Placement of File Replicas in Data Grid Environments

J.H. Abawajy

Carleton University, School of Computer Science, Ottawa, Ontario, Canada, K1S 5B6
abawjem@scs.carleton.ca

Abstract. In this paper, we address the problem of file replica placement in Data Grids given a certain traffic pattern. We propose a new file replica placement algorithm and compare its performance with a standard replica placement algorithm using simulation. The results show that file replication improve the performance of the data access but the gains depend on several factors including where the file replicas are located, burstness of the request arrival, packet loses and file sizes.

1 Introduction

Grid computing is a type of parallel and distributed system that enables sharing, selection, and aggregation of geographically distributed resources. One class of grid computing and the focus of this paper is Data Grids that provide geographically distributed storage resources to large computational problems that require evaluating and mining large amounts of data [5],[8]. In a multi-user and multi-owned systems such as Grids, the resource managers (RMS) are expected to provide good resource utilization and application response time. There is an added problem of managing several petabytes of data in Data Grid environments with high availability and access optimization being some of the key challenges to be supported. This is because, in addition to the high latency of wide-area network, Data Grid environments are prone to node and link failures [2], [5].

One way of coping with some of these problems is to distribute multiple copies of a file across different sites in the system. It has been shown that file replication can improve the performance of the applications [8], [6], [5], [12]. There is a fair amount of work on file replication in Grid environments. However, most of the work done so far relates to creating the underlying infrastructure for replication and mechanisms for creating/deleting replicas [12]. We believe that, in order to obtain the maximum possible gains from file replication, a strategic placement of the file replicas in the system is the key. To this end, we propose a replica placement service called Proportional Share Replication (PSR) algorithm. PSR places file replicas in sites such that failures are masked and network load is minimized. Moreover, it distributes the load of data requests as evenly as possible over all the replicas.

The rest of the paper is organized as follows. Section 4 presents the proposed replica placement algorithm. This section also discusses a baseline policy used to compare the performances of PSR. Section 5 discusses the performance evaluation framework and the results of the proposed policy. Conclusion and future directions are presented in Section 6.

2 System Model

In this paper, we use a hierarchical Data Grid model (see Figure 1), which is one of the most common architectures in current use [11], [5], [6] [2]. The sites are classified into four regions: local sites, regional sites, national sites and international sites [5]. The connectivity bandwidth and delays between different levels of the sites is shown on Figure 1. Since the the load on the network is shared, the bandwidth can vary unpredictably in practice. Each Data Grid user is assigned a weight relative to the percentage of overall traffic in the system. Also, each site has an associated weight that represents the rate of data request expected to traverse it. Each higher level node is assigned a weight representing the sum of the traffic from its children, and the edge distance represents the one-way link transfer delay. The cumulative cost from a child node to a parent node is the sum of the edge distances.

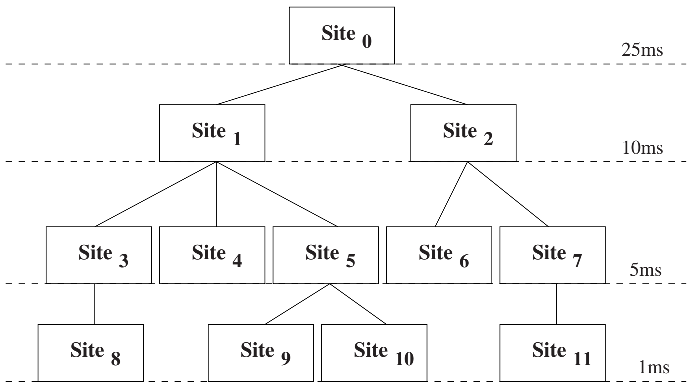

Fig. 1. Hierarchical Data Grid system model

We assume that once created, data is mostly read-only [5]. Also, as in [6] and [2], we assume that initially, only one copy (i.e., the master copy) of the files exists at the root site ($Site_0$) while all requests for file are generated at the leaves. Hence, in the hierarchical Data Grid system model, request for files travels upward from the leave node and any node along the path of the request in the hierarchy that has the requested file services the request. In other word, when a node v on level k requires a file, it sends a request to its parent node u at level $k+1$. Upon receiving the request, if u does not have the requested file, it forewords the request to its parent in turn. This process recursively continues up the hierarchy until a node that has the requested file is encountered or the request reaches the root node.

In the absence of file replicas, the root node services all data requests. Note that when there are more than one replicas of a file in the system, depending where the replica is located in the hierarchy, a site may be blocked from servicing requests for file. For example, if there are file replicas in $Site_0$, $Site_1$ and $Site_2$, the file replica at $Site_0$ cannot be used as all requests for file can be serviced by $Site_1$ and $Site_0$, respectively. Therefore, it is important to place file replicas at appropriate locations in the hierarchical Data Grid systems to reap the benefits of replication.

3 Problem Statement

The goal of replication is to allow researchers to use computing power and data storage with a reasonable response time rather than access all the data at the root. Therefore, given the network topology and data request patterns shown in Figure 1, the goal is to determine an optimal placement of a set of f file replicas on the n sites such that data access delays are reduced, data access availability is improved and the load of data request within the system both on the network and site level are balanced.

In this paper, we assume that both f and n are known. Note that there are many ways we can determine the number of replicas (i.e., f). For example, we can determined it based on the popularity of the file as in [9]. It has been noted that the access pattern in Data Grid environments shows considerable repetitiveness and locality [3], [5]. Therefore, we can use the intensity of access patterns from a given locations. Alternatively, the number of replicas can be based on a cost model that evaluates data access costs and performance gains of creating each replica. The estimation of costs and gains is based on factors such as run-time accumulated read/write statistics, response time, and bandwidth.

Broadly speaking, file replication makes identical files available at multiple locations within the Data Grid environments. The overall file replication problem includes making the following decisions: (1) when file replicas are created and deleted; (2) how many file replicas are created; (3) where these replicas should be placed in the system; and (4) how data consistency is enforced. Data consistency problem dictates that multiple copies of file must appear as a single logical data object. This problem has been address in [13] and [3]. As most of data in Data Grids are read-only [3],[5], data consistency is not a serious problem. An economic model for replica creation and deletion is discussed in [2]. Also, a scheme that is based on a cost model that evaluates data access costs and performance gains of creating each replica is discussed in [12] whereas replication of a file is done based on the popularity of a file in [9] and [14].

Although, data replication is one of the major optimization techniques for promoting high data availability, low bandwidth consumption, increased fault tolerance, and improved scalability, the problem of file replica placement as defined in this section has not been well studied for large-scale grid environments.

The common thread among existing work is that they focus on which replica should be used and not where should the file replicas are placed. Our work differs from these studies in that we are interested in determining the best location for the file replica. Therefore, the work described in this paper will help build on a high level tool above the replica management software. This tool will make replica placement decisions to increase the overall efficiency in terms of availability, response time and fault-tolerance of the Data Grid.

4 Replica Placement Service

Replica placement service is one components of the data grid architecture that decides where in the system a file replicas should be placed. In this section, we discuss a new and a baseline file replication strategies.

4.1 Proportional Share Replica Location Policy

The proportional share replica (PSR) policy works as shown in Figure 2. The main idea underlying the PSR policy is that each file replica should service approximately equal number of request rates in the system. The objective is to place the replicas on a set of sites systematically selected such that file access parallelism is increased while the access costs are decreased. For the Data Grid architecture shown in Figure 1, PSR selects $Site_1$ as the location of the new replica when the number of replica is one. However, when the number of replicas is set to two, the algorithm selects $Site_5$ and $Site_2$ as optimal location for the two replicas.

Note that collectively the sites will be able to offload about $\frac{1}{\text{root copy}+f}$ data requests from the root node. Also note that none of the nodes hinder any other node where a replica of the file is located from receiving its share of file requests. Note also that a given request for file reaches the root node only if there is no node along the path traveled by the request.

4.2 Affinity Replica Location Policy

The affinity replica placement algorithm replicates data on or near the client machines where the file is accessed most. In other words, a copy of the file will be placed near the client hat generates access traffic the most. In Figure 1, for example, when the number of replica is one, the algorithm selects $Site_6$ as the placement of the new replica. Most of existing replica management systems employs a variation of this type of replication scheme [1], [2], [12]. This algorithm is similar to the cascading replica placement algorithm discussed in [6].

5 Performance Analysis

In this section, we compare the proportional and affinity file replica placement schemes discussed in previous section on simulated Data Grid environments. We

```
1. Select a set of $V$ sites, $V \in \{Site_1, \ldots, Site_n\}$ such that each site in $V$ meets the
   following conditions:
   a) does not have $f$ already.
   b) does not block any site where a replica of the file exits from servicing its share
      of file requests.
2. Compute an ideal load distribution over the replicas as follows:
      $load = \frac{1}{\text{root copy} + f} \times R$    (1)
   where R is the total number of data access requests in the system.
3. Place replicas as follows:
   REPEAT
   a) Select a site $v \in V$ that is able to service a set of requests slightly greater than
      or equal to the ideal load.
   b) Place a replica of the file in $v$.
   UNTIL all replicas are placed.
```

Fig. 2. Pseudo-code of the proportional policy

used ns2 network simulator but modified it such that we capture grid specific components in a similar manner to [12], [1], [2]. We also added some extra nodes whose sole purpose is to generate background traffic at a specific rate such that we model a shared interconnection network, as is the real grid environment. In all experiments reported in this paper, we run a total of 30 simulations. As in [5], [12], [1], [2], we used the mean response time as a chief performance metric.

5.1 Experimental Setup

The grid topology, number of sites, connectivity bandwidth and the number of users is as shown in Figure 1. Each request is a 4-tuple field with the arrival time, a source node, a server node and file size. The request arrival process is modeled as exponential distribution with mean of 30 requests per time period. The file size is drawn from a hybrid distribution, with a log-normal body, and a Pareto tail. The median transfer size is 5MB. The largest transfer is 53MB, while the smallest is 3MB bytes. The source and server nodes are generated at random based on the desired request rate for each user. The default server for all requests is the root node.

5.2 Results and Discussions

Figure 3 shows the performance of the proportional and affinity replication policies as a function of the number of replicas. We observe that replication do improve the mean response time as shown by both policies. The main observation is that the proportional algorithm performs better than the affinity algorithm.

In proportional policy, the tendency is to choose sites that serve multiple clients and offload traffic from the busiest network links whereas this is not the case in affinity algorithm. Also the secondary influence in proportional algorithm appears to favor the placement of file replicas in sub-trees that reflect the highest traffic generating clients. This observation is not surprising considering that this sub-tree generates most of the load on upstream links. The Affinity policy cannot process requests as quickly as it is receiving them. When the node is unable to handle the load of requests it is receiving, the size of its message queue begins to increase. Therefore, balancing the load among the replicas is important as it is done in proportional policy.

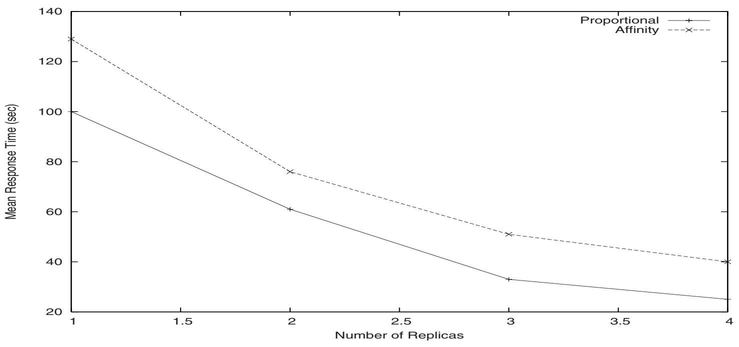

Fig. 3. Performance of the replication policies

It is important to investigate the sensitivity of the replication policy to the network congestion and packet loss factors. This can be modeled by varying the request arrival rate while keeping the file size to 20MB. Figure 4 shows the mean response time (y-axis) of the three policies as a function of the request arrival rate (x-axis). Note that 120 request per second produces an average of 80% utilization on the busiest link in the network. At this level, we observed that about 0.2% of packets were lost when all the requests are serviced from the master copy (i.e., NoReplica case), which required retransmission of the lost packets. This partially explains the performance of the NoReplica case. In contrast, the packet loses in Affinity is 0.064819% while that of Proportional is 0.03511% which is far less than both cases. The conclusion is that network related factors such as the number of packet losses have significant impact on the performance of the replica locations.

6 Conclusions and Future Direction

Many scientific and engineering applications perform large amount of data analysis on increasingly large datasets. There is a need for efficient tools and middleware infrastructures that allow secure access to massive amounts of data, to

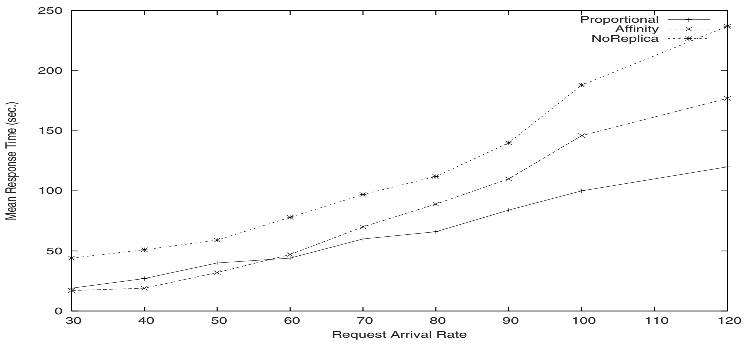

Fig. 4. Sensitivity of the replication policies to arrival rate

move and replicate data and to manage coherence of data in Data Grid environments [8]. In this paper, we addressed one of these data management problems, which is the file replication problem while focusing on the problems of strategic placement of the replicas with the objectives of increased availability of the data and improved response time while distributing load equally. We proposed a new replica placement algorithm that distributes the request to all replicas in such a way that data request processing parallelism is obtained. To the our knowledge, no replication approach balances the load of data requests within the system both on the network and host levels as well as to improve reliability. Our results show that:

1. Replication improves the data access performance of the overall system as measured by the response time. This observation concurs with results in the literature.
2. Many factors including network related issues such as the rate of packet losses affect the performance of replication approaches.
3. The placement of replicas in the system matters and it depends on several factors including the rate of data request arrival and the size of files.
4. Balancing the data request among the replicas is important in improving the performance.

The replica selection by individual users may dependent on a number of factors, including the security policies of the sites which contains these replicas, queuing and network delays, ease of access and so forth. These issues will be addressed in future. The simplifying read only assumption makes it possible to initially address the following question: What are efficient algorithms for creating and disseminating replicas in a Data Grid. We are currently working with both read and write aspects of the file replication strategy discussed.

References

1. David, W. B., Cameron, D. G. , Capozza, L., Millar, A. P., Stocklinger, K., Zini, F.: Simulation of Dynamic Grid Replication Strategies in Optorsim. In Proceedings of 3rd Int'l IEEE Workshop on Grid Computing (2002) 46-57
2. David, W. B.: Evaluation of an Economy-Based File Replication Strategy for a Data Grid, International Workshop on Agent based Cluster and Grid Computing (2003) 120-126
3. Stockinger, H., Samar, A., Allcock, B., Foster, I., Holtman, K., Tierney, B.: File and Object Replication in Data Grids. In 10th IEEE Symposium on High Performance and Dis-tributed Computing (2001) 305-314
4. Li, B., Colin, M., Deng, X., Sohraby, K.: The Optimal Placement of Web Proxies in the Internet. In Proceedings of IEEE INFOCOM (1999) 1282-1290
5. Hoschek, W., Janez, F. J., Samar, A., Stockinger, H., Stockinger, K.: Data Management in an International Data Grid Project. In Proceedings of GRID Workshop (2000) 77-90
6. Ranganathana, K., Foster, I.: Identifying Dynamic Replication Strategies for a High Performance Data Grid. In Proceedings of the International Grid Computing Workshop. (2001) 75-86
7. EU Data Grid Project. http://www.eu-datagrid.org.
8. Chervenak, A., Foster, I. , Kesselman, C., Salisbury, C., Tuecke, S.: The Data Grid: Towards an architecture for the distributed management and analysis of large scientific datasets. Journal of Network and Computer Applications (2000) 187-200
9. Allocck, W., Foster, I., Nefedova, V., Chervnak, A., Deelman, E., Kesselman, C. Lee, J., Sim, A., Shoshani, A., Tierney, B., Drach, B., Williams, D:. High-Performance Remote Access to Climate Simulation Data: A Challenge Problem for Data Grid Technologies (2001) 46-46
10. Particle Physics Data Grid (PPDG); http://www.ppdg.net.
11. Grid Physics Network (GriphyN); http://www.griphyn.org.
12. Lamehamedi, H., Szymanski, B., Shentu, Z., Deelman, E.: Data Replication Strategies in Grid Environments. In Proceedings of 5th International Conference on Algorithms and Architecture for Parallel Processing (2002) 378-383
13. Deris, M.M., Abawajy J.H., Suzuri, H.M.: An Efficient Replicated Data Access Approach for Large-Scale Distributed Systems. To appear in IEEE International Sympo-sium on Cluster Computing and the Grid (2004), April 19-22, Chicago, Illinois, U.S.A.
14. Abawajy J.H.: An Integrated Resource Scheduling Approach on Cluster Computing Systems. In the Proceedings of the IEEE International Parallel and Distributed Processing Symposium (2003) 251-256
15. Ranganathan, K., Iamnitchi, A., Foste, I.T.: Improving Data Availability through Dynamic Model-Driven Replication in Large Peer-to-Peer Communities. In 2nd IEEE/ACM International Symposium on Cluster Computing and the Grid (2002). 376-381
16. Holtman, K.: CMS Data Grid System Overview and Requirements. The Compact Muon Solenoid (CMS) Experiment Note 2001/037, CERN, Switzerland (2001)