ISSN: 2321-7782 (Online)
Volume 3, Issue 4, April 2015
International Journal of Advance Research in Computer Science and Management Studies
Research Article / Survey Paper / Case Study
Available online at: www.ijarcsms.com

Full and Partial Replication in Data Grid Using Gridsim

Rajshri S. Patil[1]
Master of Engineering Scholar
Information Technology Department
Sipna College of Engg and Technology
Amravati, India

L.K.Gautam[2]
Professor
Guide
Amravati, India

Abstract: ***In this paper in a data grid large quantities of data files are produced and data replication is applied to reduce data access time. The file replication is an effective functionality in Data Grid that not only minimizes total access time by replicating most accessed data file at appropriate location but also improve data files availability in a grid environments. The agent is deployed at each site holding the master copies of the shared data files. we design a centralized replication algorithm that reduces the total data file access delay by at least half of that reduced by the optimal replication solution. The simulation results show that the aggregated data transfer time and the execution time for jobs at various resources is less for agent based replica placement.***

Keywords- Data Grid, Data Replication, Replication Strategies, Candidate Site, Simulation.

I. INTRODUCTION

Data Grid is composed of set of sites and each site contains multiple computing, storage and networking resources. All sites are geographically connected to manage and store large data files of size Gigabytes and terabytes throughout the world. There are two aspects to a grid: sharing of data and sharing of resources. Data Grid provides an important service of data and/or data file replication in multiple locations, so that, it helps user not only to speed up data file access but also increases data file availability. A community of researchers distributed worldwide can access and share these replicated data files. In Data Grid, each data files are initially produced and stored in Grid sites. A Grid site may contain multiple data files and will be replicated in appropriate location in Data Grid to reduce access cost.

In data grid environment a large number of similar or equivalent resources that the grid users can select and uses them for the execution of their applications. Examples include human genome mapping , high energy particle physics and astronomy, and climate change modelling[3][5]. An important technique that speeds up the data access in data grid systems is replication of data in multiple locations. A 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 viewpoint.

Many time Replication is confused with caching as they have multiple copies of file, and they have some differences. Replication is a server side occurrence whereas caching is related with a client. A server decides when and where to replicate files. A client request for a file and stores a copy of the file locally for use. Any other nearest client can also request for that cached copy. The other advantages of replication are that it helps in load balancing and improve reliability by creating multiple copies of the same data. Static replication can be used to achieve some of the above-mentioned gains but the drawback with static replication is that it cannot adapt to changes in user performance. The replicas have to be manually created and managed if one were to use static replication.

Replication is that, it can enhance data availability and network performance. The replication of files in Data Grid follows the full or partial replication strategy. In full replication all files are replicated to all resources where as in partial replication

files are replicated to some resources in the Data Grid. There are two replication schemes depending on the use access pattern: 1. Static Replication: in which replicas are kept until it is deleted. 2. Dynamic Replication: in which replicas are created and destroyed or replaced according to variation access of the pattern or environment behaviour. In data replication there are three issues: 1. Replica Management- create, delete, move & modify replica. 2. Replica Selection-selecting appropriate replica across grid. 3. Replica Location-selecting physical locations of several replicas of desired data.

As the data is large, the cost of maintaining local copies of data at each site is also expensive. Although a substantial amount of work has been done on data replication in grid environments, most of it has focused on the mechanism for creating/deleting replicas. In this approach, an agent based replica placement algorithm for making a decision to select a candidate site for replica placement. The agent is autonomous, self-contained software capable of making independent decisions. In replica placement strategy considers two important issues. First issue in choosing a replica location is to place a replica at sites that optimize the aggregated response time. This issue can be addressed by placing replica in a proper location so that the time taken for obtaining all the files required by the job is minimized. Second issue in choosing a replica location is to place a replica at sites that optimize the total execution time of the jobs executed in the grid. Response time is calculated by multiplying the number of requests at site with the transmission time between the nearest replication site to the requester. The sum of the response times for all sites constitutes the aggregated response time.[3]

In this model 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 capacity limitation. 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. Specifically the formulation of this problem is as design a centralized greedy data replication algorithm, which provably gives the total data file access time reduction at least half of that obtained from the optimal replication algorithm.

II. Related Work Literature Survey

Kavitha and Foster [5] discuss various replication strategies for hierarchical data grid architecture. They proposed six different replication strategies such as no replication, best client, cascading, plain catching, caching plus cascading and fast spread for three different kinds of access patterns.

1)No Replication-The base case against which we compare the various strategies is when no replication takes place. The entire data set is available at the root of the hierarchy when the simulation starts. We then run the set of access patterns and calculate the average response time and bandwidth consumed when there is no replication involved. This gives us the base performance and any strategy that performs worse than this is not worth considering.

2)Best client- Each node maintains a detailed history for each file that it contains [5] indicating the number of requests for that file and the nodes that each request came from. The replication strategy then works as follows: At a given time interval each node checks to see if the number of requests for any of its file has exceeded a threshold. If so, the best client for that file is identified. The best client is the one that has generated the most number of requests for that file.

3)Cascading- The advantage of this strategy is that storage space at all tiers is used. Another advantage is that if the access patterns do not exhibit a high degree of temporal locality, geographical locality is exploited by this strategy. By not replicating at the very source of requests but at a higher level the data is brought closer to other nodes in the same sub-tree[5].

4)Plain catching-This combines strategy three and four The client caches files locally. The server periodically identifies the popular files and propagates them down the hierarchy. Note that the clients are always located at the leaves of the tree but any node in the hierarchy can be a server. Specifically, a Client can act as a Server to its siblings[5].

5)Caching plus cascading- This combines strategy three and four The client caches files locally. The server periodically identifies the popular files and propagates them down the hierarchy. Note that the clients are always located at the leaves of the tree but any node in the hierarchy can be a server. Specifically, a Client can act as a Server to its siblings[5].

6)Fast spread- The replacement strategy we employed takes care of both these aspects and is a combination of least popular and the age of the file. If more than one file are equally unpopular, the oldest file is deleted. One detail to be noted here is that the popularity logs for all the files are cleared periodically. Thus the dynamic aspect of changing user patterns is captured[5].

1.1Agent Based Replica Placement in a Data Grid Environment [3].

The author proposed an agent based replica placement algorithm for making a replica decision to select „candidate site" for replica placement to reduce access cost, network traffic, and aggregated response time for the applications. To select a candidate site for a replica, an agent is deployed at each site that holds master copies of the files for which the replicas are to be created. The agent in this approach is autonomous, self-contained software capable of making independent decisions. Replica placement strategy considers two issues in choosing replica location: (1) placing a replica at proper site so that times taken for obtaining all files required by jobs are minimized. (2) Place a replica at sites that optimizes total execution time of the jobs executed in Data Grid. The author extended the GridSim toolkit [9] for decision making process for selection of candidate site by implementing Replica Catalogue and Replica Manager to maintain and control all replicas.

A software agent can be defined as a software entity which functions continuously and autonomously in a particular environment and which is able to carry out activities in a flexible and intelligent manner that is responsible for changes in the environment. The main objective of an agent is to select a candidate site for the placement of a replica that reduces the access cost, network traffic and aggregated response time for the applications. To select a candidate site for a replica, an agent is deployed at each site that holds the master copies of the files for which replicas are to be created.

An agent at each site makes the decision based on resource factors that influence the data transmission time between the sites. The factors include baud-rate between the sites, CPU Rating, CPU Load, Site Storage Capacity and Local Demand of the replicas at each site. The agent uses a Multi-Dimensional Ranking (MDR) function to evaluate the resource properties and grade with an appropriate rank. The agent preferences are represented by a set of factor weightings, which allow resource rank to be tailored to the current resource characteristics. In this paper we conducted the simulation for EU- Data Grid Tesebed1 in which master files are initially place data CERN storage.

a) **Baud-rate :** This parameter represents the configuration of the resource communication channel in the grid network. It influences the transmission time for the file from the replica location to the requesting site.

b) **Load :** This parameter represents the peak load of the CPU in the resource. The CPU Load is generally measured in terms of number of processes waiting in the queue. The CPU load varies at different intervals of time. A highly loaded resource responds slowly compared to the lightly loaded resource.

c) **Local Demand :** This parameter represents the list of files needed for the jobs execution. When all the required files are available on the local storage, the job is scheduled for execution. It is recommended to place a replica at a site that has greater number of access to the replicated files.

d) **CPU Rating :** The processing capability of a resource's CPU is modeled in terms of MIPS (Million Instructions per Second). Higher CPU rating substantially reduces the average execution time of the submitted jobs.

e) **Storage capacity :** An individual storage is responsible for storing, retrieving and deleting files. For data intensive applications, larger storage resources are preferred as they involve large volumes of data sets.

1.2 Data Replication in Data Intensive Scientific Application with Performance Guarantee [1].

This paper deals with scientific data in the form of data files are produced, stored and replicated if necessary. The author proposed a centralized data replication algorithm (Greedy), it places one data file into the storage space of one site and algorithm terminates when all storage space of sites has been replicated with data files to minimize total access cost in the Data Grid. This algorithm that not only has a provable theoretical performance guarantee, but can be implemented in distributed and practical manner Specifically, the author designed a polynomial time centralized replication algorithm that reduces total access cost by at least half of reduced by the optimal replication solution. Based on this centralized algorithm a localized distributed data caching algorithm is designed to make intelligent caching decisions. It is composed of Centralized Replica Catalogue (CRC): maintained at top level sites, which is essentially a list of replica sites list for each data file. Nearest Replica Catalogue (NRC): maintained at each sites which contains information of replica copy and nearest sites, and any changes made to NRC will be updated in CRC by sending message to top level site. Simulation results shows centralized greedy algorithm performs quite close to optimal algorithm

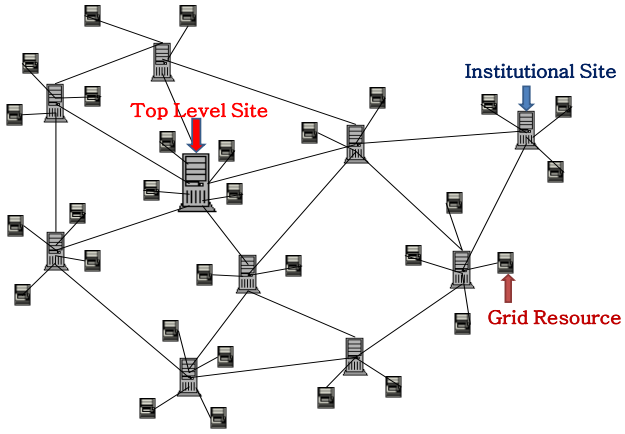


Figure:1.1 shows Top Level site, Institutional site, Grid Resource in Data Grid Model in Data Replication Environment.

We consider a Data Grid model as shown in Fig. 1.1. A Data Grid consists of a set of sites. There are institutional sites,

Fig. 1.1. Data Grid model. 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 [8]. 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.[1]

III. PERFORMANCE EVALUATION

This sections includes, performance analysis of centralized replication algorithm and Agent Based algorithm. The simulation was performed for small scale environment, random file size and large scale environment with various components of Grid such as sites, files, storage capacity, file size and network bandwidth. In this section, we demonstrate the performance of our designed algorithms via extensive simulations. First, we compare the centralized greedy algorithm (referred to as Greedy) with agent based algorithm. Then, using GridSim [5], a well-known Grid simulator, we evaluate our distributed algorithm.

A. Comparison of Centralized Algorithms

In comparisons of centralized algorithms, we assume that all file sizes are equal and all edge bandwidths are equal.Since the total data file access cost between sites is defined as the total transmission time spent to obtain the data file. For a Data Grid with *n* sites and *p* data files of unit size, and each site has storage capacity *m*.

Varying Number of Data Files.

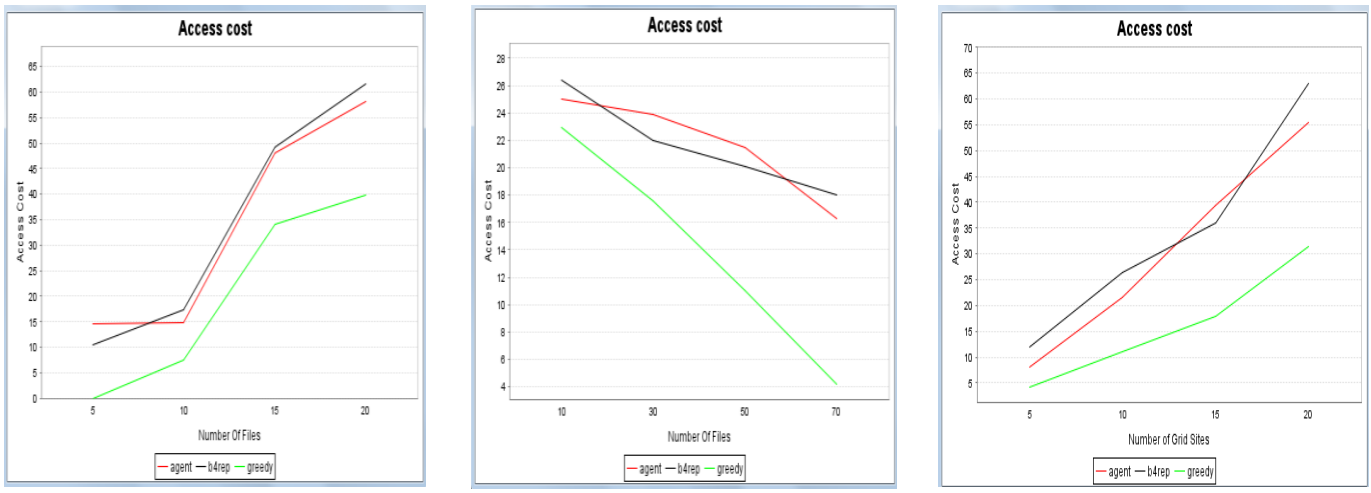
First, we vary the number of data files in the Data Grid while fixing the number of sites and the storage capacity of each site. is shown in Figure 1 (a). In terms of the total access cost, we observe that Due to the high time complexity of the optimal algorithm, we consider a small Data Grid with $n = 10$ and $m = 50$, and we vary p to be 5, 10, 15, and 20. The comparison of the Centralized, Agent and before replication algorithms Centralized < Agent < before replication. That is, among all the algorithms,Centralized performs the best since it exhausts all the replication possibilities. Furthermore, we observe that Agent performs better, which demonstrates that it is indeed a good replication algorithm.

Varying Storage Capacity.

Figure 1 (b) shows the results of varying the storage capacity of each site while keeping the number of data files and number of sites unchanged. We choose $n = 10$ and $p = 10$, and vary m from 10, 30, 50, 70, to 90 mb. It is obvious that with the increase of memory capacity of each Grid site, the total access cost decreases since more file replicas are able to be placed into the Data Grid. Again, we observe that Centralized performs comparably to Agent.

Varying Number of Grid Sites.

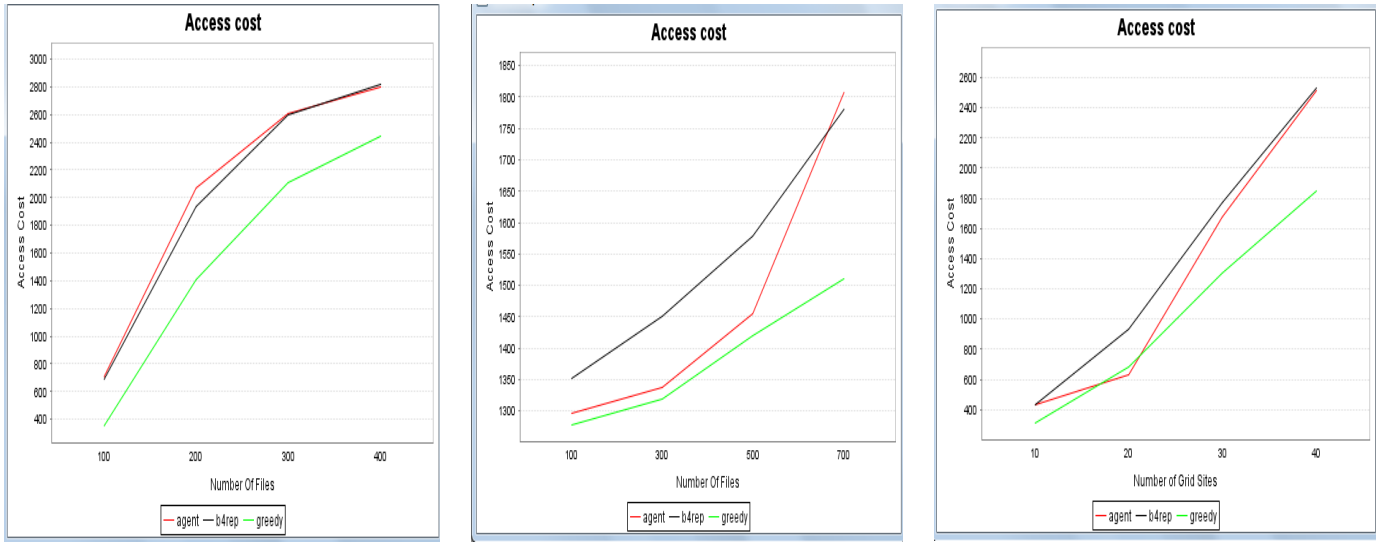
Figure 1 (c) shows the result of varying the number of sites in Data Grid. We consider a small Data Grid, choose $p = 10$ and $m = 50$, and vary n as 5,10, 15, 20, or 25. The Centralized algorithm performs only slightly better than the Agent algorithm.



Number of Files ***Storage Capacity(MB)*** ***Number of Grid sites***

(a)Varying number of data files ***(b)Varying storage capacity*** ***(c)Varying number of Grid sites***

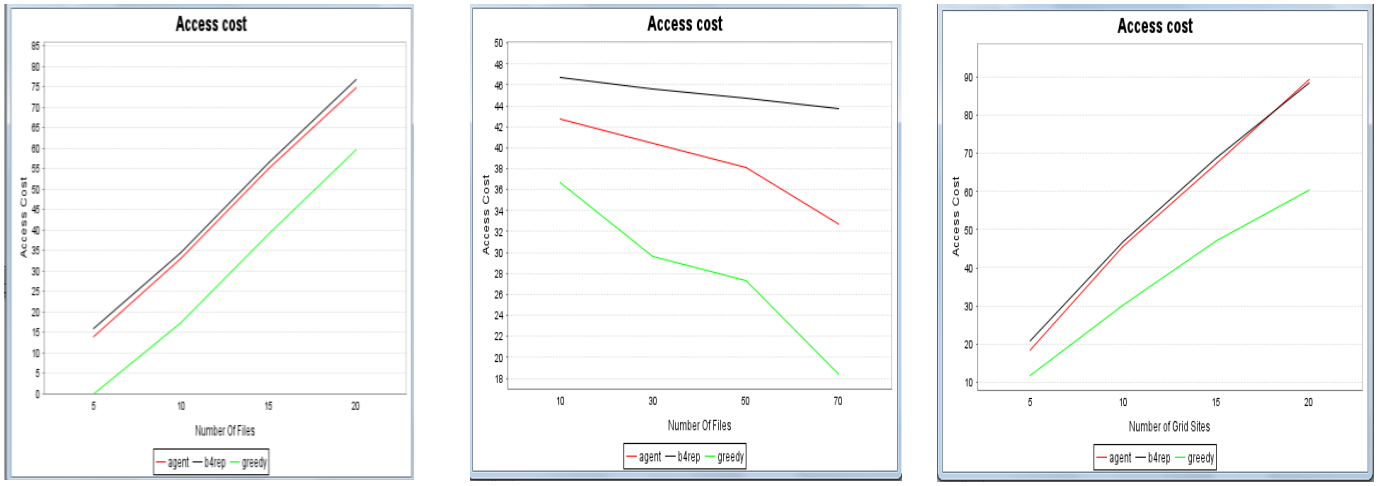
Fig. 1. Performance comparison of Centralized, Agent and Before repilcation algorithms in small scale. Unless varied, the number of sites is 10, number of data files is 10, and memory capacity of each site is 50 mb.



Number of Files *Storage Capacity(MB)* *Number of Grid sites*

(a)Varying number of data files. *(b)Varying storage capacity.* *(c)Varying number of Grid site*

Fig. 2. Performance comparison of Centralized, Agent and Before replication algorithms in large scale. Unless varied, the number of sites is 10, number of data files is 100, and storage capacity of each site is 500 mb.



Number of Files *Storage Capacity(MB)* *Number of Grid sites*

(a)Varying number of data files. *(b)Varying storage capacity.* *(c)Varying number of Grid sites*

Fig. 3. Performance comparison of Centralized, Agent and Before replication algorithms with varying data file size. Unless varied, the number of sites is 10, number of data files is 10, and storage capacity of each site is 50 mb.

IV. Conclusion

The data file replication performance depends on a variety of factors such as replica selection, placement, network traffic and bandwidth. This paper focuses on data file replication algorithms by following different file replication strategies using simulation environments. Well suited replication strategy can improve Data Grid performance depending on data file access situation.

Replication is an effective mechanism to reduce file transfer and bandwidth consumption in data grid placing most accessed data at the right location can greatly improve the performance of data access from a user's perspective.

The main objective of an agent is to select a candidate site for the placement of a replica that reduces access cost, network traffic and aggregated response time for application.

Centralized data replication algorithm reduces the file access cost & increases data availability.

References

1. Dharma Teja Nukarapu,Bin Tang, Liqiang Wang and Shiyong Lu, "Data Replication in Data Intensive Scientific Applications with Performance Guarantee", IEEE Transaction on Parallel and Distributed Systems, Vol. 22, No 8, Aug 2011.
2. D. G. Cameron, R. Carvavajal-Schiaffino, A. P. Millar, C. Nicholson, K. Stockinger and F. Zini, "Analysis of Scheduling and Replica Optimisation Strategies for Data Grids using OPtorSim ".
3. Ms. Shaik Naseer and Dr. K. V. Madhu Murthy, "Agent Based Replica Placement in a Data Grid Environment",First Int"lConference on Computational Intelligence, Communication Systems and Networks 2009.
4. Mahesh Mayura and Ketan Shah,"A Review on File Replication Algorithms", journal of Sci. & Tech. Mgt. vol 3(2), july 2011.
5. Kavita Rananathan and Ian Foster, "Identifying Dynamic Replication Strategies for High Performance Data Grid", Proc. Second Int"l Workshop Grid Computing (Grid), 2001.
6. 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.
7. The European Data Grid Project. Homepage http://eudatagrid.web.cern.ch/eu-datagrid,2005.
8. 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.
9. GridSim: A Grid Simulation Toolkit for Resource Modeling and Application Scheduling for Parallel and Distributed Computing, http://www.buyya.com/gridsim/

