Journal of Physics: Conference Series **664** (2015) 032002 doi:10.1088/1742-6596/664/3/032002

A study of dynamic data placement for ATLAS distributed data management

T Beermann[1,3], G A Stewart[2] and P Maettig[3] on behalf of the ATLAS Collaboration

1 CERN, Geneva, Switzerland
2 University of Glasgow, Glasgow, United Kingdom
3 University of Wuppertal, Wuppertal, Germany

E-mail: thomas.beermann@cern.ch

Abstract. This contribution presents a study on the applicability and usefulness of dynamic data placement methods for data-intensive systems, such as ATLAS distributed data management (DDM). In this system the jobs are sent to the data, therefore having a good distribution of data is significant. Ways of forecasting workload patterns are examined which then are used to redistribute data to achieve a better overall utilisation of computing resources and to reduce waiting time for jobs before they can run on the grid. This method is based on a tracer infrastructure that is able to monitor and store historical data accesses and which is used to create popularity reports. These reports provide detailed summaries about data accesses in the past, including information about the accessed files, the involved users and the sites. From this past data it is possible to then make near-term forecasts for data popularity in the future. This study evaluates simple prediction methods as well as more complex methods like neural networks. Based on the outcome of the predictions a redistribution algorithm deletes unused replicas and adds new replicas for potentially popular datasets. Finally, a grid simulator is used to examine the effects of the redistribution. The simulator replays workload on different data distributions while measuring the job waiting time and site usage. The study examines how the average waiting time is affected by the amount of data that is moved, how it differs for the various forecasting methods and how that compares to the optimal data distribution.

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 spread over the Worldwide LHC Computing Grid [2]. To be able to manage this data Don Quijote 2 (DQ2)[3], the collaboration's distributed data management system, was developed and has run since before the detector started data taking in 2008. A new data management system, called Rucio [4], has been under development and has taken over from DQ2 at the end of 2014, to handle the new requirements for LHC Run 2. However, the data used for the work presented in this article comes from DQ2, which was responsible for around 150PB of experiment data spread over 150 sites all over the world. The responsibilities of DQ2 included the organisation and management of primary detector and simulation data, but also all of the derived physics data used by the collaboration's physicists. The data is spread 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 the results of a study that was conducted to examine the performance of a dynamic data placement using dataset popularity forecasts based on historic accesses. This study consists of three parts: The first part introduces a mechanism to analyse the past data accesses to make forecasts of possible accesses in the near-term future. The next part then shows how these predictions can be used to redistribute data on the grid, i.e., adding and removing replicas accordingly. As it is very difficult to evaluate the benefits of such a method in a live system, where exact workload patterns never repeat, the third part introduces a simple grid simulator that is able to run the same workload on different data distributions. In the final section the evaluation of different distributions is presented together with a conclusion and a description of future work.

2. ATLAS Data Distribution

The data managed by the ATLAS DDM on the grid consists of files that contain physics events. Those files are aggregated into logical datasets, which are the unit of operation in the DDM system. It is possible to download single files from a dataset but it is only possible to move whole datasets between sites. A set of actual files on a site that belong to a dataset is called a replica. To create a new replica at a site all of the files of a dataset have to be copied to this site.

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). On top of that, there is the PD2P system [7], which triggers the creation of new replicas when a threshold number of jobs is reached that request a particular dataset.

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 sites' availability, the current workload of the sites and the job priority.

Every time a file is accessed through PanDA a trace will be sent to the DDM tracer system. 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 dataset popularity. 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 [8]. The popularity system aggregates the data from tracer system on a daily basis and provides data that is more tractable.

3. Popularity Prediction

The first step towards a better data distribution is a knowledge of future data popularity. This leads to a need for a prediction of future dataset accesses, which is described in this section.

3.1. General Concept

The idea is to take the past number of accesses to datasets on the grid and make a forecast for the near-term future. The number of accesses as recorded by the popularity system are used for this prediction. In principle these form a time-series prediction, i.e., based on the number of accesses of the last n points in time ($A_{1,2,\ldots,n}$), a prediction for the number of accesses at point $n+1$ (A_{n+1}) is calculated. In this study the interval between points is chosen to be one week. This time frame gives a reasonable period for the deletion and transfers that are necessary

for the redistribution. There exists various methods for time-series predictions. For this study a hybrid approach combining static and neural network prediction has been chosen. This is described below, along with the reason for this approach, in detail below.

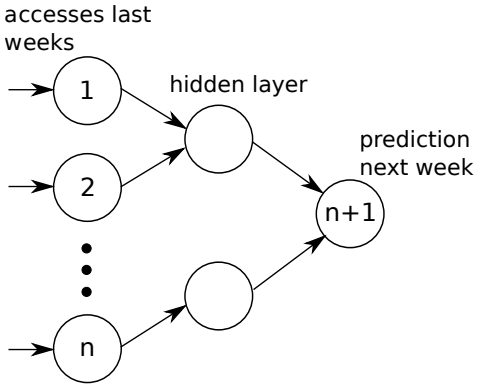
3.2. Prediction Methods

3.2.1. Static Prediction The static prediction is a very simple method. The underlying assumption is the accesses will not change a lot from one week to another and therefore the predicted accesses is the same as the last recorded access: $A_{n+1} = A_n$.

3.2.2. Neural Networks The neural network approach is a lot more sophisticated. Artificial neural networks (ANN) have been proven to be a suitable solution to time series predictions, e.g., they have been evaluated to be used to forecast the development of exchange rates [9], which is one of the reasons why they are chosen for this study. As a general introduction, [10] describes how to apply recurrent neural networks to predict time series.

In this case the ANNs are designed to have n input neurons and one output neuron. The inputs for the ANN are the accesses for the last n weeks. The output is the predicted number of accesses for week $n + 1$. An example is shown in figure 1.

Figure 1. Neural Network Example



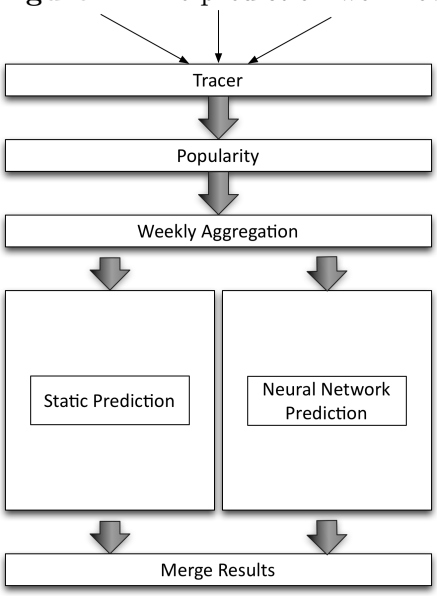
The general concept, as described above, has to be adapted for the actual implementation to make better predictions. There are different types of datasets that are used on the grid and depending on those types other user groups access the data which can result in separate access patterns. To take advantage of this the input data is split based on these datatypes, so that separate neural networks are trained for the different data types.

For the training of the neural networks the accesses data of the last n weeks is used. The inputs for the training are the accesses for the weeks $1, 2, ..., n - 1$ for each of the datasets. The output is the number of accesses for week n. After the training the quality of the prediction is evaluated by making predictions for the training data and then comparing the predicted accesses to the actual accesses. If the error is below a certain threshold the trained networks are used for the actual prediction. Here the data is now shifted and the inputs are the accesses for week $2, 3, ..., n$ and the output is the predicted access for week $n + 1$.

3.2.3. Hybrid Prediction The input can have a considerable number of datasets that are used very little or not at all. Those datasets only introduce noise to the neural networks and therefore should not be included in the training. For this a pre-filtering stage is introduced. Before the ANNs are trained the input data is filtered based on total number of accesses in the last n weeks

and number of weeks without any accesses at all. If the dataset is unpopular based on this criteria it will not be used with the neural network prediction but with the static prediction. The neural network prediction then continues only with the popular data. In the end the results of the static and neural network prediction are merged again to give the complete result set. The whole process from the collection of input data to the predicted results is illustrated in figure 2

Figure 2. The prediction workflow



4. Data Redistribution

This section describes how the predicted accesses can be used to add and remove dataset replicas to improve the data distribution. The data redistribution consists of two parts that have to work hand in hand: The cleanup of space at sites and the creation of new replicas. The deletion process has to know exactly how many bytes it has to delete and the creation of replicas has to fill up the freed space as efficiently as possible.

4.1. Replica Creation

For the creation of datasets the redistribution algorithm has to rely on the results of the dataset access prediction. The approach here is to use the access predictions to place replicas evenly on the available computing resources. Each site where replicas are hosted has a certain number of computing slots to run jobs simultaneously. If all of these slots are occupied new incoming jobs have to wait. By adding new replicas the number of job slots able to process a particular dataset becomes bigger and the WMS has more options for where to put jobs; ideally this leads to lower overall waiting times and better site utilisation.

The input metric for this approach to distribute the workload evenly is the number of accumulated accesses per job slot per site. To get the numbers of accumulated accesses for each site the numbers of predicted accesses for each replica of the site are summed up. As the number of job slots differs for each site care must be taken that small sites get proportionally fewer accesses than bigger sites. For that reason the accumulated accesses are normalised by the number of job slots per site.

4.2. Replica Deletion
Two things have to be considered when deleting replicas for the redistribution. First, there should always be a minimum number of replicas available per dataset. The actual number depends on the type of the dataset and is defined by CREM. These are typically two replicas on disk for current data types and one replica for the rest. This is to make sure that no dataset will be deleted completely by the redistribution and that data safety for precious data is assured (data is continually lost at a low level on the grid due to storage system failures, so a single replica is particularly vulnerable to accidental loss at a site). Second, only the number of replicas for datasets that have been unpopular in the last weeks should be reduced.

4.3. Workflow
For the redistribution the following inputs are needed:

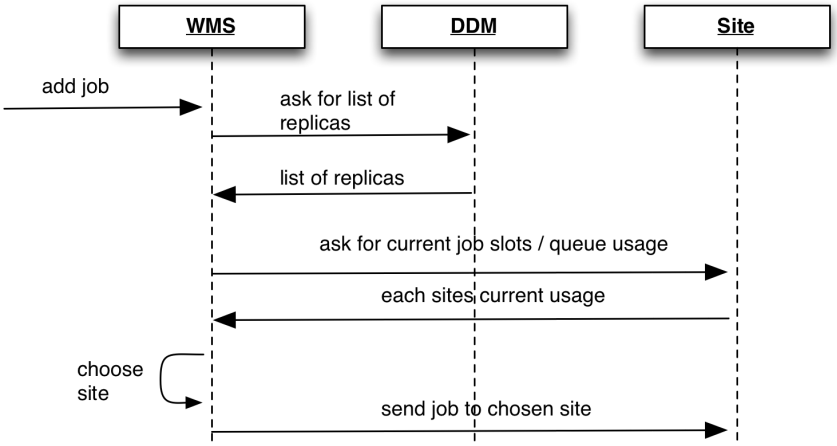
- **Current Data Distribution:** The data catalogue with the current dataset replica distribution.
- **Current Site Configuration:** The configuration of the sites containing information about the size of the site's disks and the number of available job slots, i.e., the number of jobs that can be run concurrently.
- **Past Dataset Popularity:** Aggregation of weekly accesses for datasets in the last weeks.
- **Predicted Accesses:** The predicted dataset accesses for the next week.
- **Maximum Bytes:** The maximum number of bytes the algorithm is allowed to use for redistribution, i.e., it must not delete and add more than the given number of bytes.

The first step before the redistribution begins is to calculate the accumulated accesses for the replicas that are already available. So for all sites a list of replicas is iterated and if a replica is predicted to have accesses the number of accesses is added to the sum. After this, the redistribution begins by iterating over all datasets from the prediction. The datasets are sorted beginning with those with the most accesses. For each dataset a list of possible sites for a new replica is created. This list is based on the ratio of accumulated accesses and the job slots with the sites with lowest ratio first. All sites that already host a replica for this dataset are excluded. In the next step the list of new sites is iterated and an attempt is made to place the new replica at the first site. If not enough space is available for the new replica the deletion mechanism is started and tries to delete replicas from the site until enough space is available. If it is not possible to clean up enough space the algorithm continues with the next site. In the end, if no site has enough space available no new replica will be added. If a replica has been added the algorithm continues with the next dataset. This process continues either until replicas are created for all predicted datasets or the maximum number of bytes is reached.

5. Grid Simulator
Testing and evaluating the impact of this redistribution algorithm on the waiting times and computing resource utilisation of the real system is not trivial. In order to evaluate the benefits of redistributing data the actual grid workload, extracted from the WMS, is replayed using a grid workload simulator. To compare different data distributions the system has to run multiple times with the same workload, i.e., it would have to be stopped after some time and set back to the original state and start from there again. This would not be possible to do in the real system, so a simplified grid simulator was developed for that purpose. The simulation can run on a real workload extracted from the job history from the WMS and the data distribution from DDM system and measure the waiting times and usage of job slots.

It simulates the following parts of the real system:

Figure 3. The simulation workflow



- **Sites**: they provide different storage endpoints to save data and computing slots to send jobs to this data.
- **DDM system**: This system keeps track of the location of replicas and it is used to add and remove replicas to the system.
- **WMS**: This system brokers the jobs to the sites and uses the DDM and the Sites to find a suitable place.

The workload is inserted into the simulation as separate jobs using one dataset for a predefined amount of time. The WMS then tries to pick a site based on data locality from DDM and available computing resources from the sites and sends the job there. Either there is a free job slot at the site, and the job can start running right away, or it will have to wait in a queue until another job finishes and a job slot is freed. When the job finished it saves the time it was inserted into the system and the time when it actually started using the job slot. The difference then gives the waiting time per job, which then can be used to calculate the average waiting time for all jobs. Furthermore, the simulation also regularly saves the number of free and used job slots on the sites, so that is possible to determine the slot utilisation. The whole workflow can be seen in figure 3

6. Evaluation

In this section the results of the evaluation together with the parameters for the simulation are described.

6.1. Setup

Below the most important parameters of the simulation are listed:

- The simulations run over a total period of 13 weeks.
- The evaluation of the waiting time is done per week. Three typical weeks have been picked, simulating high, medium or low load on the system.
- The workload is extracted from the DQ2 traces and the PanDA job history. Only analysis jobs are simulated.
- The number of job slots has been determined from WMS history for each week separately, based on the maximum number of simultaneously running jobs.

Load	Date	Job Slots	Total Replicas	Total Disk Space	DATADISK Replicas	DATADISK Space
High	17.-24.02.2014	24205	526511	42.3 PB	210784	27.4 PB
Medium	24.-31.03.2014	18177	534838	43 PB	211798	27.5 PB
Low	17.-24.03.2014	15274	535584	43 PB	211798	27.5 PB

Table 1. Simulation parameters for the three weeks

- The data distribution is extracted from DDM system and includes datasets of detector and Monte Carlo data of the datatype that is used for user analysis.
- The original average waiting time was 18999s (∼5:15h) for the high load week, 16656s (∼4:30h) for the medium and 7182s (∼2h) for the low load week.

6.2. Results

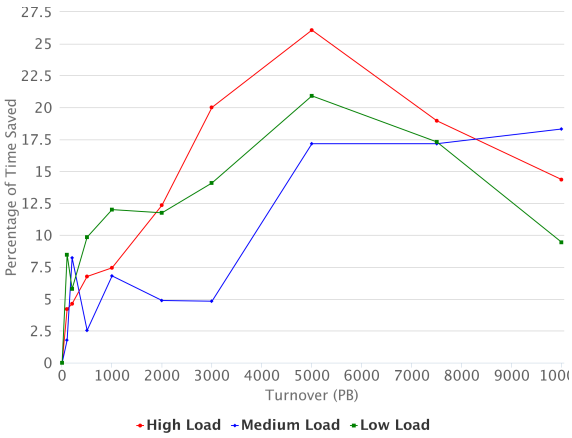


Figure 4. Figure showing the percentage of waiting time saved for different turnovers compared to the original distribution

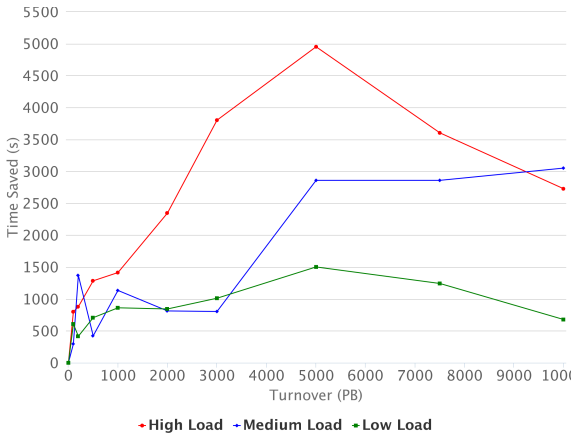


Figure 5. Figure showing the difference in waiting time for different turnovers compared to the original distribution

Figure 4 shows the waiting time benefit in percentage of the original waiting time. The overall trend for all weeks shows that the more data is moved the bigger the benefit is. For two out of three weeks this holds true until a certain point is reached when the benefit goes down again. Something interesting happens for the medium load week. The waiting time is going up and down again until it hits it maximum around 5PB, when it becomes constant.

The reason that benefit is going down again at some point is that the redistribution is deleting data, which reduces replicas for some datasets, which have not been used for a period of time. Initially, the chances are very low that this dataset will be accessed and, even if there are accesses there are still replicas left. So the waiting time will not be negatively influenced at all or in the worst case only a little bit. On the other hand the new replicas for the popular datasets mean that jobs can be sent to sites which otherwise would have been idle because the lack of popular data at that site. This will positively affect the waiting time. For the smaller turnovers this leads to an overall reduced waiting time.

For the bigger turnovers the effect changes. Replicas are removed starting with the datasets that have not been used for the longest time. At some turnover, here it is 5PB, the redistribution will start deleting replicas, that might be used. Furthermore the added replicas cannot balance that out anymore, leading to a rise of the average waiting time again.

The effect on the medium load week could be explained as follows: The prediction for this week is not as good as for the other two weeks and does not get the order for the popular datasets completely right. So it starts adding less popular datasets first and later it comes to the popular ones. Together with the deletion, which probably deletes data that could be used, this behaviour is the result. This indicates that there is still room for improvement of the prediction in future developments.

7. Conclusion and Outlook

The results show that it is possible get a significant gain in user mean waiting times, even with small amounts of data moved. The gain may vary for different weeks, but overall the results show the same trends:

- It is possible to make forecasts of future data accesses that are usable to redistribute data for better waiting times.
- The more data is moved the bigger the benefit until a turning point of around 5PB where the benefit goes down again or stays constant.
- The maximum benefit for all weeks is between 18% and 26%, which, depending on the week, can be up to an average of more than 70 minutes of waiting time saved per job.

This study used data that was extracted from the system used for Run 1. For Run 2 new concepts and systems have been introduced, which include different datatypes and data life-cycle models. The system introduced in this contribution has to be adapted and evaluated to work with the new systems as well. Furthermore the results indicate that the prediction can be improved. Therefore other time-series prediction algorithms will be evaluated.

[1] ATLAS Collaboration 2008 *JINST* **3** S08003
[2] Eck C, Knobloch J, Robertson L, Bird I, Bos K, Brook N, Düllmann 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) URL `http://cds.cern.ch/record/840543`
[3] Branco M, Zaluska E, de Roure D, Lassnig M and Garonne V 2009 *Concurrency and Computation: Practice and Experience* **22** ISSN 15320626 URL `http://dx.doi.org/10.1002/cpe.v22:11 http://doi.wiley.com/10.1002/cpe.1489`
[4] Garonne V, Stewart G A, Lassnig M, Molfetas A, Barisits M, Beermann T, Nairz A, Goossens L, Barreiro Megino F, Serfon C, Oleynik D and Petrosyan A 2012 *Journal of Physics: Conference Series* **396** 032045 ISSN 1742-6588 URL `http://stacks.iop.org/1742-6596/396/i=3/a=032045`
[5] Jones R and Barberis D 2008 *Journal of Physics: Conference Series* **119** 072020 ISSN 1742-6596 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 and Smith D 2011 *Journal of Physics: Conference Series* **331** 072024 ISSN 1742-6596 URL `http://stacks.iop.org/1742-6596/331/i=7/a=072024`
[7] Maeno T, De K and Panitkin S 2012 *Journal of Physics: Conference Series* **396** 032070 ISSN 1742-6588 URL `http://stacks.iop.org/1742-6596/396/i=3/a=032070`
[8] Molfetas A, Lassnig M, Garonne V, Stewart G, Barisits M, Beermann T and Dimitrov G 2012 *Journal of Physics: Conference Series* **396** 052055 ISSN 1742-6588 URL `http://stacks.iop.org/1742-6596/396/i=5/a=052055`
[9] Refenes A N, Azema-Barac M E, Chen L and Karoussos S A 1993 *Neural Computing Applications* **1** 46–58 ISSN 09410643 URL `http://www.springerlink.com/index/G810W7118916R733.pdf`
[10] Connor J T, Martin R D and Atlas L E 1994 *IEEE transactions on neural networks / a publication of the IEEE Neural Networks Council* **5** 240–54 ISSN 1045-9227 URL `http://www.ncbi.nlm.nih.gov/pubmed/18267794`