Managing ATLAS data on a petabyte-scale with DQ2

Miguel Branco[1], David Cameron[2], Benjamin Gaidioz[1], Vincent Garonne[1], Birger Koblitz[1], Mario Lassnig[13], Ricardo Rocha[1], Pedro Salgado[1], Torre Wenaus[4], on behalf of the ATLAS Collaboration

[1] European Organization for Nuclear Research (CERN), CH-1211 Genève, Switzerland
[2] Department of Physics, University of Oslo, P.b. 1048 Blindern, N-0316 Oslo, Norway
[3] Distributed and Parallel Systems, University of Innsbruck, A-6020 Innsbruck, Austria
[4] Physics Application Software, Brookhaven National Laboratory, NY-11973, USA

E-mail: `mario.lassnig@cern.ch`

Abstract. The ATLAS detector at CERN's Large Hadron Collider presents data handling requirements on an unprecedented scale. From 2008 on the ATLAS distributed data management system, Don Quijote2 (DQ2), must manage tens of petabytes of experiment data per year, distributed globally via the LCG, OSG and NDGF computing grids, now commonly known as the WLCG. Since its inception in 2005 DQ2 has continuously managed all experiment data for the ATLAS collaboration, which now comprises over 3000 scientists participating from more than 150 universities and laboratories in 34 countries. Fulfilling its primary requirement of providing a highly distributed, fault-tolerant and scalable architecture DQ2 was successfully upgraded from managing data on a terabyte-scale to managing data on a petabyte-scale. We present improvements and enhancements to DQ2 based on the increasing demands for ATLAS data management. We describe performance issues, architectural changes and implementation decisions, the current state of deployment in test and production as well as anticipated future improvements. Test results presented here show that DQ2 is capable of handling data up to and beyond the requirements of full-scale data-taking.

1. Introduction

1.1. Overview

The ATLAS Distributed Data Management (DDM) project was established in spring 2005 to develop the system Don Quijote 2 (DQ2), drawing on operational experience from a previous generation of data management tools. The foremost design objective was to achieve the scalability, robustness and flexibility required to meet the data handling needs of the ATLAS Computing Model. This means the complete dataflow of the experiment from raw data archiving through global managed production and analysis to individual physics analysis at home institutes. The ATLAS Computing Technical Design Report [5] defines the scope of the DDM system, henceforth DQ2, as:

The scope of the system encompasses the management of file-based data of all types (event data, conditions data, user-defined filesets containing files of any type).

Within this scope, the distributed data management system needs to make the distinction between two types of data handling: production and users data. This distinction is necessary because of their different usage patterns in the system. DQ2 is therefore the primary responsible for bookkeeping of file-based data. Other systems may extend DQ2 functionality, such as an external metadata system, but synchronisation may be necessary as DQ2 is the ultimate and reliable source for ATLAS file-based data. These distinct activities on file-based data are therefore defined the following.

- Production activities are well-defined and include data acquisition and export, rereconstruction and Monte Carlo production and is managed either at the collaboration or physics group level.
- User activities include retrieval of subsets of production data as well as management of data produced by individual users.

Consequently the responsibilities of DDM are

(i) to provide the functionality required for bookkeeping of all file-based data.
(ii) manage movement of experiment data across sites and optionally user data, thus implementing the the production dataflow described in the ATLAS computing model.
(iii) enforce access controls and manage user quotas.

DQ2 owns all ATLAS storage areas dedicated for production and optionally some areas reserved for users. Individual users and production managers own data itself but DQ2 is always able to change any existing control rights for data under DQ2-managed areas.

DQ2 is also the primary user and main responsible for managing all data handling resources, that is storage areas and network bandwidth, in order to guarantee an acceptable service level for its activities while allowing a minimal share for unmanaged activities.

1.2. Requirements

DQ2 operates on the worldwide LHC computing grid infrastructure (WLCG). It consists of three different grid flavours, the Open Science Grid (OSG) in North America, EGEE LHC Computing Grid (LCG) in Europe and NorduGrid (NDGF) in Scandinavia. WLCG spreads over 10 large Tier-1 centres globally and their associated smaller Tier-2 centres, constituting more than 200 sites with more than 3000 users. Since these grid flavours all have different middlewares DQ2 must provide a layer to interface with all of them.

The amount of data produced by the ATLAS detector is expected to be around 1 terabyte per day, with the whole experiment summing up to nearly 20 petabytes per year and increasing. This includes data from the detector, reprocessed data as well as user-generated data. The following entries in DQ2 are expected:

- O(4000) new datasets from trigger and data acquisition (TDAQ) with datasets defined per run and physics stream as well as datasets for logfiles per day.
- O(1500) new datasets from monte-carlo simulation per day. User data is less clear – a very rough estimate is in the order of monte carlo numbers.
- O(1) versions per dataset.
- O(100) files per dataset.
- It is expected that the majority of the datasets will be on a state where the latest dataset version is closed or the dataset is frozen.
- O(100) data-transfers at any moment per site.
- O(100) sites providing storage for ATLAS as dataset locations.
- O(100) ATLAS users and O(10) groups.

In the following section we present the basic concepts of the system and its constituents to match these requirements. After that we give a detailed look of the implementation of all the components and show their performance.

2. Concepts

2.1. Datasets

The primary concept of the distributed data management system is the dataset, as shown in figure 1. A dataset is an aggregation of data, typically spawning more than one physical file, that are processed together and serve collectively as input or output of a computation or data acquisition process.

Datasets have the following distinct advantages. Datasets provide the granularity of data handling that users typically manipulate thus providing a primary grouping for data. The alternative would be knowing individually the state, location or metadata associated with each constituent, that is a file, of a set of datafiles. This would require additional work for the user and would be more error prone. By having the dataset as the primary data handling concept, the architecture enforces this unit of data to be handled, that is transferred, together and consistently, thus improving scalability. This accommodates expectations that the system will have less datasets entries than data files. It is often necessary to have efficient usage of network transfer channels and interaction with external storage layers. This means having bulk operations and datasets therefore provide the unit for the bulk. An example is storing RAW data – during reprocessing each dataset of RAW data must be recalled from tape together.

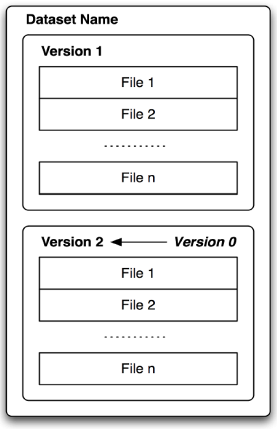

Figure 1. Dataset schema

2.2. Dataset properties

Datasets as used for ATLAS also have certain distinct properties:

Versioning Datasets always have a version. When the dataset is first created, the initial version is automatically created and assigned to the dataset. Datasets may have multiple versions to support discrete changes, for example by addition of new data, to its content. A fixed version 0 (zero) always points to the latest version.

Immutability Datasets hold a mutability state which can be open, closed or frozen. A dataset is said to be open when the latest version of the dataset is open. New data can be added to the dataset, that is new content is appended to the latest version of the data set. Existing content, part of the latest or previous version, can also be removed. A dataset is said to be closed when the latest version of the dataset is closed. No content can be added while a new version is not created for the dataset and left open again. A dataset is said to be frozen when the latest version is closed and no new versions can be added. The dataset is completely immutable and cannot be versioned anymore to support discrete changes in its content.

2.3. Dataset operations

Constituents of a datasets are files. Changes to a dataset are typically done by:

- Appending new content to the latest open dataset version.
- Removing existing content of the latest open dataset version.
- Adding new dataset versions.

Appending new content The typical use-case is to append new content to a dataset. Datasets always have a latest version, called version 0. We assume all updates to the dataset are done to the latest version, similar to a versioning control system. If the dataset is closed, a new version is created first before any new content is added. The latest dataset version is assumed to contain the most up-to-date validated content for the given dataset.

Removing existing content It is possible to remove content from a dataset. This is required when, between versions of the same dataset, the user wishes to remove some of its constituents, for example some files in a dataset were found to be invalid and should be removed from all future versions, starting from the version where the removal was requested. Removing content from a dataset is only possible to the latest open dataset version. The request to remove a file is not restricted to the newly added files, but may be for any file currently part of that dataset version. The restriction to removing content is that content can only be removed from the latest open dataset version. Nevertheless, the user may remove files that are not only in that particular open dataset version, but are part of any of the previous versions. It is possible to re-add an element to a dataset which had been removed from a previous version. This is equivalent to adding new content to the dataset. Since dataset-transfers can be in progress whilst the latest dataset version is open it is possible for the system to schedule the movement of a particular file, which in the meantime has been removed from the latest open dataset version. The movement will still be done but the file may be discarded later by external consistency checks.

Dependency on a dataset version Dependency on a particular version of a dataset does not imply that the latest version of a dataset is usable for the same purposes. Dataset contents can change, files added or removed, between versions of a dataset so locating the latest version of a dataset does not necessarily retrieve all files in the dataset. If a dataset version contains a certain file it is not guaranteed the latest version will still contain it. Additionally, the same file can be removed and re-added between dataset versions. Nevertheless a user or application can always depend on the contents of a closed dataset version or frozen dataset.

2.4. Files

Files are the basic system units and they are immutable. They are identified by their respective globally unique identifier (GUID) and human-readable logical file names (LFN).

Files cannot be updated. For each update, a new file must be written and a new GUID and LFN must be created if the file is already registered on DQ2.

A one-to-one mapping must exist between an LFN and a GUID in DQ2. Any attempt to register a new GUID with a different LFN will fail if the LFN was already registered, is associated with a different GUID as being part of a datasets in the system.

A file may be added to a dataset version and removed in a later version. Nevertheless, it exists in the system and therefore it is possible to retrieve the old version. Therefore a user cannot re-use the same LFN for newer versions of the same file. The new version will have a new GUID. To avoid this problem the user can create a new LFN, for example by appending an attempt number to the LFN.

Similarly a user cannot re-use a dataset name in DQ2. If a dataset is removed the name becomes unusable in the future. If this were not the case it would not be possible to guarantee provenance for any dataset: the dataset name would not refer to the initial dataset but to a newly created one. To avoid this problem the user can either create a new dataset version and disregard the previous version or create a new dataset with a new name.

2.5. Distribution

DQ2 seamlessly uses multiple Grid resource providers, minimising dependencies on grid-specifics to facilitate deployment and operation. Therefore, DQ2 requires knowledge of all DDM managed storage, including the role of each site (Tier-1, Tier-2) as well as the mapping between sites (Tier-1 and Tier-2 association as well as Tier-1 to Tier-1 pairing for reprocessing). Tier-3 sites are not managed by DQ2. These are only used within the scop of end-user activities and not as final storage areas for ATLAS. Each storage area managed by DQ2 is identified uniquely as a DDM site, for example CERNPROD, LYONDISK or NDGFT1TAPE.

3. System architecture

The design as shown in figure 2 layers a set of loosely coupled components over a foundation of basic file handling Grid middleware. These provide logical organisation at dataset level, supporting in a flexible way the data aggregations by which data is replicated, discovered and analysed globally. A combination of central bookkeeping services, distributed site services and agents handle data transferring, bookkeeping and monitoring. Implementation approaches were carefully chosen to meet performance and robustness requirements. Fast and lightweight remote procedure calls to web-services to the bookkeeping catalogues facilitate this requirement on the central service.

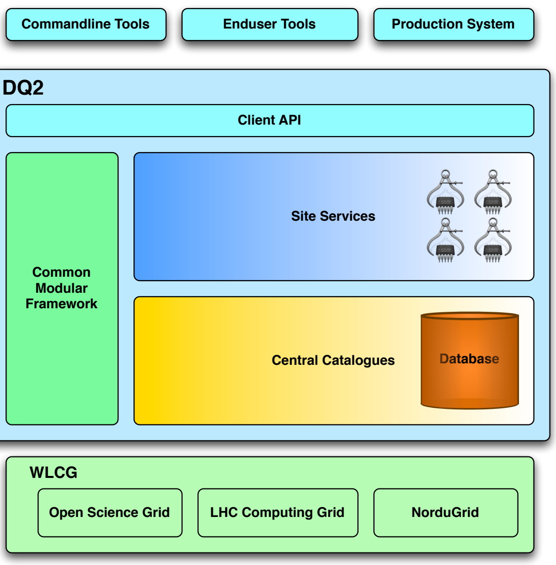

Figure 2. System overview

The local site services transfer datasets based on information polled from the central catalogues and feedback status of ongoing transfers to the monitoring services. Users interact mostly with the dataset catalogues through end-user tools, to define and lookup datasets as well as request subscriptions to datasets. The monitoring services provide user tools, including web interfaces, to follow up the status of each individual file and dataset transfer and provide an overall view of the status of datasets in the distributed system. Monitoring services also provide the mechanisms to deduce the overall state of the DQ2 services at each site.

3.1. Central catalogues

The content catalogue stores the mapping between files and dataset versions and also relevant information regarding files such as its GUID, LFN, size and checksum. This information is particularly important when finding good sources of a file or when verifying the integrity of a file transfer. This central storage of file metadata makes it possible to enforce a LFN-ATLASwide uniqueness and also to enforce a unique mapping between a GUID and a LFN. Internally the content catalogue only keeps a record of the changes done on each dataset version. This reduces significantly the amount of table and index disk space used by the database.

The repository catalogue stores information regarding dataset attributes such as name, owner, creation date, state and its versions.

The location catalogue stores the dataset replicas. The dataset replica is managed at the version level since the files contained in them may differ. If the site has all files of a version the replica is registered as complete. Otherwise it is marked as incomplete if it only has part of the dataset version.

The subscription catalogue stores users requests for dataset transfers to their sites. This catalogue is polled regularly by the site services. Afterwards the site services will look for sources to copy the files and issue transfer requests.

The central catalogues had to deal with a massive content increase in the last 6 months. The number of files increased by 15% to over 19.5 million and the number of datasets increased by 45% to over 300.000. Also, the emergence of very large datasets with more than 20000 files posed a problem. This forced to change the implementation significantly. Each of the catalogues is implemented as a standalone web service with its own relational database back-end.

The web-services installation depends on the Apache web server and its Python and GSI modules (mod python and mod gridsite). Clients can contact the catalogues through regular HTTP requests. All requests which add or modify information are done through a secure request and authorisation is given according to the users grid certificate attached to the request. The relational database back-end is implemented using Oracle and runs on the ATLAS Oracle production service at CERN.

We have a redundant load-balanced installation of the web-services. Two machines, running centrally at CERN, are hosting the web services plus a third one which provides the load balancing service.

Each of the web-services only have one endpoint. The translation of the client request is done through the inspection of the following parameters:

API This parameter is used to provide backwards compatibility for older clients.

operation This parameter contains the operation the client wants the catalogue to perform.

tuid This is a transaction universally unique identifier so that client requests can be mapped more accurately to server-side operations in the log files.

The response of the server is in accordance with the HTTP standard codes. The 200 status code is returned to the client unless an error occurs in which the status code 500 is returned instead. In case of success, the response is a regular Python structure, either a list or a dictionary; otherwise the error response will be a pickled Python exception class. We also provide client components to contact these catalogues directly. The client components are Python classes which expose the catalogue API to the user. Their main advantage is that the user can use the catalogues just like he was using a regular Python module. The client components are implemented using Python classes which construct the correct HTTP request for the user. There is a generalised client class which provides functionality needed by the clients to build HTTP requests. Those are

- Build GET and POST HTTP requests.
- Load GRID certificate for secure requests.
- Map DELETE request into a secure GET request.
- Map PUT request into a secure POST request.

This generalised client class then calls a curl/pycurl flavoured component whose responsibility is to

(i) Convert the request into a curl command.
(ii) Send the request to the server.
(iii) In case of success convert the response to a Python structure.
(iv) In case of failure pickle the response to build the exception class instance and use a Python raise instruction to return the error to the user.

3.2. Site services

The local site services are an autonomous agent based framework with concurrent Python agents acting on an MySQL database with an InnoDB backend. One instance of the site services can serve multiple sites if needed. In the current setup at CERN one installation serves a cloud, that is a Tier-1 and all its Tier-2s, for production use. A separate installation is providing all Tier-0 to Tier-1 export transfers.

All managed data movement in DQ2 is automated using a subscription methodology. The idea is that a site subscribes to a dataset and DQ2 site service agents resident locally act to pull a new dataset as well as keep the sites copy of the dataset up to date with respect to any changes that might be made to the dataset over time globally. Data movement is triggered from the destination side, such that local uploading can be done using site-specific mechanisms if desired, with no requirement that other sites are aware of these specialised mechanisms. A number of independent agents per site are involved in the data movement process to perform the principal steps of the movement process:

- Fetching the set of logical files to be transferred on the basis of the dataset content minus any files already present locally.
- Replica resolution to find the available file replicas via the dataset location catalogue, content catalogue and local replica catalogues.
- Allocation of the transfers across available sources according to policy into file transfer service (FTS) channels.
- Bulk reliable transfer using FTS.
- File- and dataset-level verification of the transfers.

The site services utilise a non-hierarchical fairshare algorithm that tries to divide slots according to the number of file transfers and not according to filesize. This would cause very large files to congest the channels. The algorithm then measures the state of active transfers against its own queue for FTS channel allocations.

A particularity in ATLAS is that there are a lot of small files so complete fairsharing while guaranteeing high throughput can not be trivially implemented using traditional queuing algorithms. At times there are over 2 million files in the queue ready to be transferred which need to be reorganised if files can be transferred on empty channels. This process is in contrast to short timeouts and short queues but enables the site services to keep the queues full and fair whenever possible. This process is called late reshuffling.

If errors occur, the failover mechanism is a retrial strategy with exponential backoff. The element causing the problem is taken off its queue and reinserted at a later time at its previous position, regardless of how the queue changed in the meantime.

Allocation to FTS channels always tries to perform on connected sites according to the ATLAS Computing Model. Connected sites are Tier-1 to each Tier-1 and Tier-2s to each Tier-1. The allocation agent chooses the source-destination channel that has the least number of attempts from that particular source and if there are multiple replicas then decides to take the one with the shortest queue. If that is not possible we fallback to multi-hop transfers.

In accordance with the ATLAS TDAQ DQ2 supports a special case in data integrity. DQ2 supports the insertion of data movement requests even before the data actually exists. The production system will provide the data at a certain point time and therefore early optimisation of the FTS channel allocation queue is made possible.

3.3. Monitoring

The monitoring service is a contribution by the ARDA project. It is based on a centralized service that receives HTTP requests, storing the monitoring events, per dataset, on a relational backend, based on Oracle. Please refer to the publication by Rocha et al. (2007).

4. DQ2 in production

The DDM operations group must keep data integrity and provide routine data transfer, data monitoring and data access to the ATLAS physics community. The data transfer priorities are defined by the collaboration management, data preparation and physics coordinators. The data transfer between the ATLAS detector and Tier-0, though, is not the responsibility of the DDM operations group. The regional DDM operations group includes the regional DDM operations coordinator and people working for each Tier-2 associated with their Tier-1. The group structure and organisation can be different from region to region but each group provides help to the ATLAS physics community, manages DQ2 and does DQ2 deployment. Together they are responsible for

- data transfer monitoring and control.
- data rerouting in case of Tier-1/Tier-2 or transfer channels instability.
- resolving data transfer errors and providing first line expertise.
- reporting data transfer problems to the Computing Operations Coordinator.
- data exchange between Tier-1s during RAW data reprocessing
- deployment of DQ2 releases
- 24/7 support of central DDM operations facilities
- support of the DDM Savannah user-requests portal.
- keep data integrity, in particular clean obsolete datasets and files entries from LFC/LRC and DDM catalogues.
- help ATLAS users to transfer their data. Regional teams in particular for the physicists from geographically closed Universities and Laboratories. communication with developers and providers (both ATLAS and WLCG) and CERN
- Networking personnel (NetOps).

DDM operations is considered as a primary job for the above teams during data-taking and global tests. Primary means that during data-taking and tests the operations team supports ATLAS Computing Shifts and provides help in case of data-transfer failures, contacts networking and computing experts to solve the problems with data transfer channels, storage system, disk space, daily checks of data transfer logs, defines tools necessary for DDM operation and monitoring and communicates with the DQ2 development team.

5. Conclusion and future work

DQ2 v0.2.10 was put into production June 2006. It featured a large rewrite of the site services and a complete change of site services database schema as well as prototype of the monitoring service. DQ2 v0.2.12 followed soon afterwards in production in September 2006. The main focus was on more stable site services, in particular agent handling and replica resolution. DQ2 v0.2.13 was put into production before Christmas 2006. It featured the migration to the new monitoring service, better source discovery and transfer retrials. DQ2 Version 0.3 was deployed for testing in mid-February 2007 and moved into Production on June 19 2007. The main difference from 0.2 is in the central catalogues where we move away from the POOL FC interface and use direct Oracle interfaces. The REST interface to the central catalogues has changed significantly, using a single RPC endpoint per catalogue. We have added server side logging, a lot more validation and means of measuring performance metrics. Site services also profit from faster diff-queries to the content catalogue. Site services have also changed with improved transfer retrials, proritization and a much better set of interactions with the ARDA monitoring. DQ2 Version 0.4 is moving towards release. Nevertheless it is used in testing already for all LCG sites since it has proven to be able to cope better with higher subscription loads. It introduces

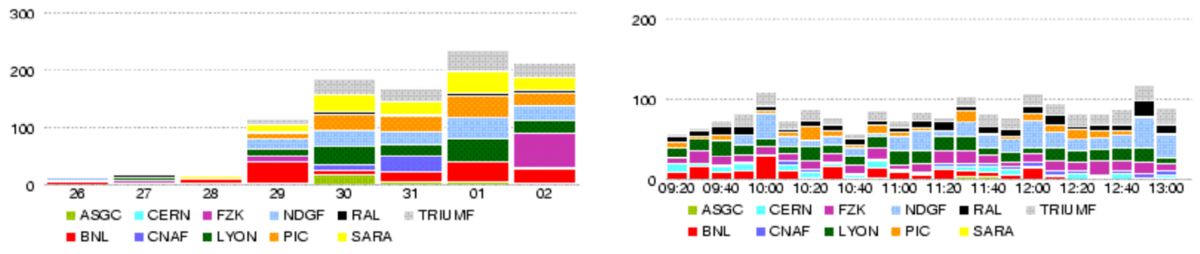

Figure 3. Ramp-up of M4 cosmics runs and production rates (MB/sec.)

completely new database interactions by removing the SQLObject dependency and changes to the transfer logic.

The Tier-0 Export Exercise during Service Challenge 4 where CERN as Tier-0 shipped simulated data to all Tier-1 centres globally showcased the systems capability. During this month-long challenge beginning April 2007, many factors in micromanagement of data-transfers were discovered which helped to the milestone of sustaining file transfer rates at 1 GB/sec. Figure 3 on the right shows day-by-day average production rates. It is therefore ready for Tier-0 Export. Additionally, another metric was introduced, namely the number of files transferred, in accordance with the usage of standard production. Here, the size of files is the bottleneck so high throughput cannot be achieved. Nevertheless the system has shown that it can handle transfers in the order of O(10000) files per site per day and is therefore ready for production. In September 2007 the cosmics M4 export run proved that the whole dataflow works, as shown in figure 3 on the left. About two million muons were measured over two weeks using the ATLAS detector and acquired data was shipped to facilities worldwide using DQ2, reprocessed, analysed and transferred again for archiving. The dataflow was successful without human intervention.

Therefore it can be concluded that DQ2 is capable up to and beyond the requirements of the ATLAS Computing Model and can be considered feature-complete, high-performance and reliable.

6. Acknowledgements

We gratefully acknowledge the contribution of Torre Wenaus to the design of DDM.

References

[1] Miguel Branco, David Cameron, Pedro Salgado, DDM Scenarios and Principles, CERN Technical Report, Revision 1.6, 2006
[2] Miguel Branco, David Cameron, Pedro Salgado, DDM High-level User Manual, CERN Technical Report, Revision 1.1, 2006
[3] Miguel Branco, David Cameron, Pedro Salgado, DDM Design and Implementation, CERN Technical Report, Revision 1.3, 2006
[4] DDM Review Board, Review, CERN Technical Report, June 2007
[5] ATLAS Computing Technical Design Report, CERN Technical Report, July 2007, http://atlas-projcomputing- tdr.web.cern.ch/atlas-proj-computing-tdr/PDF/Computing-TDR-final-July04.pdf
[6] ATLAS Computing Model, CERN Technical Report, July 2005
[7] ATLAS - The data chain works, International Science Grid This Week, 19th September 2007, http://www.isgtw.org/?pid=1000673
[8] Ricardo Rocha et al., Monitoring the ATLAS distributed data management system, Journal of Physics: Conference Series, IOP Publishing Ltd., Victoria BC Canada, 2007
[9] Dario Barberis et al., ATLAS Distributed Data Management Operations, CERN Technical Report, ATLAS Note draft, 03 August 2006