Maintaining Consistency during Data Replication in Grid Environment

M.Suguna
II M.E
Department of Computer Science and Engg
Dr.Mahalingam College of Engineering and Technology, Pollachi

J.Bhavithra M.E
Assistant Professor (SS)
Department of Computer Science and Engg
Dr.Mahalingam College of Engineering and Technology, Pollachi

ABSTRACT

Data replication has to reduce data file transfer time, bandwidth consumption and maintain the consistency between the data and replica nodes. The centralized replication that reduces the total data file access delay and caching algorithm is used for any replica server is easily joining and leaving from the main server. An Integrated File Replication and Consistency Maintenance mechanism algorithm is used to achieve high efficiency in data replication and consistency maintenance at a low cost. Each replica server determines data replication and update polling by dynamically adapting to time varying file query and file update rates. Poll reduction process is to avoid unnecessary updates.

Keywords

Grid, Replication, Consistency Maintenance, Limd**.**

1. INTRODUCTION

In today's world of high speed computing, computers have become powerful, even home based desktops are enough to run complex applications. But still use complex scientific experiments, advanced modeling scenarios, genome matching, astronomical research, a wide variety of simulations, complex scientific & business modeling scenarios and real time personal portfolio management, which require huge amount of computational resources. [1]

Grid computing is becoming an important and interesting new field of research, mainly because it offers such a large variety of applications. Informally, grid is a parallel and distributed system that enables dynamic sharing, selection, and aggregation of geographically distributed autonomous resources, depending on their availability, capability, performance, cost, and user quality of service requirements. So, the main aim of a grid is to connect geographically dislocated resources into one large system that enables users to have transparent access to data and computing resources across the grid. These resources are usually much bigger and powerful than the resources available at the user's local sites. Data grid is that they produce and manage very large amount of data sets.

The major goal of data grid is to provide interoperability among many data sources to support seamless virtual organizations. It also aims to provide highly available, secure, and efficient access to data for widely distributed users and intensive access applications. Availability and efficient accesses are critical requirements in many data intensive applications and delayed accesses due to the availability problem and non responsiveness may cause severe consequences. These characteristics of data grids make data confidentiality is highly challenging. Security solutions for data grids should be specifically adapted to work in such dynamic environments.

It uses every resource. This costs a huge file transfer time and bandwidth. So, trying to reduce both the file transfer time and bandwidth is one of the important challenges.

Replication is an effective mechanism to reduce file transfer time and bandwidth consumption in Data Grid. Placing most accessed data at the right locations can greatly improve the performance of data access from a user's perspective. Design a centralized greedy data replication algorithm, which gives the total data file access time reduction and also design a distributed caching technique based on the centralized replication algorithm, it can be easily adopted in a distributed environment.

Data consistency maintenance is to maintain the consistency between a data and its replicas is necessary to data replication. To maintain the consistency in all replicas in grid system using IRM (Integrated File Replication and Consistency Maintenance mechanism) that achieves high efficiency in data replication and consistency maintenance at a significantly lower cost. Each node decides to create or delete a replica and to poll for update based on file query and update rates in a totally decentralized and autonomous manner. It replicate highly queried files and polls at high frequency for frequently updated and queried files.IRM avoids unnecessary file replications and updates by dynamically adapting to time-varying file query and update rates. It improves replica utilization, file query efficiency, and consistency fidelity. The remainder of this paper is organized as follows. Section 2 reviews related work in the field. Section 3 details the design principles and implementation for data replication. Section 4 discusses the consistency between the data and replica nodes and Section 5 execution phase and Section 6 conclusion of our work.

2. RELATED WORKS

The data grid is produced and stored large amount of data. The large data sets are weather report, climate change etc., and the main aims of using replication are to reduce access latency and bandwidth consumption. The other advantages of replication are that it helps in load balancing and improves reliability by creating multiple copies of the same data. Here we going to use frequently used data is replicated and place replica node in user perspective. 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 behavior. The replicas have to be manually created and managed if one were to use static replication. [2]

The data grids are differentiated by the requirement for transfer of massive datasets. The area used in data grids are storage management, data discovery and data replication. The replica manager directs the creation and management of replicas according to the demands of the users and the availability of storage, and a catalog keeps track of the replicas and their locations [3].

The main task of a data grid is to manage huge amounts of data and data intensive tasks. Data replication on data grids is a static optimization problem. In this paper described a combinatorial optimization problem which models the problem of replicating objects on data grids. Data grids are becoming a very interesting field of research, data replication will necessarily receive increasingly more attention. The limitation of the static approach is that the replication cannot adjust to the dynamically changing user access pattern [4].

When replicate data must ensure that when one copy of the data is updated all the other copies are updated too. Depending on how and when these updates are executed and get inconsistencies in the data store. There are two ways in which the data store can be inconsistent. First the data could be stale, that is, the data at some replicas has not been updated, while others have. Staleness is typically measured using time or versions. As long as updates are reliably propagated to all replicas, given enough time stale data will eventually become up to date. The other type of inconsistency occurs when operations are performed in different orders at different replicas [5].

Replication is a strategy in which multiple copies of some data are stored at multiple sites by storing the data at more than one site, if a data site fails, a system can operate using replicated data, thus increasing availability and fault tolerance. At the same time, as the data is stored at multiple sites, the request can find the data close to the site where the request originated, thus increasing the performance of the system. But the benefits of replication, of course, do not come without overheads of creating, maintaining and updating the replicas. If the application has read-only nature, replication can greatly improve the performance. But, if the application needs to process update requests, the benefits of replication can be neutralized to some extent by the overhead of maintaining consistency among multiple replicas [6].

The current generation of peer to peer network share predominantly static files, future peer to peer networks will support sharing of files that are modified frequently by their users. In such applications, files may be modified and updated during their lifetime. To handle such dynamic content, P2P networks must evolve read only system to one where files can be both read and written. Since files may be widely replicated in a P2P system, handling dynamic files requires consistency techniques to ensure that all replicas of a file are temporally consistent with one another [7].

The data are read only for many data grid applications, we do not consider consistency maintenance between the master file and the replica files.

3. EXISTING SYSTEM

Replication is an effective mechanism to reduce the file transfer time and bandwidth consumption in data grids placing most accessed data at the right locations. It improves the performance of data access.

3.1 Construction of Data Grid Model

Consider a Data Grid model as shown in Fig. 1. A Data Grid consists of a set of sites. There is one top level site, which is the centralized 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 mapping between each data file and all the institutional sites where the data is replicated. Each site may contain multiple grid resources.

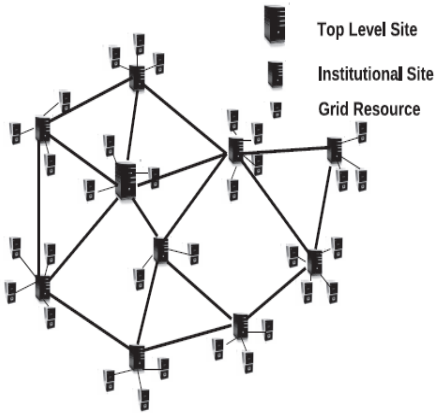

Fig-1 Construction of data grid model

A grid resource could be either a computing resource, which allows users to submit and execute jobs, storage resource, which allows users to store data files. For the data file replication problem addressed, there are multiple data files, and each data file is produced by it's the top level site or the institutional site. Each Grid site has limited storage capacity.

There are p data files; the files are splitted into part of the files $D = \{D_1, D_2,...,D_p\}$ in the Data Grid. Each part of the file is placed into each replica servers.

3.2 Centralized Data Replication in Data Grids

The centralized data replication algorithm is implemented in central server. The central server maintains the whole data file. This algorithm helps to replicate the frequently used data to place the correct location. Centralized replica catalogue contains the information about data and replica it is used to locate and maintain the replica site.

The user wants data from the any node; the node contains that particular data in that file to display the content to the user, otherwise to check the replica node. The replica server checks that data is available in which file then the main server retrieve the data from replica server and so on.

If that data is not containing in any replica node then the request is forwarded to the main server and retrieve the data from the file.

Algorithm 1.centralized data replication algorithm

Begin

$M=A_1=A_2=...=A_p=\Phi$ (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 on site m gives the maximum $r(G,\{A_1,A_2,....A_i,.....,A_p\}) - r(G,\{A_1,A_2,....A_i\ U\{m\},...A_p\})$;

$A_i=A_i\ U\ \{m\}$

end while ;

Return $M = \{A_1, A2,....,A_p\}$;

End.

3.3 Distributed Data Caching in Data Grids

Using distributed data caching algorithm, each node joining or leaving automatically. Each node maintains the nearest replica catalogue. The nearest replica catalogue maintains the information about the nearest node. Each node should maintain the nearest replica catalogue. This data caching algorithm is mainly implemented in the central or main server to identify which node is joining or leaving from data grid model.

Distributed algorithm have two important components: nearest replica catalog (NRC) maintained at each site. The main server maintains a Centralized Replica Catalogue (CRC), which is maintains the list of all replica servers. If one replica server is joining in the main server, that replica server information is entered into list. If any replica node is leaving from the main server, to delete the information from the list.

Nearest Replica Catalog (NRC)

Each server maintains the nearest replica catalog. It contains the information about the nearest replica and where the file is too placed. 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.

Maintenance of NRC

Each server should maintain the replica site. If anyone server is leaving from the data grid site, the site sends message to the top level site. The top level site removes the information from the list. If any new server is joining to the data grid site, message is broadcast to the grid sites.

4. PROPOSED SYSTEM

In the proposed system an algorithm for Integrated File Replication and Consistency Maintenance Adaptive File Consistency Maintenance Algorithm is used. The integrated file replication and consistency maintenance adaptive file consistency maintenance algorithm is used to manitain the consistency between the data.in this algorithm used two method.poll reduction and polling frequency determination .In polling frequency determination used linear increase multiplicative decrease algorithm.in this algorithm to set the time-to-refresh value with each replica.if the file is changed frequently in the main server,the TTR value is decreased by ultiplicative factor.otherwise increased the TTR value is linearly.In poll reduction,each replica node consider the file query rate and change rate.based on file query rate the time-to-refresh value is changed.when a file changes frequently,if a replica node does not receive queries for the file during a time period,it is overhead waste to poll the main server for validation during the time period.

IRM employs adaptive polling for file consistency maintenance to provide file replication dynamism.

4.1 Polling Frequency Determination

Consider a file's maximum update rate is $1/\Delta t$, which means it updates every Δt time units, say seconds, in the highest update frequency. In this case, a file replica node can ensure that a replica is never outdated by more than Δt seconds by polling the owner every Δt seconds.

Polling also refers to the situation where a device is repeatedly checked for readiness, and if it is not, the computer returns to a different task. Although not as wasteful of CPU cycles as busy waiting, this is generally not as efficient as the alternative to polling, interrupt-driven I/O.

In polling frequency determination replica node should be able to adapt the its polling frequency in response to the variations.IRM uses a Linear Increase Multiplicative Decrease algorithm.TTR denotes the next time instant a node should poll the server to keep its replica updated.

When each polling the data is doesn't change; the TTR value is linearly increased

$$TTR=TTR_{old}+\alpha \quad (1)$$

The file is updated the last poll, the TTR value is reduced by a multiplicative factor

$$TTR=TTR_{old}/\beta \quad (2)$$

The algorithm takes as input two parameters: TTRmin and TTRmax, which represent lower and upper bounds on the TTR values. Values that fall outside these bounds are set to

$$TTR=\max(TTR_{min},\min(TTR_{max},TTR)) \quad (3)$$

The bounds ensure that the TTR is neither too large nor too small. Typically, TTRmin is set to Δt, since this is the minimum interval between polls necessary to maintain consistency guarantees. The algorithm begins by initializing

$$TTR = TTRmin = \Delta t. \quad (4)$$

4.2 Poll Reduction

In poll reduction, the file change rate and file query rate is also main factor to consider in consistency maintenance. When a file changes frequently, if replica node does not receive queries for the file during time period, it is an overhead waste to poll the server for validation during the time period.

When the file change rate is higher than the file query rate, there is no need to update the replica at the rate of file change rate.

if $TTR \leq T_{query}$ **then** (5)

$TTR_{poll}=T_{query}$

The file is queried at a high rate than change rate, and then the file should be updated timely based on TTR.

if TTR >Tquery **then**

TTRpoll=TTR (6)

Algorithm 2.IRM adaptive file consistency maintenance algorithm

if there is a query for the file **then**

include a polling request into the query for file f
else
send out a polling request
if get a validation reply from file owner **then**{
if file is valid **then**

[α=0.5, β=1.5]

$TTR=TTR_{old}+\alpha$
if file is stale **then**{
$TTR=TTR_{old}/\beta$

Update file replica}

if $TTR>TTR_{max}$ or $TTR<TTR_{min}$ **then**

$TTR=\max(TTR_{min},\min(TTR_{max},TTR))$

if $TTR \leq T_{query}$ **then**

$TTR_{poll}=T_{query}$

else

$TTR_{poll}=TTR$}

It is highly effective in reducing file query latency, the number of replicas, and file consistency maintenance overhead. The proposed mechanism performance is better than other consistency maintenance algorithms in terms of low cost and file update rate [10].

5. PERFORMANCE EVALUATION

IRM adapts its update rate to replica query rate in consistency maintenance. In polling slow query rate is based on slow update rate.IRM update rate decreases with the increase of the query interval time. The number of IRM's updates messages decrease as the query interval time increases. This paper proposes the effectiveness of IRM in reducing consistency maintenance overhead by flexibly adjusting a replica's update rate to its query rate.

5.1 Screen Shots

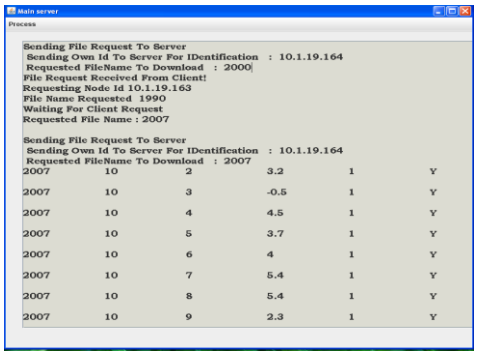

Fig-2 Main server

Fig-3 Replica server1

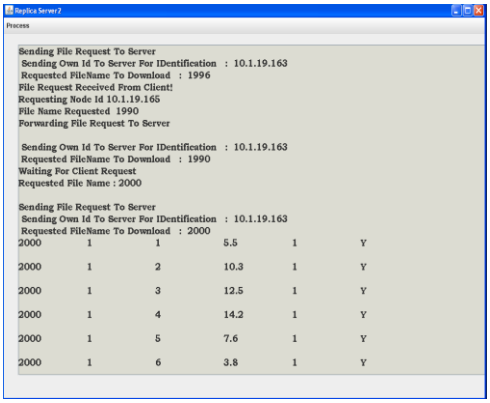

Fig-4 Replica server2

6. CONCLUSION

We proposed both read and write data. An Integrated File Replication and Consistency Maintenance algorithm achieve high efficiency at a lower cost in replica node. File replication and consistency based on file query rate and update rate. It reduces redundant file replicas, consistency maintenance overhead, and unnecessary file updates in replica nodes.

7. ACKNOWLEDGMENT

I take this opportunity to express my profound gratitude and deep regards to my guide Mrs.J.Bhavithra, Assistant Professor(SS),Department of Computer Science and Engg for her exemplary guidance, monitoring and constant encouragement throughout the course of this project. The blessing, help and guidance given by her time to time shall carry me a long way in the journey of life on which I am about to embark.

8. REFERENCES

[1] Satyarth Kumar Singh 2008, Efficient grid scheduling algorithm based on priority queues, Thapar University, Patiala.

[2] Kavitha Ranganathan and Ian Foster 2007, Identifying Dynamic Replication Strategies for a High-Performance Data Grid, the University of Chicago.

[3] Srikumar Venugopal, Scheduling Distributed Data-Intensive Applications on Global Grid, the University of Melbourne, Australia.

[4] Ihor Kuz, 2000 Replication and Consistency Maintenance, the University of New South Wales.

[5] Sushant goel, Rajkumar buyya, 2006 Data Replication Strategies in Wide Area Distributed Systems, Grid computing and distributed systems (grids) laboratory, p.211-241.

[6] Jiang Lan, Xiaotao Liu, Prashant Shenoy and Krithi Ramamritham,2003 Consistency Maintenance in Peer-to-Peer File Sharing Networks, in proc. of 3rd intl. internet applications, WIAPP ,p.90-94.

[7] D.Dullmann and B. Segal, 2001 Models for Replica Synchronization and Consistency in a Data Grid, IEEE.

[8] U. Cibej, B. Slivnik, and B. Robic, 2005 The Complexity of Static Data Replication in Data Grids, Parallel Computing, Vol.31, 8-9, p.900-912.

[9] R. Al-Khannak, B. Bitzer, 2006 Modifying Modern Power Systems Quality by Integrating Grid Computing Technology, Germany.

[10] Haiying (Helen) Shen, 2010 IRM: Integrated File Replication and Consistency Maintenance in P2P Systems, IEEE Transactions on Parallel and Distributed Systems, Vol 21 Issue 1, p.100-113.

[11] James Gwertzman and Margo Seltzer, 1996 World Wide Web Cache Consistency, Proceedings of the USENIX 1996 Annual Technical Conference, p.12-12.

[12] Dharma teja nukarapu, liqiang Wang, 2011 Data Replication in Data Intensive Scientific Applications with Performance Guarantee, IEEE transactions on parallel and distributed systems, Vol.22, Issue 8, p.1299-1306.