AsyncStageOut: Distributed user data management for CMS Analysis

H Riahi[1], T Wildish[2], D Ciangottini[3], J M Hernández[4], J Andreeva[1], J Balcas[5], E Karavakis[1], M Mascheroni[6], A J Tanasijczuk[7], E W Vaandering[8]; on behalf of the CMS Collaboration

[1]European Organization for Nuclear Research, IT Department, CH-1211 Geneva 23, Switzerland
[2]Princeton University, Princeton, NJ 08544 USA
[3]Università and INFN Perugia, Via Alessandro Pascoli, 06123 Perugia, Italy
[4]CIEMAT, Madrid 28040, Spain
[5]DiSCC, Vilnius University, LT-01513 Vilnius, Lithuania
[6]INFN Milano-Bicocca, Piazza della Scienza, 3 - I-20126 Milano, Italy
[7]University of California, San Diego, La Jolla, CA 92093-0354, USA
[8]Fermi National Laboratory, Batavia, IL 60510, USA

E-mail: Hassen.Riahi@cern.ch, Tony.Wildish@cern.ch, Diego.Ciangottini@pg.infn.it

Abstract. AsyncStageOut (ASO) is a new component of the distributed data analysis system of CMS, CRAB, designed for managing users' data. It addresses a major weakness of the previous model, namely that mass storage of output data was part of the job execution resulting in inefficient use of job slots and an unacceptable failure rate at the end of the jobs. ASO foresees the management of up to 400k files per day of various sizes, spread worldwide across more than 60 sites. It must handle up to 1000 individual users per month, and work with minimal delay. This creates challenging requirements for system scalability, performance and monitoring. ASO uses FTS to schedule and execute the transfers between the storage elements of the source and destination sites. It has evolved from a limited prototype to a highly adaptable service, which manages and monitors the user file placement and bookkeeping. To ensure system scalability and data monitoring, it employs new technologies such as a NoSQL database and re-uses existing components of PhEDEx and the FTS Dashboard. We present the asynchronous stage-out strategy and the architecture of the solution we implemented to deal with those issues and challenges. The deployment model for the high availability and scalability of the service is discussed. The performance of the system during the commissioning and the first phase of production are also shown, along with results from simulations designed to explore the limits of scalability.

1. Introduction

The analysis model of CMS[1], the Compact Muon Solenoid experiment located at CERN (Geneva, Switzerland), foresees activities driven by data location: data are distributed over many computing centers according to CMS data placement policies and the users' processing takes place at the sites where data are located. The users' results are stored and made available for later access in the storage element of the user-designated site. Figure 1 shows schematically the distributed data analysis model in CMS. The distributed analysis represents a complex task in a chaotic environment with low latency requirement. It involves more than 60 computing centers, geographically distributed all around the world used by more than 1000 individual users per month. The average load reached in Run 1 was 20k concurrently running jobs and 200k completed jobs per day. Each job produces typically one output and one log file. This load is expected to double in Run 2 along with the increase of the data rate. This creates challenging requirements for the data management system scalability, performance and monitoring during the LHC Run2.

AsyncStageOut (ASO) is a central service handling the transfer and bookkeeping of users' outputs to the final storage element. It is originally designed as a thin application relying only on the NoSQL database (CouchDB)[2] as the input and data storage. The highly adaptable design of the ASO has made it easy its evolution to deal with those issues and challenges.

In this paper, we present the asynchronous stage-out strategy and the architecture of the solution we implemented. The deployment model for the high availability and scalability of the service is discussed. The performance of the system during the readiness challenge of CMS for LHC Run2, namely Computing, Software, and Analysis (CSA14), and the first phase of production are also shown, along with results from simulations designed to explore the limits of scalability.

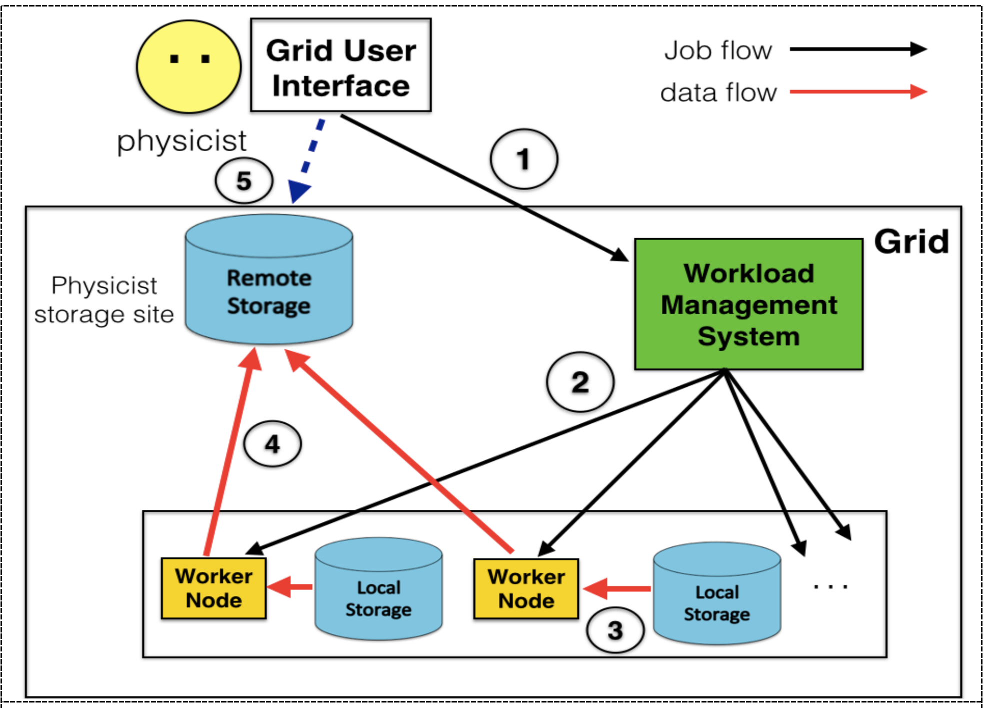

Figure 1. Distributed data analysis model in CMS

2. Overview of the AsyncStageOut model

The direct remote stage-out of users' output files has been used in CMS since the first versions of the CMS Remote Analysis Builder (CRAB)[3]. As shown in figure 1, within this approach, the jobs execute the analysis code on the worker nodes (WN) of the site where the data reside and then copy the produced output files from the WNs to the user pre-defined location. This strategy had lead to 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 after multiple retries, so the overall CPU loss is even higher than 10-15 %. Moreover, this architecture of unscheduled remote transfers often results in Distributed Denial of Service (DDoS) of the CMS storage systems.

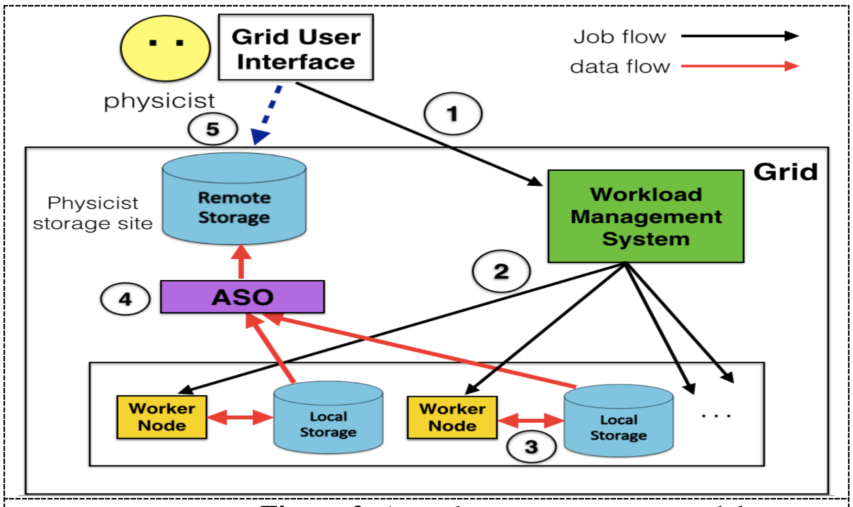

Figure 2. Asynchronous stage-out model

To overcome these issues, an asynchronous strategy has been embraced for the remote stage-out. This has required the design and implementation of an architecture able to stage-out the outputs locally, in a temporary storage area of the site where the code is executed, followed by a subsequent outputs harvesting step where the user's outputs are copied to his home site. It enables to:

- reduce the remote stage-out failure-rate hitting the analysis jobs;
- avoid storage-related problem 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 the management of users' files.

The asynchronous stage-out model is shown in figure 2.

3. Architecture

ASO is implemented as a standalone tool with modular application architecture. It relies on CouchDB as the input and data storage. ASO exposes the native REST interface of CouchDB to communicate with external tools. The ASO Web monitoring is built on top of CouchDB. An overview of ASO architecture is shown in figure 3.

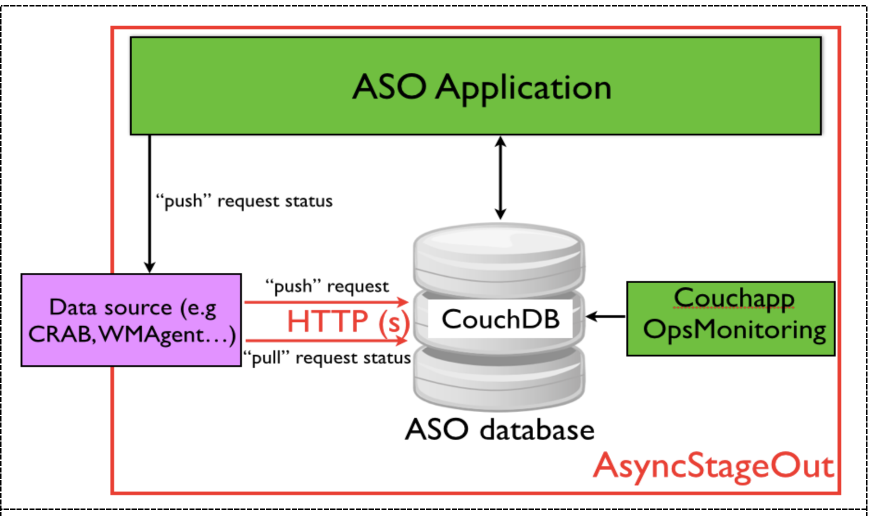

Figure 3. ASO architecture overview

3.1. CouchDB
The REST interface of CouchDB that ASO exposes has made it easy the usage of the application by new clients and so the evolution of the tool to support new workload management systems.

To expose the ASO database to the Grid, a custom handler in CouchDB is implemented to support X.509 proxy certificate for users authentication. For the authorization, CouchDB provides a database function called "validate_doc_update" to validate the requests to create or update documents in the database. So a dedicated "validate_doc_update" function is implemented in the ASO database to authorize users' operations.

ASO is a new system for CMS. Its implementation as well as its evolution was fast using CouchDB since no particular database design was required. The schema-less model of this technology satisfies the need to constantly and rapidly incorporate new types of data to enrich the application.

Given the complexity of CMS computing operations, the monitoring represents a crucial task for the computing group. The integrated Web server of CouchDB has facilitated the prototyping and implementation of the ASO Web monitoring by serving the application directly to the browser. The deployment of the monitoring application does not require any particular operation effort since it is encapsulated into CouchDB by means of CouchApp.

Moreover, the easy replication and the integrated caching of CouchDB make ASO a highly scalable and available system.

3.2. Application components
The Core ASO components are described below:

- Transfer: this module is tasked to manage files transfer relying on the File Transfer Service (FTS3)[4]. It is divided into 3 sub-modules communicating via the file system:
    - Submitter: it retrieves from the database the transfer request and then submits it to the Grid;
    - PhEDex Monitor: it tracks the transfer requests in the Grid. This component is the same used in CMS official data movement tool, PhEDEx[5];
    - Reporter: it updates the request status in the database once completed.
- Retry Manager: it implements the retry algorithm for failed transfers. In particular, a custom retry algorithm could be implemented and plugged into the system.
- Cleaner: it removes the files from the temporary storage area of the site, where the job ran, once the transfer to user's site has been completed.
- Publication: it publishes the user's files in the CMS Catalogue, named Dataset Bookkeeping System (DBS), once transferred to the permanent storage for further analysis by the collaboration.
- Analytics: it generates reports from the meta-data of completed requests for later analysis and stores them in the database.

3.3. Monitoring
The Worldwide LHC Computing Grid (WLCG) relies on FTS3 as the data movement middleware for moving sets of files from one site to another. The FTS Dashboard[6] provides a cross-experiment monitoring and statistics platform for the WLCG data movement. Within FTS3, enabling the monitoring and statistics of a new data movement tool becomes trivial. FTS3 provides a meta-field in the transfer request that could be used by experiments' data movement tools to identify later in the FTS Dashboard the transfers per submission tool.

CMS Task monitoring Dashboard[7] provides a user-friendly Web interface to the analysis users for the monitoring of their jobs. To provide a real-time monitoring of the transfers of files produced by the jobs, new APIs have been implemented in the backend of Task monitoring and FTS Dashboard to retrieve transfers status per job. The status per job is shown then in the Web interface of the Task monitoring Dashboard.

The detailed architecture of the AsyncStageOut is shown figure 4.

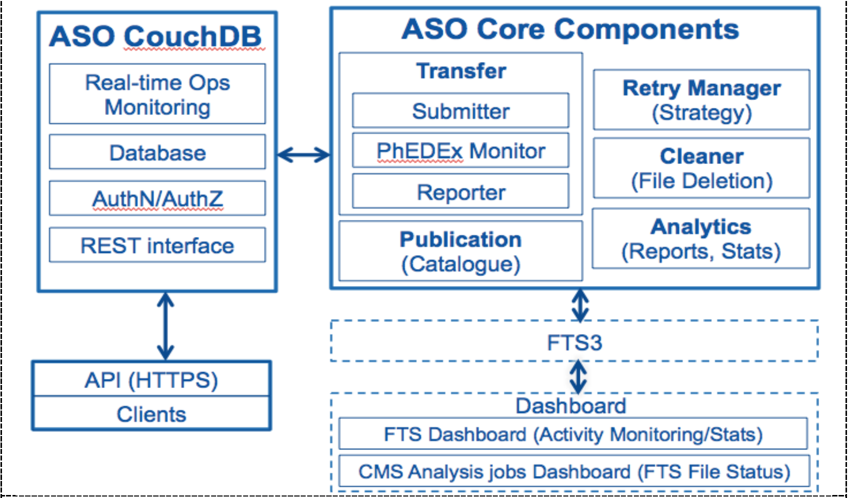

Figure 4. ASO architecture

4. Integration and deployment models

ASO has shown that it is a highly versatile tool by being integrated with various Grid workload management tools, namely CRAB2, WMAgent, and PanDA. CRAB3 is the new distributed data analysis system of CMS. The design integration within CRAB3 is shown in figure 5. The user's analysis workflows are split into chunks, namely jobs, and then submitted to the Grid by CRAB3. Then once the execution of the user's code in the WN is completed, transfer requests of user's produced files are uploaded into the ASO database. At this point, ASO starts the management of the new requests. Asynchronously, the user can retrieve the status of their files from the REST interface of the ASO database via CRAB3.

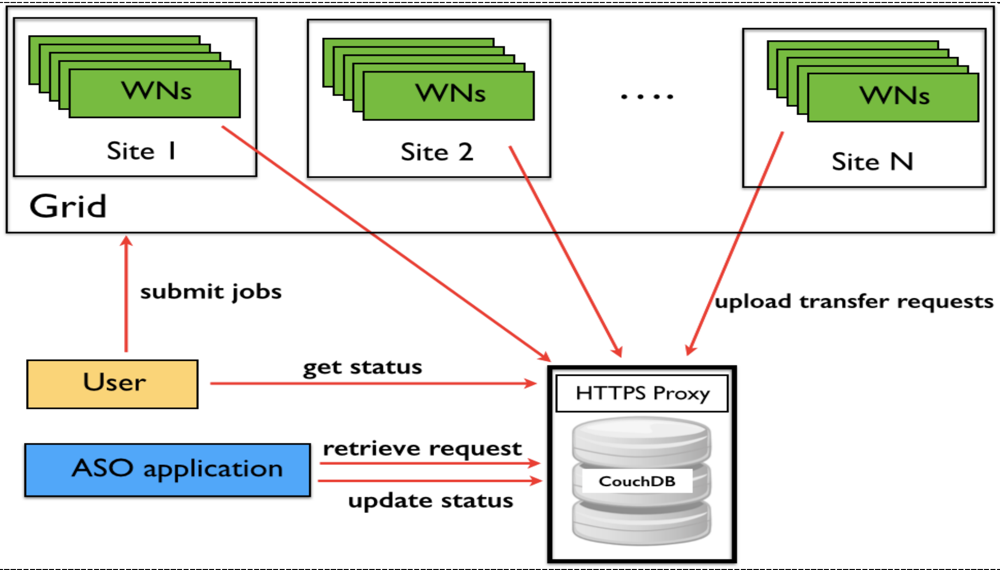

Figure 5. ASO integration in the distributed data analysis system of CMS

Currently only one ASO application and one ASO database instances are deployed in production. However as shown in figure 6, the ASO application is highly distributed so several application instances could be deployed over the same database. In addition, within CouchDB 2.0, the database can be natively distributed across many servers, adding horizontal scaling capability to this technology.

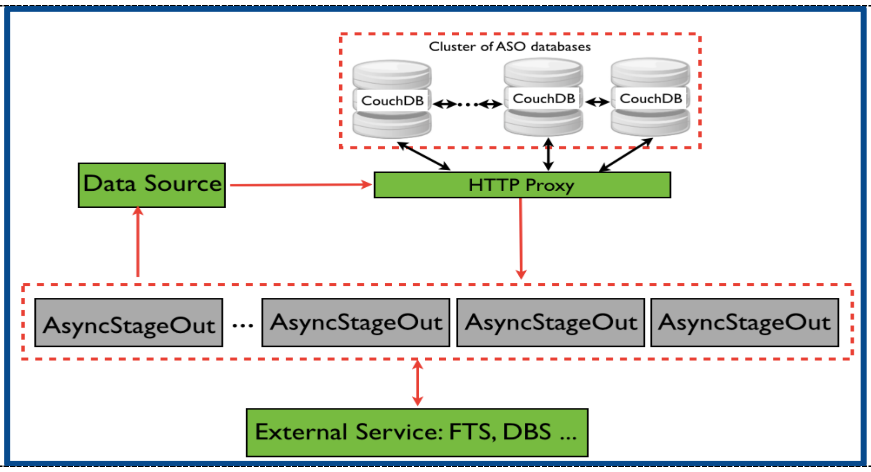

Figure 6. Highly scalable ASO deployment model within CouchDB2.0

5. Results

5.1. CSA14

In 2014, CMS ran a computing exercise, 'CSA14', to ensure readiness for LHC Run 2. The goal of this exercise was to expose problems that might prevent CMS from successfully and speedily analyze the first data from LHC Run 2. In particular for the CMS Analysis activity, the goal was to test CRAB3 analysis at a scale as close as possible to the scale necessary for LHC Run-2, and to explore its limits and weaknesses in time to apply corrective measures before the run starts. The CSA14 ran from July to September 2014.

The aim of CSA14 was to explore a higher scale of concurrently running analysis jobs than in Run 1 (20k). Figure 7 shows the volume of data transferred and number of files managed by ASO for one month during the CSA14. In particular, during this month, more than 300k files per day were produced and managed successfully by ASO during a peak day of nearly 30k concurrently running jobs.

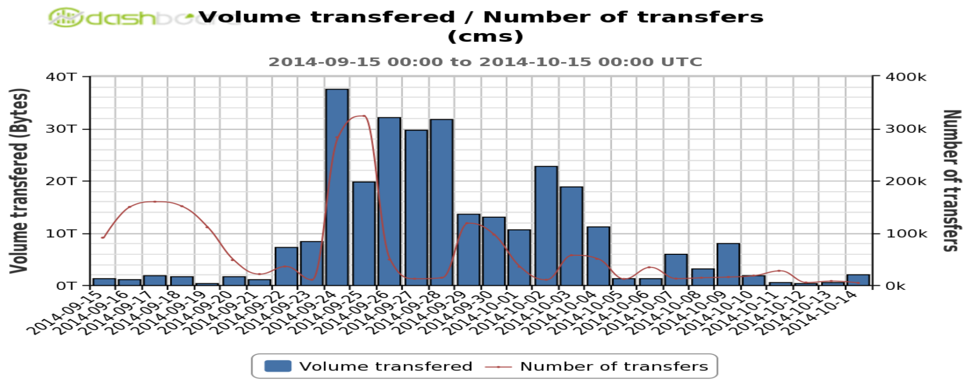

Figure 7. Volume transferred and number of files managed by ASO for 1 month during CSA14 exercise

5.2. Production

ASO was deployed in production in June 2014 and it has not shown any particular issues so far. As shown in figure 8, during the last six months, ASO has transferred more than 2 PB of data in 7 M files.

It has succeeded to manage peaks of nearly 700k transfers per week, which represents a third of the design load.

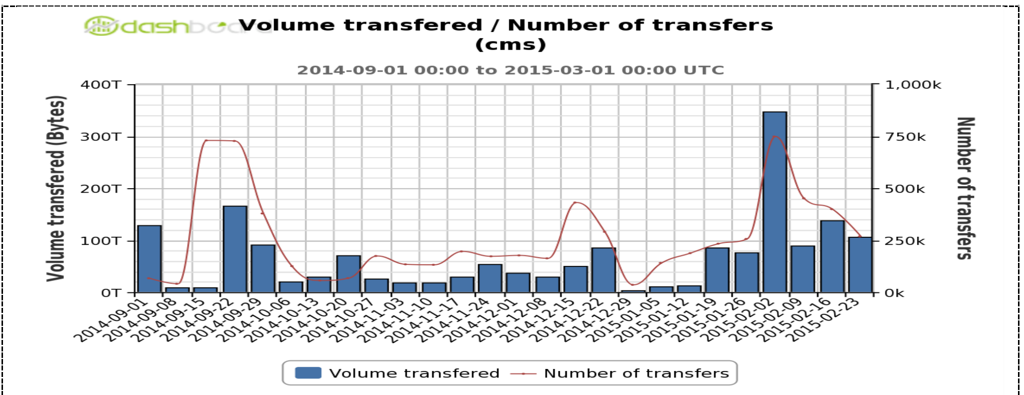

Figure 8. Volume transferred and number of ASO transfers during the last six months of production

5.3. *Scalability tests*

The ASO design load of users' analysis files is around 200k per day requiring minimal transfer delays. A given user can also ask the transfer of the log archive produced by each of the analysis jobs. Estimating that only half of users request also the transfer of the log file brings the design load to 300k files per day. Internal scalability tests have been performed to study the ASO response to a realistic workload without involving the underlying services (FTS and DBS). PhEDEx LifeCycle[8] Agent has been used to simulate real data transfers and ASO has been fed with fake requests.

Two different sets of tests have been performed, studying the capability of ASO to manage two high load scenarios. The first one was performed comparing two different versions of CouchDB, namely 1.1 and 1.6, in a 2 times design scale scenario (600k files per day). In the second test, only the newer version of CouchDB has been used in a 4 times design scale scenario (1.2M files per day).

The first test has been performed using the following setup:

- 200 parallel users;
- 100 files per FTS job;
- 60 sites randomly selected for source and destination;
- 20 seconds average file transfer times;
- 1-minute injection interval.

In figure 9, the results for a 16 hours injection of 600k files daily load using Couch 1.1 is shown. At this load the expected number of completed files per hour is around 25k; the figure shows a stable behavior of ASO under this load with no relevant issue during the whole injection time. The same test has been performed with Couch 1.6 and the results are similar.

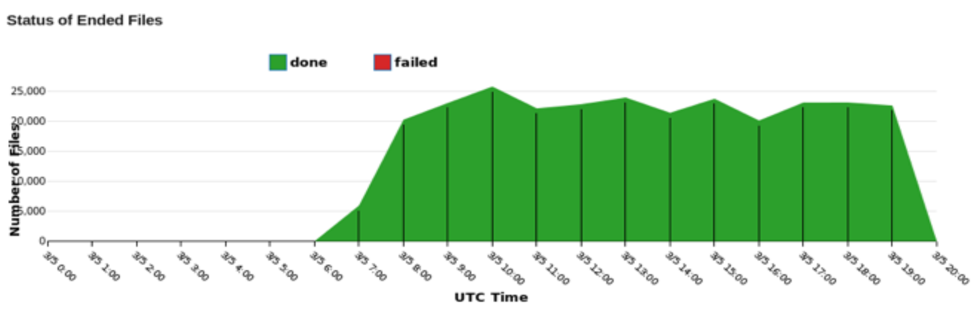

Figure 9. Performance of CouchDB 1.1 managing 2 times the design scale

Both versions of CouchDB seem to provide a stable performance in two times design scale scenario. Next we proceeded to evaluate Couch 1.6 performance under 1.2M files daily load.

In figure 10, the results for 12 hours injection of 1.2M files per day load using CouchDB 1.6 are presented. At this scale, the expected number of completed files per hour is around 50k; the figure shows some dips that indicate ASO limits under this load. In correspondence of those dips, the PhEDex Monitor shows core dump errors that suggested the necessity for tuning the parameters of this component. Actually, it succeeds to retrieve the status of FTS jobs only after several retries delaying the report of the actual status of the transfers to the database. In figure 11, the same test has been performed with CouchDB 1.6 for 6 hours with a less aggressive number of parallel processes in the PhEDex Monitor component for retrieving the status of FTS jobs. The situation was improved: no core dump error appeared but it was not enough to complete the files processing with a reasonable amount of delay. Mainly, in this case, CouchDB was not fast enough in processing data to cope with this load.

We can conclude that ASO with a proper tuning can manage almost 3 times the design scale without particular issues.

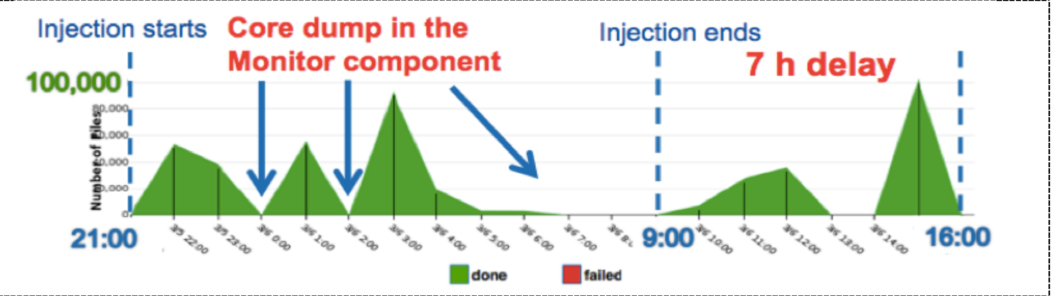

Figure 10. ASO limits for the management of 4 times the design load

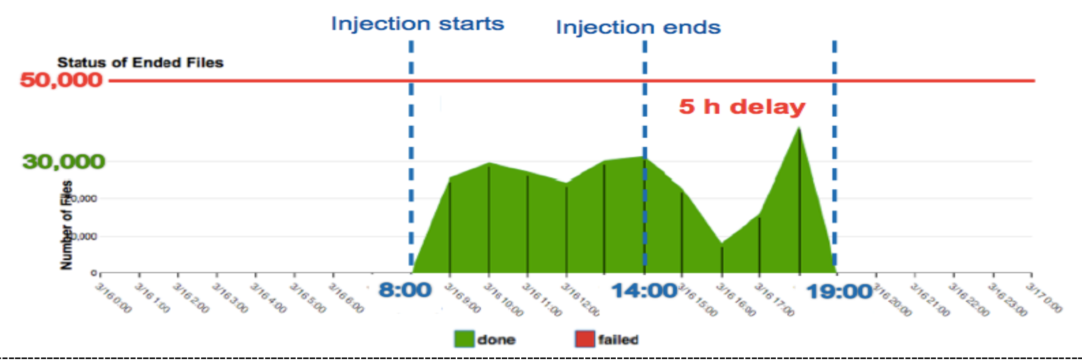

Figure 11. ASO performance for the management of 4 times the design load with tuned parameters

6. Conclusions

ASO has evolved from a limited prototype to a highly modular, versatile and scalable service for the distributed user data management of CMS Analysis for LHC Run 2. It has shown good performance during CSA14 challenge as well as during the first phase of production. The results from simulations have shown that ASO can manage 2 times the design load while the management of 4 times requires accurate tune of ASO and system parameters.

The re-use of design and components from PhEDEx points the way to a more modular architecture for data-management tools in CMS. This will lead to longer-term maintainability, performance and adaptability.

Acknowledgments
This work has been partially funded under program PRIN 20108T4XTM.

References

[1] CMS Collaboration, Adolphi R, *et al.* 2008 The CMS experiment at the CERN LHC. JINST **0803** S08004

[2] Apache CouchDB database: http://couchdb.apache.org

[3] Mascheroni M *et al.* CMS Distributed Data Analysis with CRAB3. Proceedings of CHEP'15 to be published by IOP *J. Phys. Conf. Ser.*

[4] Ayllon A A *et al.* 2014 FTS3: New Data Movement Service for WLCG. *J. Phys.: Conf. Ser.* **513** 032081

[5] Wildish T *et al.* Virtual Circuits in PhEDEx, an update from the ANSE project. Proceedings of CHEP'15 to be published by IOP *J. Phys. Conf. Ser.*

[6] Riahi H *et al.* FTS3: Quantitative Monitoring. Proceedings of CHEP'15 to be published by IOP *J. Phys. Conf. Ser.*

[7] Karavakis E *et al.* 2010 CMS Dashboard Task Monitoring: A user-centric monitoring view. *J. Phys.: Conf. Ser.* **219** 072038

[8] Boeser C *et al.* 2014 Integration and validation testing for PhEDEx, DBS and DAS with the PhEDEx LifeCycle agent. *J. Phys.: Conf. Ser.* **513** 062051