A New Performance Guarantee Approach for Data Replication in Data Intensive Scientific Applications

Aswadhati Sirisha[#1], M.V.Rajesh[*2]

[#]M.Tech Scholar
Department of CSE ,
Pragati Engineering College, Surampalem,
Kakinada, AP, India.
[1] aswadhati.sirisha@gmail.com

[*] Associate Professor & HOD,
Department of IT,
Pragati Engineering College, Surampalem,
Kakinada, AP, India.

Abstract

Replication in computing involves sharing information so as to ensure consistency between redundant resources, such as software or hardware components, to improve reliability, fault-tolerance, or accessibility. Data replication has been well adopted in data intensive scientific applications to reduce data file transfer time and bandwidth consumption. However, the problem of data replication in Data Grids, an enabling technology for data intensive applications, has proven to be NP-hard and even non approximable, making this problem difficult to solve. In this paper, we propose a data replication algorithm that not only has a provable theoretical performance guarantee, but also can be implemented in a distributed and practical manner. Specifically, we design a polynomial time centralized replication algorithm that reduces the total data file access delay by at least half of that reduced by the optimal replication solution. Based on this centralized algorithm, we also design a distributed caching algorithm, which can be easily adopted in a distributed environment such as Data Grids. Extensive simulations are performed to validate the efficiency of our proposed algorithms.

Keywords: Data intensive applications, Data Grids, data replication, algorithm design and analysis, simulations.

1. Introduction

Replication is an effective mechanism to reduce file transfer time and bandwidth consumption in Data Grids—placing most accessed data at the right locations can greatly improve the performance of data access from a user's perspective. DATA intensive scientific applications, which mainly aim to answer some of the most fundamental questions facing human beings, are becoming increasingly prevalent in a wide range of scientific and engineering research domains. Examples include human genome mapping [2], high energy particle physics and astronomy [1], [3], and climate change modeling [4]. In such applications, large amounts of data sets are generated, accessed, and analyzed by scientists worldwide. The Data Grid [5], [6], [7] is an enabling technology for data intensive applications. It is composed of hundreds of geographically distributed computation, storage, and networking resources to facilitate data sharing and management in data intensive applications. One distinct feature of Data Grids is that they produce and manage very large amount of data sets, in the order of terabytes and petabytes.

Our model is as follows: Scientific data, in the form of data files, are produced and stored in the Grid sites as the result of scientific experiments, simulations, or computations. Each Grid site executes a sequence of scientific jobs submitted by its users. To execute each job, some scientific data as input files are usually needed. If these files are not in the local storage resource of the Grid site, they will be accessed from other sites, and transferred and replicated in the local storage of the site if necessary. Each Grid site can store such data files subject to its storage/memory capacity limitation. We study how to replicate the data files onto Grid sites with limited storage space in order to minimize the overall file access time, for Grid sites to finish executing their jobs.

We formulate this problem as a graph theoretical problem and design a centralized greedy data replication algorithm, which provably gives the total data file access time reduction (compared to no replication) at least half of that obtained from the optimal replication algorithm. We also design a distributed caching technique based on the centralized replication algorithm, and show experimentally that it can be easily adopted in a distributed environment such as Data Grids. The main idea of our distributed algorithm is that when there are multiple replicas of a data file existing in a Data Grid, each Grid site keeps track of (and thus fetches the data file from) its closest replica site. This can dramatically improve Data Grid performance because transferring large-sized files takes tremendous amount of time and bandwidth [8]. The central part of our distributed algorithm is a mechanism for each Grid site to accurately locate and maintain such closest replica site. Our distributed algorithm is also adaptive—each Grid site makes a file caching decision (i.e., replica creation and deletion) by observing the recent data access traffic going through it. Our simulation results show that our caching strategy adapts better to the dynamic change of user access behavior, compared to another existing caching technique in Data Grids [9].

The main results and contributions of our paper are as follows:

1. We identify the limitation of the current research of data replication in Data Grids: they are either theoretical investigation without practical consideration, or heuristics-based implementation without a provable performance guarantee.

2. To the best of our knowledge, we are the first to propose data replication algorithm in Data Grid, which not only has a provable theoretical performance guarantee, but can be implemented in a distributed manner as well.

3. Via simulations, we show that our proposed replication strategies perform comparably with the optimal algorithm and significantly outperform an existing popular replication technique [9].

4. Via simulations, we show that our replication strategy adapts well to the dynamic access pattern change in Data Grids.

2. Background Theory

In this section, we first review related works addressing the attention of new researchers towards the data replication. Replication has been an active research topic for many years in World Wide Web [10], peer-to-peer networks [11], ad hoc and sensor networking [12], [13], and mesh networks [14]. In Data Grids, enormous scientific data and complex scientific applications call for new replication algorithms, which have attracted much research recently.

The most closely related work to ours is by Cibej et al. [15]. The authors study data replication on Data Grids as a static optimization problem. They show that this problem is NP-hard and non approximable, which means that there is no polynomial algorithm that provides an approximation solution if $P \neq NP$. The authors discuss two solutions: Integer programming and simplifications. They only consider static data replication for the purpose of formal analysis. The limitation of the static approach is that the replication cannot adjust to the dynamically changing user access pattern. Furthermore, their centralized integer programming technique cannot be easily implemented in a distributed Data Grid. Moreover, Baev et al. [16] show that if all the data have uniform size, then this problem is indeed approximable. And they find 20.5-approximation and 10-approximation algorithms. However, their approach, which is based on rounding an optimal solution to the linear programming relaxation of the problem, cannot be easily implemented in a distributed way. In this work, we follow the same direction (i.e., uniform data size), but design a polynomial time approximation algorithm, which can also be easily implemented in a distributed environment like Data Grids.

Raicu et al. [17], [18] study both theoretically and empirically the resource allocation in data intensive applications. They propose a "data diffusion" approach that acquires computing and storage resources dynamically, replicates data in response to demand, and schedules computations close to the data. They give a O (N M) competitive ratio online algorithm, where N is the number of stores, each of which can store M objects of uniform size. However, their model does not allow for keeping multiple copies of an object simultaneously in different stores. In our model, we assume each object can have multiple copies, each on a different site.

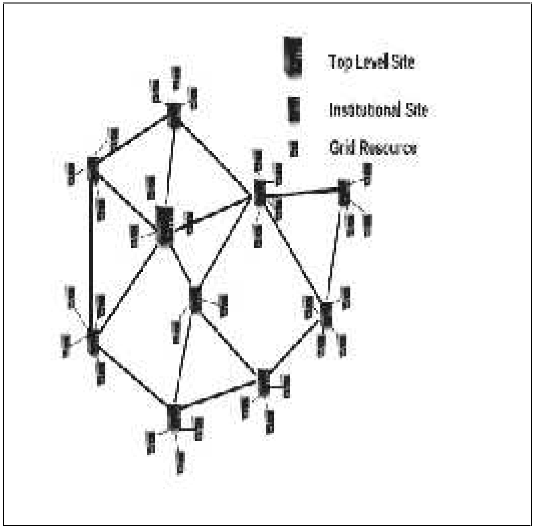


Fig 1. Data Grid Model

As demonstrated by the experiments of Chervenak et al. [19], the time to execute a scientific job is mainly the time it takes to transfer the needed input files from server sites to local sites. Similar to other work in replica management for Data Grids [20], [21], [22], we only consider the file transfer time (access time), not the job execution time in the processor or any other internal storage processing or I/O time. Since the data are read only for many Data Grid applications [23], we do not consider consistency maintenance between the master file and the replica files. For readers who are interested in the consistency maintenance in Data Grids, please refer to [24], [25], [26].

3. Data Grid Model and Problem Formulation

As shown in Fig. 1. A Data Grid consists of a set of sites. There are institutional sites, which correspond to different scientific institutions participating in the scientific project. There is one top level site, which is the centralized management entity in the entire Data Grid environment, and its major role is to manage the Centralized Replica Catalogue (CRC). CRC provides location information about each data file and its replicas, and it is essentially a mapping between each data file and all the institutional sites where the data is replicated. Each site (top level site or institutional site) may contain multiple grid resources. A grid resource could be either a computing resource, which allows users to submit and execute jobs, or a storage resource, which allows users to store data files.3 We assume that each site has both computing and storage capacities, and that within each site, the bandwidth is high enough that the communication delay inside the site is negligible. For the data file replication problem addressed in this article, there are multiple data files, and each data file is produced by its source site (the top level site or the institutional site may act as a source site for more than one data files). Each Grid site has limited storage capacity and can cache/store multiple data files subject to its storage capacity constraint.

Data Grid Model: A Data Grid can be represented as an undirected graph G (V, E), where a set of vertices V = {1,2, . . . n} represents the sites in the Grid, and E is a set of weighted edges in the graph. The edge weight may represent a link metric such as loss rate, distance, delay, or transmission bandwidth. In this paper, the edge weight represents the bandwidth and we assume all edges have the same bandwidth B (in Section 6, we study heterogeneous environment where different edges have different bandwidths). There are p data files D = {$D_{1,}$ $D_{2,}$.................. D_p } in the Data Grid, Dj is originally produced and stored in the source site Sj ε V . Note that a site can be the original source site of multiple data files. The size of data file Dj is sj. Each site i has a storage capacity of mi (for a source site i , mi is the available storage space after storing its original data). We begin this section by considering an illustrative example which serves as the basis of our problem statement and will be used throughout the paper to demonstrate the main features of our system.

Users of the Data Grid submit jobs to their own sites, and the jobs are executed in the FIFO order. Assume that the Grid site i has n_i submitted jobs { t_{i1} , t_{i2}, ...t_{ini} }, and each job ti_k (1 ≤ k ≤ n_i) needs a subset F_{ik} of D as its input files for execution. If we use w_{ij} to denote the number of times that site i needs Dj as an input file.

$$\sum_{i=1}^{n}\sum_{j=1}^{p} w_{ij} \times d_{iS_j} \times s_j/B.$$

4. Centralized Data Replication Algorithm in Data Grids

In this Paper, we have proposed a Centralized data replication algorithm is a greedy algorithm. First, all Grid sites have all empty storage space (except for sites that originally produce and store some files). Then, at each step, it places one data file into the storage space of one site such that the reduction of total access cost in the Data Grid is maximized at that step. The algorithm terminates when all storage space of the sites has been replicated with data files, or the total access cost cannot be reduced further. Below is the algorithm.

```
Algorithm 1. Centralized Data Replication Algorithm
BEGIN
    $M = A_1 = A_2 = \cdots = A_p = \emptyset$ (empty set);
    While (the total access cost can still be reduced by
            replicating data files in Data Grid)
        Among all sites with available storage capacity and
        all data files, let replicating data file $D_i$ on site $m$
        gives the maximum $\tau(G, \{A_1, A_2, \ldots, A_i, \ldots, A_p\})$
        $-\tau(G, \{A_1, A_2, \ldots, A_i \cup \{m\}, \ldots, A_p\})$;
        $A_i = A_i \cup \{m\}$;
    end while;
    RETURN $M = \{A_1, A_2, \ldots, A_p\}$;
END.
```

The total running time of the greedy algorithm of data replication is O (p^2n^3m), where n is the number of sites in the Data Grid, m is the average number of memory pages in a site, and p is the total number of data files. Note that the number of iterations in the above algorithm is bounded by nm, and at each stage, we need to compute at most pn benefit values, where each benefit value computation may take O (pn) time.

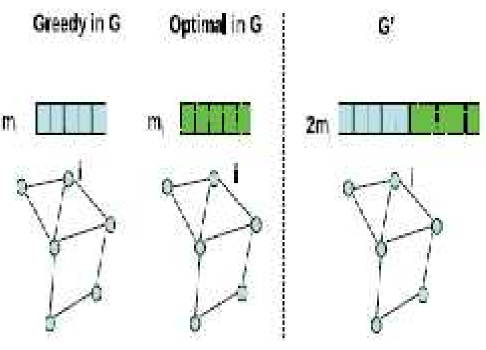


Fig 2. Each site i original graph G has m_i memory space; each site in modified G^1 has $2m_i$ memory space.

5. Distributed Data Caching Algorithm in Data Grids

In this section, we design a localized distributed caching algorithm based on the centralized algorithm. In the distributed algorithm, each Grid site observes the local Data Grid traffic to make an intelligent caching decision. Our distributed caching algorithm is advantageous since it does not need global information such as the network topology of the Grid, and it is more reactive to network states such as the file distribution, user access pattern, and job distribution in the Data Grids. Therefore, our distributed algorithm can adopt well to such dynamic changes in the Data Grids. The distributed algorithm is composed of two important components: nearest replica catalog (NRC) maintained at each site and a localized data caching algorithm running at each site.

Nearest Replica Catalog (NRC). Each site i in the Grid maintains an NRC, and each entry in the NRC is of the form (D_j , N_j) where Nj is the nearest site that has a replica of Dj. When a site executes a job, from its NRC, it determines the nearest replicate site for each of its input data files and goes to it directly to fetch the file (provided the input file is not in its local storage). As the initialization stage, the source sites send messages to the top level site informing it about their original data files. Thus, the centralized replica catalog initially records each data file and its source site. The top level site then broadcasts the replica catalogue to the entire Data Grid. Each Grid site initializes its NRC to the source site of each data file. Note that if i is the source site of Dj or has cached Dj, then Nj is interpreted as the second nearest replica site, i.e., the closest site (other than i itself) that has a copy of Dj. The second nearest replica site information is helpful when site i decides to remove the cached file Dj. If there is a cache miss, the request is redirected to the top level site, which sends the site replica site list for that data file. After receiving such information, the site will update correctly its NRC table and sends the request to the site's nearest cache site for that data file. Therefore, a cache miss takes much longer time. The above information is in addition to any information (such as routing tables) maintained by the underlying routing protocol in the Data Grids.

Localized data caching algorithm. Since each site has limited storage capacity, a good data caching algorithm that runs distributedly on each site is needed. To do this, each site observes the data access traffic locally for a sufficiently long time. The local access traffic observed by site i includes its own local data requests, nonlocal data requests to data files cached at i, and the data request traffic that the site i is forwarding to other sites in the network. Before we present the data caching algorithm, we give the following two definitions:

- **Reduction in access cost of caching a data file**.
  Reduction in access cost as the result of caching a data file at a site is the reduction in access cost given by the following: access frequency in local access traffic observed by the site × distance to the nearest replica site.
- **Increase in access cost of deleting a data file.**
  Increase in access cost as the result of deleting a data file at a site is the increase in access cost given by the following: access frequency in local access traffic observed by the site × distance to the second-nearest replica site.

Cache replacement policy. With the above knowledge, a site always tries to cache data files that can fit in its local storage and that can give the most reduction in access cost. When the local storage capacity of a site is full, the following cache replacement policy is used. Let |D| denote the size of a data file (or a set of data files) D. If the access cost reduction of caching a newly available data file Dj is higher than the access cost increase of some set D of cached data files where |D| > |Dj|, then the set D is replaced by Dj.

6. Conclusion

Through this paper, we study how to replicate data files in data intensive scientific applications, to reduce the file access time with the consideration of limited storage space of Grid sites. Our goal is to effectively reduce the access time of data files needed for job executions at Grid sites. We propose a centralized greedy algorithm with performance guarantee, and show that it performs comparably with the optimal algorithm. We also propose a distributed algorithm where in Grids sites react closely to the Grid status and make intelligent caching decisions. Using GridSim, a distributed Grid simulator, we demonstrate that the distributed replication technique significantly outperforms a popular existing replication technique, and it is more adaptive to the dynamic change of file access patterns in Data Grids.

7. Future Enhancement

In the future, we plan to design and develop data replication strategies in the scientific workflow [27] and large-scale cloud computing environments [28]. We will also pursue how provenance information [29], the derivation history of data files, can be exploited to improve the intelligence of data replication decision making.

8. References

[1] The Large Hadron Collider, http://public.web.cern.ch/Public/ en/LHC/LHC-en.html, 2011.

[2] A. Rodriguez, D. Sulakhe, E. Marland, N. Nefedova, M. Wilde, and N. Maltsev, “Grid Enabled Server for High-Throughput Analysis of Genomes,” Proc. Workshop Case Studies on Grid Applications, 2004.

[3] J.C. Jacob, D.S. Katz, T. Prince, G.B. Berriman, J.C. Good, A.C. Laity, E. Deelman, G. Singh, and M.-H Su, “The Montage Architecture for Grid-Enabled Science Processing of Large, Distributed Datasets,” Proc. Earth Science Technology Conf., 2004.

[4] M. Mineter, C. Jarvis, and S. Dowers, “From Stand-Alone Programs towards Grid-Aware Services and Components: A Case Study in Agricultural Modelling with Interpolated Climate Data,” Environmental Modelling and Software, vol. 18, no. 4, pp. 379-391, 2003.

[5] S. Venugopal, R. Buyya, and K. Ramamohanarao, “A Taxonomy of Data Grids for Distributed Data Sharing, Management, and Processing,” ACM Computing Surveys, vol. 38, no. 1, 2006.

[6] B. Allcock, J. Bester, J. Bresnahan, A.L. Chervenak, C. Kesselman, S. Meder, V. Nefedova, D. Quesnel, S. Tuecke, and I. Foster, “Secure, Efficient Data Transport and Replica Management for High-Performance Data-Intensive Computing,” Proc. IEEE Symp. Mass Storage Systems and Technologies, 2001.

[7] I. Foster, “The Grid: A New Infrastructure for 21st Century Science,” Physics Today, vol. 55, pp. 42-47, 2002.

[8] A. Chervenak, R. Schuler, M. Ripeanu, M.A. Amer, S. Bharathi, I. Foster, and C. Kesselman, “The Globus Replica Location Service: Design and Experience,” IEEE Trans. Parallel and Distributed Systems, vol. 20, no. 9, pp. 1260-1272, Sept. 2009.

[9] K. Ranganathan and I.T. Foster, “Identifying Dynamic Replication Strategies for a High-Performance Data Grid,” Proc. Second Int’l Workshop Grid Computing (GRID), 2001.

[10] L. Qiu, V.N. Padmanabhan, and G.M. Voelker, “On the Placement of Web Server Replicas,” Proc. IEEE INFOCOM, 2001.

[11] A. Aazami, S. Ghandeharizadeh, and T. Helmi, “Near Optimal Number of Replicas for Continuous Media in Ad-Hoc Networks of Wireless Devices,” Proc. Int’l Workshop Multimedia Information Systems, 2004.

[12] C. Intanagonwiwat, R. Govindan, and D. Estrin, “Directed Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks,” Proc. ACM MobiCom, 2000.

[13] B. Tang, S.R. Das, and H. Gupta, “Benefit-Based Data Caching in Ad Hoc Networks,” IEEE Trans. Mobile Computing, vol. 7, no. 3, pp. 289-304, Mar. 2008.

[14] S. Jin and L. Wang, “Content and Service Replication Strategies in Multi-Hop Wireless Mesh Networks,” Proc. ACM Int’l Conf. Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM), 2005.

[15] U. _Cibej, B. Slivnik, and B. Robi_c, “The Complexity of Static Data Replication in Data Grids,” Parallel Computing, vol. 31, nos. 8/9, pp. 900-912, 2005.

[16] I. Baev and R. Rajaraman, "Approximation Algorithms for Data Placement in Arbitrary Networks," Proc. ACM-SIAM Symp. Discrete Algorithms (SODA), 2001.

[17] I. Raicu, I. Foster, Y. Zhao, P. Little, C. Moretti, A. Chaudhary, and D. Thain, "The Quest for Scalable Support of Data Intensive Workloads in Distributed Systems," Proc. ACM Int'l Symp. High Performance Distributed Computing (HPDC), 2009.

[18] I. Raicu, Y. Zhao, I. Foster, and A. Szalay, "Accelerating Large- Scale Data Exploration through Data Diffusion," Proc. Int'l Workshop Data-Aware Distributed Computing (DADC), 2008.

[19] A. Chervenak, R. Schuler, C. Kesselman, S. Koranda, and B. Moe, "Wide Area Data Replication for Scientific Collaboration," Proc. IEEE/ACM Int'l Workshop Grid Computing, 2005.

[20] M. Lei, S.V. Vrbsky, and X. Hong, "An Online Replication Strategy to Increase Availability in Data Grids," Future Generation Computer Systems, vol. 24, pp. 85-98, 2008.

[21] F. Schintke and A. Reinefeld, "Modeling Replica Availability in Large Data Grids," J. Grid Computing, vol. 2, no. 1, pp. 219-227, 2003.

[22] D.G. Cameron, A.P. Millar, C. Nicholson, R. Carvajal-Schiaffino, K. Stockinger, and F. Zini, "Analysis of Scheduling and Replica Optimisation Strategies for Data Grids Using Optorsim," J. Grid Computing, vol. 2, no. 1, pp. 57-69, 2004.

[23] A. Ramakrishnan, G. Singh, H. Zhao, E. Deelman, R. Sakellariou, K. Vahi, K. Blackburn, D. Meyers, and M. Samidi, "Scheduling Data-Intensive Workflows onto Storage-Constrained Distributed Resources," Proc. Seventh IEEE Int'l Symp. Cluster Computing and the Grid (CCGRID), 2007.

[24] D. Du¨ llmann and B. Segal, "Models for Replica Synchronisation and Consistency in a Data Grid," Proc. 10th IEEE Int'l Symp. High Performance Distributed Computing (HPDC), 2001.

[25] H. Stockinger, A. Samar, K. Holtman, B. Allcock, I. Foster, and B. Tierney, "File and Object Replication in Data Grids," Proc. 10th IEEE Int'l Symp. High Performance Distributed Computing (HPDC), 2001.

[26] J. Pe´rez, F. Garcı´a-Carballeira, J. Carretero, A. Caldero´ n, and J. Ferna´ndez, "Branch Replication Scheme: A New Model for Data Replication in Large Scale Data Grids," Future Generation Computer Systems, vol. 26, no. 1, pp. 12-20, 2010.

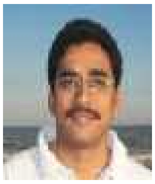
[27] C. Lin, S. Lu, X. Fei, A. Chebotko, D. Pai, Z. Lai, F. Fotouhi, and J. Hua, "A Reference Architecture for Scientific Workflow Management Systems and the View Soa Solution," IEEE Trans. Services Computing, vol. 2, no. 1, pp. 79-92, Jan.-Mar. 2009.

[28] I. Foster, Y. Zhao, I. Raicu, and S. Lu, "Cloud Computing and Grid Computing 360-Degrees Compared," Proc. Grid Computing Environments Workshop, pp. 1-10, 2008.

[29] A. Chebotko, X. Fei, C. Lin, S. Lu, and F. Fotouhi, "Storing and Querying Scientific Workflow Provenance Metadata Using an Rdbms," Proc. IEEE Int'l Conf. e-Science and Grid Computing, 2007.

9. About the Authors

Aswadhati Sirisha is currently pursuing her M.Tech in Computer Science & Engineering at Pragati Engineering College, Surampalem, Kakinada. Her area of interests is Distributed & Parallel Systems and Networks.



M.V.Rajesh is currently working as an HOD in Department of Information at Pragati Engineering College, Surampalem, Kakinada. He completed his M.Tech in Computers Science His research interests include the Cloud Computing and program slicing.