The International Journal Of Engineering And Science (IJES)
||Volume||2 ||Issue|| 3||Pages|| 95-103||2013||
ISSN: 2319 – 1813 ISBN: 2319 – 1805

Optimal Replica Placement in Graph Based Data Grids

[1,]N. Devakirubai, [2,] A. Kannammal
[1,2,]Department of Computer Science and Engineering, Jayam College of Engineering and Technology

Abstract

The replica algorithms are built based on the topology of the data grid. The real representation of grid is general graph. So we propose an algorithm called Replica Placement in Graph Topology Grid (RPGTG) to place replicas on graph based data grids. In this paper, we address three issues concerning data replica placement in graph-based data grids. The first issue is how to minimize the data access time. The second issue is how to ensure load balance among replica servers. The third issue is how to minimize the unnecessary replications. To solve these entire issues, we propose a replica placement algorithm that finds the optimal locations for the replicas, which minimizes the data access time, ensure load balance of replica servers and determines the minimum number of replicas required.

Keywords **-** Data Grid, Data Replication, Graph topology grid, Replica Catalog, Replica Manager, Replica Placement.

Date Of Submission: 26th February 2013 Date Of Publication: 13th March 2013

1. INTRODUCTION

Grid can be defined as utilizing the unused resources of computers in a network, to solve a single problem that is too intensive for any stand-alone machine. Grid computing is commonly used for scientific or technical problem which requires huge number of computer processing cycles and need to access very large amounts of data. The real environment architecture of grid is general graph [1]. When a single computer is not able to perform a large computational task, it is divided up among individual machines, which may span multiple administrative domains and geographically distant. These machines run calculations in parallel and return back the results to the original machine. This is cost effective than Supercomputers and called as Computational Grid.

Data grid is an architecture or group of services which enable users to access, modify and transfer very huge amount of data which are geographically distributed. This architecture provides the services like storage systems, data access and meta data services [2]. The main consideration of the data grid is to maximize the data availability to users. The major issues while accessing data in a data grid is network latency and bandwidth problems. Due to the dynamic changes in the size of the grid, maximum data availability becomes the challenging issue.

This challenge can be overcome by Data Replication, which increases data availability, reduced bandwidth consumption, increased fault tolerance, improved scalability and improved response time. Copying data files at more than one place in a grid is called as Data Replication. Data Replication is required for fast and efficient data access and also ensures storage availability and network bandwidth availability [3]. Issues to be considered in data replication are balancing the number of replicas in grid sites, placing replicas in the appropriate site, maintaining the consistency of the replicated data and avoiding unnecessary replications. The purpose of data replication is to place the data so that the user can access the data fast and efficiently. We are proposing an algorithm called Replica Placement on Graph Topology Grid (RPGTG) to meet the replica placement issues. As a first step, the communication with in the architecture used by our algorithm is discussed. The replication technique is completely based on the architecture of the grid. In all the grid architectures, like Multitier architecture (tree architecture), peer to peer architecture, graph architecture and hybrid architecture, the nodes are considered as sites. Each site is composed of two basic elements: Storage Element (SE) and Computing Element (CE). The SE follows the storage load of the site and CE follows the access load of the site. Storage load is monitoring the storage capacity to not to exceed its storage capacity [4].

Placing replica in the parent node of a node which generates the maximum request is the common practice. So the access load can be defined as the cumulative total number of requests responded by a node, where the requests are received from all of its children. Though in the real environment the sites are arranged in the graph topology, for better managing of replicas, the graph structure will be viewed as tree structure, with the help of Breadth First Search (BFS) and Depth First Search (DFS) algorithm. BFS determines the level of each node and DFS determines the adjacent node list of each node. Then the replica selection and placement algorithms are implemented on the resulted tree structure. The replacement algorithm monitors the storage load of each replica server.

II. Literature Survey

In [5] Ding et al. proposed Data Placement algorithm and self tuning data replication algorithm for general grid topology for improved load balancing, reduced response time and conserved network bandwidth. In this model, grid is composed of clusters, with each cluster having different storage and computational capabilities. The resources in the cluster sites and data access patterns keeps on changing. So a self tuning data replication algorithm is proposed to automatically adjust such changes. The new replication algorithm outperforms the general threshold based algorithms in terms of efficiency and load balancing.

In [6] Rasool et al. proposed a two way replication strategy. The multi-tier sibling tree architecture is used which is a mixture of the architectures, presented by Ranganathan and Lin. It's a hierarchical model in which all the siblings are connected to each other as well. In this two way replication (TWR) scheme the most popular data is identified and placed to its proper host in a bottom up manner in this they are closer to the clients. In top down manner the less popular files are identified and are placed to one tier below the root node, in this way they are close to the roots. In this approach, replica selection is done by using the closest policy which tries to provide the data from the nearest site. The drawback of the research is that it only considers the homogeneous data grid nodes and cannot be applied to heterogeneous nodes while the nodes in a data grid are normally heterogeneous.

In [7] Lin et al. have addressed the problem of placement of a new replica in a proper place by considering a priority list. The proposed replica placement algorithm finds out the minimum number of replicas when the maximum workload capacity of each replica is given. The hierarchical model is different from other related works, because in this model they assume a logical connection between all siblings of a parent and a request can be served from a node present in sibling ring. If requested data is not present in sibling ring then parent ring is searched. This architecture is called a Sibling Tree model, which is an extension of a normal tree structure. The hierarchical model assumes a logical connection between the siblings and actually all connections from one sibling to another physically involves the parent i.e. at most two hops. This means the actual time taken to serve a request is infected more than it is presented, as this logical connection is assumed physical and already the time complexity is too high. The problem of network congestion or bandwidth consumption is not mentioned in proposed model.

In [8] Golda. J et al. they have worked for ensuring data availability, improving fault tolerance, reducing file access time, minimizing file transfer cost and controlling network bandwidth usage. They proposed two data replication algorithms in their paper. First, when geographically distributed storage servers are concerned for data replication, Centralized Data Replication algorithm is used. Second when the proximity among the storage servers are really high, then the Distributed Data Replication algorithm is used. In these two cases, the final replication decision is based on the characteristics such as need count of the files, minimum number of hops, size of the data files, bandwidth of file transmission, request rates of files across the sites, file location and capacity of storage servers. The drawback in this research is grid topology is not graph based.

In [9] Garmehi. M et al stated that data replication is one of the best-known strategies to achieve high levels of availability and fault tolerance, as well as minimal access times for large, distributed user communication using a worldwide data grid (DG). One of the challenges in data replication is to select the candidate sites where replicas should be placed and which are known as optimal placement of replicas (OPR). In that paper an algorithm is proposed, which is formulated by using dynamic programming method to find optimal placement k replicas of an object over DG systems, such that the overall cost (i.e. storage cost plus read cost) is minimized.

In [4] Jianzhong Li et al. they proposed a load balancing replication strategy, Fair-Share Replication (FSR) that takes into account both the access load and storage load of the replica servers before placing a replica. They use P2P concepts in multi-tier data grid with unique path for data search, and leverage the neighborhood of replica servers in replication process. A sibling node of a candidate replica server is selected for replica placement if its access and storage loads are less than the candidate. Experiment results show that FSR performs better than fast spread replication using random access pattern.

In [10] Tang et al. present a dynamic replication algorithm for multitier Data Grids. They propose two dynamic replica algorithms: Single Bottom Up and Aggregate Bottom Up. ABU achieved great performance improvements for all access patterns even if the available storage size of the replication server is very small. They compared the two algorithms to Fast Spread dynamic replication strategy and found ABU proves to be superior. As for SBU, although the average response time of Fast Spread is better in most cases, Fast Spread's replication frequency is too high to be applicable in the real world. Performance results show both algorithms reduce the average response time of data access compared to a static replication strategy in a multitier Data Grid.

In [11] Yaser. N et al. proposed a dynamic data replication policy. They developed a replication strategy for 3-level hierarchical structure, called HRS (Hierarchical Replication Strategy). In this replication algorithm, replicated file stored in the site with the largest number of access for the file and main factor for replica selection is bandwidth. The replication strategy proposed in this paper allows placing data in a manner that will suggest a faster access to files require by grid jobs, therefore there is an increase in the job execution's performance. If the available storage for replication is not sufficient, the proposed algorithm only replicates those files that do not exist in the local LAN.

In [12] Abdullah et al. presented a P2P model for higher availability, reliability, and scalability. Then have developed their own data grid simulator to test the proposed replication strategy, taking response time, number of hops and average bandwidth consumption as basic parameters for evaluation. In this research they are studying four replication strategies, out of which two are existing strategies: "requester node placement strategy" and "path node placement strategy", and two are newly proposed in this research: "path and requester node placement strategy", and "N-hop distance node placement". In the "requester node placement strategy", when a required file is found then it is only replicated to the requester node.

In the "Path node placement strategy" the file is replicated to all the nodes on the path from the requester node to provider node. The newly proposed strategy "Path and requester node placement strategy" is a combination of the first two strategies. In "N-hop distance node placement" a file is replicated to all neighbors' of provider nodes within an n hop distance. The results of their simulation show that new strategies have shown better performance than existing ones in terms of performance, success rates and response time. However, the proposed strategies use more bandwidth than the existing strategies. The drawback of the research is that the storage loads of replica servers are not considered in their strategies, because the file is replicated to all the nodes on the path from the requester node to provider node.

In [13] Sashi et al. presented a modified form of BHR (Bandwidth Hierarchy Replication) to overcome its limitations. In the modified BHR model a network region is defined as a network topological space where sites are located closely. Whenever the required replica is present in the same region, the job completion will be fast. Again, the Modified BHR model is based on tree structure which is not very suitable in real data grid environment.

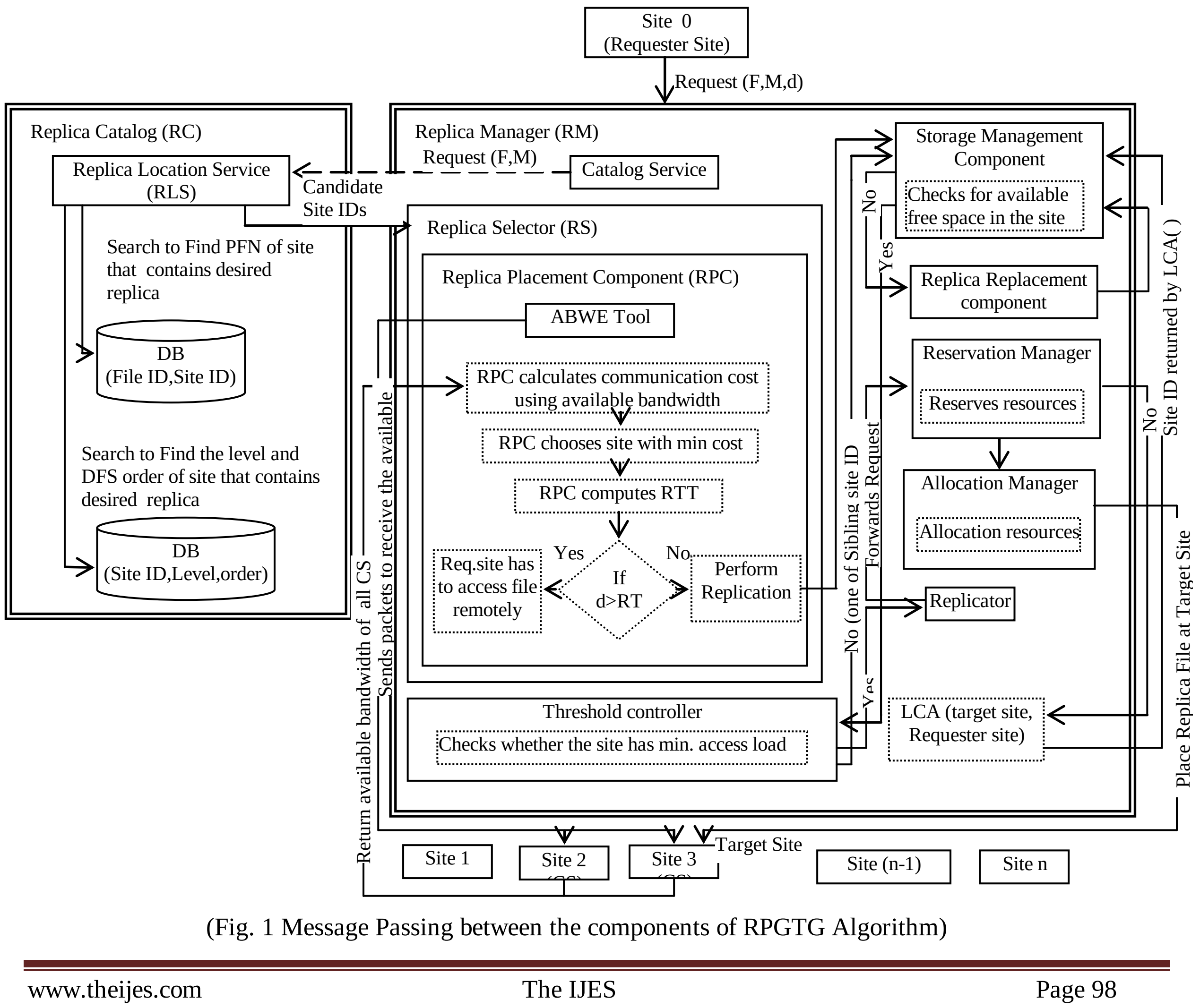
III. ARCHITECTURE OF REPLICA PLACEMENT IN GRAPH TOPOLOGY GRID ALGORITHM

Architecture for the proposed algorithm Replica Placement in Graph Topology Grid (RPGTG) is explained in this section. Since the true representation of grid is general graph, where there is no root node, we consider the grid topology as graph. Due to better managing of replica servers and their related nodes, we convert the graph structure to hierarchical structure. Here the data grid is modeled to have three tiers, where the tier 0 machines have enormous storage capacity. The tier 1 machines are called as Regional Servers have computing and storage resources. The tier 2 machines are called as Local Servers and tier 3 machines are workstations.

A network topological space where sites are closely located is called as network region. Job completion will be fast, if the required replica is found within the network region. To facilitate dynamic file replication in the multi-tier data grid, the following services are available in the system [14]. The Replica Catalog (RC) and Replica Manager (RM) are located at Regional servers and the Local Replica Catalog (LRC) and the Local Replica Manager (LRM) are the local services which are distributed on every machine in the system. The RC and RM at the Regional Servers manage the LRC and LRM of sites which are connected to them. Replica Catalog helps to recognize the physical locations of data files.

Physical File Name (PFN) is a URL pointing to a physical copy of the file. The Logical File Name (LFN) is a system-wide unique identifier of PFN. Replica Catalog has a database that stores the mappings between LFN and PFN. This database acts as a registry and keeps track of where files are placed in the grid. Another database in Replica Catalog stores the breadth-first and depth-first order of each node, by which each node can be aware of the nodes which are located at its sub tree. The level indicates the distance of a particular node from the root node. Replica Location Service (RLS) in Replica Catalog invokes the above two databases to store mapping between LFN and PFN. The function of Replica Manager is to perform replication and creating replicas.

Replica Manager contains the following components: Reservation Manager, which authenticate the reservation requests for resources based on the resource availability. Allocation Manager is a component that communicates with the resource manager to allocate the resources based on the reservations. Replica selector, selects the replica with the minimum communication cost, where communication cost is obtained by dividing the size of replica i with available bandwidth between grid site ‘a’ and ‘b’. Catalog Service is a component invoked by optimizer, which interacts with RLS to get the information from the databases in Replica Catalog. The component Replicator replicates the chosen replica on the optimal place found by our proposed algorithm.


Site 0
(Requester Site)
Request (F,M,d)
Replica Catalog (RC)
Replica Location Service (RLS)
Candidate Site IDs
Search to Find PFN of site that contains desired replica
DB
(File ID,Site ID)
Search to Find the level and DFS order of site that contains desired replica
DB
(Site ID,Level,order)
Replica Manager (RM)
Request (F,M)
Catalog Service
Replica Selector (RS)
Replica Placement Component (RPC)
ABWE Tool
RPC calculates communication cost using available bandwidth
RPC chooses site with min cost
RPC computes RTT
Yes
No
If d>RT
Req.site has to access file remotely
Perform Replication
Threshold controller
Checks whether the site has min. access load
Storage Management Component
Checks for available free space in the site
No
Yes
Replica Replacement component
Reservation Manager
Reserves resources
Allocation Manager
Allocation resources
Replicator
LCA (target site, Requester site)
No (one of Sibling site ID
Forwards Request
Yes
No
Site ID returned by LCA( )
Place Replica File at Target Site
Return available bandwidth of all CS
Sends packets to receive the available
Site 1
Site 2 (CS)
Site 3 (CS)
Target Site
Site (n-1)
Site n

(Fig. 1 Message Passing between the components of RPGTG Algorithm)

Replica Selection component has the following components: Available Bandwidth Estimation (ABWE) is a monitoring tool used to estimate the Round Trip Time (RTT) and available bandwidth between the host pair of nodes. The source node uses ABWE for sending continuously examined packets to the nodes which has the required replica. These nodes in turn reply about the delay and available bandwidth. The source node collects the replies and chooses the target node with the minimum communication cost. The second component Threshold Controller [15] determines the threshold value by comparing the average aggregated access request count and available server capacity.

The average aggregated access count is calculated by dividing the total number of aggregated access counts for a file at the replica server at each level with the number of the replica servers at their level. File is called as “popular file” when its access count exceeds the threshold value. The number of replica creation is reduced when the threshold value is greater and when the request rate decreases. The third component in Replica Selector is the Replica Placement Management. The proposed algorithm uses this component to find the minimum distance between source and destination. After considering the access rate of each replica server, this component chooses the replica server with minimum access load. Target nodes which have their access rate lesser than the threshold value are selected for replication.

IV. Replica Placement in Graph Topology Grid (RPGTG) Algorithm

4.1 Determining the level and adjacent nodes of each node

Every node in the graph structure is a site in real data grid structure, and every edge demonstrates the relations between sites. Graph structure is traversed by BFS algorithm by which the level of each node can be determined. Tree ‘T’ is obtained after traversing our graph structure by the BFS algorithm. The tree ‘T’ is traversed by DFS algorithm which determines the number of adjacent nodes of each node. List L is maintained to store the DFS order of nodes. The identification number given to any node is smaller than the identification number given to its descendents. The above mentioned identification number is assumed to be the grid site IDs. After traversing the tree by the BFS and DFS algorithms, one of the database in the RC get data about SiteID, level, depth_order of each node. So now the Local Server and Regional Server of our architecture are aware of the gird sites that are placed on its sub tree.

4.2 Finding the Lowest Common Ancestor (LCA)

To find the LCA(u,v) in List L, find the first occurrence of u and v, this part of numbers form the interval I. The minimum number in I is the value for LCA(u,v).

4.3 Popular Files

The RM in Regional Servers aggregate the access record of each file from lower to upper level to determine the threshold value. The most ‘Popular’ files are the one which are frequently accessed by clients and whose access count exceeds the threshold value. The threshold value is set based on the average aggregated access count at the replica servers in each level. The average aggregated access count is calculated by dividing the total number of aggregated access count for a file at the replica servers at a level by the number of replica servers at that level.

4.4 Functioning of Proposed Algorithm

When a grid site needs a popular file which is not stored locally, the request will be sent to the Local Server with information like requested file (F), requesting machine (M) and deadline (d). Whenever Local Server receives a file request, it stores “LFN of that file, PFN of the requestor site” at its RC in the database that contains (File ID, Site ID). If the requested file is not available in Local Server, the request is forwarded to Regional Server.The request is forwarded to the Catalog Service in RM for determining which grid sites have the requested replica. Catalog Service contacts the RLS at the RC, which invokes the two databases: database which stores File ID, Site ID and database which stores Site ID, Level, Order information.

The RLS searches in database and returns back the site information that contains the desired replica and these sites are called as candidate sites. Among these candidate sites, the site with minimum distance from the requester site is chosen as target site. This is done by the Replica Selector Component, which selects the target site with minimum communication cost. The Replica Placement component in Replica selector, computes the minimum distance between the requester site and the candidate sites. Replica Placement Component computes Round Trip Time (RTT) through ABWE (Available Bandwidth Estimation) Tool.

ABWE is a low network intrusive monitoring application based on packet pair techniques and designed to work in continuous mode. The requester site use ABWE for sending continuously probe packets to all other candidate sites. The candidate sites reply the available bandwidth to the Replica Placement Component, which in turn uses the available bandwidth information to calculate the communication cost of each candidate site. The communication cost is calculated by dividing the $size_i$ with the $bandwidth_{ab}$ (available bandwidth between grid site 'a' and grid site 'b'). The Replica Selector now chooses the site with the minimum communication cost. As each request has the deadline, our proposed algorithm now decides whether to perform replication or to access the file remotely by comparing the request's deadline. If (request's deadline > RTT) then the requester can access the file remotely or else replication will be performed.

Consider if the above condition results false, then replication has to be performed; now the question arises about where to replicate the file. To replicate on the requester site the Storage Management Component is contacted to check for the availability of space, if there is space for storing replication the Replica Placement Component contacts the Threshold controller to decide whether this site has minimum access load, the request is sent to replicator to perform the replication process. Replicator forwards this request to Reservation Manager through Storage Management Component. If Reservation Manager succeeds in making reservations, RM calls allocation manager. Once Allocation Manager finishes allocating the resource reservation, RM starts the file transferring from the identified target machine to the requestor site. If Reservation Manager does not succeed in making reservations on requester site, the proposed algorithm finds LCA of the identified target site and requester site.

To replicate on the site identified by LCA algorithm, the Storage Management Component is contacted to check for the availability of space, if there is space for storing replication, the Replica Placement Component contacts the Threshold controller to decide whether this site has minimum access load, if yes, replication is performed or else replication is performed on one of its sibling node whose access load is lesser than the threshold value. If no grid sites within the Local Server and Regional Server contain the desired replica, the request will be sent upward towards root. Replacement is performed when a remote replica has been selected for replication to the requester site's storage element and the storage element might not have sufficient space. In this case one or more replica must be deleted by LRU algorithm.

4.5 Algorithm To Determine The Level And Adjacent Nodes Of Each Node

4.5.1 Run the BFS algorithm to traverse the graph structure.
4.5.2 Let *T* denotes a tree which traversed by the Step 1.
4.5.3 Maintain the adjacent of each node on the tree *T*.
4.5.4 Run DFS algorithm to traverse the tree *T*.
4.5.5 Maintain the List *L* of nodes in the same order that they are visited.
4.5.6 Keep the sub tree of every node.

4.6 Algorithm To replicate and replica placement

4.6.1 Submit jobs to grid.
4.6.2 Every request sent to Replica manager of Regional servers.
4.6.3 Replica manager query Replica Catalog to determine which grid site contains the desired replica (Candidate sites).
4.6.4 If the file not found in lower level its Manager send Request to upper level.
4.6.5 Determine the communication cost between requester site and candidate sites.
4.6.6 Compute the Round Trip Time (RTT).
4.6.7 If (d>RTT) then access the file from the remote place or else replicate.
4.6.8 Check the storage element of the site selected for replication. If no storage space available invoke the Replica Replacement algorithm, otherwise
4.6.9 The Threshold Controller checks whether the site has minimum access load, if yes, it communicates with Reservation Manager.
4.6.10 If Reservation Manager succeeds in making reservations, Allocation Manager is called to allocate resources.
4.6.11 Once Allocation Manager allocated resources, Replication Placement is performed.

4.6.12 If Reservation Manager is not succeeded, then the Lowest Common Ancestor algorithm (LCA) is invoked.
4.6.13 LCA returns a site, with this site ID, repeat steps 8 and 9.
4.6.14 If the Threshold Controller results maximum access load, choose one of the sibling node and continue step 10 and 11.

4.7 Replica Replacement

4.7.1 The threshold controller controls the threshold value at each level *i*
4.7.2 for each (node j in level *i*)
4.7.3 checks the sl(j)
sl is (the desired percentage of storage use / the actual percentage of storage used) at each replica server.
4.7.4 The threshold value on level *i* is updated
4.7.5 T = $\sum_{j=1}^{n} sl(i)$ (where n is number of node at level i)
4.7.6 for each (node j in level i)
4.7.7 if (sl (j) > T) Then Do Replacement
4.7.8 If (target site doesn't have enough free space) Then Do Replacement
4.7.9 Replacement:
Sort Files in SE using LRU and delete the files in ascending order until the space found for the new replica.

V. PERFORMANCE EVALUATION

5.1 Simulation tool

OptorSim is used as the simulator tool to evaluate the performance of our proposed algorithm. OptorSim [16] is a simulation package written in Java. It was developed to study the effectiveness of replica optimization algorithms within a Data Grid environment [17] and to represent the structure of a real European Data Grid. The simulation was constructed assuming that the Grid contains several sites; each consists of zero or more Computing Elements (CEs) and zero or more Storage Elements (SEs). CEs run jobs by processing data files, which are stored in the SEs. A Resource Broker (RB) controls scheduling of jobs to Grid Sites, and schedules jobs to CEs according to scheduling algorithm. Each site handles its file content with Replica Manager (RM), within which a Replica Optimizer (RO) contains the replication algorithm which drives automatic creation and deletion of replicas [18].

There are two types of algorithms in OptorSim: the scheduling algorithm used by the RB to schedule jobs to CEs and the replication algorithm used by RM at each site to manage replication. Each scheduling and replication algorithm is implemented as a separate Resource Broker and Replica Optimizer class respectively. We have made changes only in Replica Optimizer Class and the default Resource Broker class is used. There are three options for Replication Algorithms in OptorSim. First, one can choose No Replication which never replicates a file and all replicas are taken from the master site where the data were produced at the beginning of the simulation and the distribution of files does not change during simulation. Second, one can use LRU or LFU algorithm that always tries to replicate and, if necessary, deletes Least Recently Used files or Least Frequently Used files.

Third, one can use an economic model in which algorithm only deletes files if they are less valuable than a new file. There are currently two types of the economic model: the binomial economic model, where file values are predicted by ranking the files in a binomial distribution according to their popularity in the recent past, and the Zipf economic model, where a Zipf-like distribution is used instead [19].We have compared our proposed algorithm with all of these algorithms.

5.2 Simulation Results

The RPGTG algorithm was compared with No Replication, LRU (Least Recently Used) and LFU (Least Frequently Used algorithms). Final results and discussion The simulated grid used in our experiments has 20 sites, 18 of them have Storage Element (SE) and Computing Element (CE) and 2 of them have only SE. The capacity of sites 14 and 19 that only have SE's are 100 GB (all master files are stored in these two sites at the beginning of simulation) and the other ones are 70 GB and 50 GB. The SE's of Regional Server are 70 GB. Also there are 8 routers that do not have SEs and CEs. We have compared our proposed algorithm with 3 of existing algorithms: No replication, LRU and LFU. The three algorithms are implemented in optorsim. The simulation results for the different access patterns are shown in Figures 1and 2.We ran six jobs totally 100 times and evaluated the impacts of file access pattern. We tested RPGTG and the other algorithms in 2 types of access pattern: 1. Random Access, 2. Random Zipf Access. The performance evaluation metrics that we used in our simulation are: Mean Job Execution Time and Effective Network Usage

5.2.1 Mean Job Time of all Jobs on Grid

The mean job time of all jobs on grid is defined as the combined total time in milliseconds of all the jobs run divided by the number of jobs completed. Mean Job Time of all jobs on Grid is calculated as dividing the sum of the difference in Job arrival time and Job departure time with number of jobs completed. Note that for all the components, total job time is defined as the sum of the entire individual job times, including their queuing times [20]. We have compared Mean Job Time of our proposed algorithm with other existing ones. The comparison results are shown in fig 2. The simulation results show that RPGTG has the lowest value of Mean Job Execution Time in both Random and zipf access patterns.

The reason is because of the nodes at upper level like: the Local Server and the Regional Servers can be aware of the location and the level of their children and descendent nodes. So every Regional server aware of the grid sites which are placed in its Region. According to our proposed algorithm, the most popular file is chosen to replicate. On the other hand the minimum distance between the requester site and target sites is computed. Then the proposed algorithm computes the RTT between these sites, and estimates the ability of responding the job before its deadline. By considering these factors, the algorithm tries to replicate the desired file on site which is located nearby the client, and could respond the job before its deadline. So at the time of execution, jobs will have their required files locally.

One of the important factors that decrease the grid site's job execution time is having their required files locally stored on their storage element. It should be noted that, according to zipf access pattern, a few files are requested many times. So, as mentioned before, in our proposed architecture the physical location of sites and the files that requested by them are registered in specific databases. Additionally, the proposed algorithm selects the most popular file. By this features the proposed algorithm has the lowest value of Mean Job Execution Time in comparison with LFU, LRU, and No replication. But when Random access pattern is used, LFU and LRU have shorter mean job. As Mean Job Execution Time is the most important evaluation metric, RPGTG can be considered as the superior strategy.

5.2.2 Effective Network Usage (ENU)

This is effectively the ratio of files transferred to files requested, so a low value indicates that the optimization strategy used is better at putting files in the right places. It ranges from 0 to 1. It can be measured by using equation. ENU is calculated by dividing the sum of remote file accesses and file replications with the sum of remote file accesses and local file accesses. Through the graph search algorithms like: BFS and DFS, which traverse our graph structure, every node knows its location among its sibling and child nodes, so the required file could be replicated from the sites which are nearest to it. By this assumption the bandwidth consumption is minimized and used effectively. The No Replication strategy performs the worst because it always accesses files remotely.

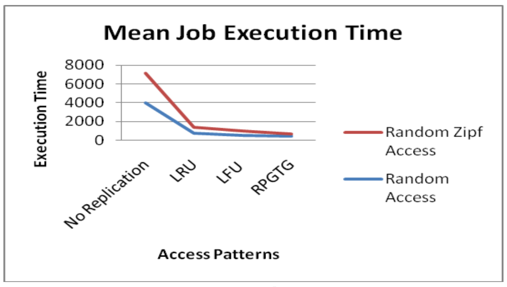


Fig. 2 Average Mean Job Execution Time

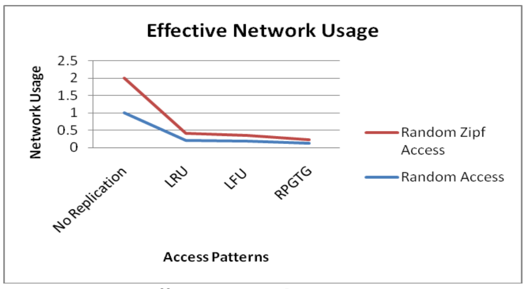


Fig. 3 Average Effective Network Usage

6 CONCLUSION

In this paper we described our proposed architecture, Replica Placement on Graph Topology grid Architecture. As a true representation of a grid is a general graph in which there is no central node designated as a root node, and each node can be connected with any number of nodes. Therefore in this paper, we consider a grid with graph-based topology. But at the first step of our proposed algorithm, this structure is converted to hierarchal tree structure due to better managing of replica servers and their related nodes. Next we determined a suitable selection and placement of replicas on this hierarchical structure. The proposed algorithm takes into account both the access load and the storage load of the replica servers and their sibling nodes before placing the replicas. So the workloads among these nodes are balanced and data availability is increased. It also reduces unnecessary replication and improves our performance. The comparison of our algorithm with LRU and LFU demonstrates that our algorithm has better performance in average storage usage when zipf access pattern is used.

REFERENCES

[1] Zeinab Fadaie, Amir Masoud Rahmani, A new Replica Placement Algorithm in Data Grid, International Journal of Computer Science, Vol 9, Issue 2, (2012), Pp 491-507

[2] J. Zhang, B.S. Lee, X. Tang, C.K.Yeo, A Model To Predict The Optimal Performance Of The Hierarchical Data Grid, Future Generation Computer Systems 26 (2010), Pp 1-11

[3] Y. Yuan, Y. Wu, G. Yang, F. Yu, Dynamic Data Replication Based On Local Optimization Principle In Data Grid, Grid And Cooperative Computing, (2007) GCC 2007, Sixth International Conference On.

[4] Q. Rasool, J. Li, G. S. Oreku, Sh. Zhang, D. Yang, A Load Balancing Replica Placement Strategy In Data Grid, Digital Information Management, ICDIM 2008. Third International Conference, Nov. (2008), Pp. 751-756

[5] Y. Ding, Y. Lu, Automatic Data Placement And Replication In Grids, In: High Performance Computing, Hipc, International Conference On, (2009), Pp. 30–39

[6] Q. Rasool, J. Li, S. Zhang, Replica Placement In Multi-Tier Data Grid, In: 2009 Eighth IEEE International Conference On Dependable, Autonomic And Secure Computing, (2009), Pp. 103–108.

[7] Y.F. Lin, J.J. Wu, P. Liu, A List-Based Strategy For Optimal Replica Placement In Data Grid Systems, In: 37th International Conference On Parallel Processing, (2008), Pp. 198–205.

[8] P. Golda Jeyasheeli, K. Ramar, And M. Archanaa Centralized And Distributed Replica Placement Algorithms For Data Grid, European Journal Of Scientific Research, Vol. 79, No. 1, (2012), Pp. 68-81.

[9] M. Garmehi And Y. Mansouri, Optimal Placement Replication On Data Grid Environments, 10th International Conference On Information Technology, ICIT 2007.

[10] M. Tang B.S Lee, C.K. Yeo and X. Tang, Dynamic Replication Algorithms For The Multi-Tier Data Grid, Future Generation Computer Systems, Vol. 21, (2005), Pp. 775-790.

[11] Yaser Nemati, Faramarz Samsami, Mehdi Nikhkhah (2012), A Novel Data Replication Policy In Data Grid,, Australian Journal Of Basic And Applied Sciences, 6(7): 339-344.

[12] Abdullah, M. Othman, H. Ibrahim, M.N. Sulaiman, A.T. Othman, (2008) Decentralized Replication Strategies For P2P Based Scientific Data Grid, Information Technology, Itsim, International Symposium On, Pp. 1–8.

[13] Sashi. K, A.S. Thanamani, (2011) Dynamic Replication In A Data Grid Using A Modified BHR Region Based Algorithm Future Generation Computer Systems 27 (2), Pp. 202–210.

[14] Dogan, A Study On Performance Of Dynamic File Replication Algorithms For Realtime File Access In Data Grids Future Generation Computer Systems, September (2009), Pp. 829-839.

[15] M. Shorfuzzaman, P. Graham, R. Eskicioglu, Adaptive Replica Placement In Hierarchical Data Grids, Journal Of Physics: Conference Series 256 (2010).

[16] Optorsim – A Replica Optimiser Simulation, Http:// Grid-Datamanagement.Web.Cern.Ch/Grid-Data Management/ Optimization/Optor

[17] W.H. Bell, D.G. Cameron, L. Capozza, A.P.Millar, K. Stockinger, F. Zini, Simulation Of Dynamic Grid Replication Strategies In Optorsim, Int. Journal Of High Performance Computing Applications, 17(4), (2003).

[18] R. M. Vozmediano, Application Layer Multicast For Efficient Grid File Transfer, International Journal Of Computer Science And Applications, Technomathematics Research Foundation, (2009), Pp.70-84.

[19] D.G. Cameron, A.P. Millar, C. Nicholson, Optorsim: A Simulation Tool For Scheduling And Replica Optimization In Data Grids, Proc. Computing In High Energy And Nuclear Physics (CHEP), (2004).

[20] D.G. Cameron, R. Schiaffino, J. Ferguson, A.P. Millar, C. Nicholson, K. Stockinger, F. Zini, Optorsim V2.1 Installation And User Guide, October (2006).