWAT has also introduced passive service monitoring via instrumented jobs with the idea to replace the existing monitoring probes by sampling and extracting the information from regular batch jobs run daily by the WLCG pilot analysis frameworks. Whilst messaging was used before by SAM as a transportation layer, SWAT has shown that it can be used even for large scale collection as well as a temporary buffer and a simple aggregation layer.

Continuous running of SWAT over the years has shown also some of its limitations. The extensive number of messages transferred via message bus was reaching some of the limits in the relational database processing, which had to store, sample and process large amounts of data. With the evolution of the grid middleware, relying on gLite has also shown to be a limitation as other middleware become more frequent such as ARC, UNICORE, DesktopGrid, etc. For those, SWAT didn't offer an appropriate solution.

To address the previously mentioned limitations, we have investigated how we can improve the overall storage and processing capabilities of SWAT by introducing non-relational databases along with the map-reduce framework. In particular, we have explored if the existing aggregation algorithms can be implemented in the map-reduce way and let them process the sample information already collected by both SAM and SWAT. Since SWAT contained a very large sample of job workflows that we can now collect from the pilot and analysis frameworks, we have tried to determine if non-relational storage and processing can offer the means to aggregate, sample and extract information necessary to compute availability and reliability of sites and services.

Non-relational databases were recently introduced as an alternative way for the storage and distributed processing of large scale data sets, such as those seen in SWAT. The main characteristics of the non-relational databases are their horizontal and vertical storage scalability, support for less expressive languages and more relaxed support for transactions. In general, following nomenclature of CAP theorem[2], they favor partitioning tolerance to consistency, which makes them good potential candidates for the cases seen in SWAT. In order to perform our investigation, we had to first select potential candidates. Since there are many existing non-relational databases today[3], it was very difficult to choose the appropriate ones for our evaluation. We have determined the following characteristics that we were looking into while deciding on the candidates:

(i) Expressivity - we were primarily looking for column type and document-based stores as there was no need to represent graphs or any complex schemas.

(ii) Persistency - since our primary aim was to improve storage capabilities of SWAT we adhered strictly to persistent databases, avoiding in-memory solutions such as Redis[4].

(iii) Processing - current status and availability computation requires to process large amounts of data and therefore we were looking for databases offering, apart from storage, general data processing capabilities such as built-in map reduce or an easy integration with Hadoop.

[2] http://en.wikipedia.org/wiki/CAP_theorem

[3] http://en.wikipedia.org/wiki/Comparison_of_structured_storage_software

[4] http://redis.io/

Database	Type	License	Persistance	Transactions	Replication	High Availability
Cassandra	Column	Apache 2.0	Yes	No	Yes	Distributed
HBase	Column	Apache 2.0	Yes	Yes	Yes	Yes
MongoDB	Document	GNU AGPL 3.0	Yes	Partial	Yes	Fail-over

Table 1. A feature-matrix of chosen non-relational databases.

(iv) Community support - since our effort was limited, we were looking for big support communities and well tested systems in preference to early prototypes driven by small groups of developers.

(v) Commodity - we were primarily looking into open source systems to continue in the line of existing SAM developments and to ease the integration with existing components.

Based on these characteristics we chose *Cassandra, HBase* and *MongoDB* as initial candidates for our feasibility study. Both *Cassandra* and *HBase* are key value stores supporting persistence, replication and high availability [1, 3]. *HBase* offers in addition an easy integration with Hadoop as well as support for transactions. MongoDB is a document store supporting persistence, replication and partial atomicity [6]. Tab. 1 shows a comparison of the key technical features of the chosen databases.[5]

2. Approach

SAM map-reduce framework (SAM-MR) is an extension of SAM based on non-relational storage interface supporting distributed processing with map-reduce. Fig. 1 shows how SAM-MR extends the existing architecture and how it connects to the existing SAM components. Distributed model of SAM currently provides two deployment models: a centralized instance that computes overall availability and reliability and a set of regional instances that gather data from WLCG experiments and National Grid Initiatives (NGIs) [4]. Both models are connected to a message bus that offers efficient transport of data between all the instances. SAM-MR extends this architecture in two aspects. First, motivated by the SWAT approach, gathers environment information and job workflow directly from the worker nodes via instrumented jobs. This information is transported by messaging and stored in the non-relational database together with existing metric results coming from NGIs and experiments.

SAM-MR data analysis framework implements algorithms to re-process, merge and extract the meaningful information to compute *status and availability of services*. This is performed by a sequence of map-reduce jobs that store intermediate results in the non-relational database while querying topological and service meta-data information from already existing SAM components. The resulting *status and availabilities* computed in SAM-MR can then either be fetched directly and presented on the web interface or uploaded to the relational database to re-use existing functionality of SAM such as notifications, report generation, etc.

This approach offers several benefits to the existing SAM and SWAT implementations. It can use existing pilot frameworks to gather monitoring information and doesn't need any specific deployment on the worker nodes. It reuses messaging to buffer and aggregate data from different sources and introduces non-relational storage that offers scalable persistent store that can cope with the large amount of metric data and meta-data coming from the instrumented jobs. Existing functionality can then be instantly used by reconnecting the framework back to SAM.

[5] http://en.wikipedia.org/wiki/Comparison_of_structured_storage_software

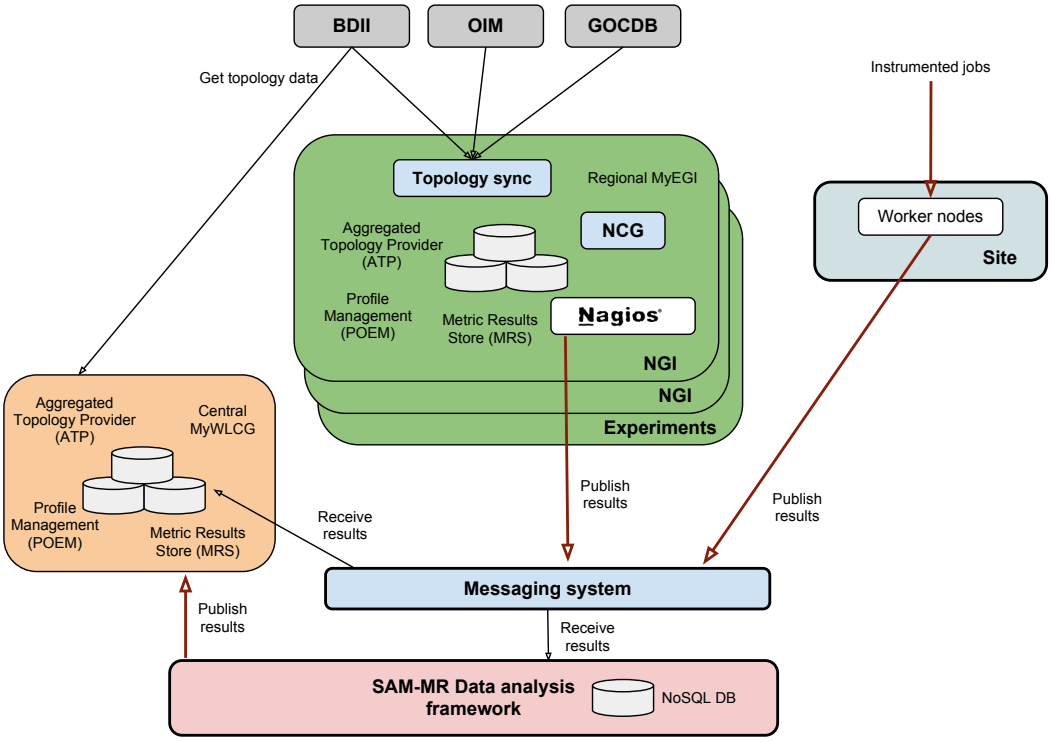

Figure 1. Overview of the SAM-MR data analysis framework.

3. Summary

SAM-MR prototype has been implemented, deployed and tested in a simulated environment reusing production data from SAM and SWAT. The prototype has shown that the proposed solution is feasible and offered an insight and first-hand experience in developing and operating non-relational databases for monitoring purposes. While prototyping SAM-MR we have evaluated non-relational databases to understand their basic development and operational characteristics as well as tried to introduce pluggable interfaces that would later enable us to run a performance comparison.

The evaluated non-relational databases have shown several strengths and weaknesses during the development and testing periods that can be summarized as follows: *HBase* has shown to be a mature solution offering very good integration with *Hadoop* (map-reduce framework) and several very useful tools from its ecosystem such as Hive, Pig and Cascading.[6] With its quite complex implementation, HBase was often difficult to use and configure. Since it's based on HDFS[7], it does have a single point of failure and doesn't scale very well to large clusters spanning hundreds of nodes. Since it has a limitation on the set of column families, it doesn't offer the expressive level of other systems[8]. *MongoDB* has an excellent documentation and a very big community and support including many tools for its operations. In addition, it offers many official drivers for different languages, which makes it very easy to develop stable clients. Apart from built-in map-reduce it also offers flexible Javascript-like query language. The only drawback we have found was its rather difficult integration with Hadoop as well as only partial

[6] http://hive.apache.org/, http://pig.apache.org/, http://www.cascading.org/
[7] http://hadoop.apache.org/hdfs/
[8] http://hbase.apache.org/book/number.of.cfs.html

support for transactions[9]. *Cassandra* was the last evaluated store that was mature and very well tested. It offers configurable CAP properties [10] and has no single point of failure. Similarly to *HBase* it is quite complex to maintain and operate especially if there is a need to balance the cluster or add new nodes. Overall, all three systems have performed very well and proved to be good candidates for future development.

We have also carried out an initial performance evaluation comparing *Cassandra* and *MongoDB*. Our testbed was composed of 6 large Amazon EC2s (7.5GB, 4 EC2 compute units 64 bits) and a read/write benchmark was run to simulate the load of the status and availability computation algorithm. We received mixed results where *Cassandra* performed better in writes while *MongoDB* performed better in reads. The evaluation has also shown that both systems would be able to cope with the load as both performed well enough to support current message throughputs in SAM and SWAT.

Since our evaluation, several perspective initiatives have emerged that can influence and further evolve monitoring with non-relational databases. Processing time series data with non-relational databases is in the scope of several open source and commercial projects including OpenTSDB[11], Cube[12], Sensu[13] and Splunk[14]. Operations and maintance has been greatly improved by commercially supported distributions such as Cloudera[15], Datastax[16] and MAPR[17]. Due to lack of effort, the continuation of SAM-MR at CERN will be primarely driven by the already established Agile Monitoring Infrastructure project, that has non-relational database as its central store [7]. Several interfaces and components proposed within Agile Monitoring Infrastructure projects were in turn motivated by the work on SAM, SWAT and SAM-MR. The prospective work for the future, as part of Agile Monitoring Infrastructure, can be a broader evaluation of the existing non-relational databases as well as understanding the requirements and needs of other monitoring frameworks to support more flexible interfaces and functionality reuse.

References

[1] Capriolo E 2011 *Cassandra High Performance Cookbook* (Packt Publishing, Limited, Community experience distilled, 2011, ISBN 978-1849515122)

[2] White T 2010 *Hadoop: The Definitive Guide* (O'Reilly Media, Definitive Guide Series, 2010, ISBN 978-1449389734)

[3] George L 2011 *HBase: The Definitive Guide* (O'Reilly Media, Definitive Guide Series, 2011, ISBN 978-1449315221)

[4] P Andrade, M Babik, K Bhatt, P Chand, D Collados, V Duggal, P Fuente, E Imamagic, P Joshi, R Kalmady, U Karnani, V Kumar, J Tarragon, W Lapka, and C Triantafyllidis 2012 *Service Availability Monitoring Framework Based On Commodity Software* (Computing in High Energy and Nuclear Physics 2012, New York)

[5] L Cons, M Paladin 2012 *The WLCG Messaging Service and its Future* (Computing in High Energy and Nuclear Physics 2012, New York)

[6] Chodorow K and Dirolf M 2010 *MongoDB: The Definitive Guide* (O'Reilly Media, O'Reilly Series, 2010, ISBN 978-1449381561)

[7] P Andrade, T Bell, J van Eldik, G McCance, B Panzer-Steindel, M Coelho dos Santos, S Traylen and U Schwickerath 2012 *Review of CERN Data Centre Infrastructure* (Computing in High Energy and Nuclear Physics 2012, New York)

[9] This has changed recently with 10gen's release of https://github.com/mongodb/mongo-hadoop.

[10] http://en.wikipedia.org/wiki/CAP_theorem

[11] http://opentsdb.net/

[12] http://square.github.com/cube/

[13] http://www.sonian.com/cloud-monitoring-sensu/

[14] http://www.splunk.com/

[15] http://www.cloudera.com

[16] http://www.datastax.com

[17] http://mapr.com