Data Allocation based on Evolutionary Data Popularity Clustering

Ralf Vamosi[1], Mario Lassnig[1], and Erich Schikuta[2]

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

Abstract. This study is motivated by the high-energy physics experiment ATLAS, one of the four major experiments at the Large Hadron Collider at CERN. ATLAS comprises 130 data centers worldwide with datasets in the Petabyte range. In the processing of data across the grid, transfer delays and subsequent performance loss emerged as an issue. The two major costs are the waiting time until input data is ready and the job computation time. In the ATLAS workflows, the input to computational jobs is based on grouped datasets. The waiting time stems mainly from WAN transfers between data centers when job properties require execution at one data center but the dataset is distributed among multiple data centers. The proposed novel data allocation algorithm redistributes the constituent files of datasets such that the job efficiency is increased in terms of a cost metric. An evolutionary algorithm is proposed that addresses the data allocation problem in a network based on data popularity and clustering. The number of expected job's file transfers is used as the target metric and it is shown that job waiting times can be decreased by faster input data readiness.

Keywords: Grid computing, Data layout, Distributed data management, Data allocation, Data placement, Popularity, Data clustering

1 Introduction

Grid computing aggregates distributed computing, storage, and network resources to support unified, secure, and coordinated high-level access to the combined capabilities [9].

The ATLAS experiment at CERN acquires, stores, and processes data for detector operation and physics analysis. For these purposes, it utilizes part of the worldwide Large Hadron Collider (LHC) Computing Grid (WLCG), which is here referred to as the *grid*.

The data acquisition process deals with huge quantities of information. After passing a hardware trigger, event data from the experiment is immediately stored in form of files at the local CERN data center. Experimental data is usually packed into files in the GB range, which are subsequently transferred to other data centers. Computational jobs, which are referred to as *jobs*, read sets of files which contain different numbers of files. Those file collections are referred to as

ICCS Camera Ready Version 2018
To cite this paper please use the final published version:
DOI: 10.1007/978-3-319-93698-7_12

datasets. Each job is assigned to a data center in order to minimize the expected total waiting time. A job must wait for a free job slot, i.e. until the computation can commence on a worker node of the data center. After a data center is chosen, the job has to wait for the input data, i.e. its input dataset.

2 Motivation

In the LHC Computing Grid, physicists from institutions all over the world submit tasks to process stored data on the grid. To perform analysis, users prepare tasks on specific sets of data, which are then executed in form of jobs at the shared resources of the grid.

In such data-intensive workflow, the network represents a bottleneck for data transfer due to the high amount of file transfers triggered by jobs: Every time the corresponding computation takes place at a node, missing input data must be shipped over the network to the target node. The posed network load resulting in transfer delays represent a time-consuming part of any data-intensive job in the grid. This is especially problematic due to the fact that computational job properties, e.g., memory requirements, do not conform to the amount of available storage. There can be data centers with large storage systems but few processing capabilities, and vice versa.

The current operating standard procedure for storing data across storage resources is to distribute data *uniformly* in free storage resources. To cope with this situation, users interact with the grid storage system, cleaning and moving data with the aim to keep work-in-progress data sets on available data centers and to remove obsolete data sets to free storage.

The following simplified example provides an incentive for the optimization process. Three data centers supplying computing and storage resource are assumed. Without loss of generality, each up- and downlink shall be symmetrical. The time period in which a job waits for its input dataset is referred to as *'waiting time'* T_{wait}. In this scenario, the average T_{wait} between data center 1 and 2 is 2 h, between data center 2 and 3 is 0.5 h, and between data center 1 and 3 is 1 h yielding approximated T_{wait} coefficients of

$$\overline{\boldsymbol{T}}_{wait} = \begin{bmatrix} 0 & 2 & 1 \\ 2 & 0 & 0.5 \\ 1 & 0.5 & 0 \end{bmatrix} h$$

where we assume that data center internal LAN links do not pose delays. The first data center has a large storage capacity, whereas the others don't. In general, statistics must be gathered about the usage of the network, i.e. jobs, and their access patterns in order to compute expectation values:

- Datasets and their files are associated with access likelihoods
- For a job, likelihoods emerge for running on data centers

Now estimations for the waiting time for two different scenarios are given. The first scenario covers jobs with high memory requirements running at data center

ICCS Camera Ready Version 2018
To cite this paper please use the final published version:
DOI: 10.1007/978-3-319-93698-7_12

state	data center 1	data center 2	data center 3	T_{wait}
scenario 1				
uniform	33 % files	33 % files	33 % files	2 h
optimized	50 % files		50 % files	1 h
best	100 % files			0 h
scenario 2				
uniform	33 % files	33 % files	33 % files	1.66 h
optimized	50 % files		50 % files	1.33 h
best			100 % files	0.5 h

Table 1. File allocation and the effect of optimization in two use cases.

1. Only data center 1 is capable of running this job type. Different possibilities of data allocation for these jobs and respective waiting times are illustrated in Table 1.

Scenario 2 considers jobs that are being balanced out across the computing resources. In this case, 33 % at each of the three data centers. Data allocation and observed values are depicted in Table 1.

In *scenario 1*, data center 1 is able to accommodate the set of jobs, thus the best outcome can be achieved through placing all possible input files to the corresponding storage node. Upon assignment of these jobs, the job broker places them at data center 1. The job execution can launch promptly since the input files are available at this site. T_{wait} vanishes. Even in the *scenario 2*, where the considered jobs are being mapped equally likely across all data centers, the best optimization results in a reduction of waiting time from 1.66 h to 0.5 h. This example illustrates the basic idea although the network load was assumed constant with fixed delay times over network links. The considered jobs represent a small amount of the total quota.

3 State of the Art

In the context of data sharing, there are two shortcomings that have been studied and trialed extensively: *Data allocation* and *duplication of data*. Data allocation should reduce default communication cost. Replication results in multiple data source nodes to enable routing from these sources to the target node. Replication is not addressed in this work.

File allocation problem and data allocation problem is here used interchangeable, even though this may not be true from a strict semantic point of view and both generally differ from each other. In the data allocation process, the objects dealt with are not necessarily fixed, or unknown a priori. The relationship between these objects can be complex and access to some of the data may demand several transmissions across involved data.

The data allocation problem has been analyzed in the past when distributed databases were studied and parallelization had to be utilized. In [8], a linear

ICCS Camera Ready Version 2018
To cite this paper please use the final published version:
DOI: 10.1007/978-3-319-93698-7_12

model for file allocation with storage and transmission costs is elaborated, and the simulation covers five files and three computers.

In [11], a store-and-forward computer network is investigated. Network constraints are considered to be availability and delay, which is demonstrated in a model with a total of 10 files.

Data placement is modeled on different abstract levels in [3] and it is proven that the data placement problem is extremely difficult to solve. It is shown that data placement problem is NP-Complete.

The file allocation problem is discussed under concurrency constraints to build a model with storage cost and communication cost in [14]. Constraints of the model are the multiplicity of databases, variable routing of data, and available capacity of the network nodes.

Attempting to cluster datasets according to their interdependency and subsequently to store the clusters on separate machines have been investigated [9].

The problem has been coined as solving a multi-constraint hypergraph partitioning problem for task and data assignment [6].

A further clustering algorithm is described in [19] that uses k-means algorithm for finding locations for the clustered data, resulting in task allocation to the data centers with most of the input dataset. This is comparable to the ATLAS workflow.

Evolutionary algorithms were also applied to this kind of problem. In [10], 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.

Replication and placement of files across different nodes can lead to improvements in job execution, makespan, and bandwidth [7]. However, caution is advised since a priori movement of data to improve the access to data can lead to a bottleneck in network performance between data centers. To avoid drops in performance, data replication is used in the ATLAS grid very rarely and selectively [12].

Further efforts have been undertaken in previous studies for database optimizations. The authors of [18] discuss database allocation optimization and propose a mathematical model concerning average waiting time. Other database approaches attempt to arrange data effectively over the network nodes, such as in [1,2,4]. 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. Room for improvement would comprise user behavior and workflow characteristics. Analysis of access patterns can 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* [5]. In [5], 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 [13,17].

ICCS Camera Ready Version 2018
To cite this paper please use the final published version:
DOI: 10.1007/978-3-319-93698-7_12

Summing up, the actual research on data partitioning and allocation techniques is specific or limited. Due to complexity and variety, simplified models and local optimizations were studied and applied. A thorough analysis of data usage and data optimization point of view for data grids is necessary. Storage resources and the use of data has to be appropriately treated in the process of file placement [16,15]. The lack of information has prompted this work, where grid computational application uses data in big quantities. Research on data management will have a strong impact on usability, performance, cost and the acceptance of worldwide spanning grids on a large scale.

4 An Evolutionary Data Allocation Approach

4.1 Idea

For data-intensive workflows, an evolutionary data cluster and allocation algorithm is proposed to minimize the overall network load and thereby reducing average waiting times across the network.

The approach is based on an evolutionary algorithm generating solutions to the allocation problem built upon two heuristics to improve the decision-making process. Thus, the data allocation approach rests on two solid pillars which are combined in a novel way:

The algorithm's *first pillar* is calculating the data dependency and thus possibly clustering the dataset on correlated features. These features are subject to big data studies. Today, machine learning algorithms classify and assess the multi-dimensional datasets of ATLAS already. Filters and clustering can be applied to extract parameters and collect statistics.

The *second pillar* of the proposed algorithm is the popularity of datasets that stipulates how likely the dataset will be accessed in the near future. The best practice would be to allocate more popular files to more efficient, higher performing data centers.

4.2 Data Dependency

Data dependency is used for estimating the likelihood of files being in the same input dataset essential to perform a job. Files within datasets with high dependency provide more similar features, e.g. type of contained data, naming etc., and are more likely to be part of a new common dataset processed together by a job.

In this context, a simple idea relates to clustering highly dependent datasets together such that new datasets for upcoming jobs are more likely to find the majority of those files at a single node. Thus, fewer files have to be transferred to complete the datasets at the target node. The fewer WAN transfers per job, the more LAN transfers per job occur. In general, these data center internal transfers occurring between the storage node and the worker node at the same site are, however, not the focus of attention, since they are sufficiently and quickly

ICCS Camera Ready Version 2018
To cite this paper please use the final published version:
DOI: 10.1007/978-3-319-93698-7_12

executed, thus they may be ignored in terms of performance. The use of internal LAN transfers rather than WAN transfers reduces the bottleneck effect on the global WAN network.

In the implemented simulation, dataset dependency (DSD) values are mapped to pairs of datasets, i.e., $DSD : \{DS\} \times \{DS\} \rightarrow [0, 1], (DS_n, DS_k) \mapsto DSD_{n,k}$. The mapping can be normalized to 1 for each sample set.

This is represented as a symmetrical matrix with 1s in the main diagonal and values < 1 otherwise.

4.3 Data Popularity

Data popularity describes the usage importance of files by the number of occurred accesses to them. The chosen time window and the weighting for counting are not subject here. It is used as a measure for how likely a data element, here files in datasets, will be accessed in the near future. More often accessed datasets will be more likely to be accessed in the near future. In the simulation, normalized data popularity values are assigned to all datasets, i.e., $Pop : \{DS\} \rightarrow [0, 1], DS_n \mapsto Pop_n$.

Figure 1 depicts popularity values of datasets in the test cases with 20 % respectively 35 % access rate to popular data. The leftmost bin comprises 80 % respectively 65 % accesses to data with popularity value 0, i.e., the portion of data for which no conclusion about popularity could have been drawn. It can be seen that very popular data is used more often than less popular data.

4.4 Data Arrangement S

The solution of the allocation problem is denoted as a *storage matrix* of a set of possible storage matrices, $S \in \mathcal{S}$, by clustering and storage allocation. The storage matrix S is a *permutation matrix* specifying the mapping of files to storage nodes. The set $\mathcal{S}$ contains the matrices which obey the storage constraints in the inequality 1. Jobs read file collections expressed by so-called file datasets. Files of those datasets are arranged across storage nodes as defined by the output of the algorithm, S.

4.5 Algorithm

Every file in the grid is assigned to a respective dataset. Overlaps are omitted. However, if original datasets have overlaps, these can be eliminated before applying test jobs: Intersections become new sub-datasets with, for example, composed values for popularity and dependencies. For test job samples, datasets are generated out of existing ones based on popularity, so that more popular files are more likely moved into new datasets. In this analysis, approximately a half of the files have a popularity value of 0 which means there is no corresponding information on these files. So far, these files would be considered unclassifiable if, for instance, they were of a type never seen before.

ICCS Camera Ready Version 2018
To cite this paper please use the final published version:
DOI: 10.1007/978-3-319-93698-7_12

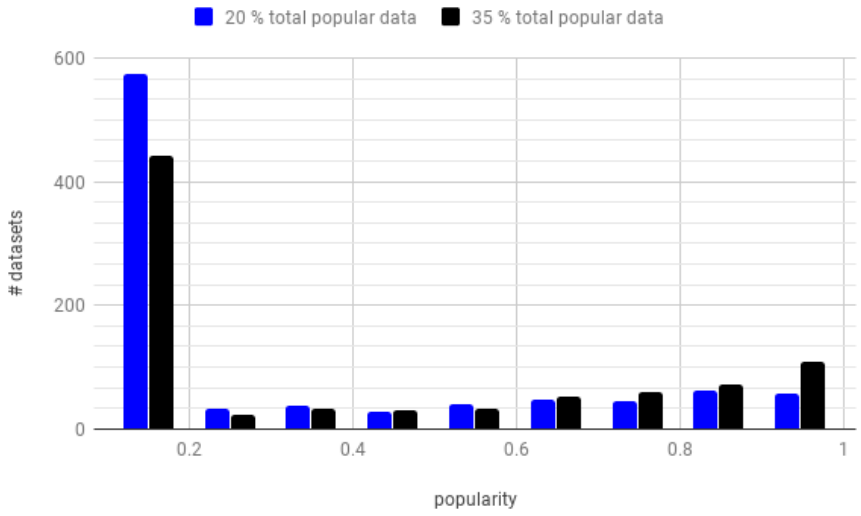

Fig. 1. Access frequencies of datasets with and without popularity. Datasets with popularity are accessed with a rate of 20 %, and 35 %. The leftmost bin implies the majority with 0 popularity.

The chosen data centers provide all storage capacities and computing power, i.e. they are represented as computing and storage nodes in the simulation. The simulation can be parametrized with special cases: For example, if a data center represents a data warehouse or cloud storage, the workload manager could be set to omit this node and no WAN transmission would occur to this node in Equation 1.

The proposed algorithm generates a file arrangement S with the goal to minimize the expected number of WAN transfers per job. After each evolutionary iteration, which produces a version of S, the target metric is evaluated again and the evolution takes place as described in details below. Given a test job sample, the problem can be formalized with S as a parameter and the expected cost functional $\#tx_{WAN}$ as a target metric. $\#tx_{WAN}$ can be decomposed into single $\#tx_{WAN}(i)$ giving the triggered WAN transfers for job i:

$$\begin{aligned} &\underset{\boldsymbol{S}}{argmin}\ \mathbb{E}[\#tx_{WAN}] \approx \underset{\boldsymbol{S}}{argmin} \sum_{j \in \{jobs\}} \#tx_{WAN}(j) \\ &such\ that \\ &\boldsymbol{S}^T \times \mathbf{w}_{file} \leq \mathbf{w}_{storage} \end{aligned} \tag{1}$$

where $\mathbf{w}_{file}$ is a column vector which represents the file sizes for $file_1$,...,$file_M$ and $\mathbf{w}_{storage}$ is a column vector which represents the storage capacities of data

ICCS Camera Ready Version 2018
To cite this paper please use the final published version:
DOI: 10.1007/978-3-319-93698-7_12

centers 1, 2, ..., N. The expectation value is denoted as $\mathbb{E}$. A normalization is not needed for *argmin* seeking a minimum of the expression.

The Algorithm 1 illustrates the working principle of the approach. **Step 1**, building the DSD matrix for the given datasets, must be performed only once. **Step 2.1** covers the evolutionary part of the file partitioning. A clustering algorithm of the algorithm evaluates the file affinity to the constructed partitions $S_1, ..., S_N$ by the dataset dependency matrix. Datasets with higher DSD belong together and their files should be placed in one common partition if possible. The coarseness on the dataset level rather than file level is beneficial for a quicker evaluation. Datasets are logical clusters themselves, whose properties allow comparison to each other. In each solution, i.e., an entity in the population, a random number of partitions prefer high popularity data, whereas the rest of the partitions do not. Files are collected in a random trajectory which determines the aimed solution in the solution space.

Step 2.2, after constructing the partitions $S_1, ..., S_N$, they are logically mapped to the data center storages in terms of minimization of the target metric in Equation 1. If a partition does not fit into a data center, the drop-out files will be mapped to other free gaps of storages not yet completely filled under the application of the target metric as well.

Step 2.1 rapidly results in generations with a fixed number of partitions. The storage constraint holds in *step 2.2* if a total set of files is picked such that there is sufficient total storage capacity under the considered circumstances. Should the capacity constraint not be obeyed, only a subset may be allocated in place of the optimized storage nodes, and the complementary subset, which must be discarded, are allocated to storage nodes outside of the optimized storage nodes. The complementary subset consists of the files less worthy according to the heuristics.

A less strict optimization goal could be addressed by dealing with a subset of the total files, but a deeper analysis would exceed the scope of this work.

Each entity in the population is evaluated by its achieved performance in terms of the metric in Equation 1. It is evaluated by simulating a job sample set and summing up incurred WAN data transfer values for each computing job. Over several iterations, evolution continues until no further improvements can be achieved. The process comprises selection, mutation, and crossover steps which are iteratively applied in-between:

- Mutation partially alters clustering of files and mapping those partitions to the storages as described above. The prior outcome S transforms into another $\tilde{S}$.
- Crossover merges the initial clustering and mapping of two entities into the common offspring. If both entities share overlapping files mapped to different storages, not yet assigned files of those overlaps are assigned uniquely for a single target storage. This avoids illegal double use since files can only be selected once in the allocation.
- Selection keeps the best entities. If the population starves, then new entities are added to enrich the pool of solutions.

ICCS Camera Ready Version 2018
To cite this paper please use the final published version:
DOI: 10.1007/978-3-319-93698-7_12

```
// Generate dataset dependency values (DSD) , i.e. pairwise
   correlation values (interdependencies) (step 1)
DSD_{n,k} ← cor(n,k)  ∀n,k ∈ {datasets}
N ← number of data centers
// Generate population (step 2)
while {termination condition not met} do
    while {population starved} do
        // Add one entity S ∈ 𝒮 with N random dataset clusters S_n ∈ S
        k ← random[1,N]
        S ← (S_1 = {}, S_2 = {}, ..., S_N = {})
        datasets ← available datasets on storage_{1,..,N}
        while {datasets.nonempty()} do
            foreach {S_{n=1,...,k}} do
                S_n.add(datasets.get(S_n.dependency(high), popularity(high)))
            end
            foreach {S_{n=k+1,...,N}} do
                S_n.add(datasets.get(S_n.dependency(high), popularity(low)))
            end
        end
        population.add(S)
    end
    // Make compatible partitions allocated to network storages for
       each entity
    foreach {S ← population.next()} do
        (S_1, ..., S_N) ← (S.S_1, ..., S.S_N)
        foreach {S_n} do
            foreach {storage_{1,...,N}} do
                storage_k.add(S_n.top(popular)) // add datasets
                S_n.remove(storage_k) // remove allocated datasets
            end
        end
        foreach {S_n.nonempty().top(size)} do
            // not allocated
            foreach {storage_k.nonfull().top(size)} do
                // not yet completed storage, biggest first
                storage_k.add(S_n.top(popular)) // add datasets
                S_n.remove(storage_k) // remove allocated datasets
                if {S_n = {}} then
                    break // inner loop
                end
            end
        end
    end
    population.apply({selection, mutation, crossover})
end
return population.best();
```

Algorithm 1: Popularity-based data allocation algorithm

ICCS Camera Ready Version 2018
To cite this paper please use the final published version:
DOI: 10.1007/978-3-319-93698-7_12

5 Evaluation

This section shows the efficiency of the algorithm on the basis of the simulated network of data centers. The data centers should represent the most important ones of the grid in reality, i.e., they provide most of the total capacity. This set of data centers can be extended. However, from the perspective of data management, the rest of the grid holds less significance, since the situation would considerably improve through better management just in these data centers. Datasets are present with the following characteristics:

- Datasets comprise 10 to 40 files to reflect dynamic dataset sizes.
- A popularity value is assigned to each dataset.
- A dataset dependency value (DSD) is assigned to each pair of datasets.

A job sample is generated in the following way:

- Each new job is assigned to a newly generated dataset derived from the pre-defined datasets covering all files. Thereby random files are selected from the pre-defined datasets.
- The workload manager, i.e., job broker, is set to aim at WAN transfer minimization.

Each of the data centers can hold between 200 and 2000 files for the sake of run time of the simulation on a single personal computer. Capacities are chosen randomly at the start of the simulation. This adds up to approximately 20 k files and 1 k 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 use significance that jobs have an expected 20 % access rate to datasets with some popularity and 80 % to purely random datasets, including the majority, i.e. datasets with popularity=0. Highly popular datasets are accessed more likely than others, as seen in Figure 1. Figures 2 and 3 present the results of the optimized file allocations for different configurations:

- The *random case* shows the initial random dataset spread across available network storages.
- The *optimized case* gives the achieved outcomes from the optimization algorithm in the experiment.

It is evident that the utility of popularity declines as the number of data storages in the network increases. This could contribute to a weakness of the algorithm's evolution process in varying the target popularity of the different partitions. In addition, the number of WAN transfers per job increase by the number of possible network nodes. Input data is spread across more network nodes. Furthermore, there are more network nodes where a job can execute, increasing the uncertainty for target nodes where the input files are requested and transmitted to.

A better outcome can be achieved in networks with a smaller number of data storages. The complexity of the optimization problem grows due to bigger

ICCS Camera Ready Version 2018
To cite this paper please use the final published version:
DOI: 10.1007/978-3-319-93698-7_12

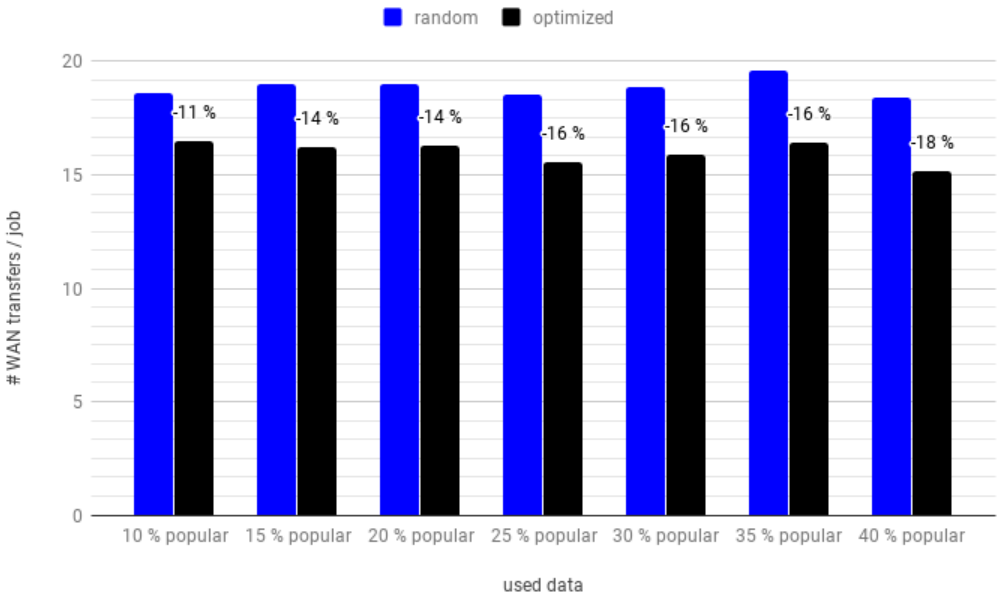

Fig. 2. Runs with 10 data centers comprising different job configurations in terms of used data popularity

networks and so the rise in the number of possibilities in the solution space worsens the convergence rate. The descent of the cost function towards higher total popularity rate is explained by more predictability on some part of the file population. In any case, a strong deviation from this pattern can be observed by the randomness of the load and the data access in the experiment. Random effects in choosing worker nodes and associated data may lead to distorted results.

The workload manager plays a central role in the technique proposed in this work. A network metric, such as average network load, depends directly on the task of the workload manager. The job has to be placed most likely to the node with the majority of input files. In the hypothetical worst case, jobs would run just remotely from the input data. This situation would bear the maximum load on the network due to the fact that all the input files have to be transferred.

This goes in hand with the described data management which is rooted in the described DSD clustering. The used clustering adjusts to the stochastic usage of global data. Hashing algorithms take the reverse way which provides data arrangements with efficient parallelism to maximize throughput. However, in this consideration, since all network nodes provide some portions of data, parallelization occurs through the workload manager combined with the data management of interest.

ICCS Camera Ready Version 2018
To cite this paper please use the final published version:
DOI: 10.1007/978-3-319-93698-7_12

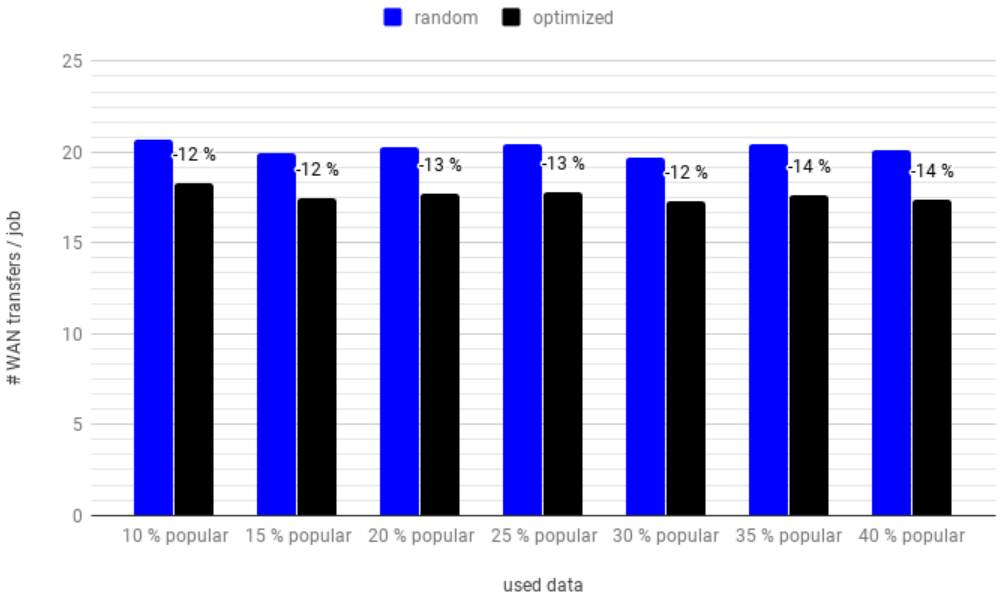

Fig. 3. Runs with 20 data centers comprising different job configurations in terms of used data popularity

6 Conclusion

The proposed algorithm clusters and allocates grid data in a more sophisticated way by focusing on the importance of data and minimizing the effort by changing the solution slightly. More popular data should be held together at better performing data centers. With this form of data allocation policy, less overall network load and shorter network waiting times occur, which reduces the cost of computational jobs in terms of expected WAN transfer time. The adaptable model uses a rapid popularity-based partitioning for optimized clustering that in general can be applied to other workflow environments and used for various control and optimization tasks in dynamic systems.

However, so far, the proposed model represents an idealized toy model which was optimized in terms of the overall inter-data center communication on a basis of transparent data in a uniform grid. Further constraints must be imposed in order for the model to cope with a real case. Real data may not necessarily be transparent in the first step. The evaluation of access patterns could be extended over time which would yield more information. Also, the effects of replication of files and dataset overlaps were not investigated in this work. Equal copies of files should be declustered in the network. This means, different storage nodes with sufficient distance between one another are selected for storing equal copies of files. Furthermore, data sets placed by the user should not be moved to another

ICCS Camera Ready Version 2018
To cite this paper please use the final published version:
DOI: 10.1007/978-3-319-93698-7_12

storage location. This would restrict the total optimization. However, optimization on the enabled subsets can be carried out by the proposed algorithm. Fixed subsets can be flagged accordingly and must not be moved automatically. User interactability can be incorporated into a grid service of the proposed algorithm to enable or disable datasets. Users could identify interesting datasets that would be the focus of computation, or alternatively single out the uninteresting datasets that should no longer be in use. So far, ATLAS users and users of other research collaborations move datasets with rigid policies or by hand on a dataset basis. The implementation of this feature into a service could replace the manual way by an at least semi-automatic mechanism of rebalancing data. *Future research* will focus on:

- The algorithm will be refined in terms of flexibility for use in various environments with even more input parameters. One additional feature will be enabling/disabling datasets, allowing the user to control whether a dataset is able to move or not. This makes the algorithm more efficient and reliable.
- The algorithm is highly parallelizable. Hence, runtime improvements are expected to be achieved in order to apply it to more complex use cases. The current code base represents a first version which proves the concept. It runs on a single standard personal computer.
- Further constraints should be refined in the model. On the network side, I/O bandwidth between data centers can be defined into the cost metric. Various job classes can be considered, an issue that has not yet been covered in this work.

References

1. Abdel-Ghaffar, K.A., El Abbadi, A.: Optimal allocation of two-dimensional data. In: International Conference on Database Theory. pp. 409–418. Springer (1997)
2. Atallah, M.J., Prabhakar, S.: (almost) optimal parallel block access to range queries. In: Proceedings of the nineteenth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems. pp. 205–215. ACM (2000)
3. Bell, D.A.: Difficult data placement problems. The Computer Journal **27**(4), 315–320 (1984)
4. Berchtold, S., Böhm, C., Braunmüller, B., Keim, D.A., Kriegel, H.P.: Fast parallel similarity search in multimedia databases. In: Proceedings of the 1997 ACM SIGMOD International Conference on Management of Data. pp. 1–12. SIGMOD '97, ACM, New York, NY, USA (1997). https://doi.org/10.1145/253260.253263, `http://doi.acm.org/10.1145/253260.253263`
5. Bonacorsi, D., Boccali, T., Giordano, D., Girone, M., Neri, M., Magini, N., Kuznetsov, V., Wildish, T.: Exploiting cms data popularity to model the evolution of data management for run-2 and beyond. In: Journal of Physics: Conference Series. vol. 664, p. 032003. IOP Publishing (2015)
6. Campello, R.J.G.B., Hruschka, E.R.: On comparing two sequences of numbers and its applications to clustering analysis. Information Sciences **179**(8), 1025–1039 (2009)
7. Chang, R.S., Chang, H.P.: A dynamic data replication strategy using access-weights in data grids. The Journal of Supercomputing **45**(3), 277–295 (2008)

ICCS Camera Ready Version 2018
To cite this paper please use the final published version:
DOI: 10.1007/978-3-319-93698-7_12

8. Chu, W.W.: Optimal file allocation in a multiple computer system. IEEE Transactions on Computers **100**(10), 885–889 (1969)
9. Foster, I., Kesselman, C., Tuecke, S.: The anatomy of the grid: Enabling scalable virtual organizations. The International Journal of High Performance Computing Applications **15**(3), 200–222 (2001)
10. Guo, W., Wang, X.: A data placement strategy based on genetic algorithm in cloud computing platform. In: Web Information System and Application Conference (WISA), 2013 10th. pp. 369–372. IEEE (2013)
11. Laning, L.J., Leonard, M.S.: File allocation in a distributed computer communication network. IEEE Transactions on Computers (3), 232–244 (1983)
12. Lassnig, M., Garonne, V., Branco, M., Molfetas, A.: Dynamic and adaptive data-management in atlas. In: Journal of Physics: Conference Series. vol. 219, p. 062054. IOP Publishing (2010)
13. Megino, F.B., Cinquilli, M., Giordano, D., Karavakis, E., Girone, M., Magini, N., Mancinelli, V., Spiga, D.: Implementing data placement strategies for the cms experiment based on a popularity model. In: Journal of Physics: Conference Series. vol. 396, p. 032047. IOP Publishing (2012)
14. Ram, S., Marsten, R.E.: A model for database allocation incorporating a concurrency control mechanism. IEEE Transactions on Knowledge and Data Engineering **3**(3), 389–395 (1991)
15. Sato, H., Matsuoka, S., Endo, T.: 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. pp. 204–211. IEEE Computer Society (2009)
16. Sato, H., Matsuoka, S., Endo, T., Maruyama, N.: Access-pattern and bandwidth aware file replication algorithm in a grid environment. In: Proceedings of the 2008 9th IEEE/ACM international Conference on Grid Computing. pp. 250–257. IEEE Computer Society (2008)
17. Spiga, D., Giordano, D., Barreiro Megino, F.H.: 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)
18. Wang, J.Y., Jea, K.F.: A near-optimal database allocation for reducing the average waiting time in the grid computing environment. Information Sciences **179**(21), 3772–3790 (2009)
19. Yuan, D., Yang, Y., Liu, X., Chen, J.: A data placement strategy in scientific cloud workflows. Future Generation Computer Systems **26**(8), 1200–1214 (2010)

ICCS Camera Ready Version 2018
To cite this paper please use the final published version:
DOI: 10.1007/978-3-319-93698-7_12