CMS users data management service integration and first experiences with its NoSQL data storage

H Riahi[1], D Spiga[2], T Boccali[3], D Ciangottini[4], M Cinquilli[2], J M Hernàndez[5], P Konstantinov[6], M Mascheroni[7] and A Santocchia[4]

[1]INFN Perugia, Via Alessandro Pascoli, 06123 Perugia, Italy
[2]European Organisation for Nuclear Research, IT Department, CH-1211 Geneva 23, Switzerland
[3]INFN Pisa, Edificio C - Via F. Buonarroti 2, 56127 Pisa, Italy
[4]Universita‘ and INFN Perugia, Via Alessandro Pascoli, 06123 Perugia, Italy
[5]CIEMAT, Av Complutense, 40, 28040 Madrid, Spain
[6]INRNE, 72, Tzarigradsko Chaussee blvd, BG-1784, Sofia, Bulgaria
[7]INFN Milano-Bicocca, Edificio U2, Piazza della Scienza, 3 - I-20126 Milano, Italy

E-mail: `Hassen.Riahi@pg.infn.it`

Abstract. The distributed data analysis workflow in CMS assumes that jobs run in a different location to where their results are finally stored. Typically the user outputs must be transferred from one site to another by a dedicated CMS service, AsyncStageOut. This new service is originally developed to address the inefficiency in using the CMS computing resources when transferring the analysis job outputs, synchronously, once they are produced in the job execution node to the remote site.

The AsyncStageOut is designed as a thin application relying only on the NoSQL database (CouchDB) as input and data storage. It has progressed from a limited prototype to a highly adaptable service which manages and monitors the whole user files steps, namely file transfer and publication.

The AsyncStageOut is integrated with the Common CMS/Atlas Analysis Framework. It foresees the management of nearly nearly 200k users' files per day of close to 1000 individual users per month with minimal delays, and providing a real time monitoring and reports to users and service operators, while being highly available. The associated data volume represents a new set of challenges in the areas of database scalability and service performance and efficiency. In this paper, we present an overview of the AsyncStageOut model and the integration strategy with the Common Analysis Framework. The motivations for using the NoSQL technology are also presented, as well as data design and the techniques used for efficient indexing and monitoring of the data. We describe deployment model for the high availability and scalability of the service. We also discuss the hardware requirements and the results achieved as they were determined by testing with actual data and realistic loads during the commissioning and the initial production phase with the Common Analysis Framework.

1. Introduction

CMS[1], the Compact Muon Solenoid experiment located at CERN (Geneva, Switzerland), has defined a model where end-user analysis jobs running on a worker node at a site, where the data reside, store and publish their outputs in the storage element of a user-designated site for later access. The AsyncStageOut (ASO)[2] is a central service that handles the transfer and publication of users outputs to the final storage element. It is designed as a thin application relying only on the NoSQL database (CouchDB)[3] as input and data storage. The highly adaptable design of the ASO has made easy its integration with the CMS/ATLAS Common Analysis Framework (CAF)[4]. In this paper, an overview of the ASO model and the integration strategy with the CAF are presented. The motivations for using the NoSQL technology as well as the deployment model for the high availability and scalability of the service are described. The hardware requirements and the results achieved are also discussed.

2. Overview of the AsyncStageOut model

The direct remote stage-out has been used in CMS since the first versions of the CMS Remote Analysis Builder (CRAB)[5]. Within this approach, the jobs execute the analysis code on the worker nodes of the site where the data resilde and copy from there each output file to a user pre-defined location at the end of the execution. This strategy has caused the most common failure mode for analysis jobs. Overall, about 25 to 30 % of the analysis jobs fail and about 30 to 50 % of the failures are due to remote stage-out, so, between 10 and 15 % of the CMS analysis jobs fail in the remote stage out. Those jobs fail at the end of the processing, so the overall CPU loss is even higher than 10-15 %.

To address the synchronous stage-out issues, it was decided to adopt an asynchronous strategy for the remote stage-out. This has required the design and development of a machinery able to stage-out the outputs locally, in a temporary area of the site storage where the code is executed, followed by a subsequent outputs harvesting step where the users outputs are copied to a remote storage using the ASO tool. It has allowed to:

- reduce the remote stage-out failure rate affecting the analysis jobs
- avoid the inefficiency of storages due to the unscheduled and potentially concurrent stage-out approach by preventing overloads
- limit the CPU wasting caused by the remote synchronous stage-out of outputs
- improve automation in users file management tasks

The ASO tool is implemented as a standalone tool with a modular architecture. Its core components are described as follows:

- The TransferDaemon module retrieves data from the *files_database* and then instantiates, for each user, a TransferWorker object which manages File Transfer Service (FTS)[6] jobs.
- The DBSPublisher retrieves the files ready for publication from the *files_database*. It calls a PublisherWorker object for each user to interact with the Dataset Bookkeeping system (DBS)[7] for the files publication. The file metadata required for the publication are retrieved from the cache area specified in the configuration.
- The Analytics component contacts the source across the dedicated plugin to communicate the status of the files management request.

The ASO Web monitoring is built on top of CouchDB. Figure 1 shows schematically the ASO architecture and the interactions between their components.

3. Integration strategy

For the distributed analysis tool sustainability, CRAB3[5] is integrated with the distributed analysis tool of ATLAS, named PanDA[8] into a Common Analysis Framework. Since the management of the users outputs is mandatory for the execution of the CMS analysis workflow, the ASO has been also integrated into CAF.

A dedicated plugin, named CMSAdderComponent, has been implemented in PanDA. It is called after the execution of the analysis job on the worker node. It pushes file documents into

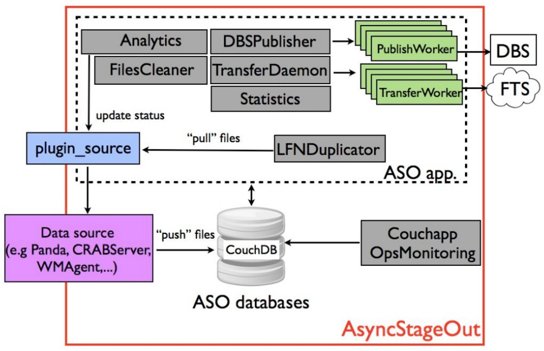

Figure 1. AsyncStageOut architecture and interactions between their components.

the ASO database, *files_database*, across the CouchDB REST interface. In addition, it uploads into a cache area the file metadata required for the publication. Then, the ASO machinery described previously can start.

To make the ASO communicating with the CAF, a dedicated plugin source has been implemented into ASO. Since the push mode is used for the communication CAF $\rightarrow$ ASO, this plugin needs just to callback the CAF to update the job status. The message provider, maintained centrally at CERN, namely ActiveMQ[9], has been used to achieve that. The interactions between CAF and ASO are shown in Figure 2.

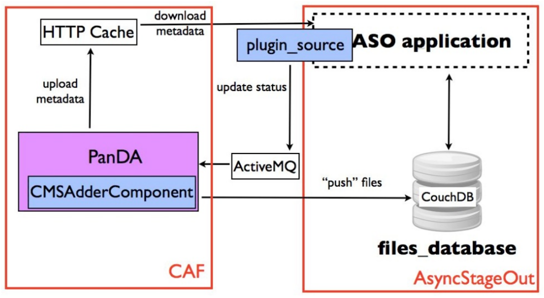

Figure 2. AsyncStageOut and CAF integration.

4. The motivations of using NoSQL

The NoSQL database, in particular CouchDB, is extensively used in the Data Management/Workload Management (DMWM) tools of CMS. This technology is used to persist the details of user outputs to manage.

The users data management system, ASO, is a new system for CMS. Its implementation was fast using this technology since no particular database design has been required. Moreover, the schema-less nature of NoSQL models satisfies the need to constantly and rapidly incorporate new types of data to enrich the new applications as the ASO.

CouchDB exposes natively a REST interface. This feature makes the communication of the ASO with external tools easy. Giving also that monitoring is an important component for CMS computing group, the CouchDB′s integrated Web server has facilitated the prototyping and implementation of the ASO Web monitoring by serving the application directly to the browser. Furthermore, no particular deployment of the monitoring application is required since it is encapsulated into CouchDB by means of CouchApp technology.

The easy replication and the integrated caching of NoSQL databases, such as CouchDB, influences highly the system scalability and availability. As shown by Figure 3, the replication represents the core of the ASO deployment model to guarantee high scalability and availability of the service.

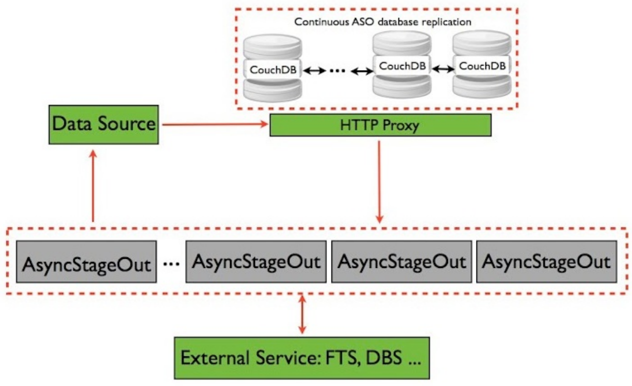

Figure 3. AsyncStageOut deployment model.

5. Scalability test

The ASO foresees the management of nearly 200k of users' analysis files with minimal delays. During the commissioning of the CAF, scalability test is performed to study the ASO response to a high workload independently of the underlying services, namely FTS and DBS. The ASO has been loaded with nearly 300k files per day.

Various scripts have been implemented for this test, in particular, a fakeFTS.pl script in PhedexLifeCycle[7] Agent to simulate the FTS server operations. One instance of ASO and CouchDB have been deployed in the same box.

The machine used in this test is a Virtual Machine with 8 VCPU, 15 GB of RAM and 200 GB of disk storage. It is managed by a Xen Hypervisor. No other machine runs in this Hypervisor to avoid resources overcommitment.

By design, ASO performances depend highly on the disk-based database, CouchDB. Within such databases, the views are always cached in the filesystem so the more RAM available the better. In addition, when a document is updated in CouchDB, the new version gets a new revision number, while the old one stays in the database. This caching and update mechanisms make CouchDB a very disk hungry system. The CPU Cores are used in particular for view building.

6. Results

In this test, we injected 100k files every 8 hours. Initially, we saw that ASO spends too much time to retrieve the views emitting the value of several parameters. So the views used by the TransferWorker and PublisherWorker components have been split into smaller ones emitting the value of only one parameter. Furthermore, views have been cached in memory via crontab jobs.

The first injection of 100k files has started at 16:00. As shown in Figure 4, the ASO has succeeded the transfer of this load in 7 hours, in particular, before the start of the next injection at 00:00.

After the second injection, the database starts to be slower retrieving views information, maybe due to the number of updates made in the file collection. Probably that is the reason for the second main dip found, at 6:00, in the temporal plot in Figure 4. In any case, ASO did not show any issue other than a reduced processing rate.

The average processing rate for this test is approximatively 9.6k files per hour with maximum of more than 20k files in an hour.

In this test, nearly 60 GB of disk storage has been consumed by the CouchDB. Only 9 % of 60 GB has been used for the storage of databases while the rest for views caching. The average CPU idle time was almost stable at 90 %. The RAM has been almost always fully consumed during the processing time of the ASO.

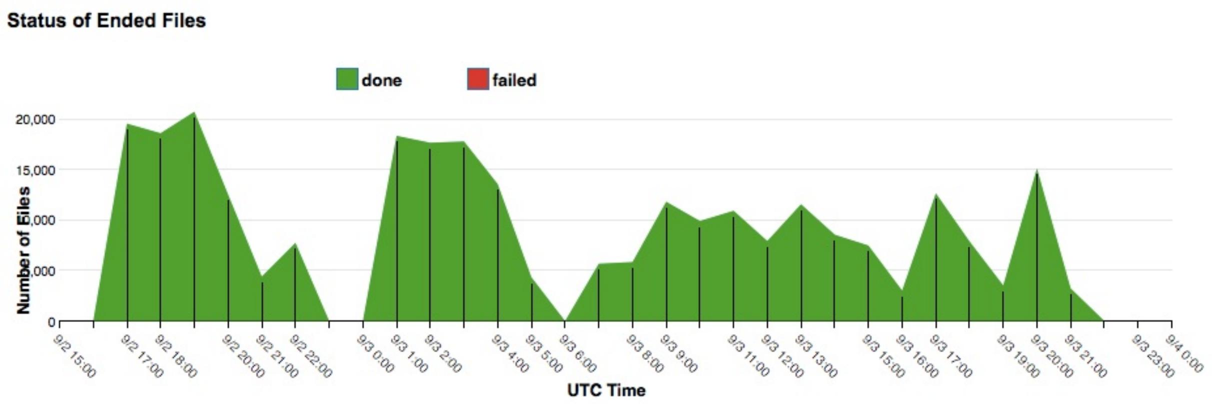

Figure 4. Files transferred per hour.

After this test, we can assert that ASO with the hardware configuration used appears to be stable at this daily workload despite some delays due to file collections update in CouchDB. Most probably these delays would be reduced by increasing the RAM dedicated to the Virtual Machine.

7. Conclusions

The ASO design based on the NoSQL database, CouchDB, has allowed an easy integration of the tool with CAF. Scale tests of the ASO independently of the underlying services have been performed and have shown satisfactory performances of the tool. In particular there are some issues that have been investigated and, so far, well understood.

Actually, the ASO can interact only with the Workload management tools such as PanDA. In the future, it will be exposed also to the users. The scalability test infrastructure is ready to be used for the highest load expected.

Acknowledgments

This work has been partially funded under contract 20108T4XTM of ”Programmi di Ricerca Scientifica di Rilevante Interesse Nazionale (Italy)”.

References

[1] CMS Collaboration, Adolphi R., *et al.* 2008 The CMS experiment at the CERN LHC, *JINST* 0803 S08004.
[2] H Riahi 2012 Design optimization of the Grid data analysis workflow in CMS, *Ph.D Thesis, University of Perugia, Italy.*
[3] Apache CouchDB database `http://couchdb.apache.org`.
[4] M Girone, *et al.* The Common Analysis Framework Project, in these proceedings.
[5] M Cinquilli, *et al.* 2012 CRAB3: Establishing a new generation of services for distributed analysis at CMS, *J. Phys.: Conf. Ser.* **396** 032026.
[6] A Frohner, *et al.* 2012 Data management in EGEE, *J. Phys.: Conf. Ser.* **219** 062012.
[7] T Wildish, *et al.* Integration and validation testing for PhEDEx, DBS and DAS with the PhEDEx LifeCycle agent, in these proceedings.
[8] T Maeno 2008 PanDA: distributed production and distributed analysis system for ATLAS, *J. Phys.: Conf. Ser.* **119** 062036.
[9] ActiveMQ `http://activemq.apache.org`.