ATLAS Data Management Accounting with Hadoop Pig and HBase

Mario Lassnig, Vincent Garonne, Gancho Dimitrov, Luca Canali, on behalf of the ATLAS Collaboration
European Organization for Nuclear Research (CERN), 1211 Geneva, Switzerland
E-mail: mario.lassnig@cern.ch

Abstract. The ATLAS Distributed Data Management system requires accounting of its contents at the metadata layer. This presents a hard problem due to the large scale of the system, the high dimensionality of attributes, and the high rate of concurrent modifications of data. The system must efficiently account more than 90PB of disk and tape that store upwards of 500 million files across 100 sites globally. In this work a generic accounting system is presented, which is able to scale to the requirements of ATLAS. The design and architecture is presented, and the implementation is discussed. An emphasis is placed on the design choices such that the underlying data models are generally applicable to different kinds of accounting, reporting and monitoring.

1. Introduction

ATLAS is a global collaboration of more than 3000 persons to conduct high-energy physics research. The collaboration uses the ATLAS particle detector, which is installed at the Large Hadron Collider (LHC), to record the results of particle collisions. Both the LHC and ATLAS detector are located in Geneva, Switzerland at the European Organization for Nuclear Research (CERN). The data that is recorded by the detector is then processed by globally distributed computing centres for analysis. Additionally, the computing centres produce simulated particle collisions for comparison with detector data. The aggregate data volume amounts to 20 Petabytes per year, not including temporary and transient data, and is currently at 90 Petabytes. This rate is likely to grow in correspondence with the energy and luminosity increases of the LHC in the coming years.

ATLAS uses a distributed data management (DDM) system, called Don Quijote 2 (DQ2), to organise experiment data[1]. This includes the registration of files, the aggregation of files into data sets, the transfer of data sets between computing centres, consistency verification and repair, monitoring and reporting, quotas, and permissions. More importantly however, the DQ2 system provides a platform as a service (PaaS) for
members of the collaboration to address specific data management needs. For example, if a physics subgroup of the collaboration needs to reorganise some data in a different way for analysis, they can use the platform programming interface in the DQ2 system to achieve this.

Knowledge of how storage space is used by DQ2 on the grid is vital for the effective management of computing operations. This problem is complicated by the number of different dimensions along which ownership can be measured, e.g., datatype (RAW, ESD, AOD) is orthogonal to project (data11_7TeV, mc10_7TeV), which are both orthogonal to accounting by the data owner. For this reason the DQ2 accounting system, which was previously based on pattern matching, has been replaced with a system based on key-value pairs. Thus a wildcard query which previously might have been two strings '{data10.*.ESD}' + '{CERN}' can now be expressed as a set expression '{'project':'data10', 'datatype':'ESD', 'location':'CERN'}'. The new system allows arbitrary combinations of key-value pairs to be specified, which offers considerably more flexibility than the old pattern based approach. Once a set of key-values has been established the system will query these periodically. We call such a set an **accounting summary**. In this way historical data will be built up automatically, and trends can be analysed. An initial implementation is available based on an Oracle backend, but an assessment of structured storage systems, which are inherently more suitable to such key-value stores, is underway. This paper describes the new accounting system based on such a structured storage system, namely Hadoop. The Hadoop ecosystem provides two products, Pig and HBase, which are especially designed for large data analytics, and are used to build the new accounting system. The remainder of the paper is structured as follows: first, a description of the accounting use case is given, then the technologies used are described. The following sections then explain the actual implementation and give first results.

## 2. Accounting

The traditional definition of accounting stems from business and financial domains:

**Definition 1 (Accounting).** The system of recording and summarizing business and financial transactions and analyzing, verifying, and reporting the results. (Merriam-Webster, 2012)

In DQ2 however, such business transactions are data transfers and placement. This data has metadata attributes attached to it, which are to be analysed, summarised and reported. Metadata attributes associated with a dataset or file are represented using key-value pairs. The set of available keys is however restricted due to operational constraints. Metadata attributes are classified into four categories:

(i) **System-defined attributes:** e.g., size, checksum, creationtime, modificationtime, status

(ii) **Physics attributes:** e.g., number of events

(iii) **Production attributes:** e.g., task and job ID that produced the file, processing campaign ID

(iv) **Data management attributes:** necessary for the organisation of data on the grid
An accounting system must support different combinations of metadata attributes, to satisfy some higher level reporting use cases, for example:

- What is the occupancy at groups of sites? e.g., all SCRATCH at UKTIER2s
- Distributed between datatypes? e.g., RAW, ESD, NTUP
- Matching some ATLAS physics definitions? e.g., reprocessing campaigns
- Of a particular tag? e.g., f405_r120
- And how much of it is temporary user data? e.g., can be deleted

The dimensionality of this problem by number of attributes is greater than 25, which presents a serious combinatorial, and therefore time delay, problem. This would result in more than 300 possible combinations, out of which many are not useful. For this reason, up to now, we have only supported 3 dimensions: site, project and datatype. Extending the accounting for arbitrary selections of attributes poses a hard problem that cannot right now be solved by just aggregating data on the fly. However, a generic system is needed to solve this problem, as possible and plausible combinations of attributes will evolve over time. For this reason, a new smart and scalable accounting system was developed.

3. Technology

The new accounting system is based on a shared-nothing parallel data pipeline, built atop a self-configuring and self-repairing cluster of commodity hardware. Table 1 shows the hardware configuration of the cluster. Note that the overall amount of memory and CPU cores are not accessible to each node. Each node only has access to its own cores and memory. The software application stack is built as follows:

**Oracle** [2] is a relational database management system, with advanced features like clustering, partitioning, hot backup, data warehousing, and much more. DQ2 stores its transactional and relational data in the central Oracle Database 11g at CERN. This database contains the core data that needs to be accounted.
**Puppet** [3] is an IT automation software that helps system administrators manage infrastructure throughout its lifecycle, from provisioning and configuration to patch management and compliance. Currently, 12 nodes are provisioned in the CERN Computing Centre, which are managed by Puppet. This includes host configuration, software installation, and software configuration.

**Cloudera** [4] provides an enterprise-ready, commercial Distribution for Hadoop. Cloudera integrates the most popular projects related to Hadoop into a single package, which is run through a suite of rigorous tests to ensure reliability during production. The Cloudera distribution has shown to be of very high quality, and well documented.

**Apache Hadoop** [5] is a framework that allows for the distributed processing of large data sets across clusters of computers with MapReduce. It is designed to scale up from single servers to thousands of machines, each offering local computation and storage. Rather than rely on hardware to deliver high-availability, the framework itself is designed to detect and handle failures at the application layer, so delivering a highly-available service on top of a cluster of computers, each of which may be prone to failures.

**Apache Pig** [6] is a platform for analysing large data sets that consists of a high-level language for expressing data analysis programs, coupled with infrastructure for evaluating these programs. The salient property of Pig programs is that their structure is amenable to substantial parallelisation, which in turns enables them to handle very large data sets.

**Apache HBase** [7] is an open-source, distributed, versioned, column-oriented store modelled after Google’s BigTable. HBase supports random real-time read/write access and can host very large tables, with billions of rows times millions of columns.

4. Implementation

The accounting system backend works in three steps. First, it retrieves data from the operational instance of DQ2, second, it creates the accounting summaries based on the retrieved data, and third, publishes the accounting summaries for public consumption.

4.1. Prepare data for analysis

First, the relevant data is extracted from the DQ2 tables in Oracle using Oracle Active Data Guard, and then written to HDFS, the Hadoop Distributed File System. HDFS can be mounted as a POSIX filesystem via FUSE, which greatly facilitates writing data into Hadoop. This data is a simple CSV file with 25 attribute columns, with one file per day. This allows the recreation of historical accounting summaries, as old data is preserved. The size of the daily dump is currently about 6GB uncompressed for 10 million rows and 1.5GB compressed. Retrieving the dump takes about 20 minutes, so in principle could be done more often. From an operational point of view, anything more than once a day is not useful due to limitations on data transfer throughput and deletion on the grid, as the fluctuations would be too high. Additionally, in an effort to offload the Oracle database it would be imprudent to constantly retrieve rows for external processing, when the time to retrieve rows is longer than actual processing.
SELECT /*+ parallel(t 8) */ campaign,
    COUNT(duid),
    SUM(nbfiles/replicas),
    SUM(replicas),
    SUM(rnbfiles),
    SUM(length/replicas),
    SUM(rlength)
FROM dump
WHERE NOT ((name LIKE '.%_dis.%') OR
    (name LIKE '.%_sub.%') OR
    (name LIKE '.%_shadow.%') OR
    (name LIKE '.%/'))
GROUP BY campaign;

Table 2: Oracle SQL example.

d = FOREACH dump GENERATE campaign,
    duid,
    nbfiles/replicas AS xnbfiles,
    length/replicas AS xlength,
    rnbfiles,
    rlength,
    name;

t = FILTER d BY NOT ((name MATCHES '.*_dis.*') OR
    (name MATCHES '.*_sub.*') OR
    (name MATCHES '.*_shadow.*') OR
    (name MATCHES '.*/'));

g = GROUP t BY campaign;
s = FOREACH g { duids = DISTINCT t.duid;
    GENERATE FLATTEN(group),
    COUNT(duids),
    COUNT(t.duid),
    SUM(t.xnbfiles),
    SUM(t.rnbfiles),
    SUM(t.xlength),
    SUM(t.rlength);};

Table 3: Pig Latin example.

4.2. Run the analysis
Pig uses an algebraic language, Pig Latin, that describes a data-pipeline, with expressions bound to variables. For example, “use all rows from a CSV dump where column
3 must end with string AOD, and aggregate the values from columns 4, 6 and 9. Pig then creates parallel MapReduce jobs, which run on Hadoop. Pig is extremely efficient, because it can create an algebraic minimum possible access path to the data, which is important when there are multiple, similar requests to the same data. That way, redundant access to the data can be avoided. Current execution time for the daily summaries is at 8 minutes, which spawns about 1500 MapReduce jobs that are distributed on the cluster.

Tables 2 and 3 show how a traditional SQL query for accounting can be expressed as a Pig Latin program. The principle is the same in both, but note however, that the Pig version lends itself to independent parallel computation as the FOREACH and FILTER statements work line-by-line, whereas the SQL SELECT and WHERE do not impose a data access path. For this reason, the Pig Latin program requires the explicit definition of what the output of the program should be, as it is done independently in parallel.

4.3. Make analysis available
Finally, it must be possible to access the results. Pig writes the output of its summary computations into CSV files, and natively into HBase. A separate CSV file per requested summary per day is written. Pig writes both outputs directly and in parallel, and there is no intermediary step necessary. The flexible, non-relational tables of HBase allow to store the summaries in a compact and efficient way. Currently, 29 summaries are computed, with an aggregate output size of 1.8 million rows and 180 Megabytes of space for the CSV case, and 4 Megabytes in the HBase case, as it is automatically compressed.

5. Conclusion
The increasing demands of analytical workloads, like accounting, have become disruptive for the Oracle database at CERN. In an effort to offload these particular workloads from the Oracle database, structured storage systems were evaluated as potential alternatives. Out of an evaluation of several products, the most promising one, Hadoop, was put in production. The Hadoop ecosystem, especially HDFS and MapReduce, provides all features and scalability that are necessary to fulfil the analytical use cases of ATLAS DDM. HBase also encompasses the most important features of the other non-relational database products.

The new accounting system has been in production since December 2011, and so far worked without a problem. Several new summaries were added over time, with no increase in processing time. At the current rate of processing the cluster will run out of storage space within 2 years, before hitting the CPU limit for processing. Fortunately, adding new nodes to the cluster is a quick operation to ensure linear scalability. Also, the history data of the previous accounting system has been successfully migrated to the new one as well. In two occasions, mistakes in a Pig Latin program were found, which rendered two summaries useless. Due to keeping historical dumps, it was possible to correct the Pig Latin programs for this summary and re-run them on the historical data. In one case, a fatal hardware fault that destroyed the disks in 4 machines at the same time was also survived. Hadoop automatically repaired its state and the system continued working while the hardware was replaced with no downtime.

Due to our positive experience with Hadoop, Pig and HBase we are considering
constructing more analytics in this ecosystem. With the accounting system as a blueprint, we are certain that such analytics applications can be built and brought into production in a short amount of time.

Bibliography