Popularity framework to process dataset traces and its application on dynamic replica reduction in the ATLAS experiment

Angelos Molfetas, Fernando Barreiro Megino, Andrii Tykhonov, Mario Lassnig, Vincent Garonne, Martin Barisits, Simone Campana, Gancho Dimitrov, Stephane Jezequel, Ikuo Ueda, Florbela Tique Aires Viegas

CERN, Geneva, Switzerland
Jožef Stefan Institute, Ljubljana, Slovenia
Deutsches Elektronen-Synchrotron, Hamburg, Germany
Vienna University of Technology, Vienna, Austria
LAPP, Université de Savoie, CNRS/IN2P3, Annecy-le-Vieux, France
University of Tokyo, Tokyo, Japan

E-mail: angelos.molfetas@cern.ch

Abstract. The ATLAS experiment’s data management system is constantly tracing file movement operations that occur on the Worldwide LHC Computing Grid (WLCG). Due to the large scale of the WLCG, statistical analysis of the traces is infeasible in real-time. Factors that contribute to the scalability problems include the capability for users to initiate on-demand queries, high dimensionality of tracer entries combined with very low cardinality parameters, and the large size of the namespace. These scalability issues are alleviated through the adoption of an incremental model that aggregates data for all combinations occurring in selected tracer fields on a daily basis. Using this model it is possible to query on-demand relevant statistics about system usage. We present an implementation of this popularity model in the experiment’s distributed data management system, DQ2, and describe a direct application example of the popularity framework, an automated cleaning system, which uses the statistics to dynamically detect and reduce unpopular replicas from grid sites. This paper describes the architecture employed by the cleaning system and reports on the results collected from a prototype during the first months of the ATLAS collision data taking.

1. Introduction
The ATLAS experiment [1] uses the WLCG to store and process data in numerous sites around the world. The experiment’s distributed data management operations are undertaken by the DQ2 system [2], which manages petabytes of experiment data on over 750 storage end points. DQ2 is built on top of the EGEE/EMI gLite [3], NorduGrid/EMI ARC [4] and Open Science Grid (OSG) [5] middleware. The DQ2 system provides a middleware interface allowing users to interact with the data on the grid without requiring an awareness of the heterogenous systems that make up the grid. Furthermore, the DQ2 system maintains a central catalogue of all data stored on the ATLAS grid, which is organised into logical collections of files called datasets. DQ2
ensures consistency between the central catalog, local file catalogs in sites, and storage systems. Datasets that are replicated on sites in the grid are referred to as *replicas*.

This paper describes the DQ2 popularity framework, and demonstrates its implementation as a DQ2 subsystem responsible for providing usage statistics on datasets on the grid. Furthermore, this paper demonstrates an application of this system, a new DQ2 service, the replica reduction agent, which is responsible for removing unpopular replicas from the grid.

The ATLAS experiment has a large number of participants, large namespace for collected data, and applications that generate high numbers of data events on the grid. The motivation of this work is to provide a facility to query user activity occurring on the grid and use it as feedback to optimise data distribution for the experiment.

The paper continues as follows: Section 2 discusses the DQ2 tracer system which collects raw usage data occurring on the grid. Section 3 describes DQ2 popularity. The paper then discusses an application of DQ2 Popularity, the Replica Reduction Agent (section 4), before concluding.

### 2. DQ2 Tracer system

All applications that move data on the ATLAS grid use a tracer client to submit a trace for every file transfer event occurring between hosts to a central web service. The trace information for an event is inserted into a hash table, which is transmitted by a tracer client to the central tracer web service. A list of attributes and corresponding descriptions is defined in table 1.

As numerous traces originating from different end points on the grid all converge on the central tracer web service, this causes concurrent tracer insertions into the database. This concurrency is managed by the web service, by placing all tracer requests into a queue, which feeds a session pool, thus limiting the number of concurrent insertions occurring on the database. Today, 52 traces per second are inserted into the database.

| Attribute     | Description                                                                 |
|---------------|-----------------------------------------------------------------------------|
| UUID          | Unique identifier shared by traces originating from the same operation        |
| Event Type    | Type of application initiating data transfer, e.g. an analysis job           |
| Tool Version  | Release version of the grid application                                      |
| Remote Site   | Site where data is being downloaded/uploaded to                             |
| Local Site    | Site where data is being downloaded/uploaded from                           |
| Time Start    | Time operation started                                                       |
| Time End      | Time operation ended                                                         |
| DUID          | Unique dataset identifier                                                    |
| Version       | Version of dataset                                                          |
| Client State  | Final state of client; either done or error code                             |
| File Size     | Size of file                                                                 |
| GUID          | Unique identifier for file                                                   |
| Catalog Time  | Time stamp when file catalogs were queried                                   |
| Transfer Time | Time stamp when first byte arrives                                           |
| Validation Start | Time stamp when validation began                                        |
| Application ID| Operation ID, assigned by the application                                    |
| Hostname      | Name of host                                                                |
| IP            | Host’s IP address                                                           |

Table 1. DQ2 Tracer attributes and corresponding descriptions.
3. Popularity
As all processing operations on the grid require data movement, the popularity system can be used to determine how often datasets were used in a given time period, and can provide this information in terms of dataset name, remote site, local site, and user. Even though the tracer service collects information at the file event level, it also records an operation’s unique identifier (see table 1), allowing popularity to report not only on how many file requests were performed, but also how many distinct grid operations were conducted for a dataset or set of datasets.

Figure 1 shows the interaction of DQ2 Popularity with the rest of the DQ2 System.

![Figure 1. Popularity and the Replica Reduction Agent.](image)

The popularity service collects data for a number of attributes (dataset name, day, eventtype, duid, version, local site, remote site, user). Thus, a user can request how many times datasets were requested on the grid using a number of search criteria; e.g. by specifying a particular remote site or application. In addition, the system calculates these statistics for all existing combinations of the search attributes. This allows the user to launch queries involving any number of attributes. For example, a user can request the popularity for datasets that have a dataset name matching the wild card string `data10.7TeV.*` and used by a certain application.

It is not feasible to directly query the tracer database for dataset popularity statistics due to the large number of traces that are collected. The high dimensionality of tracer entries means that establishing indexes for all searchable column uses a lot of database storage. Furthermore, the low cardinality of parameters in some columns renders many b-tree indexes ineffective.

To resolve this problem, the popularity framework aggregates traces using a daily granularity. In addition, since traces are collected at the file level, the popularity framework is able to further reduce the number of rows inserted into the database by aggregating all transactions at the dataset level. For example, if a particular dataset is downloaded on ten different occasions (between two sites, by a specific user) resulting in 10,000 file transfers, this will create 10,000 rows in the tracer system. In the popularity system, however, this will aggregated into a single
row with two additional column entries, specifying that there were ten requests for this dataset and 10,000 file transfers (for the given set of attribute parameters).

Figure 2 illustrates a simplified example of how multiple traces are aggregated into popularity entries. In this example, the first three rows share the same dataset, local site, remote site, and user, and consequently these three traces are combined into a single popularity entry. As all three rows share the same operational id, they all belong to the same transaction. Thus, in the popularity row the number of dataset events is set to 1. The number of files parameter is set to three, as there are three different files downloaded. The following three rows also share common parameters (dataset, local site, remote site, and user) and involve the movement of three files on the grid. However, there are two separate operation ids, so they are two separate transactions. Consequently, as these rows get aggregated, the number of dataset events is set to two instead.

4. Replica Reduction Agent
Figure 3 shows the evolution of storage space usage of large (Tier 1) sites at a critical time in the ATLAS experiment. The upper green line shows the available space and the stacked coloured plot is the used space. The figure demonstrates that storage resources have been rapidly used in the ATLAS grid, the total of which is illustrated by the blue line which shows the total pledged space by the sites. The Replica Reduction Agent was developed to free storage space on the grid by deleting unpopular datasets from sites that have low available space.

Replicas on the grid are classified into two broad categories – primary and secondary replicas. Primary replicas on the grid are mandated by static policy, which define replication factors for types of experimental data. Secondary replicas are datasets that have been replicated in addition to what is mandated by the static policy to facilitate its access for analysis, and consequently are
capable of being deleted in case of lack of free space to host new data. The Replica Reduction Agent improves the usage of storage resources by removing unpopular replicas from the grid. It acquires the replica popularity information from the DQ2 popularity system. It facilitates data analysis at sites by allowing generous data distribution while keeping accessible space for data that has proven to be popular for analysis.

![Cumulative evolution for DATADISK by site (SRM)](image_url)

**Figure 3.** Available evolution of storage usage at large (T1) sites in the ATLAS Grid. The upper green line shows the available space and the stacked, colored plot is the used space.

### 4.1. Method

The replica reduction system has to perform three steps, as illustrated in figure 4:

(i) Selection of sites where disk space is nearly full
(ii) Selection and deletion of unpopular secondary datasets
(iii) Publication of actions

#### 4.1.1. Selection of full site

The storage space information for each site, used and available storage space is retrieved from the DQ2 Accounting subsystem and it is checked if the used space exceeds certain thresholds. This allows datasets, even if they are unpopular, to remain on sites in case they are required by analysis processing jobs at a future date, and are only deleted if free space is required. This model discourages large amounts of unused space in ATLAS grid sites. Operational experience has shown that the threshold has to be set between 10% and 24%, depending on the endpoint type and the activity that it serve in order to insure efficient utilisation of storage resources. The Replica Reduction Agent has been implemented in a flexible manner allowing thresholds to be set to targeted storage elements in specific sites if required.

#### 4.1.2. Selection and deletion of unpopular secondary datasets

The next step is to select the replicas that are going to be deleted from the full sites. This is done by an incremental algorithm, where numerous deletion checks are made, each being more permissive and allow the deletion of more popular datasets until the site can be cleaned or a maximal popularity parameter has been reached. The datasets in each step are sorted from older to newer replica creation date and the cleaning is constrained to secondary replicas with creation dates older than 15 days. Replicas need to be at the site for a minimum period to establish a level of popularity. Deletion of datasets is prevented if they are currently in use.

Once replica deletion list has been generated, it is passed to the DQ2 deletion service that will take care of scheduling the physical deletion of the files after a certain grace period.
4.1.3. Publication of actions Once the cleaning agent has performed the actions, they have to be published to inform the central operations team and the different cloud teams. At the moment a web page is generated on the fly, but as the use of the system is spreading and historical statistics and more advanced monitoring is required, future plans include the generation of dynamic web pages that on demand query the information from the DDM Central Catalogues and, if needed, from a dedicated database backend.

![Dynamic Replica Reduction Agent](image_url)

**Figure 4.** Dynamic Replica Reduction Agent.

5. Conclusion

This paper demonstrated DQ2 Popularity which is a system for calculating usage statistics for sets of data on the grid. This paper also demonstrated an application based on DQ2 Popularity, a replica reduction agent which reduces the number of replicas on the grid.

Popularity maintains scalability by reducing the large number of entries generated by the tracing system by aggregating traces at the dataset level with a daily granularity. The popularity framework in the ATLAS experiment has proven to be a very useful at ascertaining the popularity of the datasets in the ATLAS grid, and is currently being used to reduce secondary replicas.

Future work involves utilising the popularity framework in the ATLAS grid for the inverse purpose to the application described in this paper, that is, develop new replica distribution models based on dataset popularity to further optimise the use of storage and network resources.

References

[1] Jones R and Barberis D 2008 *Journal of Physics (Conference Series)* vol 119 (IOP Publishing Ltd) chap The ATLAS computing model

[2] Lassnig M, Garonne V, Branco M and Molfetas A 2008 *IOPscience* 119

[3] Laure E, Hemmer F, Prezl F, Beco S, Fisher S, Livny M, Guy L, Barroso M, Buncic P, Kunszt P Z, Di Meglio A, Aimer A, Edlund A, Groep D, Pacini F, Sgaravatto M and Mulmo O 2004 *Computing in High Energy Physics and Nuclear Physics* 2004 p 826 URL http://portal.acm.org/citation.cfm?id=1232286.1232292

[4] Ellert M, Gronager M, Konstantinov A, Kόnya B, Lindemann J, Livenson I, Nielsen J L, Niinemäki M, Smirnova O and Wiinblad A 2007 *Future Generation Computer Systems* 23 219–240 URL http://portal.acm.org/citation.cfm?id=1232286.1232292

[5] Pordes R, Petravick D, Kramer B, Olson D, Livny M, Roy A, Avery P, Blackburn K, Wenaus T, Würthwein F, Foster I, Gardner R, Wilde M, Blatecky A, McGee J and Quick R 2007 *Journal of Physics: Conference Series* 78 011345