Data Allocation with Neural Similarity Estimation for Data-Intensive Computing

Ralf Vamosi[1] and Erich Schikuta[2]

[1]ralf.vamosi@unet.univie.ac.at
[2]erich.schikuta@univie.ac.at

University of Vienna, Faculty of Computer Science, Vienna, Austria

Abstract. Science collaborations such as ATLAS at the high-energy particle accelerator at CERN use a computer grid to run expensive computational tasks on massive, distributed data sets.
Dealing with big data on a grid demands workload management and data allocation to maintain a continuous workflow. Data allocation in a computer grid necessitates some data placement policy that is conditioned on the resources of the system and the usage of data.
In part, automatic and manual data policies shall achieve a short time-to-result. There are efforts to improve data policies. Data placement/allocation is vital to coping with the increasing amount of data processing in different data centers. A data allocation/placement policy decides which locations sub-sets of data are to be placed.
In this paper, a novel approach copes with the bottleneck related to wide-area file transfers between data centers and large distributed data sets with high dimensionality. The model estimates similar data with a neural network on sparse and uncertain observations and then proceeds with the allocation process. The allocation process comprises evolutionary data allocation for finding near-optimal solutions and improves over 5% on network transfers for the given data centers.

Keywords: data allocation, data placement, similarity estimation, parallel computing, wide-area file transfers

1 Introduction

Physics collaborations such as the ATLAS collaboration [10] at CERN store their data as files in a worldwide computing grid. The LHC Computing Grid [3] provides distributed storage and processing for physics data from extensive experiments. Experts from all over the world submit tasks with the data stored as files across many data centers. The data centers can be considered geographically distant sites that spawn a fully connected network, as shown in Figure 1 as an example. Many data-intensive, high-performance applications use large numbers of files that are in some way labeled or categorized. In the case of ATLAS, data sets label a various number of files that jobs can collectively handle. A computational job is a process that reads a data set and eventually outputs another

data set. Data sets can be used none or several times. The population of data sets is changing. Some may be removed after their lifespan, and data sets are created from time to time to label files. Because of this changing population of data sets, they cannot be the basic unit of data placement/allocation, but the constituent files are.

Regardless of the particular content, data sets are managed by users. A user defines a class on its own, using particular data sets. Their work interest and funding correlate with data sets. Data sets in a more or less narrow range will be processed to fulfill their research. Those have some commonalities, so some aspects are equal or similar. A metric based on machine learning will be introduced later to find similarities within data sets.

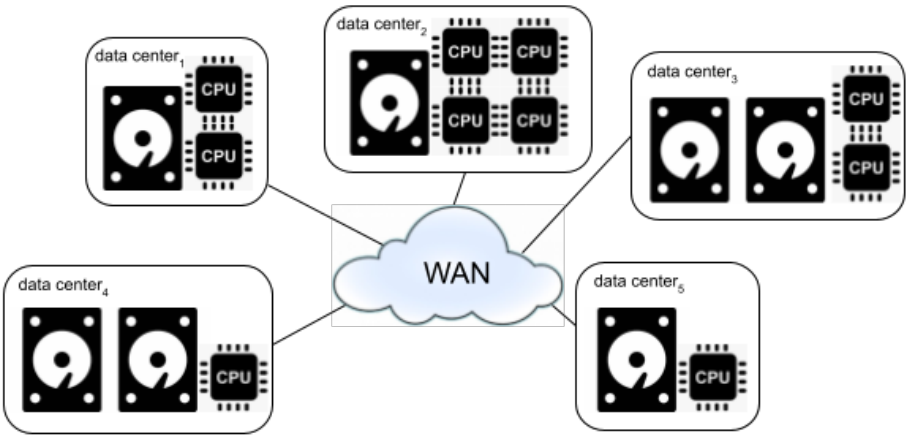

Fig. 1. Example of five data centers inter-connected with wide area network (WAN), over which each data center can read files on an external data center, which incurs costs. Each HPC site facilitates different amount of resources.

In our case, there are two types of jobs to be executed in large quantities:

- Producing final data with parameters and calibration data from the experiment. Calibration data is updated from time to time and therefore production must be repeated, for example.
- Analysis to find events in the productive data. Examples around the well-known Higgs boson H^0 are $t + \bar{t}$ into H^0, or $\gamma + \gamma$ into H^0. An H^0 boson is formed in both processes, but they are different processes with different probability amplitudes.

Jobs for production and analysis are submitted onto data sets. Computational resources are facilitated in the form of job queues at data centers whose task is to run jobs. A job requires the entire data set to be at the local data center. Missing files of the data set must be transferred to the job, that is, the location of the compute job.

The network latency causes a significant delay in the workflow. The *wide area network (WAN)*, the network between data centers, is a scarce resource and represents a bottleneck in a data-intensive workflow [14]. The way to improve the situation related to the WAN is to leave wide-area transfers out, i.e., local processing for each data set. However, jobs cannot be placed arbitrarily to the appropriate data sets: Jobs are allocated to proper sites. First, they must be compatible; the site must support the requested computation. Say, a high-memory worker is needed. Load balancing takes place according to provided computing volumes in data centers. The one with the majority of files of proper sites is preferred as the target. This process continues, and sites are filled with jobs.

On the other hand, data cannot be freely allocated as there are natural storage limitations. Our focus is to tackle data allocation under the following condition:

- Every time a job kicks in, missing input data must be transferred over the bottleneck, the inter-data center network/WAN, from the source to the target site.
- The network consists of non-uniform sites that provide resources with considerable differences. Some possess more storage capacity, and others provide more processing capabilities due to local funding constraints and even support for various user communities.

In order to deal with a large amount of data, clustering is utilized. Clustering is a broad field applied for different tasks such as classification and segmentation, matching commonalities such as identifying normal samples versus outliers.

The *novel contributions* of this paper are:

A *similarity metric*, a distance, is introduced for data sets to be able to put clustering into action and to decide on proper sub-sets.

Combined with this similarity metric, a *loss function* related to the data allocation task is induced. Based on this loss, allocation can be performed regarding spatial patterns of data set use, that is, locations at data centers. Following patterns of data use is necessary for improving data allocation because data sets will be differently used depending on jobs/tasks on distant sites. The jobs will be distributed in some way across data centers. Some of these data centers are capable of more than others. Say, some store more data sets or provision a processing capability for some job class such as large memory. Large memory jobs require larger memory on the CPU.

2 State of the Art

The data allocation problem is synonymously referred to hereto as file allocation. It was investigated when distributed databases were studied, and parallelization had to be utilized.

One of the first data placement/allocation papers was [5]. The work models the task on different abstraction levels. It is further proven that the data placement problem is very difficult to solve and generally NP-Complete.

The file placement problem is investigated under concurrency conditions to build a model with storage cost and communication cost in [17]. Constraints are the multiplicity of databases, variable routing of data, and available capacity of the network nodes.

Some work attempts to cluster data sets according to their inter-dependency [11]. Subsequently to clustering, clusters are stored on separate machines. A different clustering algorithm is described in [25] 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 data set. This is comparable to the ATLAS workflow. This paper utilizes data sets as small clusters and imposes several conditions such as uncertainty and sparsity on the data sets.

Evolutionary algorithms have been applied to almost any kind of problem, and it is no surprise that these were applied to this kind of problem as well. In [12], data allocation strategies have been investigated to reduce transaction costs. A genetic algorithm was used here to limit communication effort between data centers by balancing the load.

Placement of files and replication across different nodes may improve on metrics such as job execution, makespan, and bandwidth [8].

Further efforts have been undertaken in previous studies for database optimizations. The authors of [22] 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 [1,2,6]. 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. 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, for example, as data *popularity* [7]. In [7], a successful popularity model is established which uses historical data. A popularity model is implemented as an autonomous service for finding obsolete data and used in the cleaning process [15,20].

Due to complexity and variety, simplified models and local optimizations were studied and applied. A thorough analysis of data usage and data optimization for data grids is necessary. Storage resources and the use of data has to be appropriately treated in the process of file placement [19,18].

Summing up, the actual research on data partitioning and allocation techniques is specific, and often the models aim at a narrow case. Especially in data-intensive high-performance computing (HPC), large volumes of data are processed, and patterns for data usage have to be observed and taken into account to utilize the performance and improve the workflow fully. At this point, we would like to present our research. In general, data management strongly impacts usability, performance, and cost.

3 Methods

3.1 Background

Data sets collect files in various numbers by giving them a common name (data set name). To certain data sets, jobs will be submitted by a magnitude of 1000 users around the globe. The data set name is a high-dimensional pattern on an abstract level. It consists of a text string in variable lengths, with different sub-strings such as numbers and terms. A data set name (label) looks like

$$mc15.13TeV.362233.Sherpa.CT10.Zee.Pt70$$
$$.140.BFilter.ckkw30.evgen.EVNT.e4558.tid07027172.00$$

There are many identifiers composed into this name to describe the data set. mc15 stands for Monte Carlo; 13 TeV stands for $13 * 10^{12}$ electron volt, which implies the energy of the run; and so on. With this notation, it is challenging to find interrelations between data set names. At least, users manage jobs processing and data sets. The decision is a user-dependent process that was introduced in section 1. User's co-variates are background and funding, and they do usually not select randomly at their whim. Creating data sets and submitting jobs is a confusing process from the outside. However, the common factor is the user between task and data.

Two important points must be addressed in a data allocation policy:

- How likely is it, that data sets will be read by a job of a certain type. It is certain that data sets that are not processed, do not incur any costs from a network's point of view. The opposite case is when a data set is processed at least once or several times.
- What is the impact from the cost perspective, data sets residing on specific nodes.

In the model context, the term node is used for data centers. A node represents a black box, which is not decomposed in its parts, such as network storage, different computing nodes, etc., but rather is described on a top-level with the provision of resources from a job's perspective.

With data sets also comes a lack of information. The use of data sets is uncertain two-folds:

- Data sets change over time. Some of them will overlap.
- Furthermore, there is no information on how likely they will be used for most data sets. Only a tiny minority of data sets have a popularity value that indicates how likely they will be read. There are some investigations on popularity, which are not covered in this work [4].

Our model consists of two sub-models joined together to perform the two steps depicted in Figure 2:

1. Transform the variable space of data sets into a metric space. This model is introduced to generate variables of data sets that allow better handling. The first model in section 3.2 covers this part.

2. Generate disjoint file sets from data sets and allocate them to nodes. This can be pictured in two dimensions as on the right side in Figure 2. The allocation is the process that divides files, comprised in data sets, into the number of nodes and maps them to the nodes, which are data centers in grid. The necessary steps are done in the second model in section 3.3.

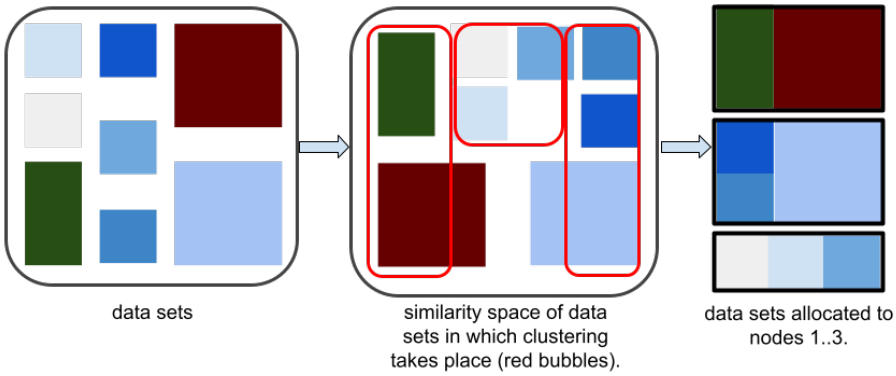

Fig. 2. Two models for two transformations of the data sets. First, the data sets are mapped into a similarity space. Second, the allocation model disjoints the data sets into proper sub-sets to allocate to nodes.

3.2 Neural Similarity Estimator

The relationship between users and data sets, and thereby data sets to others, has been discussed. The similarity model proposed in this work relates data sets with others. The neural similarity model is placed first to extend the variables-space for the allocation.

With the similarity model, given a data set as input, another similar data set may be found in the population of data sets. Even new data sets that have not yet been seen can be assigned to other data sets. This is important because new data sets are constantly being created during operation. On the one hand, similarity plays a role in the allocation in order to select a specific kind of data sets. On the other hand, data sets also depend on each other because data sets can overlap several others. They will be more or less mixed, which makes them not separable.

The virtue of such a similarity model is that it can handle complex and inter-related (text) strings. Triplet learning sometimes referred to as triplet loss or triplet comparison, comes in helpful for this task. This technique makes inferences about observations based on commonalities and non-commonalities within the variables [24]. A specific loss function is applied to learn a similarity or distance metric by recognizing similarities (and dissimilarities). It was successfully

applied for large learning applications in image similarities [9], [23] as well as a variety of digital image processing tasks such as face recognition [16].

In the case of face recognition and person re-identification, similar samples are images of the same person. Figure 3 illustrates the core idea. The so-called anchor a and the positive sample + are from the same user, and the other example - is from another user. The triplet comparison operates on three different example inputs. On average, the a and + are more similar to each other. During network training, the cost function applied to a triplet of samples is gradually decreased. The distance between the anchor and the positive sample becomes smaller, while the distance between the anchor and the negative sample increases.

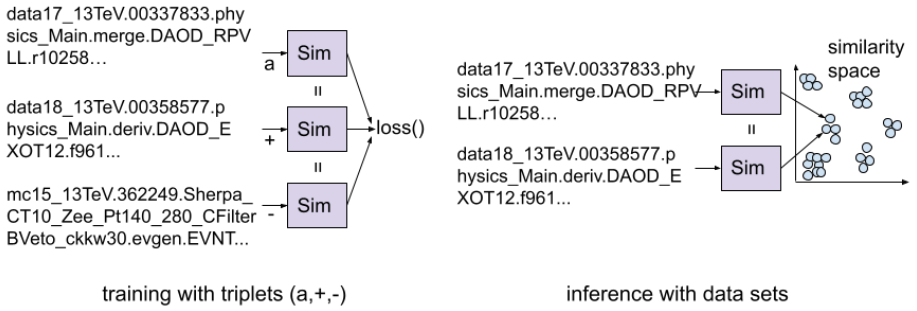

Fig. 3. Training of the model is shown on the left, in which triplets of data sets (a,+,-) are fed from the training set while the model tries to lower the loss(). The inference is shown on the right. Data sets are mapped into the latent similarity space, where next data sets have higher similarity.

In our case, data sets with their names are the input from users. As data set names are depicted in section 3.1, the first step is text pre-processing to obtain a tuple of word vectors that are the representations for the data set names. Terms and numbers occurring only once in the training set of data set names are replaced by a default value since they are not informative. For example, 933 and aabb are replaced as default value.

The model consists of several steps:

1. The data set names are tokenized. This process makes a discrete set with unique numbers from each alpha-numeric subtext. In a standard text case, this would be the word level, and each word would convert into a number in the discrete co-domain.
2. Each token is converted into a word vector.
3. In training, a latent metric space is spawned under the loss function from Equation 1. Data sets are just passed through the trained model in the inference phase. Data sets become labeled with the vector of the low-dimensional neural co-domain of the similarity model 3.

The model's network architecture is strongly inspired by image recognition projects with a triplet loss, like [13], we do not need a particular network type such as CNN for the short input vectors at hand. Even though a CNN would be possible, we go with a fully connected multi-layer feed-forward network: a multi-layer perceptron. As for other recognition tasks, as mentioned, a triplet loss is used in the network:

$$loss = max[d(\ a,\ +) - d(\ a,\ -) + margin, 0] \tag{1}$$

where a triplet is $(a, +, -)$ with some data set, a, a positive example data set from the same user $+$, and a negative example data set from some other user, $-$.

The training phase takes a magnitude of one day on a single GPU. In the prediction phase, the model maps each piece of input data set into its output space, a metric space in which similar inputs are placed closer to each other. This is depicted on the right in Figure 3. Inference on one data set is just a mapping with the fixed weights in the model and takes a magnitude of 1 millisecond.

3.3 Data Allocation Model

For the allocation process, an evolutionary clustering method has been developed. The goal is to find improved data allocations over the course of the run time in terms of a loss according to the following loss function decreases:

$$cost_{tx}(\boldsymbol{S}) = \sum_{j \in \{jobs\}} cost_{tx}(S, j) \tag{2}$$

where $cost_{tx}(S, j)$ denotes an affine function of the number of non-local files for job j given $\boldsymbol{S}$, that is, the number of triggered file transfers over the network (WAN) by j.

The data allocation model generates an allocation matrix for all files, $\boldsymbol{S}$. As explained, network transfers between nodes are expensive, which is why the total number of these transfers is regarded in the total loss denoted as $cost_{tx}(S)$. Data center-internal transfers are much faster and do not need to be considered (zero cost). Given the amount of work, i.e., a set of computational jobs, the problem can be formalized as minimization problem with parameter S specifying the allocation, and a loss function $cost_{tx}(S)$ depending on system variables and the set of jobs:

$$\begin{aligned} &\underset{\boldsymbol{S}}{argmin}\ cost_{tx}(\boldsymbol{S}) \\ &so\ that \\ &\boldsymbol{S}^T \times \mathbf{w}_{file} \leq \mathbf{w}_{store} \end{aligned} \tag{3}$$

where $cost_{tx}(\boldsymbol{S})$ is the loss function introduced before, $\mathbf{w}_{file}$ is a column vector which represents the file sizes for $file_1$,...,$file_M$ and $\mathbf{w}_{store}$ is a column vector

which represents the storage capacities of data centers 1, 2, ..., N. $cost_{tx}(\boldsymbol{S})$ penalizes external heavy transfers. $\mathbf{w}_{store}$ is needed by the model to control the cluster sizes. Each cluster becomes parameterized for the sake of collecting files for sub-sets in the allocation process. No other variables are needed except relative costs, i.e., $cost(\boldsymbol{S}_{new}) - cost(\boldsymbol{S})$ while the model probes the nodes iteratively with variations of data allocations and descent on the plane of the loss function.

The allocation process can be outlined in the following steps:

- Parameterize and prioritize clusters. Set a seed according to popularity and type variable for each cluster.
- Disjoint the data sets into the number of clusters into subsets to be allocated. This produces the allocation matrix S (Equation 3). Boundary conditions $\mathbf{w}_{store}$ are needed to control the cluster sizes.
- Calculate costs by Equation 2 and repeat the process.

The presented model here relies on work management and solves the data allocation problem. The model does not see the actual workload, but the cost that comes with the workload related to the data allocation. The consideration by only costs allows for a more flexible way of not having to expose the underlying grid and jobs.

The evolution of the model comes into the described allocation process. After finding a S which betters the situation, it is placed into an internal pool with the label of the cost that indicates the quality or fitness:

- Populate the pool and delete the bottom solutions
- Vary single solutions
- Combine parameters of two solutions into one novel solution

The model runs on a single PC with parallelization. In-memory data on files and data sets are fixed and can be shared. Each data allocation iteration is done on a worker process, which just communicates the parameters of a solution. Other workers take the work on combining solutions.

4 Justification and Evaluation

Data centers / HPC sites (in some jargon, just sites) are referred to here as nodes. The number of nodes in the optimization has been reduced from over 100 to a low 2-digit number, see Figure 4. Details are listed below. This is a feasible number, and the set can cover the most important ones, say, nodes with the highest capacities. There is an exponential drop-off from the largest to the smallest data centers. At the peak, there are large ones working on the majority of data sets, and at the tail, there are many small ones. Large ones covered in the optimization process imply a high potential for improvement.

For each iteration of the experiment, the following parameters are defined for nodes in such a way that the values are sampled in a uniformly random way:

- the provisioned work performance in terms of average jobs

- the provisioned queues for jobs for two types (A and B)
- the provisioned storage capacity for files

Data is defined in the following way:

- 10 thousand data sets from 100 users are taken. This is used for the validation of the model.
- Each data set possesses 1 to 100 files.
- For training of the similarity model, a subset of the data set user records are taken, a sample size of 2 GB plain text. This record indicates the data set, their files, and the associated user to each data set. This sub-set covers 100 users from the full user records comprising about 25 GB per month. The selected time window of one month is short for picking up short correlations for the demonstration.
- For these data sets, a popularity value is chosen between 0 and 1. Highly active users possess data sets with higher popularity on average, and more frequent data sets are represented with higher popularity. The model can only see 10 percent of the popularity values.

With the data and the provision on the computer side, the work of the model can be outlined:

- Similarity estimation is done once, and the final similarity values are kept in memory beside the original variables.
- The data allocation part produces a data distribution while observing the total costs according to Equation. 2. Over the course of evolution, the allocation model permutates the output to find a fitter solution. In other words, a solution with a smaller cost.

The improvement by the data allocation can be depicted in the Figure 4. The model runs three configurations, and in the smallest one, it performs better. This is probably due to the fact that the solution space is smaller. A smaller cardinality makes the model performs iteration quicker, and more vectors in the solution space can be probed. The improvement is shown to be at least 5.5 percent despite uncertainties in the system that the allocation model cannot control, on the input side, where sparse information comes from the data sets, and second, the work on each node that depends on the jobs and the nodes.

5 Conclusion and Outlook

Six percent improvement in the most important data centers is a perceivable saving on time and cost in terms of money. This amount corresponds to the provisioned traffic of one to two large data centers to get an idea. Provisioned bandwidth of data centers is often 10 Gb/s. Our experiment shows that savings are possible without making a structural change or manual investigation. In order to contribute to a better solution, more detailed exploitation of data centers, e.g., including different types of storage, compute nodes, etc., for example, is

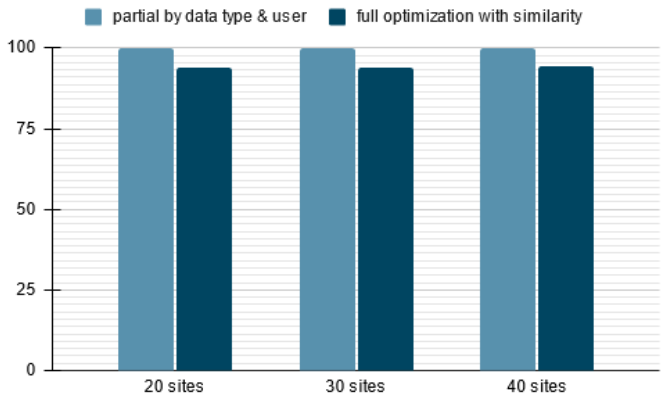

Fig. 4. On the left, data sets are allocated by clustering by type and user. The full optimization, on the right, is done with similarity model and the exploitation of popularity. It saves 6 percents, ranging from almost 6.3 percent for with 20 sites, and 5.5 percent points for 40 sites.

possible in the following way: A second optimization level can be added, and both algorithms will go hand in hand. Our model works on the top level regarding inter-data center traffic. Another model, with the proper cost metric also a variation of this model, will act on the second level, an entity for the internal costs per data center.

The first part of the composed model, the similarity model from section 3.2, can be updated from time to time. This update process would run in the background to update to the next version of this sub-model.

The core model is the allocation model from section 3.3. It can also be set up in such a way that it operates on a subset of the data sets. Say, new data sets shall be placed according to its policy. Free resources must be observed by the model. A flag would be added to distinguish between data sets to be moved and fixed data sets, i.e., data sets allocated/placed recently [21].

The volume of data sets is small compared to the population in the real grid. Anyhow, the outcome of the data allocation can be used in a top-down approach, in which missing data sets of the entire population would be clustered to the initial data sets. So, for example, 1 data set becomes 10, and 10 becomes 100, until all data sets are included. The solution contains clusters with contiguous elements, as with the solution of the '1' data sets with the heuristic variables' similarity and popularity. In other words, the sparse '1' data sets have a high dependency on the nearby data sets of the entire population.

The presented model shows the potential to perform data placement/allocation with high-dimensional data in the background. Further, the loss function allows the model to be more abstract. The model operates solely regarding the loss and only sees the data and the relevant resources; these are the capacities. The complexity of the data processing with all relevant resources hides in the loss.

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. ATLAS, C., Åkesson, T., Eerola, P., Hedberg, V., Jarlskog, G., Lundberg, B., Mjörnmark, U., Smirnova, O., Almehed, S., et al.: Atlas computing: technical design report (2005)
4. Beermann, T., Chuchuk, O., Girolamo, A.D., Grigorieva, M., Klimentov, A., Lassnig, M., Schulz, M.W., Sciabà, A., Tretyakov, E.: Methods of data popularity evaluation in the atlas experiment at the lhc. EPJ Web of Conferences (2021)
5. Bell, D.A.: Difficult data placement problems. The Computer Journal **27**(4), 315–320 (1984)
6. 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
7. 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)
8. 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)
9. Chechik, G., Sharma, V., Shalit, U., Bengio, S.: Large scale online learning of image similarity through ranking. Journal of Machine Learning Research **11**(3) (2010)
10. Collaboration, A., Aad, G., Abat, E., Abdallah, J., Abdelalim, A., Abdesselam, A., Abdinov, O., Abi, B., Abolins, M., Abramowicz, H., et al.: The atlas experiment at the cern large hadron collider (2008)
11. 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)
12. 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)
13. Hoffer, E., Ailon, N.: Deep metric learning using triplet network. In: International workshop on similarity-based pattern recognition. pp. 84–92. Springer (2015)
14. Liu, Y., Liu, Z., Kettimuthu, R., Rao, N., Chen, Z., Foster, I.: Data transfer between scientific facilities–bottleneck analysis, insights and optimizations. In: 2019 19th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID). pp. 122–131. IEEE (2019)
15. 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)
16. Parkhi, O.M., Vedaldi, A., Zisserman, A.: Deep face recognition (2015)
17. 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)

18. 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)
19. 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)
20. 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)
21. Vamosi, R., Lassnig, M., Schikuta, E.: Data allocation service adas for the data rebalancing of atlas. In: EPJ Web of Conferences. vol. 214, p. 06012. EDP Sciences (2019)
22. 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)
23. Wang, J., Song, Y., Leung, T., Rosenberg, C., Wang, J., Philbin, J., Chen, B., Wu, Y.: Learning fine-grained image similarity with deep ranking. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 1386–1393 (2014)
24. Weinberger, K.Q., Sha, F., Saul, L.K.: Convex optimizations for distance metric learning and pattern classification [applications corner]. IEEE Signal Processing Magazine **27**(3), 146–158 (2010)
25. 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)