Advances in Service and Operations for ATLAS Data Management

Graeme A Stewart[1], Vincent Garonne[1], Mario Lassnig[1], Angelos Molfetas[1], Martin Barisits[1], Donal Zhang[2], Ivan Calvet[1], Thomas Beermann[1], Fernando Barreiro Megino[1], Andrii Tykhonov[3], Simone Campana[1], Cedric Serfon[4], Danila Oleynik[5], Artem Petrosyan[5] for The ATLAS Collaboration

[1]CERN, CH-1211, Genève 23 Switzerland
[2]Institute of High Energy Physics, Chinese Academy of Sciences, Beijing, China
[3]Jožef Stefan Institute, Ljubljana, Slovenia
[4]Ludwig-Maximilians-Universität München, Munich, Germany
[5]Joint Institute for Nuclear Research, Dubna, Russia

E-mail: graeme.andrew.stewartcern.ch

Abstract. ATLAS has recorded almost 5PB of RAW data since the LHC started running at the end of 2009. Many more derived data products and complimentary simulation data have also been produced by the collaboration and, in total, 70PB 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).

DQ2 has evolved rapidly to help ATLAS Computing operations manage these large quantities of data across the many grid sites at which ATLAS runs and to help ATLAS physicists get access to this data. In this paper we describe new and improved DQ2 services:

- Popularity service, which measures usage of data across ATLAS.
- Space monitoring and accounting at sites.
- Automated exclusion service.
- Cleaning agents, which trigger deletion of unused data at sites.
- Deletion agents, to reliably delete unwanted data from sites.

We describe the experience of data management operation in ATLAS computing, showing how these services enable management of petabyte scale computing operations.

We illustrate the coupling of data management services to other parts of the ATLAS computing infrastructure, in particular showing how feedback from the distributed analysis system in ATLAS has enabled dynamic placement of the most popular data, helping users and groups to analyse the increasing data volumes on the grid.

1. Introduction

The ATLAS experiment[1] at the Large Hadron Collider (LHC) is a general purpose particle physics detector designed to investigate physics at the energy frontier. The ATLAS detector is capable of operating at the 20MHz collision rate of the LHC, however, online trigger systems reduce the rate of data taking to offline to 200-400Hz. Even with this drastic reduction ATLAS records a huge amount of data – more that 5PB of raw collision data have been taken since the LHC started running. This data is first processed by the ATLAS Tier-0[2], which runs first

pass reconstruction of events and writes the data to tape at CERN. It then registers this data in the ATLAS Distributed Data Management system (DDM)[3]. DDM organises, transfers and manages ATLAS data across more than a hundred individual grid sites that are part of the Worldwide LHC Grid[4] in accordance with the policies established in the ATLAS Computing Model[5]. This not only covers raw data, but the entire lifecycle of derived data products for the collaboration physics groups and individual physicists.

In this paper we give an overview of ATLAS Distributed Data Management, covering core concepts (§2), performance with LHC data (§3) and then, in some detail, new services and features (§4).

2. ATLAS Distrubuted Data Management

2.1. Data Model

ATLAS data, like most high energy physics experiments, mostly consists of persistified C++ objects. These objects are stored together in files, usually corresponding to many detector events. DDM considers files to be its elemental unit (thus, DDM has no knowledge of sub-file structures). However, files themselves are usually defined by operational considerations and rarely correspond to a complete set of data of interest to the user of the system. For this reason DDM allows the aggregation of files into *datasets*. Datasets are the operational unit of replication for DDM – they may be transferred to grid sites, whereas single files may not. Datasets in DDM may contain files that are in other datasets, i.e., datasets may overlap. This is illustrated in Figure 1.

Datasets may be *open* (able to have new files added to them) or *closed* (no new files may be added). Closed datasets may have new versions created, which might have a different file content; however, datasets which are *frozen* cannot have files added or new versions created. These datasets are immutable (excepting that files which have been permanently lost can be removed).

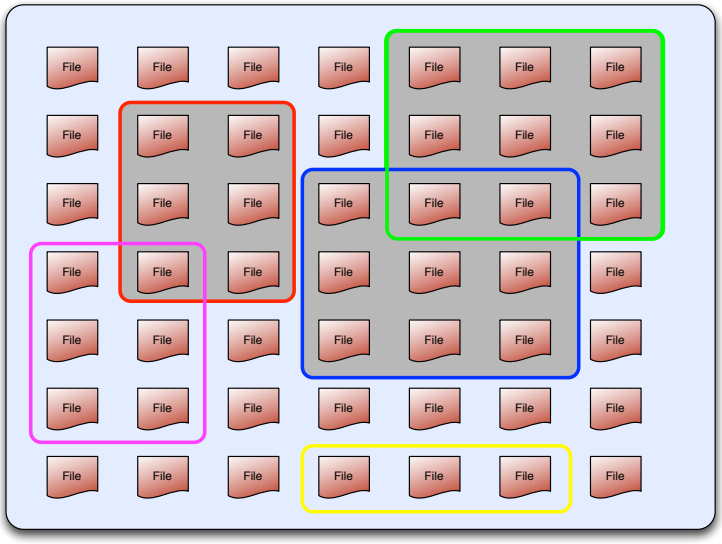


Figure 1. ATLAS Data Management core concepts. Individual files are collected into datasets (coloured outlines). Datasets may overlap and contain the same files (e.g., green and blue). Datasets themselves may be aggregated into containers, e.g., the red, blue and green datasets form the grey container.

There is a further level of aggregation provided by DDM. This is the concept of a *container*,

which is a collection of datasets (see Figure 1). Containers are not units of replication, but allow large blocks of data, which could not be replicated to single site, to be described in the system.

In practice, most dataset overlaps and aggregation into containers are hierarchical: e.g., monte-carlo production will use small datasets, referring to a few jobs processed at a site; these are then aggregated into the main dataset, which refers to a particular task in the ATLAS production system (all of the small datasets overlap with this main dataset). Then these main datasets may be added to a container, where the output of several similar simulation tasks is gathered.

2.2. Data Management Responsibilities

The ATLAS Distributed Data Management system is charged with managing and organising ATLAS data for the collaboration. In particular this means:

- Registering and cataloging ATLAS data.
  - Registration of datasets and containers.
  - Registration of files into datasets.
  - Registration of datasets into containers.
- Transferring data between sites.
  - Registering the data transfer request.
  - Allowing requests to be queried and cancelled.
- Delete replicas from sites.
- Ensure dataset consistency on sites.
  - In particular manage file loses on sites.
- Enforce ATLAS Computing Model policies.

The current implementation of DDM is called DQ2 (*Don Quixote 2*).

2.3. DQ2 Architecture

The architecture of DQ2 is based on a service stack – clients are at the top and use a well defined API to inteact with the system. Internally, the system core utilises an Oracle database for all state. The access points for the central serices are stateless apache web servers, which then talk to the database, which allows horizontal scaling of this component. To interact with the grid, the *Site Services* components shield the central services of the system from dependencies on any particular grid implementation or technology. However, there are certain basic common libraries which are shared between DQ2 components. This is illustrated in Figure 2.

2.4. Scope and Users

As ATLAS is a collaboration of many thousands of physicists, managing data across hundreds of sites and running hundreds of thousands of jobs a day, DDM has a wide base of clients. The majority of calls to DDM (see §3) are from other ATLAS systems, e.g., the ATLAS PanDA workload management system[6]. Thus, a few clients which produce the majority of the DDM load. However, DDM also supports a user base of thousands of individuals: about 500 unique users each day, 1000 each week and 1500 every month.

3. Scaling and Performance

Since the start of LHC data taking the total amount of data managed by DQ2 has grown steadily, from about 10PB in 2008 to more than 70PB in summer 2011 (Figure 3).

Likewise, in Figure 4, the evolution of the total number of managed files is shown.

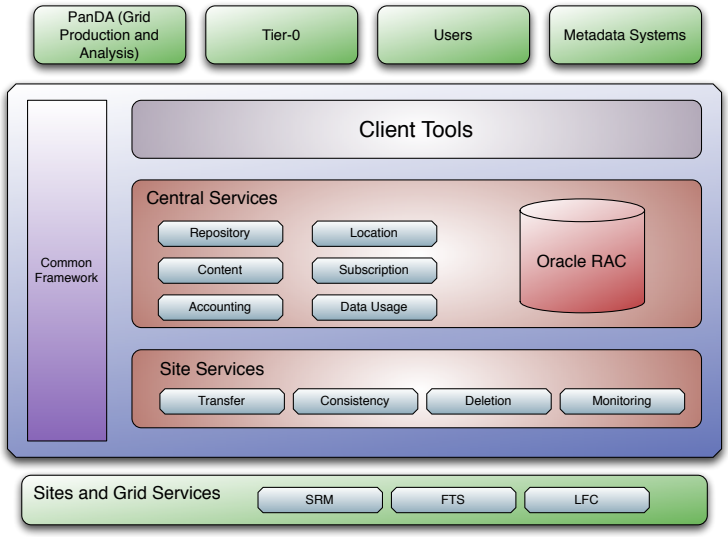


Figure 2. Schematic illustration of DQ2 architecture.

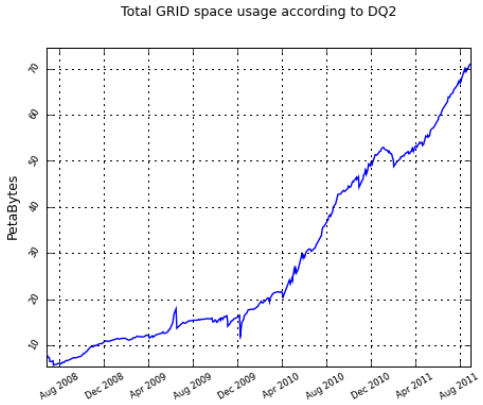


Figure 3. Total size of data managed by DDM from LHC statup. Note the sharp increase in data taking rate once the LHC centre of mass energy reached 7TeV in spring 2010.

The figures show that the current implementation is scaling with data volume. However, this scaling is achieved at the cost of considerable tuning efforts, especially in the area of interactions with Oracle. Indeed, to achieve sufficient performance with Oracle, a de-normalised schema has had to be adopted, where columns are replicated using triggers in order to reduce the number of table joins needed for popular queries. The queries also require significant numbers of Oracle hints to ensure that the database executes them in an efficient manner.

Currently DQ2 has a load of about 14M reads and 0.6M writes per day.

As the amount of data managed by DQ2 has increased, so have the transfer rates of data around the grid. In Figure 6 it can be seen that transfer rates average 2GB/s continuously for ATLAS. Peaks of up to 10GB/s have been observed after reprocessing campaigns, when large amounts of data need to be replicated across the grid.

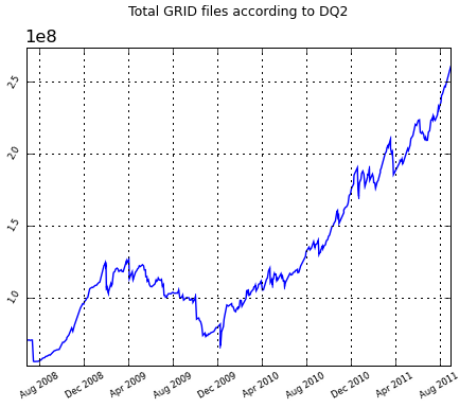


Figure 4. Total number of files managed by DDM from LHC statup. Note the sharp increase in data taking rate once the LHC centre of mass energy reached 7TeV in spring 2010.

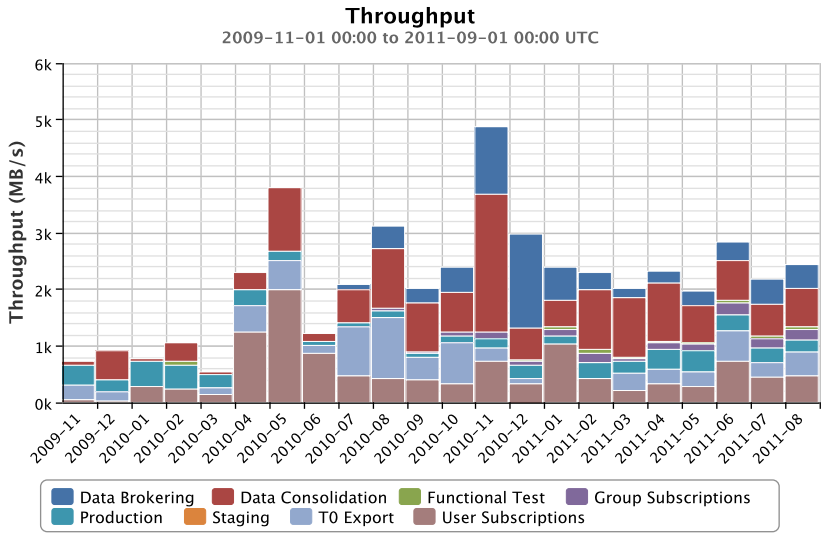


Figure 5. ATLAS grid data transfer rates for different activities since LHC startup. An increase in data rate is observed once 7TeV data taking began.

4. New Services and Features

As DQ2 has evolved to become a more complete system for managing ATLAS data throughout its life a number of new services have been introduced, which we now describe.

4.1. Tracer

One of the key problems facing a large collaboration like ATLAS in an inhomogeneous resource scarce environment is to determine which data should be widely replicated because it is, or it likely to be, popular and which data should replicated less.

In order to achieve an objective measure of this the DDM team introduced the tracer system[7, 8]. The tracer has an API library that is used by DQ2 clients to send a callback when data is accessed on the grid. Data pertaining to the dataset, file, site (both local and remote), user, activity and timestamps are included in the tracer message, which is sent via http. These messages are collected via the tracer service, which then inserts them into the DQ2

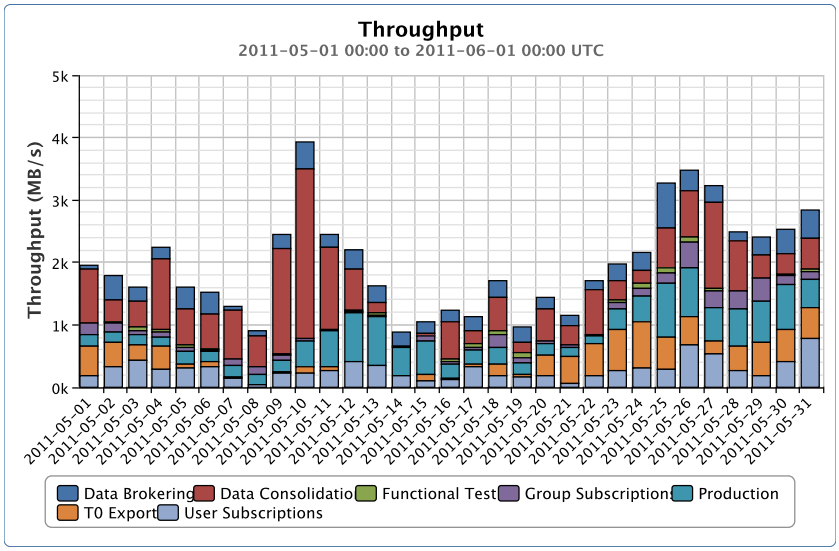


Figure 6. Data transfer rates per activity for May 2011. The system has a continuous substantial workload.

database.

The tracer service is optimised to accept a high rate of data insertion (more than 5M per day) and recent work has investigated the use of the Cassandra NoSQL database to buffer insertions before doing a bulk insert of data into Oracle. This helps the system to scale.

4.2. Popularity

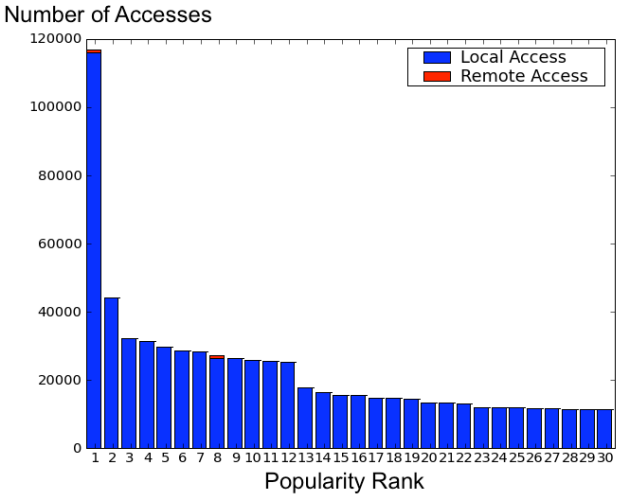
The DQ2 tracer keeps a very fine granularity for data access – each single access results in multiple database row entries. This makes it unsuitable for querying directly, as data aggregation would be very slow. For this reason the DQ2 *popularity* service[7] was introduced. This service runs over the tracer data each day and provides summaries of data access per local or remote site, user, dataset, etc.

The popularity service can be queried by clients via a web interface or an API to discover popular and unpopular data on sites (e.g., see Figure 7). When taken globally such information is valuable for knowing what data might be deleted across the grid in order to free up space on sites. This is used to form part of the ATLAS's dynamic data placement system. The ATLAS PanDA workload management system[6] measures the instantaneous popularity of data based on the jobs that users have submitted and, if the number of replicas held in the system is below a threshold, extra replicas are created via a DDM subscription request. Conversely, the popularity service is used to decide which replicas of unpopular data to delete at each site, allowing more popular data to be sent to replace it.

4.3. Deletion

While deletion might seem like a fairly straight-forward 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.



Dataset listing

No	Dataset	Accesses
1	mc11_7TeV.105200.T1_McAtNlo_Jimmy.merge.AOD.e835_s1272_s1274_r3043_r29	116916
2	mc11_7TeV.107650.AlpgenJimmyZeeNp0_pt20.merge.AOD.e835_s1299_s1300_r30	44277
3	mc09_7TeV.105830.JF17_herwig_jet_filter.merge.AOD.e507_s765_s767_r1215	32284
4	mc11_7TeV.107660.AlpgenJimmyZmumuNp0_pt20.merge.AOD.e835_s1299_s1300_r	31253
5	data11_7TeV.00191190.physics_Muons.merge.AOD.f413_m1019	29835
6	mc11_7TeV.107670.AlpgenJimmyZtautauNp0_pt20.merge.AOD.e835_s1299_s1300	28593
7	mc11_7TeV.106046.PythiaZee_no_filter.merge.AOD.e815_s1272_s1274_r3043_	28211
8	user.olezenin.mc11_7TeV.126185.Pythia8Zmumu.merge.AOD.e984_a131_s1353_	27312
9	mc11_7TeV.126184.Pythia8Zee.merge.AOD.e984_a131_s1353_a141_r2900_tid59	26518

Figure 7. Results of a typical query to the popularity service, giving the grid-wide popularity of AOD datasets over a 30 day period, ranked by number of accesses.

Each of these steps might fail, so it is necessary for the deletion service to have an internal state engine that 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 can reach significant levels, with millions of file deletions per day and terabytes of data being cleaned. Figure 8 shows a typical example.

Analysis of this level of deletion shows that it primarily arises from the use of intermediate files in processing chains. e.g., simulation can only produce small detector hits files as jobs are limited by cpu/runtime considerations to only process about 50 events; these small hits files are then merged to larger files and afterwards the original hits are deleted. A considerable load also comes from the deletion of user outputs from transient disk areas.

4.4. Consistency

Grid operations across the large number of sites used by an experiment such as ATLAS are far from trivial (see, e.g., [9]). Data loss at sites occurs frequently and recovery from this state must be an automated process to be efficacious. In addition consistency problems can occur between the different catalog layers in the system, which need to be rectified.

The consistency service[10] has been developed in order to achieve this goal. This service operates by allowing storage URLs which have been lost to the site to be declared via a dedicated

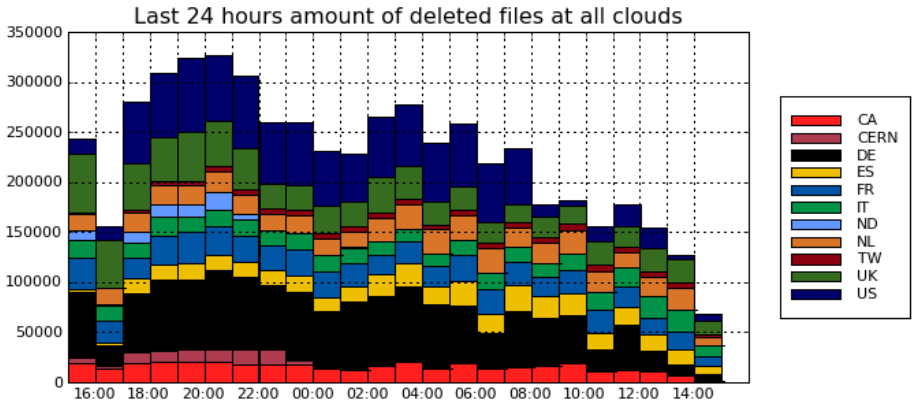


Figure 8. Deletion across the ATLAS clouds over a 24 hour period. Deletion rates of more than 300k per hour are frequent.

API. The consistency service will then set about re-establishing the consistency of the data on the site by:

- Declaring affected datasets on a site to be *incomplete*, so that they will not be used for data processing.
- Cleaning the storage and file catalog namespaces of lost files.
- Re-subscribing files which have other replicas elsewhere back to the site.
- Deleting files which have no other replicas from the definition of the affected datasets (thus permanently lost).
- Declaring datasets to be *complete* again as a site after consistency is re-established.

The consistency service also keeps records of which files have been lost and which recovery actions have been taken. This allows a historical view of data loss on the grid and for sites with particular problems to be identified.

4.5. Accounting

Knowledge of how storage space is used 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., data type (RAW, ESD, AOD) is orthogonal to project (data11_7TeV, mc10), which are both orthogonal to accounting by the data owner.

For this reason the 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 `'data10.*.ESD.*' + 'CERN'` can be expressed as `{'project':'data10', 'type':'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. In this way historical data will be built up and trends can be analysed.

An initial implementation is available based on an Oracle backend, but an assessment of NoSQL backends, which are inherently more suitable to such key-value stores, is underway. This approach would offer the additional benefit of reducing the overall load on Oracle.

5. Conclusions and Future Directions

The ATLAS Distributed Data Management project has provided ATLAS with a functioning system in which to organise and manage data during LHC running. New services have been introduced to help automate the data lifecycle and to improve the robustness of the system to real operating conditions. Introducing and managing these new features, while keeping the system stable have taken considerable and continual efforts of developers and database experts, but the multi-petabytes of data moved and stored shows that the system has scaled well to date.

However, it is also true that some conceptual and design limitations have arisen in the current system, which now hamper future development. For example, dataset versions (§2.1), while supported, are largely unused and have not proved to be really necessary; however, the inclusion of versioning in the current schema introduces an indirection between a dataset and its constituent files that requires a table join with a consequent performance impact in the database layer. Therefore the ATLAS DDM group are undertaking a review of DQ2, looking forward to the long LHC shutdown of 2013-2014. A new version of DDM, *Rucio*, is planned for 2013 to take ATLAS forward into the next years of high luminosity LHC running.

References

[1] The ATLAS Collaboration 2008 *Journal of Instrumentation* **3** S08003 URL `http://stacks.iop.org/1748-0221/3/i=08/a=S08003`

[2] Elsing M, Goossens L, Nairz A and Negri G 2010 *Journal of Physics: Conference Series* **219** 072011 URL `http://stacks.iop.org/1742-6596/219/i=7/a=072011`

[3] 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`

[4] Knobloch J et al 2005 LHC Computing Grid Technical Design Report Tech. Rep. CERN-LHCC-2005-024 CERN

[5] Jones R and Barberis D 2008 *Journal of Physics: Conference Series* **119** 072020 URL `http://stacks.iop.org/1742-6596/119/i=7/a=072020`

[6] Maeno T, De K, Wenaus T, Nilsson P, Stewart G A, Walker R, Stradling A, Caballero J, Potekhin M, Smith D and Collaboration T A 2011 *Journal of Physics: Conference Series* **331** 072024 URL `http://stacks.iop.org/1742-6596/331/i=7/a=072024`

[7] Molfetas A, Megino F B, Tykhonov A, Lassnig M, Garonne V, Barisits M, Campana S, Dimitrov G, Jezequel S, Ueda I and Viegas F T A 2011 *Journal of Physics: Conference Series* **331** 062018 URL `http://stacks.iop.org/1742-6596/331/i=6/a=062018`

[8] Lassnig M, Garonne V, Branco M and Molfetas A 2010 *Journal of Physics: Conference Series* **219** 062054 URL `http://stacks.iop.org/1742-6596/219/i=6/a=062054`

[9] I Ueda and the ATLAS collaboration 2011 *Journal of Physics: Conference Series* **331** 072034 URL `http://stacks.iop.org/1742-6596/331/i=7/a=072034`

[10] Serfon C, Garonne V and The ATLAS Collaboration 2011 *Journal of Physics: Conference Series* **331** 072065 URL `http://stacks.iop.org/1742-6596/331/i=7/a=072065`