C3PO - A Dynamic Data Placement Agent for ATLAS Distributed Data Management

T Beermann¹, M Lassnig¹, M Barisits¹, C Serfon² and V Garonne²
on behalf of the ATLAS Collaboration

¹ CERN, Geneva, Switzerland
² University of Oslo, Norway
E-mail: thomas.beermann@cern.ch

Abstract. This paper introduces a new dynamic data placement agent for the ATLAS distributed data management system. This agent is designed to pre-place potentially popular data to make it more widely available. It therefore incorporates information from a variety of sources. Those include input datasets and sites workload information from the ATLAS workload management system, network metrics from different sources like FTS and PerfSonar, historical popularity data collected through a tracer mechanism and more. With this data it decides if, when and where to place new replicas that then can be used by the WMS to distribute the workload more evenly over available computing resources and then ultimately reduce job waiting times. This paper gives an overview of the architecture and the final implementation of this new agent. The paper also includes an evaluation of the placement algorithm by comparing the transfer times and the new replica usage.

1. Introduction
The ATLAS [1] collaboration is one of the four major experiments at the Large Hadron Collider at CERN. The detector, as well as the Monte Carlo simulations of physics events, create vast amounts of data that are distributed across the Worldwide LHC Computing Grid [2]. To be able to manage this data, originally Don Quijote 2 (DQ2) [3], the collaboration’s distributed data management system, was developed and it has run since before the detector started data taking in 2008. A new data management system, called Rucio [4], has been developed and has taken over from DQ2 in the end of 2014 to handle the new requirements for LHC Run 2. The work presented in this paper is an extension of Rucio. Rucio is responsible for over 250PB of experiment data across 130 sites all over the world. The responsibilities of Rucio include the organisation and management of primary detector and simulation data, but also all of the derived physics data that is used by the collaboration’s physicists. The data is distributed over the grid according to the ATLAS Computing Model [5]. If users want to analyse this data they have to send jobs to the workload management system (WMS), called PanDA [6], which schedules the jobs at the sites where the data is available.

This contribution presents a newly developed tool, called C3PO, for the ATLAS data management that dynamically creates new replicas that then can be used by the workload management system in future scheduling decision, helping spreading the workload more evenly over existing computing resources. This paper has several parts: First, an overview of the ATLAS data management is given followed by a description of what dynamic data placement
is and how the system can benefit from it. Next, the architecture is described as well as the workflow of the currently used placement algorithm. Then, the next section shows how the monitoring of the system is implemented. The paper finishes with a brief discussion of first results and an outlook for future improvements.

2. ATLAS Data Management

The data that is stored on the grid consists of files, which usually contain many physics events. The distributed data management system operates on these files. Events that are taken under the same detector conditions can be spread over several files so that some kind of aggregation is needed. For this reason the concept of datasets exists in Rucio. Datasets are logical units in Rucio that combine one or more files that share some common characteristic (e.g., belonging to the same physics stream and taken in the same LHC fill). Datasets represent the operational unit in Rucio. Even though it is possible to retrieve single files from a dataset or to transfer them between sites, usually complete datasets are used in transfer operations. A set of actual files on a site corresponding to a dataset is called replica. To create new replicas at a site all of the files in a dataset have to be copied to the new site over the grid and will be registered in the system. Therefore, one logical dataset in Rucio can have multiple physical replicas on the grid.

Data transfer requests are registered in Rucio using the replication rule system [7]. If users or systems want to transfer datasets between two endpoints they create a new rule specifying the dataset(s) to be transferred and the destination endpoint. The system then creates the actual transfer and monitors the progress until the file have been successfully transferred.

The way that datasets are spread over the grid is based on policies defined by the Computing Resource Management (CREM), which is guided by the ATLAS computing model and the operational constraints. Based on the age and the type of a dataset the minimum number of replicas is defined and where these replicas have to be stored (Tier-1 or Tier-2, disk or tape).

If a user wants to run a job on the grid it is done in multiple steps involving both the WMS and DDM system. The user defines a job that will run on one or multiple datasets and sends it to the WMS. The WMS asks the DDM system on which sites replicas of the requested datasets are hosted. In the next step the WMS schedules where the jobs will run based on the site’s availability, the current workload of the sites and the jobs’ priority.

EverytimeafileisaccessedthroughPanDA,atracewillbesenttotheDDMtracerystem[8]. The trace contains information about the corresponding dataset, the involved site, the user, the starting and ending time and whether the access was successful or not. One application of this information is the analysis of the popularity of given dataset. Since the system was introduced in 2008 it has already collected more than 7 billion traces, which makes it impractical to use directly. That is the reason for the development of the popularity system [9]. The popularity system aggregates the data from tracer system on a daily basis and provides data that is more tractable.

3. Dynamic Data Placement

On top of the rather static replication policies that make sure that the data is well spread over the grid to make them available for analysis by users, a dynamic data placement can help exploiting computing and storage resources by removing replicas of unpopular datasets and creating extra replicas of popular ones.

During Run-1 PD2P [10] existed inside the PanDA system that waited for incoming jobs and then created new replicas if a threshold of queued jobs was exceeded, taking into account available resources and past dataset popularity. The system presented in this paper extends this concept by also using network metrics to make sure that the new replicas are created quickly. Overall the system is designed in a way that allows for it to be extended easily in the future to also include network metric and popularity forecasts.
On centrally managed data endpoints, there are mainly two different types of replicas: primaries and secondaries. Primary replicas are those that are created based on the beforementioned policies and are protected from automatic deletion. Secondary replicas on the other hand can be deleted at any time when space is needed on a storage element. The Rucio deletion agent, called Reaper, is automatically triggered for centrally managed storage elements when a threshold of either 10% or 400TB of free space is reached. The replicas are first sorted based on their last access date, which is continuously updated with the incoming traces, and their creation date. So it starts with the least recently used data first and deletes them until enough free space is available.

With this deletion policy a dynamic data placement can be achieved by simply adding extra replicas of popular data into free space. As soon as a site becomes full the automatic deletion will make sure to free up space again and the newly created replicas will stay on disk as long as they are popular.

4. Architecture

![Figure 1. Architecture](image.png)

The system for deciding when and where to put new replicas is split in three different types of components:

- **Collectors**: The collectors gather information from different sources and make it available to the redistribution algorithms. Currently, the system supports collectors to get newly defined jobs and their input datasets from PanDA, free space for all storage endpoints, topology information and network metrics.

- **Placement Algorithms**: The placement algorithms can use the data gathered by the collectors to decide when and where to make new replicas.

- **Utilities**: The utilities are tools that provide needed helper functions for the algorithms, e.g., an expiring dataset cache to store which datasets have been newly created, so that they will not be considered again for some pre-defined time.
Figure 1 shows an overview of the architecture and the currently implemented collector components:

- **Job input datasets:** The main collector is the workload management job collector which periodically scans for incoming jobs in the job definition database and extracts the input dataset to make them available for the placement algorithm.

- **Replica information:** This collector retrieves the currently available replicas from the Rucio database.

- **Network metrics:** This collector retrieves bandwidth information and queued file for all available links. The metrics are based on numbers reported by FTS, FAX or perfSonar.

- **Dataset popularity:** This collector retrieves access numbers from the popularity system. The popularity system creates daily aggregations for each dataset. The aggregations are needed because direct queries to the traces would be too slow.

The main component then is the actual placement algorithm that takes the inputs and then makes a decision if and where to put a new replica. This component is exchangeable allowing to test and use different placement strategies. The currently used algorithm is described in the next section.

The last part of the architecture is the output. There are two different kinds of outputs: The first one is the actual Rucio replication rule so that the new replica will be created. The rules are created with a lifetime of 7 days, i.e., during the 7 days the new replica is guaranteed to stay on disk. After that the automatic deletion can delete it if space is needed and it is not accessed anymore. The second output are detailed descriptions of the decision of the algorithm. This information is then written to an Elasticsearch\(^1\) cluster where it then can be used for debugging and monitoring, which will be explained in more detail later.

5. **Workflow**
This section describes the workflow of one of the many possible placement algorithms. This currently used algorithm concentrates on free space and network connectivity between sites, so it weights the sites based on those criteria to find a suitable storage endpoint but also makes sure not to put too much stress on single sites.

(i) The tool constantly scans incoming user jobs and collects the input datasets, the placement algorithm runs for every dataset containing official Monte Carlo or detector data.

(ii) First the algorithm checks if there has already been a replica created in the past 24 hours, if yes it will not continue.

(iii) It then checks how many replicas already exist and exits if a threshold of more than 4 replicas is met.

(iv) The algorithm has a configurable threshold of files and bytes that it can create per hour and day and per destination site, if the datasets would exceed this it will not continue.

(v) It then checks the popularity of the dataset in the last 7 days, if it has not been popular enough it will again exit.

(vi) It will then continue to check network metrics for links between sites having an existing replica and possible destination sites

(vii) The sites are ranked based on free space, bandwidth and queued files and they are down-ranked if a replica of a previous job has been recently created there.

(viii) If a suitable site has been found, the algorithm submits a replication request to Rucio, which will then take care of the transfer.

---

\(^1\) https://www.elastic.co/products/elasticsearch
(ix) In a last step detailed information about this decision is written to Elasticsearch for further analysis.

6. Monitoring

![Figure 2. Kibana Monitoring Overview](image)

The monitoring is based on two parts: The first one is the monitoring of the Rucio rules, which is used to check the status of requested replicas and to make sure that there are no problems during the transfer. The second one uses Elasticsearch and Kibana\(^2\) to visualise the decisions made by the placement algorithm.

Elasticsearch is an analytics engine which makes it possible to easily search for documents and aggregate them. Kibana is a front-end for Elasticsearch which is able to visualise those aggregations. Together those two tools are very helpful to quickly build a monitoring system.

Every time the algorithm goes through the process described in the previous section a document is written to Elasticsearch, which includes different entries depending on whether the algorithm decided to create a new replica or not. There is a common part which is always included and contains:

- Name of the dataset.
- Size of the dataset.
- Number of files in the dataset.
- Number of already existing dataset replicas.
- Past dataset popularity.

\(^2\) [https://www.elastic.co/products/kibana](https://www.elastic.co/products/kibana)
• Name and parameters of the placement algorithm, e.g., maximum number of files / bytes.

Then, depending on whether a new replica was created or not the document contains either of the following entries:

• If decided not to create a new replica:
  – The reason why the replica was not created, e.g., already created a replica in the past 24 hours or hourly threshold is exceeded.

• If decided to create a new replica:
  – The location of the source replica.
  – The location of the new replica.

With this information written to Elasticsearch it is possible to create an overview monitoring page as shown in Figure 2. The overview shows the percentage of created to not created replicas together with a break-down of the reason why a new replica has not been created. Then, there is the volume of the newly created replicas per hour and the number of already available replicas. Next, a break-down of transferred volume per source site and destination site to find possible imbalances and last there is an overview of the total volume, the number of new datasets and the number of transferred files for the monitored time period. Using Elasticsearch and Kibana for this makes it possible to get a live overview of the behaviour of the daemon as well as the possibility to look at specific time frames in the past.

7. First Results

![Figure 3. Transfer times per dataset](image)

The algorithm has been running in a pre-production mode since July 2016 already creating new replicas. Safe-guards have been put in place to make sure that the daemon is not flooding the grid with too many replication requests and that it does not stress the sites. Therefore, a limit of 100TB or 100,000 files per hour has been set for the whole grid. For single sites a
maximum of 50TB or 10,000 files has been implemented. The algorithm has been set to only replicate data to Tier-2 DATADISKs.

The following results are from a test run in August 2016. Overall, during that period the algorithm decided to create 7500 new replicas, which accounts to 1.8PB of data. The plot in Figure 3 shows that the algorithm manages to pick source and destination sites with a good connectivity to be able to create the replicas quickly. Most of the replicas are created in less than 30 minutes, so they are quickly available for the workload management system.

![Figure 4. Accesses per dataset](image)

Of those newly created replicas, 60% have then been used by the WMS. Considering that the current algorithm does not take into account the available computing resources at the site this value is already good. Also when looking at the plot in Figure 4, it can be seen that more than half of the datasets that are accessed are accessed more than once, i.e., the algorithm created replicas that are popular for a longer period of time.

8. Conclusion & Outlook
This paper presented a new tool for the ATLAS distributed data management that helps to spread popular data across the grid to make it available for the workload management scheduling decisions. The results produced in the first test runs are promising and extensibility makes it possible to quickly implement different placement algorithms and evaluate them. This leads to several points that can be improved in future developments:

- The current algorithm includes network metrics and free space for the destination site selection. This can be improved by also taking into account the available computing resources.
- The WMS tries to reschedule jobs after a waiting time of 24 hours. Only after this time the new replicas will be included in scheduling. Instead, a notification could be sent of the
WMS that a new replica has been created. The WMS can then decide if a rescheduling is necessary.

- Currently, the algorithm only has past popularity information available for the decision process. A previous study showed [11] that a forecast of future popularity of datasets is possible. This information could be included for better dataset selection.
- Also the algorithm relies on network metrics for the last 1 hour / 6 hours / 24 hours and currently queued files. Currently, work is undergoing to predict as well the time-to-complete for transfers and include them.

References

[1] ATLAS Collaboration 2008 *Journal of Instrumentation* **3** S08003 ISSN 1748-0221

[2] Eck C, Knobloch J, Robertson L, Bird I, Bos K, Brook N, Dillmann D, Fisk I, Foster D, Gibbard B, Grandi C, Grey F, Harvey J, Heiss A, Hemmer F, Jarp S, Jones R, Kelsey D, Lamanna M, Marten H, Mato-Vila P, Ould-Saada F, Panzer-Steindel B, Perini L, Schutz Y, Schwickerath U, Shiers J and Wenaus T 2005 *LHC computing Grid: Technical Design Report. Version 1.06 (20 Jun 2005)* Technical Design Report LCG (Geneva: CERN)

[3] Garonne V, Stewart G A, Lassnig M, Molfetas A, Barisits M, Beermann T, Nairz A, Goossens L, Barreiro Megino F, Serfon C, Olevnik D and Petrosyan A 2012 *Journal of Physics: Conference Series* **396** 32045 ISSN 1742-6588

[4] Garonne V, Vigne R, Stewart G, Barisits M, Beermann T, Lassnig M, Serfon C, Goossens L and Nairz A 2014 *Journal of Physics: Conference Series* **513** 042021 ISSN 1742-6588

[5] Jones R and Barberis D 2008 *Journal of Physics: Conference Series* **119** 72020 ISSN 1742-6596

[6] Maeno T, De K, Wenaus T, Nilsson P, Stewart G A, Walker R, Stradling A, Caballero J, Potekhin M and Smith D 2011 *Journal of Physics: Conference Series* **331** 72024 ISSN 1742-6596

[7] Barisits M, Serfon C, Garonne V, Lassnig M, Stewart G, Beermann T, Vigne R, Goossens L, Nairz A, Molfetas A and the Atlas Collaboration 2014 *Journal of Physics: Conference Series* **513** 42003

[8] Zang D, Garonne V, Barisits M, Lassnig M, Stewart G A, Molfetas A and Beermann T 2012 *Journal of Physics: Conference Series* **396** 32119

[9] Molfetas A, Lassnig M, Garonne V, Stewart G, Barisits M, Beermann T and Dimitrov G 2012 *Journal of Physics: Conference Series* **396** 52055 ISSN 1742-6588

[10] Maeno T, De K and Panitkin S 2012 *Journal of Physics: Conference Series* **396** 32070 ISSN 1742-6588

[11] Beermann T, Stewart G A and Maettig P 2015 *Journal of Physics: Conference Series* vol 664 (IOP Publishing) p 32002