XRootD popularity on hadoop clusters

Marco Meoni¹, Tommaso Boccali², Nicolò Magini³, Luca Menichetti⁴ and Domenico Giordano⁴ for the CMS collaboration

¹ University of Pisa & INFN, Italy
² INFN, Pisa, Italy
³ Fermilab, Batavia IL, USA
⁴ CERN, 1211 Geneva 23, Switzerland

E-mail: marco.meoni@cern.ch, tommaso.boccali@pi.infn.it, nicolo.magini@cern.ch, luca.menichetti@cern.ch, domenico.giordano@cern.ch

Abstract. Performance data and metadata of the computing operations at the CMS experiment are collected through a distributed monitoring infrastructure, currently relying on a traditional Oracle database system. This paper shows how to harness Big Data architectures in order to improve the throughput and the efficiency of such monitoring. A large set of operational data - user activities, job submissions, resources, file transfers, site efficiencies, software releases, network traffic, machine logs - is being injected into a readily available Hadoop cluster, via several data streamers. The collected metadata is further organized running fast arbitrary queries; this offers the ability to test several Map&Reduce-based frameworks and measure the system speed-up when compared to the original database infrastructure. By leveraging a quality Hadoop data store and enabling an analytics framework on top, it is possible to design a mining platform to predict dataset popularity and discover patterns and correlations.

1. Introduction
The CMS [1] experiment has recorded huge volumes of data from collisions at the Large Hadron Collider, and produced a similar amount of simulated data. CMS data are recorded as files, which are grouped into datasets by physics content, e.g. if they were produced at the LHC accelerator in the same running period, or simulated with a given set of parameters. The datasets are replicated in multiple copies and distributed among the computing centers of the Worldwide LHC Computing Grid [2] for further processing and for analysis: currently more than 60 PB are placed on disk and more than 100 PB on tape.

Since 2011, the CMS Popularity [3] service has been monitoring the access to datasets. The collected metrics are used to optimize data distribution, ensuring that the most used data are replicated and accessible on the constrained disk resources, while cleaning up replicas of unused or less used data.

Both processing jobs and storage systems have been instrumented to report file accesses back to the monitoring system. In particular, storage servers accessed through the XRootD [14] protocol (such as the EOS [4] storage at CERN, or any storage accessed remotely through the AAA federation [14]), send UDP monitoring packages to report any file access. These packages are collected by the GLED collectors [5] and relayed to the CERN IT centralized monitoring infrastructure [12] through a message bus.
The file access metrics are collected in an Oracle database at CERN, and aggregated for different monitoring services: measuring global data transfer rates through XRootD, as well as providing dataset popularity statistics. The jobs running on Oracle used to aggregate popularity time series have been in production for several years, but are now showing a non-optimal scaling with the increase of monitoring data.

In order to overcome this limitation, in 2015 CMS and CERN IT started to migrate the popularity aggregations to the Hadoop [9] cluster at CERN, using it as a testbed in order to quantify future user activity and predict dataset popularity. This work constitutes the basis to exploit Big Data architectures in order to discover patterns and correlations that can improve the throughput and the efficiency of the CMS computing model.

We proceed by collecting XRootD metrics from the message bus in Hadoop, and benchmarking their aggregation at the level of dataset popularity, thus proving how dashboard and monitoring systems can benefit from Hadoop parallelism. Big Data architectures scale with data volume and make reprocessing any user-defined time interval very effective. The entire set of existing Oracle queries is replicated in the Hadoop data store and result validation is performed accordingly.

The aforementioned results constitute the set of features on top of which to build a future mining platform for the prediction of the popularity of a new dataset, the best location for replicas or the proper amount of CPU and storage in future time-frames. Machine Learning (ML) techniques applied to Big Data architectures are pioneering the study of correlations between aggregated data and seek for patterns in the CMS computing ecosystem. Examples of this kind are primarily represented by operational information like file access statistics or dataset attributes, which are organized in samples suitable for feeding several binary classifiers.

2. System Environment

The Hadoop environment is a highly flexible ecosystem in terms of both hardware requirements and software components. CERN IT provides a set of clusters using commodity hardware, where different open-source tools are installed; accounts for free until a certain quota of disk space or execution time is reached are provided. XRootD popularity is hosted on one cluster, called analytix, composed by 38 heterogeneous nodes, with 2 TB of total memory, around 1000 virtual cores (number of CPU cores allocated to run jobs) and nearly 2 PB of raw storage. The cluster currently runs CentOS and Scientific Linux 6.

Spark and Pig are provided as user level tools. The latter is an engine that allows users to write Map&Reduce programs though a simple ad-hoc scripting language, avoiding Java code and packing Jars. Spark, instead, is an independent framework in Scala (with Python and Java APIs) that, compared to Map&Reduce, tends to cache data in memory in order to reuse them as much as possible, minimizing IO disk operations. In either case the storage layer is the Hadoop Distributed File System (HDFS), where data are written in blocks and replicated 3 times in order to obtain high availability (allowing parallel reads and write) and fault tolerance.

Other Hadoop compliant tools needed in the new XRootD Popularity calculation process, and in particular those involved in the data ingestion and presentation part, were not available on the cluster. Flume, for instance, is configured by the CERN IT monitoring team [12] in external nodes; it is a streaming tool that ships data from origin sources into HDFS. All the components are installed via Puppet (the standard software distribution tool at CERN IT) via modules written in order to install programs with all their dependencies and configurations; such modules are provided and maintained by different teams or users in need of a particular application (CMS wrote new ones where needed). Puppet deploys the CMS setup in virtual machines running in OpenStack, where the environment is already tuned to user needs. In particular, we used Maven (build manager for Java and Scala projects) and two web notebooks used for inline data inspection and raw data pre-processing with Spark, Jupyter and Zeppelin. In
future iterations this part will be replaced by the SWAN [13] service (as soon as Spark notebooks in cluster mode will be supported).

3. Data Ingestion
The current CMS distributed computing monitoring is modeled as a system able to integrate existing data sources (figure 1), classify and monitor missing sources of information, provide long-term storage of monitoring data and, ultimately, develop analysis tools.

![Hadoop Data Ingestion](image)

We refer to the data coming from Flume or Sqoop as the “raw” data. In the case of Flume, the format is JSON. Passing through a chain of components, the message of interest is embedded within an object, including transfer information not important for the next steps. On the other hand, with Sqoop, we can obtain a CSV file with a fixed schema and no field to overlook. Occasionally we may want to simplify data, changing format or deleting useless fields, in order to obtain an intermediate result that suits best. A valid alternative, leveraging Spark features, is to recreate this subset of interest caching our data inside the Spark job; this will avoid consuming space in HDFS in case there is no need to store it permanently and will increase the overall performance.

Raw data is then organized into metadata suitable for data-mining. Information of this type contains physics content (lepton, jets...), processing workflows (Monte Carlo, reconstruction, AOD...), software version, physics working groups requesting the data, log statistics (number of bytes, accesses, CPU time, individual users etc.), data provenance (SiteDB, dashboard, CRAB, PhEDEx...). Other categories can be derived, like user “social” activity (field of interests, past activities on the Grid...) or seasonality of the datasets (proximity to conferences or major seminars).

Each source has the potential to disclose innovative clues in order to predict future user activity and optimize resource allocation. They are currently spread out in several Oracle-based data sources, which we need to transfer, aggregate, map and reduce for data analytics leveraging the Hadoop cluster. Among them:

- AAA, the XRootD-based [14] federation service which holds WAN-access file information from local and remote storage systems.
- PhEDEx [15], the transfer management database which holds data-placement and data-movement catalog information.
- CRAB [6], the CMS distributed analysis tools for job submission, which holds information about who submits which type of job, where, and what data they run on.
- The Popularity DB and Dashboard, which integrate dataset-level information, hiding individual backstage sources.

All the data are stored into the Hadoop cluster at CERN, provided by the IT Department. Figure 2 outlines the overall schema:

- A batch-serving layer that stores a constantly growing dataset and exposes the ability to compute arbitrary functions and use indexing techniques to make them efficiently queryable.
- An analytic layer able to perform mining on fresh data with incremental algorithms to compensate for batch-processing latency.
- An active-reaction layer that uses the analytics layer as input and adopts a classical pattern matching approach to promptly detect errors and failures on the stream of monitoring events.

![Figure 2. Overall schema of the Dataset Popularity project at CMS](image)

At the moment, data starting from March 2015 have been injected to the Hadoop cluster for analysis. They are collected on a daily basis and transformed via Map&Reduce (MR) operations into a format that is suitable to ML studies. Per each dataset, four different metrics are selected on a daily basis: number of accesses, number of users, total number of CPU hours spent, number of bytes read. These attributes are normalized over the full number of datasets. The metrics also contain information about hostname and username for each operation on the dataset, which can be used should ML studies lead to introduce CPU quotas (adapt user priority versus consumed CPU time) or site ranking (nodes serving more dataset requests may be the first ones to satisfy future resource pledges).

Several streamers transfer to Hadoop File System (HDFS) some 10 GB of Oracle data every day, which includes between 100k and 2M entries in several formats (CSV, JSON, Parquet, AVRO). The Oracle RAC back-end features 10 GB/s links, 24 TB SATA disks with 2xSAS loop and 512 GB SSD cache per controller. The Hadoop cluster comprises 54 active nodes equipped with 4/8 cores Intel(R) Xeon(R) CPU (276 vcores), ~2 PB SATA3 HDD and 814.78 GB of total memory.
4. Data Validation

XRootD metadata is aggregated on a daily basis following the Map&Reduce programming paradigm applicable to Big Data architectures.

A number of Map&Reduce scripts extract the values of interest for each file access operation, join them with the dataset names (an HDFS duplicate of the Popularity DB catalog) and calculate utilization metrics like CPU time, bytes processed, number of user accesses, etc. Apache Pig has been used initially; it is a high-level toolkit for creating scripts that run on Hadoop which makes Java Map&Reduce programming similar to SQL. It offers a fast learning curve and allows to produce fast aggregations of file access logs and measure the performance impact. Figure 3 shows the hierarchy of queries produced from EOS and AAA XrootD sources, with intermediate views representing partial processing.

Table 4 summarizes the degree of consistency between Oracle Materialized Views (MVs) and Pig queries. Oracle and Hadoop sources are not totally identical: they go through different streamers and are implemented via different APIs, thus the final aggregations may have some limited differences. The validation process computes the deltas, which are pictured in figure 4 with respect to three of the main metrics: number of accesses, CPU time and bytes read.

The main limitation of Pig, in terms of performance, is inherited from Map&Reduce, which forces a linear dataflow on distributed programs: read input data from disk, map a function

![Figure 3. HDFS metadata aggregation for EOS and AAA XrootD logs](image)

![Figure 4. Validation between Oracle MVs and Pig queries](image)
across the data, reduce the results of the map, store reduction results on disk. Due to this, once the query results are validated between Oracle and Pig, Apache Spark is used as the ultimate analytics platform. It is a fast and general processing engine compatible with Hadoop data, thanks to its distributed memory-based architecture. It requires a cluster manager, YARN, and a distributed storage system, HDFS. On top of them, it exposes an API based on resilient distributed dataset (RDD), a read-only multiset of data. RDDs take advantage of shared memory and reduce the latency of distributed applications when compared to MR implementations like Hadoop; this allows for a faster implementation of both iterative algorithms and interactive querying. A performance comparison between Pig and Spark is shown in figure 6.

Benchmarks of popularity data on Hadoop demonstrate how dashboard and monitoring systems can benefit from parallelism. The processing time of XrootD time series logs scales better than linearly with data volume, which makes it very effective for quick re-processing of entire months. Speedup factors range between 2x for daily aggregations up to 50x on monthly time-frames.

Map&Reduce operations have measured to yield a 10x speedup compared with the equivalent Materialized Views used to analyze Oracle data. In fact, while MVs require about one hour every day to compute time series aggregations on monthly/yearly sliding windows, the equivalent outcomes can be produced by Pig queries on HDFS in less than 10 minutes (figure 7). The obtained results can be either used in order to speed up the process in a transparent way, feeding them back to the Dashboard, or in order to serve as a fast data-frame generator to feed ML algorithms.

Figure 8 shows the performance speed-up in Hadoop compared to Oracle. Each data point is averaged on 3 tests. Pig offers the possibility to customize the number of parallel reduce tasks (1 by default) by the option “parallel”, while the map parallelism is determined by the input file, precisely one map for each HDFS block, whose size is 256 MB.

One limitation we encountered with Pig is the non satisfactory handling of malformed JSON input metadata. Flume - that is used in its default configuration with no optimization - may
occasionally truncate some entries. This has made it preferable to implement data aggregation in Spark, which instead offers input data consistency check through its RDD object.

![Monthly Processing Time](image1)

**Figure 7.** Monthly processing time in Hadoop. Higher data volume allocates a higher number of mappers

![Daily Processing Time](image2)

**Figure 8.** Performance comparison between Oracle and Hadoop MR on monthly data. MR performs 5x faster

The output data is ultimately uploaded into the Oracle database used by the monitoring system and dashboard in order to replace the slower MVs.

### 4.1. Mobile Monitoring

Map&Reduce query outputs are also consumed by a mobile application for monitoring purposes: the app arranges the results by site and displays geographically the dataset access distribution by individual users as well as the data volume between the sites and CERN (figure 9).
Figure 9. Mobile app that displays the results of the MapReduce layer, maps them by site and produces several graphical distributions

5. Conclusions and Outlook
This work has demonstrated that a new implementation of the Popularity service in the Hadoop ecosystem can fully replace the previous Oracle-based implementation; we are currently setting up in Hadoop the automatic daily update and publication of the popularity metrics. Thanks to the scalability of the service we will be able to define custom metrics and calculate them as needed. This is not the only benefit, however: we can now also extend the service to provide predictions of popularity patterns using machine learning tools. In addition, we can easily integrate data from different sources in Hadoop: in this way, we will be able in the future to analyze popularity metrics simultaneously with data from other monitoring sources such as the workflow management or data placement systems. In conclusion, this work has been a successful testbed for the wider adoption of Big Data technologies in CMS computing monitoring.

References