EPJ Web of Conferences **214**, 06012 (2019) https://doi.org/10.1051/epjconf/201921406012
CHEP 2018

Data Allocation Service ADAS for the Data Rebalancing of ATLAS

Ralf Vamosi[1,*], *Mario* Lassnig[2,**], and *Erich* Schikuta[3,***] on behalf of the ATLAS Collaboration

[1]CERN, Geneva, Switzerland
[2]University of Vienna, Faculty of Computer Science, Vienna, Austria

Abstract. The distributed data management system Rucio manages all data of the ATLAS collaboration across the grid. Automation, such as data replication and data rebalancing are important to ensure proper operation and execution of the scientific workflow. In this proceedings, a new data allocation grid service based on machine learning is proposed. This learning agent takes subsets of the global datasets and proposes a better allocation based on the imposed cost metric, such as waiting time in the workflow. As a service, it can be modularized and can run independently of the existing rebalancing and replication mechanisms. Furthermore, it collects data from other services and learns better allocation while running in the background. Apart from the user selecting datasets, other data services may consult this meta-heuristic service for improved data placement. Network and storage utilization is also taken into account.

1 Introduction

Scientific collaborations provide experimental and simulated data for research and analysis. For this purpose, grids capable of storing and processing large quantities of data are established. The grid in which the scientific workflow of ATLAS[1] is established, is the worldwide Large Hadron Collider (LHC) Computing Grid (WLCG). Events from the experiment are recorded and stored as new files in the main data center CERN and shipped to outside data centers. Simulation data is generated globally to allow users parallel computation across the comprised data centers.

Usual computations in the workflow take place as *jobs* running at computing resources. These jobs arise from higher-level tasks, which process a collection of datasets, i.e. a larger set of files. The task is divided up into jobs and each job takes on a subset of the constituent files, i.e., its *input dataset* derived from the collection of datasets. A job outputs new files upon completion. Jobs are distributed to data centers following a method that minimizes the overall waiting time. A new job must join behind a waiting queue of already assigned jobs.

An important criterion for job execution is that the input dataset is locally available. After a data center has been chosen for a job, according to the policy of dynamic balancing between data availability and work load, missing files of the input dataset are copied to the data center.

*e-mail: ralf.vamosi@cern.ch
**e-mail: mario.lassnig@cern.ch
***e-mail: erich.schikuta@univie.ac.at,

During this stage-in process, file transfers to the destination contribute to the waiting time. The job broker generally tends to a majority decision with respect to the input files. In the best case, the dataset already lies at the local data center where the job is placed to. In summary, the job broker mainly attempts to balance out these two major time components: waiting time for execution versus waiting time for input data.

2 Motivation

Users from different institutions and work groups access data and submit tasks to be performed on the grid. After a task is broken down into jobs, each job is mapped onto shared computing resources. There are two major bottlenecks affecting the performance of the workflow.

- The computing power of each data center dealing with the workload: This determines how fast a job will be executed on average.
- The network comprises WAN (wide-area network) links between data centers: The bandwidth of a WAN link determines the average transfer rate of files.

Due to load balancing, different resources at computing centers, and distributed data, transfers must be triggered. Subsequently, delay times over network links affect the workflow. A priori file allocation can lead to an average reduction in file transfers in spite of significant uncertainties in the use of datasets and in the assignment of the corresponding jobs to data centers. These uncertainties may be addressed by measuring access frequencies and estimating values for upcoming jobs. Simulations based on usual access patterns can predict which datasets will be used. There are also variances in the number of jobs and their locations depending on job type. For example, a job with high memory constraint may only run at data centers complying with the requirements. Reducing the number of file transfers on jobs benefits the *workflow* in a two-fold manner: Firstly, the network load is reduced and better response times are achieved during times of heavy demand. Secondly, fewer file transfers are triggered in each job, resulting in a faster stage-in of the dataset and a reduced waiting time for each job.

The current standard procedure for file placement of ATLAS is to distribute data according to so-called 'memoranda of understanding', i.e., kinds of service level agreements, which are broken down to policies in data services. Users interact with the grid file system, cleaning and moving data with the aim to keep work-in-progress datasets on fully performing data centers and to move obsolete datasets to free storage. Furthermore, grid data services maintain and improve the file arrangement related to the workflow. Services for rebalancing and replicating data rank as the most essential.

3 State of the Art

In order to increase availability or performance in networks, effective data allocation is utilized. The data allocation problem has been analyzed in the past when distributed databases were studied and parallelization had to be utilized. A linear model for file allocation with storage and transmission costs is elaborated in [2], and the simulation covers five files in a network of three computers. Data placement is further modeled on different abstraction levels in [3]. This theoretical work shows that the data placement problem is extremely difficult to solve. The proof is given that the data placement problem is NP-Complete. Meta-heuristic algorithms were also applied to this kind of problem. In [4], data allocation strategies have been investigated to reduce transaction costs. A genetic algorithm was used here, with the goal of limiting communication effort between data centers by balancing the load.

Other database approaches attempt to arrange data effectively over the network nodes, such as in [5, 6]. However, these studies investigate idealized database cases. For example, they focus on a single query type or do not consider any constraints on communication characteristics. These studies could be improved by using behavior and workflow characteristics. Analysis of access patterns would be beneficial for network utilization.

A well-known approach is ranking data according to the number of accesses per time unit. This characteristic is referred to as data *popularity* [7]. In [7], a successful popularity model is established which uses different structured and unstructured sources for collecting historical data. A popularity model is implemented as an autonomous service for finding obsolete data and used in the cleaning process in [8, 9].

In [10], a data placement optimizer is presented for the hybrid storage network of the LHCb physics collaboration. Hybrid means that different storage technologies are used and different cost factors are thereby assigned in the machine learning algorithm, which further includes the popularity of data. The LHCb Grid Simulator provides a platform for simulating an LHCb computer model in which different types of computer jobs are distinguished [11]. For the job broker system, two pull methods are considered: In the Simple Model, a job is sent to the first available site. In the Data Availability Model, a job is sent to the first site which provides the job's input data.

Replication and rebalancing are important tasks to ensure stable operation and a flawless workflow. On the ATLAS data grid, a replication service, called C3PO, creates secondary copies of datasets and distributes them to free storage nodes [12]. Secondary copies are replicated versions of datasets that increase accessibility and performance across the grid. In order to ensure full functionality of single storage nodes on the grid, the rebalancer service, called BB8, monitors free capacities and the ratio between primary and secondary copies of files [13]. In a sub-optimal case defined by policies, it deletes local data or moves data to other storage nodes.

In summary, research and application often concentrate on models applied in particular cases. In the data allocation context, storage resources and the use of data have to be appropriately treated in the process of file placement [14]. This proceedings outlines one approach that aims to improve the data-intensive workflow. Meta-heuristic techniques shall be incorporated into a data allocation algorithm as a service which extends the possibilities for better data allocation on the grid.

4 Data Allocation Service

The aim is to carry out data allocation that uses methods that reduce time-related costs in the workflow as far as possible. When observing the workflow, two major time delays can be identified. Jobs are designated in a manner that balances out the computing and the networking load:

- The job broker system places jobs onto data centers, where jobs are in turn allocated to workers. Normally, a newly created job waits in a queue within the data center due to the constant incoming load. This wait makes up a significant component of the total job time, particularly when compared to its run time. The job broker attempts to reduce this waiting time through job allocation to worker nodes.
- The job broker also aims to effectively allocate jobs in order to reduce a second major time delay. This delay equates to waiting time for a job's necessary input data. Missing files from the job's input dataset must be transferred to the target data center, which results in waiting time. It is highly evident that data allocation has an impact on the time component.

The allocation task attempts to minimize the waiting time for input data. In order to effectively achieve this, one should consider placing files as close as possible to their corresponding jobs. In the stochastic domain of the dataset access, at least two obstacles arise:

- What are the interesting files, say, the files that are more likely to be accessed? The prediction of this is attached to uncertainties.
- Where will a job be placed and run, given an input dataset? This depends heavily on dataset distribution and the changing load on data centers. Therefore, hidden uncertainties should be taken into account when predicting the job's target node.

ATLAS is using straightforward approaches and an algorithmic placement which exhibits throughput problems. In this R&D, possibilities are explored to use machine learning as an alternative. Data allocation based on heuristics remains a difficult albeit necessary task. Such a core allocation algorithm is incorporated into a service component compatible with the data services described above and must meet the following requirements:

- Allocation optimization may be applied iteratively each time files are transferred.
- The allocation service must attend to other data services and take into account the current state of the grid. The replication service as well as the rebalancing service influences the data arrangement. Users interact continuously with the workflow management system to manipulate datasets. Already used concepts, e.g. data popularity, can be incorporated to support the data allocation.
- For automatic reallocation acting on preselected datasets, the service must have the correct transfer rate in order not to impose too many transfers on the network.

4.1 Allocation and Reallocation

The core concept of the allocation algorithm is based on a meta-heuristic algorithm such as in [15], where the datasets containing files to be optimally allocated are passed on to the optimization algorithm. This input set can be moved around in the file system, whereas the complementary set is not allowed to move. The allocation algorithm runs several optimization iterations, from which the best is carried out. For any dataset selection, the target metric indicates the improvement gained from the possible optimization step. There are two working methods within an allocation algorithm:

- *Automatic reallocation*: New files packed into datasets originating at data centers can be selected for reallocation, which will be then moved towards lower costs under the constraints. On top of new datasets, used datasets can be also included in the selection.
- *Semi-automatic reallocation:* A data service wants to copy or move a certain set of datasets.

4.2 Constraints

The allocation algorithm operates at *dataset level*. Datasets are user-defined collections of files that correspond to each other. For example, events collected in one LHC run are commonly packed into one dataset. They are tendentially not divided and placed onto several nodes. Datasets can be categorized and analyzed by their attributes via mining and machine learning techniques. Special values such as popularity and dependency can be associated with datasets. Popularity is a heuristic that predicts how likely the dataset will be accessed by an upcoming job in a certain time window. A noticeable portion of the total datasets does not provide popularity values, causing them to be ranked lower. Dependencies between datasets indicate how similar datasets are. For dataset overlaps, it can be predicted how likely files of

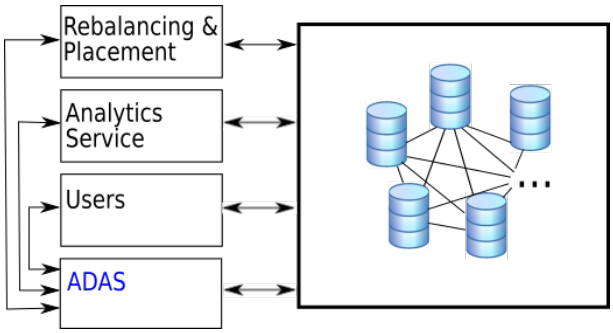

Figure 1. The learning agent ADAS plays an auxiliary role in delivering better data allocation and proposing information on it.

two datasets will be processed by an upcoming job, and popularity can be derived. Datasets can be further clustered to super-datasets for generating collections and finally allocated to free gaps at storage nodes [15].

The size of the *dataset selection* taken in one optimization step can be increased or decreased by the transfer rate according to how much network load can be put on the network. Shipping large amounts of data back and forth must be avoided. Larger subsets give more optimization potential. Due to storage and network constraints, smaller subsets are chosen in each iteration step. A global solution embracing all datasets would be incomparably more complex to achieve and vast load could not be handled by the network. For efficiency, a useful subset of *network nodes* – the most important data centers from a storage point of view – are covered by the allocation algorithm. Local optimization within these nodes leads to a strong overall improvement. In the concrete example of the ATLAS data grid, it can be seen that the ten biggest data centers hold over a half of the total data.

5 Model

The allocation algorithm shall be deployed as a service component. ADAS – the ATLAS Data Allocation Service – will run in the background, being independent of the other services. The cooperative interaction between ADAS and other data services within the workflow allows possibilities for improving the operation of these services. The service may be switched to fully automatic for temporary allocation improvements. This pushes the situation towards a more optimal arrangement. Generally speaking however, automatic reallocation is concurrent with the other data services, since they operate independently and do not pay attention to external actions. Common datasets could be transferred multiple times when interfering with another data service.

The information flow is depicted in Figure 1. ADAS occupies another place among the grid data services. Before any dataset placement, a service can consult ADAS about a possible group of datasets. ADAS replies with a list of datasets that should be moved. For instance, it is better for a dataset that has recently been moved in the allocation process to remain stationary, since it is already part of an improved, cost-reducing allocation. New promulgated locations for datasets are optional. If the service is unable to move files in the first place, or when it contradicts performed actions or internal policies, it simply ignores the proposed target. The datasets will then not be flagged as reallocated in ADAS. The interface to other components works as follows:

- The rebalancer service dynamically rebalances data between data centers in order to avoid possible overloads at the storage nodes. The intention is to keep free storage capacity or to free storage up if necessary on monitored storage nodes. One possibility is the deletion of secondary copies of datasets, i.e. copies not needed at this time. The second possibility is that the rebalancer moves datasets whose target can be determined by ADAS.
- The placement service places secondary copies of datasets to distinguished storage resources. Each time a replication of a dataset is made, the location can be determined by ADAS.
- Sometimes the user will be in a position to indicate important datasets that will be used more frequently. This information should be put into ADAS, which increases the popularity values of these datasets.
- Other services such as analytics services deliver updates on popularity values (Pop) or dataset dependency values (DSD).

For the core allocation algorithm, a *meta-heuristic evolutionary algorithm* has been deployed [15]. It implements the two previously described heuristics data popularity and dataset dependency to ensure that more similar datasets with higher access likelihoods are collected at certain single storage nodes. The evolutionary method is able to take file system snapshots across storage nodes and to find a better reallocation for a subset of these data. The solution space is constrained with the available storage spaces determined by the overall data allocated and storage capacities available. In an evolutionary iteration, swaps of allocated subsets of datasets are tested by the imposed cost metric. This process is undertaken until a good solution is found, which can be executed by moving of chopped up subsets between the storage nodes. To ensure that the file system consistently remains in a good state, file system snapshots must repeatedly be taken in order to evaluate the current state. After a file system snapshot becomes obsolete, i.e., data have been moved or deleted, the optimization method has to be triggered again to determine a new collection of datasets and its allocation. In general, this collection of datasets to be moved, can be selected based on a tournament selection of several subsets of datasets. The subsets of a tournament are initially chosen based on the following: Each subset takes high dependent datasets. Datasets with higher popularity have priority over others. Already allocated datasets are down-ranked. This evolutionary process between selected collections ensures a wider extent of the search of minima, as opposed to a more limited case.

In a simplified experiment, a model with 20 data centers in a network is implemented. Each of the data centers gets a random storage capacity and can hold between 200 and 2000 files for the sake of runtime of the simulation on a single personal computer. This adds up to approximately 20 k files and 1 k datasets. Datasets comprise different numbers of files with a variance based on real datasets. The simulation initiates multiple runs in different configurations. Jobs access different shares of the popular datasets in each configuration. A configuration with 20 % popularity comprises jobs with an expected 20 % access rate to datasets with some popularity and 80 % to purely random datasets. The majority of datasets don't possess any popularity, i.e., their popularity is 0. Highly popular datasets are accessed more likely than others. Figure 2 presents the results of the optimized file allocations for different configurations. The random case shows the initial datasets spread across available network storages. The optimized case gives the achieved outcomes from the allocation algorithm in the experiment. The horizontal axis depicts the number of popular datasets among all datasets. The vertical axis shows the number of file transfers per job. The overall network load is lessened with this data allocation algorithm, resulting in a cost reduction of computational jobs in relation to expected WAN transfer time. Even though the experiment is based on

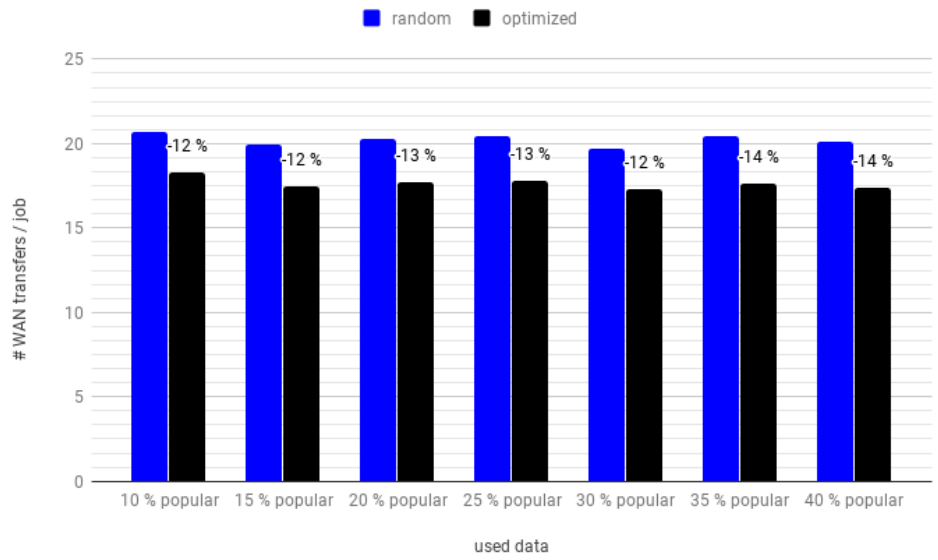

Figure 2. Runs with 20 data centers comprising different job configurations in terms of used data popularity

a simplified model, it shows how data access can be improved by data placement. However, the implemented algorithm must be upgraded, for example, with the necessary interfaces for passing enabled datasets and setting a transfer rate.

6 Conclusion

Improving the file allocation within the grid is an important task to ensure quick execution of the workflow. Regarding automatic file allocation on the grid, the following drawbacks should be noted:

- Concurrency occurring among several services.
- The complexity is high.
- Automatism might not be trusted. Users want to have the control.

This proceedings proposes an integral approach to combat these shortcomings with the service component ADAS. The approach can be integrated in the established environment with data rebalancing service and the data placement service in the distributed data management system Rucio of the ATLAS collaboration at CERN. Generally speaking, an automatic mode would not be necessary due to data services running on the grid. On the other hand, an automatic mode would prove useful when important datasets, say for an upcoming conference, are chosen and given to ADAS. The allocation service ADAS then acts pro-actively on user-enabled datasets, which are processed in the described manner of taking turns of smaller portions each. Through regulating by a transfer rate, the number of reallocated datasets per time unit can be set, to target a 'gentle' and continuous background improvement. Combined with other services, ADAS utilizes important information, e.g. the popularity of datasets, and works out an improved data allocation. The heuristic approach including network and storage

utilization in the target metric may lead to better data allocation, and enabling a faster paced workflow.

References

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

[2] W.W. Chu, IEEE Transactions on Computers **100**, 885 (1969)

[3] D.A. Bell, The Computer Journal **27**, 315 (1984)

[4] W. Guo, X. Wang, *A data placement strategy based on genetic algorithm in cloud computing platform*, in *Web Information System and Application Conference (WISA), 2013 10th* (IEEE, 2013), pp. 369–372

[5] K.A. Abdel-Ghaffar, A. El Abbadi, *Optimal allocation of two-dimensional data*, in *International Conference on Database Theory* (Springer, 1997), pp. 409–418

[6] S. Berchtold, C. Böhm, B. Braunmüller, D.A. Keim, H.P. Kriegel, *Fast Parallel Similarity Search in Multimedia Databases*, in *Proceedings of the 1997 ACM SIGMOD International Conference on Management of Data* (ACM, New York, NY, USA, 1997), SIGMOD '97, pp. 1–12, ISBN 0-89791-911-4, http://doi.acm.org/10.1145/253260.253263

[7] D. Bonacorsi, T. Boccali, D. Giordano, M. Girone, M. Neri, N. Magini, V. Kuznetsov, T. Wildish, *Exploiting CMS data popularity to model the evolution of data management for Run-2 and beyond*, in *Journal of Physics: Conference Series* (IOP Publishing, 2015), Vol. 664, p. 032003

[8] F.B. Megino, M. Cinquilli, D. Giordano, E. Karavakis, M. Girone, N. Magini, V. Mancinelli, D. Spiga, *Implementing data placement strategies for the CMS experiment based on a popularity model*, in *Journal of Physics: Conference Series* (IOP Publishing, 2012), Vol. 396, p. 032047

[9] D. Spiga, D. Giordano, F.H. Barreiro Megino, *Optimizing the usage of multi-Petabyte storage resources for LHC experiments*, in *Proceedings of the EGI Community Forum 2012/EMI Second Technical Conference (EGICF12-EMITC2). 26-30 March, 2012. Munich, Germany. Published online at https://pos.sissa.it/162/107/* (2012)

[10] M. Hushchyn, P. Charpentier, A. Ustyuzhanin, *Disk storage management for LHCb based on Data Popularity estimator*, in *Journal of Physics: Conference Series* (IOP Publishing, 2015), Vol. 664, p. 042026

[11] M. Hushchyn, A. Ustyuzhanin, K. Arzymatov, S. Roiser, A. Baranov, *The LHCb Grid Simulation: Proof of Concept*, in *Journal of Physics: Conference Series* (IOP Publishing, 2017), Vol. 898, p. 052020

[12] T. Beermann, M. Lassnig, M. Barisits, C. Serfon, V. Garonne, A. Collaboration et al., *C3PO-A dynamic data placement agent for ATLAS distributed data management*, in *Journal of Physics: Conference Series* (IOP Publishing, 2017), Vol. 898, p. 062012

[13] M. Barisits, C. Serfon, V. Garonne, M. Lassnig, T. Beermann, T. Javurek, A. Collaboration et al., *Automatic rebalancing of data in ATLAS distributed data management*, in *Journal of Physics: Conference Series* (IOP Publishing, 2017), Vol. 898, p. 062006

[14] H. Sato, S. Matsuoka, T. Endo, *File clustering based replication algorithm in a grid environment*, in *Proceedings of the 2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid* (IEEE Computer Society, 2009), pp. 204–211

[15] R. Vamosi, M. Lassnig, E. Schikuta, *Data Allocation Based on Evolutionary Data Popularity Clustering*, in *International Conference on Computational Science* (Springer, 2018), pp. 153–166