PROCEEDINGS OF SCIENCE

A Popularity-Based Prediction and Data Redistribution Tool for ATLAS Distributed Data Management

Thomas Beermann*
CERN
E-mail: thomas.beermann@cern.ch

Graeme A. Stewart
University of Glasgow
E-mail: graeme.a.stewart@gmail.com

Peter Maettig
University of Wuppertal
E-mail: peter.mattig@cern.ch

on behalf of the ATLAS collaboration

This paper presents a system to predict future data popularity for data-intensive systems, such as the ATLAS distributed data management (DDM). Using these predictions it is possible to improve the distribution of data, helping to reduce waiting times for jobs using this data.
This system is based on a tracer infrastructure that is able to monitor and store historical data accesses, which is then used to create popularity reports. These reports provide a summary of data accesses in the past, including information about the accessed files, the involved users and the sites. From this past accesses information it is possible to make near-term forecasts of data popularity. The prediction system introduced in this paper makes use of both simple prediction methods, as well as predictions made by neural networks. The best prediction method is dependent on the type of data and the access information is carefully filtered for use in either system.
The second part of the paper introduces a system that effectively places data based on the predictions. This is a two phase process: In the first phase space is freed by removing unpopular replicas; in the second new replicas for popular datasets are created. The process of creating new replicas is limited by certain constraints: there is only a limited amount of space available and the creation of replicas involve transfers that use bandwidth. Furthermore, the benefits of each replica is different. The goal is to maximise the global benefit while respecting the constraints.
The final part shows the evaluation of this method using a grid simulator. The simulator is able to replay workload on different data distributions while measuring the job waiting time. We show how job waiting time can be reduced based on accurate predictions about future accesses.

International Symposium on Grids and Clouds (ISGC) 2014,
23-28 March 2014
Academia Sinica, Taipei, Taiwan

*Speaker.

PoS(ISGC2014)004

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 it has run since before the detector started data taking in 2008. A new data management system, called Rucio [4], is currently in development and will overcome some limitations of the current system to address LHC Run 2 requirements. However, the data taken for the work presented in this article comes from DQ2. DQ2 is currently responsible for around 150PB[1] of experiment data that is spread over 150 sites all over the world. The responsibilities of DQ2 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 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 article introduces a new way to automatically and dynamically remove and add new data replicas to the system according to the future popularity of the data. This is done in multiple parts: The first part analyses the past data accesses to make forecasts of possible accesses in the near-term future. The next part then uses these predictions to redistribute data on the grid, i.e., adding and removing replicas accordingly. It is very difficult to evaluate the benefits of such a method in a live system, where exact workload patterns never repeat. Therefore the third part introduces a simple grid simulator that is able to run the same workload on different data distributions. The final section then presents the evaluation of the different distributions, the conclusions and a description of future work.

2. Related Work

In recent times there has been great interest in researching dynamic data replication strategies for the grid as well as simulation of it. A variety of grid simulators have been developed, like Monarc[7], ChicSim[8], SimGrid[9], OptorSim[10], MicroGrid[11] and GridSim[12].

MicroGrid and SimGrid were developed to simulate computational grids. They model job work loads in a heterogeneous set of computing resources, such as might be found in a grid of HPC centres. However, the simulation of data management aspects was not considered in these programs.

The Monarc project addressed data grids more directly, in order to explore different computing models for the Large Hadron Collider (LHC) experiments. It is able to evaluate transfer times and bandwidth usage for different computing models but it is missing the infrastructure for replica management and optimisation, which it is now clear that the LHC experiments need.

The original GridSim was focused on resource modelling and application scheduling for parallel and distributed computing and had no capabilities for data management. However this changes with the extensions presented in [13]. The data management is file based and allows for a complex

[1] `http://bourricot.cern.ch/dq2/accounting/global\_view/30/`

hierarchy of global, regional and local file catalogs. While such flexibility extends the scope of the application, it is overly complex for the modelling of the ATLAS DDM system.

OptorSim is another grid simulator which was originally developed for European DataGrid, but was also used to simulate the LHC computing grid (LCG). In the LCG case it was aimed at studying different replication strategies and how they would affect mean job time and the effective network usage. The model relies on autonomous agents for the replica optimisation, which is currently not the model used by ATLAS.

Currently in PanDA there exists a system to dynamically add replicas of popular datasets called PD2P[14]. PD2P triggers the creation of new replicas when a threshold of jobs is reached that request a particular dataset. However, because the creation of new replicas takes some time it is possible that a currently popular dataset might not be popular by the time the new replica is created.

In this paper, in contrast to PD2P, we attempt to forecast popular datasets in advance, so that DDM will have time to create new replicas before they will be needed. Therefore we base on many of the ideas found in the literature, but focus specifically on the use case for ATLAS when storage resources are severely constrained. Secondly, we have developed a new, lightweight, simulator so that many models of data distribution can be evaluated very quickly, while still retaining the capacity to model the real distribution of 150PB of data and inject a real workload of hundreds of thousands of jobs.

3. ATLAS Data Distribution

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 that reason the concept of datasets exist in DQ2. Datasets are completely logical units in DQ2 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 DQ2. Even though it is possible to retrieve single files from a dataset, it is only possible to transfer complete datasets between sites. A set of actual files on a site corresponding to a dataset is called replica. To create new replicas at a site all of these files have to be copied to the new site over the grid and will be registered in the system. Therefore one logical dataset in DQ2 can have multiple physical replicas on the grid.

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 which is described above.

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 request 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.

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 popularity of 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 [15]. The popularity system aggregates the data from tracer system on a daily basis and provides data that is more tractable.

The current way data is distributed is rather static, which leads to many unused replicas on the grid. The idea of this work is to automatically delete some of these unused datasets dynamically to create space for more popular data. The general assumption is that the system can benefit from more replicas of a dataset in two ways: 1) reduction of the time users have to wait until their jobs can run and 2) better overall usage of resources at sites.

4. Popularity Prediction

To distribute data better, first a knowledge of future access patterns is needed, i.e., a way to predict future dataset accesses is needed. The underlying assumption is that historic user behaviour can be used to make predictions about future dataset popularity. There are various methods than can be used to make these predictions; this section presents two possible ways.

4.1 General Concept

The popularity system records all user accesses to datasets on the grid. This data can be used as input to make predictions. In practice predictions are made on a weekly basis. This time frame is chosen as it gives a reasonable time period for the deletion and transfers necessary for the redistribution of data, but in general smaller or bigger time frames would also be possible. The input for the predictions are the accesses for each dataset of the last n weeks ($A_{1,2,\ldots,n}$). The output is the predicted number of accesses for the next week (A_{n+1}). The n weeks for each dataset are relative from week $n+1$, if a dataset is younger than n weeks the accesses for the weeks where it has not existed will be 0.

One of the simplest approaches is static prediction. The assumption here is that the accesses stay rather constant for one week to another, i.e., $A_{n+1} = A_n$. This is the approach that will be used in the evaluations below.

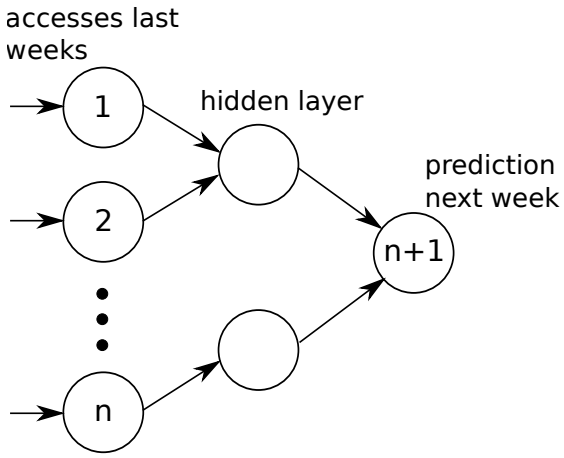
4.2 Neural Networks

Besides the rather simple approach, that is described above, there are other more complex methods available to model time series and predict future values. Artificial neural networks have been proven to be viable solution to predict time series, e.g., they have been evaluated to be used to forecast the development of exchange rates [16], and are therefore chosen for further evaluation in this use case. In [17] it is generally described how to apply recurrent neural networks to predict time series.

Following, it is described how artifical neural networks can be used here to predict future dataset accesses. The ANNs can be trained to pick up common dataset access patterns. The general idea is to build a neural network with n input neurons and one output neuron. Each input neuron

is fed with the accesses of one of the last n weeks for each dataset and the output neuron gives the predicted number of accesses for week $n+1$. An example can be seen at figure 1.

Figure 1: Neural Network Example



The general concept of the neural networks has been described above, but for the actual implementation a few more things have to be considered. The datasets that are used on the grid have different types with different user groups accessing the data and therefore different access patterns per type. These access patterns can vary significantly for the different core data types and lumping them all together makes it hard for the neural network to make good predictions. To take this into account, different neural networks are trained for the different types. Another step to help the neural networks to find patterns is to pre-filter the input before using it for the training. Some datasets have a very low number of accesses or are accessed very irregularly. For these types of datasets the static prediction gives good results, so that a threshold of minimum accesses for the preceding weeks is introduced. Everything above this threshold is used for neural network and everything below is used with static prediction.

After the data has been filtered the neural networks can be trained. For the training a set of training input and target data is needed for the accesses for the last n weeks. The idea is now to use weeks $1,2...,n-1$ as input for the training and week n as the target. After the training is done, and the quality of the network has been evaluated with the original input, the network can be used for the predictions. For that the input is shifted, i.e., now the weeks $2,3,...,n$ will be used for the input and the output will be the predicted accesses for week $n+1$.

5. 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.

5.1 Replica Creation

For the creation of datasets the redistribution algorithm has to rely on the predicted dataset accesses. The approach here is to use the access data from the predictions to place replicas evenly on the available computing resources. Each site where replicas are hosted has a certain amount 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 as to 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 accumulated accesses per job slot per site. To get the number of accumulated accesses for each site the number of predicted accesses for each replica of the site is summed up. As the number of job slots differs for each site so it has to be taken care that small sites get proportionally less accesses than bigger sites. For that reason the accumulated accesses are normalised by the number of job slots per site.

5.2 Replica Deletion

Two things have to be considered when deleting replicas for the redistribution. First, there should always be a minimum amount of replicas available per dataset. The actual number depends on the type of the dataset and is defined by CREM. These are typically 2 replicas on disk for current data types and 1 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.

5.3 Workflow

For the redistribution the following inputs are needed:

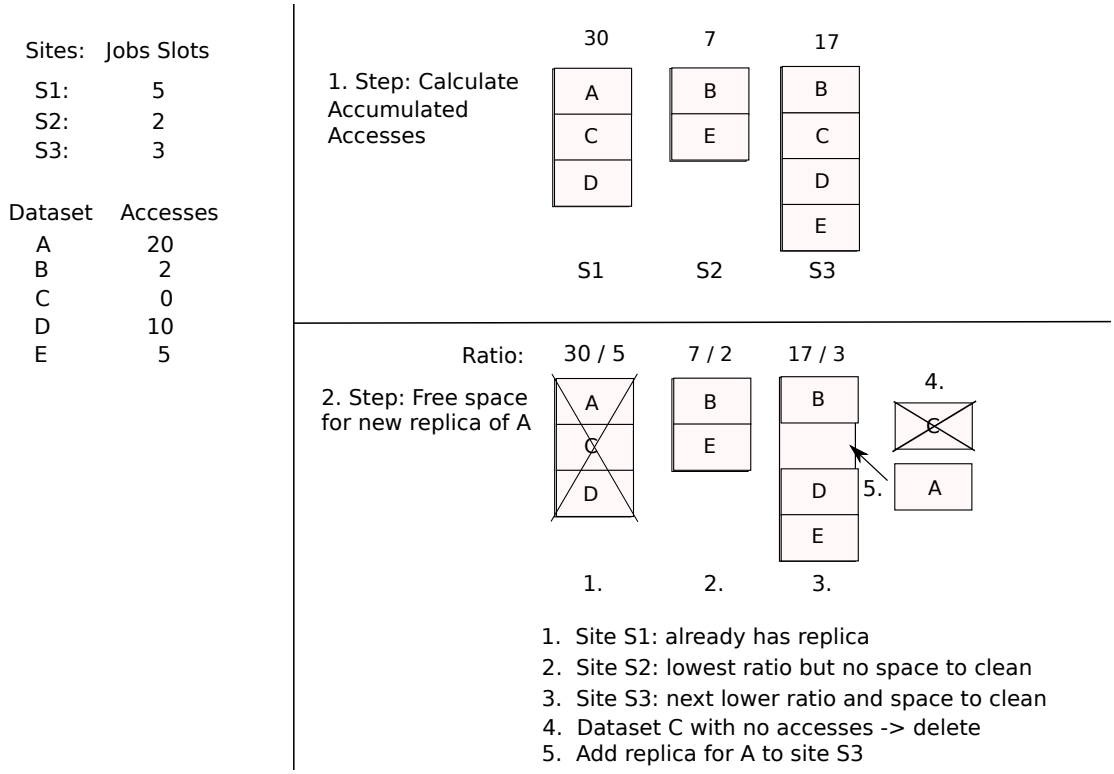
- **Current Data Distribution:** The data catalogue with the current dataset replica distribution.
- **Current Site Configuration:** The configuration of the sites containing information about the size of the site's disks and the available job slots, i.e., the amount of jobs that can be run concurrently.
- **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 that 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 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 the new replica is tried to be placed 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.

Figure 2 illustrates this process with a simple example.

Figure 2: Simple redistribution algorithm example



6. Grid Simulator

This section describes the architecture and the collected metrics of the grid simulator that was developed to evaluate the quality of the redistribution methods that are described in the previous sections. The simulator is a simple discrete event based simulator that is able to run a certain workload on a given data distribution with a predefined configuration of available sites. This makes it possible to start with a data distribution from the real system, extract the workload from the history tables and then run this workload on the data distribution and collect different metrics. Then, after redistribution of the data, the same workload can be re-run. This can be done for many

different data distribution methods and time periods and in the end the collected metrics can be compared to evaluate the redistribution impact.

6.1 Architecture

The simulator needs the following inputs to run:

- **Workload:** The workload is a list of tuples with four values for each tuple: submission time, execution time, dataset name, user name.
- **Site Configuration:** The site configuration defines which sites are available for the simulation and number of job slots for each site.
- **Data Catalogue:** The data catalogue stores information of each replica. These are dataset location, the size and the creation date.

During a simulation run the simulator generates the following outputs:

- **Job Summary:** For each job that was simulated an output is generated. This output contains the job's submission, waiting and execution time as well as the dataset name and the chosen site.
- **Site Utilisation:** The number of running and waiting jobs is monitored continuously.

Internally the simulator consists of the following components:

- **Sites:** The sites are responsible for managing the available and used disk space and the number of jobs and waiting queue slots.
- **Distributed Data Management:** The DDM component provides an interface to add and remove replicas to the central data catalogue and the sites. Furthermore, it provides information about replicas for datasets.
- **Workload Management System:** The WMS uses the workload and schedules it to the sites. To decide where to place the jobs it interfaces both with the DDM system, to get the dataset's locality information, as well as the sites for job slot and waiting queue information.

6.2 Workflow

When the simulation is started a list of workload tuples is processed in ordered by actual submission time. For each workload tuple the WMS first asks the DDM system about the sites hosting a replica of the relevant dataset. The decision where to place the job is then made based on the available job slot resources. If there is only one site with available job slots the job will be placed at that site. If there are multiple sites with open job slots the site with the lowest ratio of running jobs and overall job slots will be taken. Finally, if there are no free job slots available anywhere a brokerage decision is made based on the ratio of waiting jobs to the sites' job slot capacity. When a site is picked the job will be handed over to the site. The site then either directly runs the job, if a free slot is available, or puts it into the waiting queue. Jobs are occupying the slots

for their actual execution time (in the simulation all job slots are assumed to be homogeneous). If jobs are waiting and a slot becomes free, the first one waiting in the queue will be picked to occupy this slot. The simulator can then either run until the all workload is processed and no jobs are running at sites anymore or until a predefined number of simulation steps is reached.

6.3 Metrics

Based on the output from the simulator different metrics can be used to evaluate the redistribution algorithms. These are:

- **Average Waiting Time per Job:** The average waiting time per job gives a good first measurement of the redistribution performance from the user point of view.
- **Evolution of Job Slot / Waiting Queues:** This shows how given computing resources are used at the sites. The simulator collects the current usage of job slots and the size of the waiting queues globally for all sites combined. With this it can be evaluated how jobs slots are filled when jobs have to wait in the queues. When jobs are waiting in the queue but not all job slots are filled there might be a possibility for improvement.

6.4 Limitations

The aim of the simulator is not to fully replicate the complexity of the distributed analysis system, but to have the minimum complexity necessary to evaluate the performance of new data distributions. Therefore the simulator currently has the following limitations:

- **No PanDA priority queues:** In the real WMS jobs have different priorities that makes them advance quicker or slower in the waiting queues. This currently is not implemented in the simulator.
- **Only user analysis jobs:** At the moment no jobs running physics production are simulated.
- **No dynamic job allocation:** In the real system the computing resources are shared between production and user analysis jobs. The actual number of available jobs slots for the two job types is allocated dynamically. The simulator only works with a fixed number of job slots for analysis jobs.
- **Only separate weeks:** Currently the simulation is done for each week separately disregarding the redistributions made during previous weeks.
- **No network simulation:** The simulation currently does not take into account the bandwidth needed to redistribute data. That will be added in a future version.

Because of this limitations the measured waiting times can be different from the actual waiting times but they should still be sufficient to evaluate the performance of the algorithms.

7. Evaluation

This section evaluates the performance of the redistribution algorithm with workload based on two sample weeks.

Week	Number of Jobs
04.11.2013 - 10.11.2013	2892000
18.11.2013 - 25.11.2013	2552000

Table 1: Overview of the number of jobs for each evaluated week

7.1 Evaluation Setup

The evaluation is based on single simulation runs for two separate, unconnected weeks. The original data distribution has been extracted from DQ2's original data catalogue for each week. The ATLAS data on the WLCG is a mixture of real detector, Monte Carlo simulation and user data. For the real data and the Monte Carlo data different processing steps are stored from the raw data to the input used for user analysis jobs. For this evaluation only data that is used as analysis input is considered. This is the data which is most frequently used by many different users and therefore best suited for redistribution. The data is spread over different kinds of storage space, which are either managed centrally by ATLAS computing operations, by a particular physics group or locally. For the simulation data on all different storage types is considered but for the redistribution only data on the centrally managed space will be deleted and added. The maximum amount of disk space available is the sum of the size of replicas of the original distribution, i.e., before new data can be added existing data has to be deleted. The initial size distribution is 45PB as extracted from the DQ2 database. For each of the evaluated weeks the workload is different in regard to the number of jobs that run. Table 1 gives an overview of the number of jobs for each week.

Different distributions are evaluated. As a reference the first simulation is done with the original distribution of 45PB. Then data distribution based on the static prediction method are evaluated with different amounts of redistributed data. The evaluated values are 10TB, 50TB, 100TB, 500TB, 1000TB and 2000TB. At the end the same is done with a perfect prediction to get the maximum possible gain with the given redistribution algorithm. Perfect prediction means that the number of accesses are extracted from the actual workload. For the deletion only replicas for datasets are considered, that have not been accessed during the last 10 weeks and that are at least one week old.

7.2 Results

The first examined metric is the evolution of waiting time for the different data distributions. Figure 3 shows the results for the two different weeks both for the static and perfect prediction. It can be seen that the overall trend is that the more data is moved the bigger is the benefit. But especially for the first week the biggest difference is from 500TB to 1000TB. After that the gain is only marginal. So this means that with a turnover of 1000TB for the static prediction a reduction of waiting time from 4 hours to around 3 hours is possible. With a better prediction this can already be achieved with 500TB. The results are different for the second week, where the gain is overall more linear. Another interesting result is the difference between the static and the perfect prediction. In this week the maximum gain possible with perfect prediction and a turnover of 2000TB is a reduction of the waiting time from 3.75 hours to 2 hours. With the static prediction only a minimum waiting time of 2.6 hours is possible. For small amounts of moved data the static and perfect prediction are close but then from 1000TB upwards the perfect prediction has a clear advantage.

Figure 3: Evaluation of average waiting times for one week (l. 04.11-10.11, r. 18.11.-24.11.)

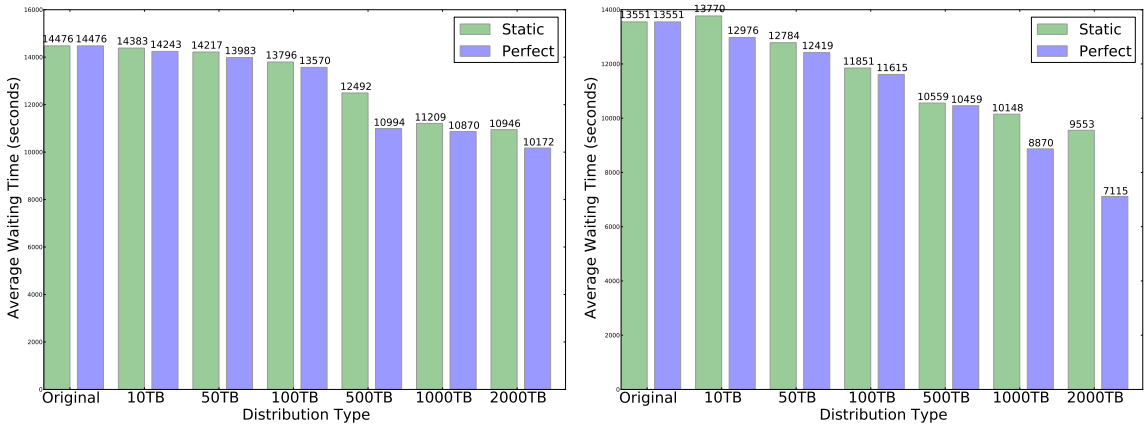
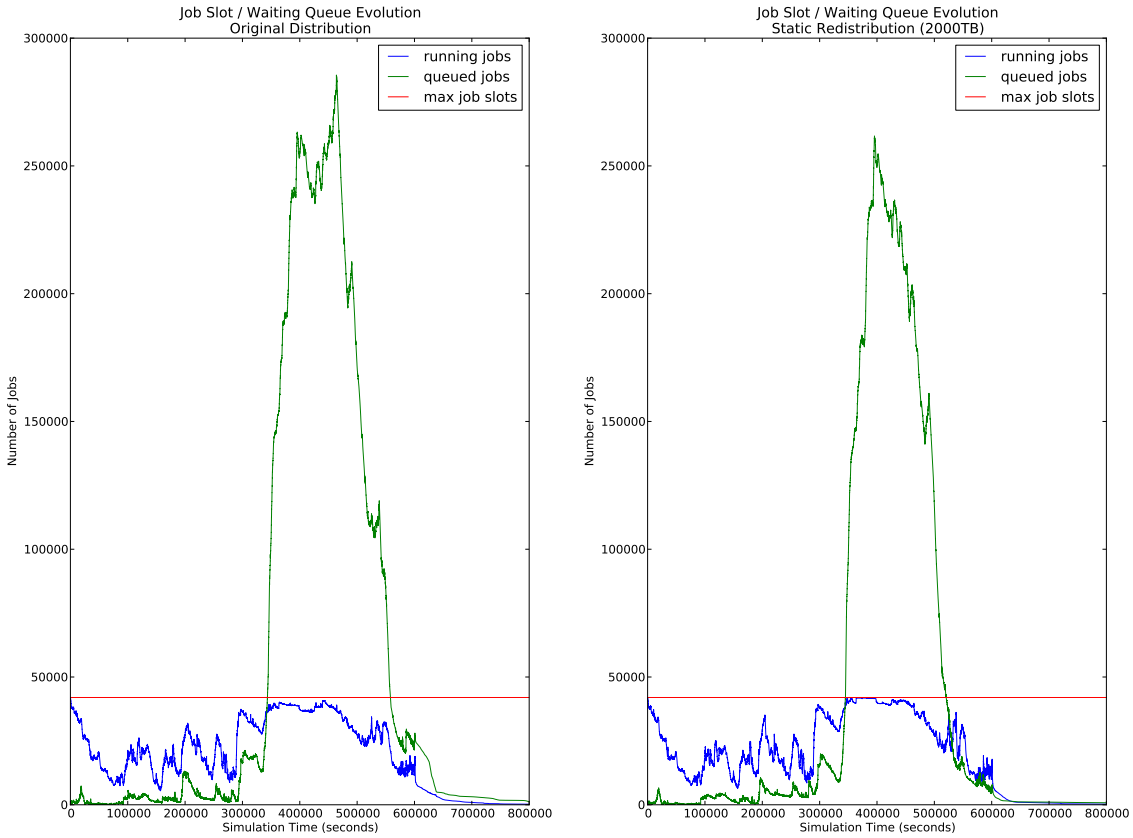
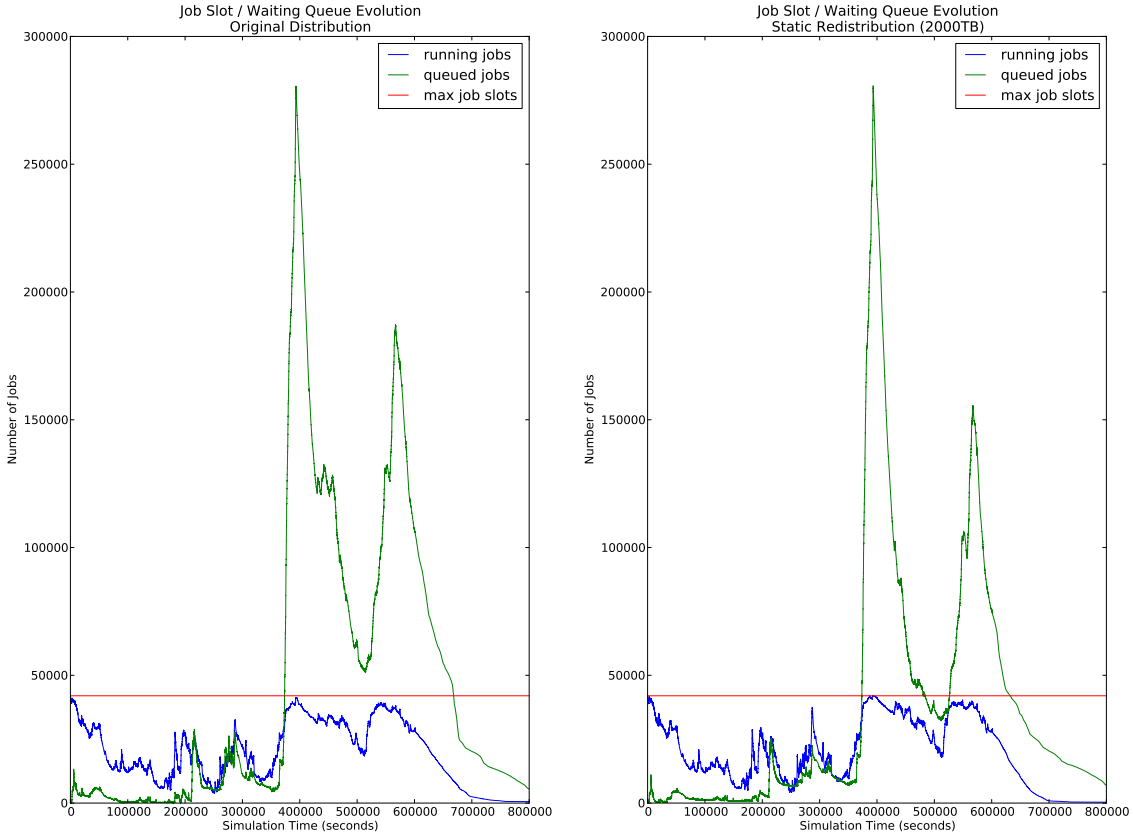


Figure 4: Evolution of occupied job slots / queue sizes, original vs. static prediction (2000TB) (04.11-10.11)



The results for the waiting times show that even with a static prediction is possible to reduce the waiting time significantly but dependent on the week the redistribution can benefit a lot from better prediction either in terms of the needed turnover or the maximum possible gain in waiting times.

Figure 5: Evolution of occupied job slots / queue sizes, original vs. static prediction (2000TB) (18.11-24.11)



The next metric that is presented here is the evolution of the job slots and waiting queues over the simulation time. The red line is the maximum number of jobs that can concurrently run. So if there are jobs waiting (green) and the number of running jobs (blue) is not close to maximum number of available job slots it means that the resources are not used optimally. In Figure 4 the two plots can be found for the first week and Figure 5 shows the same plots for the second week. The left plots shows the queues for the original distribution and the right plots show the same for the static distribution with a turnover of 2000TB. For both weeks most jobs are submitted towards the middle of the simulation. For the original distributions it can be seen that the number of running jobs raise as well as the number of queued jobs. Especially for the second week it can be seen that even though a lot of jobs have to wait, not all of the available computing resources are used, i.e., there are sites with available computing resources but they don't have the replicas for the requested datasets. Now with the redistribution this changes. Overall during mid-week there are more concurrently running jobs and less jobs waiting in the queues, i.e., some of the sites, that were not used before now have replicas of requested datasets. So overall the redistribution leads to a better usage of the available resources, but it can still be seen that jobs have to wait when not all job slots are filled, i.e., there is still room for improvements of the distribution algorithm.

8. Conclusion and Outlook

This article presented a new approach to use past data popularity to redistribute ATLAS data on the WLCG to lower waiting times for users and to make better use of computing resources. The results show that even with a static prediction it is possible to reduce waiting time. Dependent on the workload of a week it can even get close to the perfect prediction. The evolution of the queues showed that overall a better use of the computing resources is possible.

In the future more prediction algorithms have to be implemented and evaluated. The first will be the neural network prediction but also other time series prediction algorithms are possible. First internal results with the neural networks showed that the quality of these predictions will be significantly better than the static prediction. Furthermore the redistribution mechanism has to be improved to get as close to the optimal usage of the resources as possible. Besides that the simulation has to be extended to multiple weeks. At the moment the redistribution runs completely independent for each week. The simulation has to run continuously for several weeks and the redistribution will be done in the beginning of each weeks. With this setup it should be possible to evaluate how much data has to actually be moved every week and what gain could be achieved. Additionally the simulation has to be extended to support production jobs. It is important to understand if and how the redistribution would affect these jobs.

References

[1] ATLAS Collaboration, *The ATLAS Experiment at the CERN Large Hadron Collider*, *JINST* **3** (2008) S08003.

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

[3] M. Branco, E. Zaluska, D. de Roure, M. Lassnig, and V. Garonne, *Managing very large distributed data sets on a data grid*, *Concurrency and Computation: Practice and Experience* **22** (2009), no. 11.

[4] V. Garonne, G. A. Stewart, M. Lassnig, A. Molfetas, M. Barisits, T. Beermann, A. Nairz, L. Goossens, F. Barreiro Megino, C. Serfon, D. Oleynik, and A. Petrosyan, *The ATLAS Distributed Data Management project: Past and Future*, *Journal of Physics: Conference Series* **396** (2012), no. 3 032045.

[5] R. Jones and D. Barberis, *The ATLAS computing model*, *Journal of Physics: Conference Series* **119** (2008), no. 7 072020.

[6] T. Maeno, K. De, T. Wenaus, P. Nilsson, G. A. Stewart, R. Walker, A. Stradling, J. Caballero, M. Potekhin, and D. Smith, *Overview of ATLAS PanDA Workload Management*, *Journal of Physics: Conference Series* **331** (2011), no. 7 072024.

[7] C. Dobre and C. Stratan, *MONARC Simulation Framework*, *Proc. of the RoEduNet International Conference* (2011) [arXiv:1106.5158].

[8] K. Ranganathan and I. Foster, *Decoupling computation and data scheduling in distributed data-intensive applications*, in *Proceedings 11th IEEE International Symposium on High Performance Distributed Computing*, vol. 2002, pp. 352–358, IEEE Comput. Soc, 2002.

[9] a. Legrand, L. Marchal, and H. Casanova, *Scheduling distributed applications: the SimGrid simulation framework*, in *CCGrid 2003. 3rd IEEE/ACM International Symposium on Cluster Computing and the Grid, 2003. Proceedings.*, pp. 138–145, IEEE, 2003.

[10] W. H. Bell, D. G. Cameron, A. P. Millar, L. Capozza, K. Stockinger, and F. Zini, *Optorsim: A Grid Simulator for Studying Dynamic Data Replication Strategies*, *International Journal of High Performance Computing Applications* **17** (2003), no. 4 403–416.

[11] H. Song and X. Liu, *The microgrid: a scientific tool for modeling computational grids*, *Proceedings of the 2000 ACM/IEEE conference on Supercomputing* (2000) 53.

[12] R. Buyya and M. Murshed, *GridSim: a toolkit for the modeling and simulation of distributed resource management and scheduling for Grid computing*, *Concurrency and Computation: Practice and Experience* **14** (2002), no. 13-15 1175–1220.

[13] A. Sulistio, U. Cibej, S. Venugopal, B. Robic, and R. Buyya, *A toolkit for modelling and simulating data Grids: an extension to GridSim*, *Concurrency and Computation: Practice and Experience* **20** (2008), no. 13 1591–1609.

[14] T. Maeno, K. De, and S. Panitkin, *PD2P: PanDA Dynamic Data Placement for ATLAS*, *Journal of Physics: Conference Series* **396** (2012), no. 3 032070.

[15] A. Molfetas, M. Lassnig, V. Garonne, G. Stewart, M. Barisits, T. Beermann, and G. Dimitrov, *Popularity framework for monitoring user workload*, *Journal of Physics: Conference Series* **396** (2012), no. 5 052055.

[16] A. N. Refenes, M. E. Azema-Barac, L. Chen, and S. A. Karoussos, *Currency Exchange Rate Prediction and Neural Network Design Strategies*, *Neural Computing Applications* **1** (1993) 46–58.

[17] J. T. Connor, R. D. Martin, and L. E. Atlas, *Recurrent neural networks and robust time series prediction.*, *IEEE transactions on neural networks / a publication of the IEEE Neural Networks Council* **5** (Jan., 1994) 240–54.