ATLAS DQ2 Deletion Service

Danila Oleynik\textsuperscript{1}, Artem Petrosyan\textsuperscript{1}, Vincent Garonne\textsuperscript{2}, Simone Campana\textsuperscript{2} on behalf of the ATLAS Collaboration

\textsuperscript{1}Joint Institute for Nuclear Research, Laboratory of Information Technologies, Dubna, Russia

\textsuperscript{2}CERN, CH-1211 Geneva 23, Switzerland
danila.oleynik@cern.ch

Abstract. The ATLAS Distributed Data Management project DQ2 is responsible for the replication, access and bookkeeping of ATLAS data across more than 100 distributed grid sites. It also enforces data management policies decided on by the collaboration and defined in the ATLAS computing model. The DQ2 Deletion Service is one of the most important DDM services. This distributed service interacts with 3rd party grid middleware and the DQ2 catalogues to serve data deletion requests on the grid. Furthermore, it also takes care of retry strategies, check-pointing transactions, load management and fault tolerance. In this paper special attention is paid to the technical details which are used to achieve the high performance of service, accomplished without overloading either site storage, catalogues or other DQ2 components. Special attention is also paid to the deletion monitoring service that allows operators a detailed view of the working system.

1. Introduction

The ATLAS experiment [1] has recorded almost 5PB of RAW data since the LHC started running at the end of 2009. Many more derived data products and complementary simulation data have also been produced by the collaboration and, in total, more then 94PB (300M of files) is currently stored in the Worldwide LHC Computing Grid by ATLAS. All of this data is managed by the ATLAS Distributed Data Management system, called Don Quixote 2 (DQ2)[2].

While deletion which might seem like a fairly straightforward activity on the surface, in a complex distributed environment, such as that managed by DDM, it is far from trivial. Dataset deletion requests on a particular site need to be done with care to ensure that:

- The dataset replica entry is deleted from the DDM central catalog.
- Corresponding files are physically deleted from storage.
- All file replica locations are removed properly from the local file catalog.

Each of these steps might fail, so it is necessary for the deletion service to have an internal state engine, which records the state of deletion for any dataset at a particular site. It is also necessary to throttle both catalog and physical deletion requests for files in order to prevent external services from being overwhelmed. In addition, there is also the additional complexity of overlapping datasets. If two
Datasets share files on a site and only one of them is deleted then the shared files should not be deleted, otherwise the remaining dataset would become incomplete. This requires some care in mapping dataset deletion requests onto file deletions. Deletion rates in ATLAS DDM can reach significant levels, with millions of file deletions per day and terabytes of data being cleaned.

2. DQ2 Deletion Service architecture

While the number of stored data is growing steadily (fig. 1), increasing the number of sites involved in the ATLAS data processing, as well as increasing demands on the rate of release of storage space, architecture of Deletion Service shall be capable of scaling.

To provide required scalability, Deletion Service was developed as a client-server application based on Web-service technology. Main components of deletion service are: Deletion client, Deletion server, Deletion agent and Deletion monitoring. Deletion client/server and Deletion agent together are the core part of Deletion Service. Deletion monitoring is a special web-based application providing reports about deletion process. Schematic interactions between components of Deletion service are shown in Figure 2.
2.1. Deletion Client/Server
Server part of Deletion service encapsulates intercommunications with Oracle database backend. Deletion server is based on Apache web-server [3] with mod_python extension [4]. To speed up connections with the database, backend uses the connection pool component [5]. All these solutions allow to achieve high service performance. This configuration of Apache web-server makes it possible to maintain multiple concurrent requests to the database with fixed number of open connections to database.

Usual number of processed requests by web server is around 3 millions per day. There is no direct relation between performance of Deletion service and load on the Deletion server due to the fact that most of instructions are realized as bulk operations.

Deletion client is a specialized http-client designed to simplify interaction with the Deletion server. The client is integrated with the Deletion agent module on API level and completely hides the lower-level protocols.

2.2. Deletion Agent
Main executive component of the Deletion service is Deletion agent. Before describing implementation of the Deletion Agent it is necessary to explain some concepts of DDM. DDM DQ2 realizes data management at dataset level; dataset is a group of logical aggregated files. There could be from one file to dozens of thousands of files in a dataset. Deletion request, the initial instruction for the Deletion Service, contains information about which dataset should be deleted from the specified endpoint. Thus, the additional task of the Deletion Agent is to define a correct file list for removal.

Thereby, it is possible to mark out three main operations for the Deletion Agent to preform:
1. Resolving list of files for removal
2. Removing files replica locations from LFC
3. Physical files deletion from storage

Three main procedures were developed to implement these functions. The created procedures operate as three independent concurrent processes. Architecture of Deletion agent is schematically shown in Figure 3.

![Figure 3. Deletion agent architecture](image)

Procedure called “Resolver” defines the list of files to be deleted according to the request. For obtaining this list the DDM catalog is requested, the request is constructed in such a manner as to receive the list of not overlapped files only. The incoming list contains only names of the files; this information is not enough for deleting data from storage. For getting a full path to files (storage URL,
SURL) LFC is requested. All acquired information is saved in a database for the subsequent operation. After this operation a deletion request will be marked as ready for deletion.

All requests marked as ready for deletion are processed by the procedure that cleans LFC records for the corresponding files (“Catalog cleaner”) and procedure of removing files from storage (“Storage cleaner”). The main algorithm of these procedures is similar: to select the chunk of files from a database, to execute an operation, to analyze the result, to update records in the database. The difference between algorithms consists in the features of interfacing with LFC (the asynchronous interface) and with storage systems (the synchronous interface). The asynchronous interface needs an additional check of operation’s results. Procedures of cleaning of LFC and storage works at file levels, it gives the possibility to make checkpoints for each file.

“Catalog cleaner” and “Storage cleaner” manage only state of files in the database; the logic of dataset deletion operates by database triggers. This approach allows us to improve the performance of the service, and align deletion rate by eliminating the dependence on the size of the dataset.

To achieve high productivity, Deletion Agent is designed as a multithread application. Each site served by its own copy of the “Resolver”, the “Catalog cleaner” and the “Storage cleaner”. To improve the interaction with the database, bulk operations are used, and size of chunks is configurable for single sites and endpoints.

3. Deletion Monitoring

Deletion service is a high-priority service of ATLAS DDM, but most of the time only one person is engaged to control the work of the service. Tracking of removal of more than two hundred sites without the use of a specialized operator’s interface is almost impossible. To provide the operator’s information on how the service works, a special application was created – the system of monitoring of the Deletion Service [6].

![Deletion Monitoring](image-url)
The monitoring system is implemented as an application running in the Internet environment. Web-platform was chosen in order to provide access to information about how the service works from anywhere in the world and be independent of the operating system installed on the operator’s computer.

Deletion monitoring must meet the following requirements:

- high availability of the service;
- actuality of the information;
- intuitive user interface;
- summary graphical reports;
- detail of each event;
- minimum possible load on the database.

Several DDM DQ2 web-applications use Django framework [7], which, being written in Python, provides full compatibility with services system code. Therefore, Django is a natural choice of the platform for the deletion monitoring.

Deletion monitoring generates reports using jQuery [8] AJAX (Asynchronous JavaScript and XML) calls. BBQ plug-in [9] is used to maintain browser history, direct links and bookmarks.

Every report provides JSON (JavaScript Object Notation) export which can be retrieved by any third-party application.

The application is dealing with the heavily loaded database. To avoid extra load on the database, database connection pooling and Memcached [10] are used. Once being called, plot data are save in Memcached for next 10 minutes, 30 minutes, 1 hour or 4 hours depending on report period.

The application provides graphical and numeric reports representing deletion process at ATLAS sites. Screen shot of the application is shown on figure 4.

Summary information is presented at the cloud, site, and endpoint levels. Different report periods could be selected: last hour, last 4 hours, last 24 hours.

To provide detailed information about any event in the deletion process, system is instrumented by the dataset and error browsers with search interface. File reports reach highest detail level all the way to single operations as deletion from LFC and local storage system. This allows tracing of deletion events for every file in DDM.

4. Conclusion

Presented Deletion Service is used by ATLAS Distributed Computing since autumn of 2010. In 2011 some works for optimization of algorithms were performed, that led to a significant performance boost.

Deletion Service serves more than 120 sites with more than 700 endpoints. In usual operation it deletes up to 2,5M of files per day, which correspond to 250 - 300 TB per day. During the deletion campaigns when deletion was carried out on most sites, deletion rate achieved is more than 6M of files per day, reaching up to 300k files per hour (fig. 5).

Figure 5. Deletion service performance during deletion campaign
Deletion monitoring is actively used by the ATLAS Distributed Computing operation team and site administrators.

Current implementation of the Deletion Service has flexible configuration and can be easily tuned for better performance, but main limitations presently are performance and stability of storage systems and LFC Servers.

References
[1] The ATLAS Collaboration 2008 Journal of Instrumentation 3 S08003 URL
   http://stacks.iop.org/1748-0221/3/i=08/a=S08003
[2] Branco M, Cameron D, Gaidioz B, Garonne V, Koblitz B, Lassnig M, Rocha R, Salgado P and Wenaus T 2008 Journal of Physics: Conference Series 119 062017 URL
   http://stacks.iop.org/1742-6596/119/i=6/a=062017
[3] Apache http server project website, http://projects.apache.org/projects/http_server.html
[4] Mod_python Apache extension website, http://www.modpython.org/
[5] DB Connection pool component website, http://commons.apache.org/dbcp/
[6] Deletion monitoring page, http://bourricot.cern.ch/dq2/deletion/
[7] Django project web site, http://www.djangoproject.com
[8] Memcached project web site, http://memcached.org/
[9] jQuery project web site, http://jquery.com
[10] jQuery BBQ plug-in web site, http://benalman.com/projects/jquery-bbq-plugin/