RPLB: A Replica Placement Algorithm in Data Grid with Load Balancing

Kingsy Rajaretnam, Manimegalai Rajkumar, and Ranjith Venkatesan
Department of Computer Science and Engineering, Sri Ramakrishna Engineering College, India

Abstract: *Data grid is an infrastructure built based on internet which facilitates sharing and management of geographically distributed data resources. Data sharing in data grids is enhanced through dynamic data replication methodologies to reduce access latencies and bandwidth consumption. Replica placement is to create and place duplicate copies of the most needed file in beneficial locations in the data grid network. To reduce the make span i.e., total job execution time, storage consumption and Effective Network Usage (ENU) in data grids, a new method for replica placement is introduced. In this proposed method, all the nodes in the same region are grouped together and replica is placed in the highest degree and highest frequency node in the region. The node to place replica should be load balanced in terms of access and storage. The proposed dynamic Replica Placement algorithm with Load Balancing (RPLB) is tested using OptorSim simulator, which is developed by European Data Grid Projects. In this paper, two variants of the proposed algorithm RPLB, namely $RPLB_{frequency}$ and $RPLB_{degree}$ are also presented. The comparative analysis of all the three proposed algorithms is also presented in this paper. A Graphical User Interface (GUI) is designed as an interface to OptorSim to get all values for grid configuration file, job configuration file and parameters configuration file. Simulation results reveal that the performance of the proposed methodology is better in terms of makespan, storage consumption and replication count when compared to the existing algorithms in the literature.*

Keywords: *Replica placement, load balancing, ENU, data grid, data replication.*

Received June 17, 2013; accepted April 28, 2014; Published online December 23, 2015

1. Introduction

Grid computing [15] provides co-ordinated resource sharing in geographically distributed, dynamic and virtual organizations. There are two types of grid systems, namely, computational grid and data grid [42]. Data grid is an important branch of grid computing. It is a collection of geographically distributed computers and storage to share data and resources [2, 11, 23]. According to [3, 12] data grid architecture consists of the following layers, namely, fabric, connectivity, services and application. The distributed resources in the fabric layer are connected as a wide area network using high bandwidth. The fabric layer consists of computing, storage and networking resources which is analogous to physical layer. The data transfer protocol in the connectivity layer is responsible for copying data from resources in the fabrication layer. The data grid service layer provides core services such as: Resource monitoring, replication management and resource allocation.

Most of the scientific applications such as: High energy physics [39], human genome project [20], human brain project [21], earth system grid [40] and meteorology data grid system [42] are implemented in application layer. Usually these applications generate huge amount of data on daily basis. In all these applications data management is an important task for efficient data access. Data replication is a key technique to manage large data sets in a distributed job. way [28, 30]. It also reduces data access time for a grid If the required data file is stored in the job running site, then the communication delay of a data transfer is reduced; else it has to be transferred from remote site during job execution. This transfer takes long time when the size of the file is too large and the bandwidth is limited [14].

In general, data replication is broadly classified into two categories, namely, static replication and, dynamic replication. Static replication techniques store and delete replicas manually. Job scheduling is quick in static replication techniques [11, 38]. But they are not scalable when the number of users and data is drastically increased in the data grid. Dynamic replication strategies create and delete replicas dynamically depending on the resources and behavior of the users [6, 8, 17, 23, 24, 32, 45]. A dynamic replication technique can be implemented in either centralized or distributed fashion. Some of the advantages of dynamic replication strategies when compared to static replication technique are improved availability, reliability, scalability, adaptability and performance.

The proposed dynamic replica placement strategy places the replica in the job running site or nearby site depending on the load in order to provide better performance in terms of makespan, effective network and storage usage. The proposed algorithm, namely, Replica Placement with Load Balancing (RPLB) is explained in section 3.

The rest of the paper is organized as follows: The related work is discussed in section 2. Section 3, discusses the proposed algorithm, RPLB and its two variants, namely, $RPLB_{frequency}$ and $RPLB_{degree}$. The experimental results obtained using the proposed algorithms are presented in section 4. Section 5 concludes the paper and gives directions for future work.

2. Related Works

In data management, data replication is used for creating and managing multiple copies of a file. The problem of data replication in data grid is studied by many researchers. This section discusses some of the important and recent data replication and placement strategies that are available in the literature. Nukarapu *et al*. [29] have proposed a centralized data replication strategy for data intensive scientific applications. The problem is formulated as a graph and a greedy algorithm is devised for efficient replication. The total running time of the algorithm is $O(pn)$, where p: Is the total number of data files, and n: Is the number of sites in the data grid. They have also, introduced the distributed caching concept which is based on the centralized replication algorithm in order to reduce work load of the server.

Six different replication strategies, namely, no replication or caching, best client, cascading replication, plain caching, caching plug, cascading replication and fast spread are proposed in [32]. Random access, temporal locality and geographical locality are the three access patterns used for evaluating these replication strategies. From simulation results, it is observed that fast spread strategy performs the best when random access is used. The cascading replication strategy works better in geographical and temporal locality. Chang and Chang [9] have proposed a dynamic data replication mechanism called Latest Access Largest Weight (LALW) in which all historical data are assigned weight based on their data access. The most recent accessed data has the largest weight.

Park *et al*. [30] have proposed a replication algorithm in data grid called Bandwidth Hierarchy Replication (BHR). It takes the advantage of network level locality, i.e., the required file is located in a site with high bandwidth. The BHR algorithm reduces data access time by avoiding network congestions in data grid network. It is implemented using tree level hierarchical structures. Sashi and Thanamani [35] have proposed modified BHR Region Based Algorithm to overcome the limitations of standard BHR algorithm. The modified BHR algorithm is improved to reduce data access time and to avoid unnecessary replication. It replicates files in a region and stores them in a frequently accessed site. The modified BHR algorithm increases data availability and reduces unnecessary replication [35].

An optimal replica placement algorithm is proposed in [16] to select the candidate sites where the replicas are to be placed. The Optimal Placement of Replicas (OPR) is implemented using dynamic programming to find optimal placement of k replicas in data grid systems. The OPR algorithm minimizes the read and storage cost. The time and space complexity at worst case is $O(nhk)$ where n is the number of nodes, h is the height of data grid tree and k is number of replicas. Tang *et al*. [37] have proposed two dynamic replication mechanisms, namely, Simple Bottom Up (SBU) and Aggregate Bottom Up (ABU) for multi-tier data grids. SBU strategy creates replicas when the predefined threshold of a data file is exceeded. Whereas, the ABU strategy creates replicas for frequently accessed data files. The performance of SBU and ABU are compared with fast spread strategy. It is observed that ABU is superior to SBU and fast spread in terms of average response time and average bandwidth cost.

Distributed Popularity Based Replica Placement (DPBRP) algorithm is proposed by Shorfuzzaman *et al*. [36] for allocating replicas in a hierarchical data grid with minimal replication cost. The replication cost is the sum of read cost of all the decendants of a node V and the update cost for the replica on V. The popularity of a file is calculated based on access history and two preset thresholds. DPBRP reduces both execution time and bandwidth consumption when compared to the algorithms [32].

The Data Replication Service (DRS) [13] and the Physics Experimental Data Export (PheDex) [33] are real time data replication implementations. DRS replicate files in their storage and register them in replica catalog. In PheDex, data is distributed in hierarchical manner and the data transfer is based on user subscription. Hong *et al*. [18] have proposed fast cascaded replication strategy. Replicas are created on the next level when the numbers of replica requests exceed a certain threshold. OptorSim [43] is used for simulation and the results obtained using the proposed strategies are compared with fast diffusion [32], LRU and Economy-zipf [6].

Three replica placement algorithms, namely, p-center, p-median and multi-objective model have been proposed by Rahman *et al.* [31]. The p-center is used to place replica in a grid site for minimizing the response time. But the p-median model places replica to the grid sites for optimizing the weighted average response time. Both p-center and p-median objectives are combined in multi-objective model to decide where to place a new replica. A replica maintenance algorithm for relocating replicas is also proposed in this paper. A distributed approximation algorithm called Distributed Greedy Replication (DGR) is proposed by Zaman and Grosu [46]. It provides a 2-approximation solution for a distributed replication group. DGR performs 97.28% better in all cases and provides a gain of up to 26.9%.

Xiong *et al*. [44] have proposed QoS-aware replica placement for data intensive applications. It has also proposed a replica placement algorithm based on dynamic programming to minimize replication cost, storage cost and communication cost. The QoS-aware placement [44] is compared with proportional placement [1] in terms of replication cost. The proposed replica placement algorithm outperforms proportional placement in [1].

A Dynamic Hierarchical Replication algorithm (DHR) is proposed by Mansouri and Dastghaibyfard [25]. It is the extension of the work done in [19]. DHR is suitable for small storage size grid sites. A modified version of [25] is proposed in [26]. It is a combination of Modified DHR Algorithm (MDHRA) and Combined Scheduling Strategy (CSS). Network traffic is reduced in [26]. An enhanced version of [25] is proposed in [27]. It is a combination of Weighted Scheduling Strategy (WSS) and Enhanced Dynamic Hierarchical Replication (EDHR). It improves file access time. OptorSim is used to test the proposed methodologies in [25, 26, 27]. Different data placement algorithms are discussed in [4, 22]. A detailed comparison chart is given in [22] to identify the best replica placement algorithm.

3. RPLB: Replica Placement with Load Balancing

In general large number of files is required to execute a job in a data grid. Data replication is crucial to increase data availability in data grid. In [30] grid sites that are located in the same region are grouped together. The popular files are replicated many times and stored in a site which has broad bandwidth. But in [35], if the requested file is not available in the local region, it is replicated in the most frequently accessed site. Chen *et al*. [10] have proposed a methodology for creating replicas based on highest degree and highest frequency. This method reduces makespan and storage consumption. Rasool *et al.* [34] have introduced a replica placement strategy to balance the work load and storage usage. It reduces both the parameters, number of replicas to be created and mean response time. Fair share replication strategy in [34] is implemented only in local grid not in InterGrid.

In this work, replica is placed at a region which contains the job running site. The site selection for replica placement is done based on highest degree and highest frequency [10] with balanced storage and load [34]. In this work, the grid network is represented as a graph for simulation. A graph, G=(V, E), is a set of vertices (V) and set of edges (E). V is a finite non-empty set that represents grid sites. E is a set of pair of vertices representing edges connecting vertices. These edges act as communication links across sites in the grid.

At first, network graph is split into regions by removing articulation points. Let G=(V, E) be a connected, undirected graph. An *articulation point* of G is a vertex whose removal disconnects G. A *bridge* of G is an edge whose removal disconnects G [41]. A graph is split into several regions by removing articulation points in it. A connected graph is biconnected if it has no articulation point. A biconnected component of G is a maximal set of edges such that any two edges in the set lie on a common simple cycle [41].

A bi-connected component of an undirected graph G= (V, E) is a maximal subset, B, of the edges with the property that the graph G_B=(V_B, B) is bi-connected, where V_B is the subset of vertices incident to edges in B [5]. Finding articulation point can be done by using Depth-First Search (DFS). In a DFS tree of an undirected graph, a node u is an articulation point, for every child v of u, if there is no back edge from v to a node higher in DFS tree than u. That is, every node in the decedent tree of u has no way to visit other nodes in the graph without passing through the node u, which is the articulation point [5, 41].

The proposed algorithm replicates files within a region if the frequency of requested files is greater than the average frequency access of all files in the same region [34]. In a region, the best node to place the replica is selected based on highest degree and highest frequency with balanced load. If the highest degree node and the highest frequency node are the same, then, this node is selected as the best node for replica placement. If the highest degree node and highest frequency node are different, then, the node with minimum cost is selected as the best node. The cost is calculated using Equation 1.

$$C_i = NH_i \times NR_i \qquad (1)$$

Where C_i: Is the cost of a node, NH_i: Is the number of hops to the requesting site, and NR_i: Is the number of Replicas in the node A site which has at least 20% free space may become a best node. If the prospective best nodes have maximum access and storage load, then, they may not be considered as the best node. The grid topology used for simulation is shown in Figure 4. When the user submits a job to the grid, all files which are not available in the site are transferred to the site by the Replica Manager (RM) before job execution [25].

3.1. Replica Decision

The request for a file is updated in the history table. If the frequency of requested file is greater than the average access frequency of all the files, then it is decided to replicate the required file.

3.2. Replica Selection

In general, several replicas of the same file are available in the network. The RPLB selects the replica with less number of requests as the best file for replication.

3.3. Replica Replacement

If the requested file is not available in the site but available in the local region, then, it is accessed remotely. If the requested file is not available in the local region and if there is no sufficient space to store the replica, then, older files are deleted using Least Recently Used Algorithm (LRU).

The complete RPLB algorithm is presented in Algorithms 1 and 2. Table 1 gives the details of the terminologies used in the algorithms.

Algorithm 1: RPLB replica placement.

```
for each file $f_i$ in the access history table, H (nodeid, fileid, frequency)
        if ($freq(f_i) >= freq_{avg}$)
            mark the file for $f_i$ replication
        end-if
end-for
//Select the best node to replica placement
for each file $f_i$ marked to replicate
{
   if replica of $f_i$ is not available in the local region
   {
    1. find all p nodes in local region who received
       the request for file $f_i$
    2. rank all p nodes in descending order of
       frequency $f_i$ and create list $L_1$
    3. rank all p nodes in descending order of degree
       $d_i$ and create list $L_2$
    4. remove the nodes have highest access load and
       storage load from the list $L_1$ and $L_2$
    5. Select $p_k$ and $q_k$ as the highest rank node in
       both $L_1$ and $L_2$
    6. if ($p_k = q_k$)
          $BN_i = p_k$
        else
          find the smallest cost nodes $Nc_f$ & $Nc_d$ from
          both $L_1$ and $L_2$
             if ($Nc_f < Nc_d$)
                 $BN_i = Nc_f$
             else
                 $BN_i = Nc_d$
             end-if
        end-if
    7. if more than one node have highest degree and
       highest frequency rank
    8. Select the $BN_i$ as the node with smallest cost
        if (Available_Space ($BN_i$)< Size ($f_i$) )
            Evacuate() using LRU
        end-if
   }
}
      9. Replicate ($f_i$ , $BN_i$)
```

Algorithm 2: Evacuate function.

```
// Delete the LRU files from the
//  best node $BN_i$
Evacuate($f_i$)// $f_i$ – The file to be placed in $BN_i$
{
  1. Create a list of all files, L, that are available in
       $BN_i$
  2. Sort the list, L, using LRU
  3. While(List, L, is not empty)
      {
        if(Not enough space available to place $f_i$ in
            $BN_i$ )
        {
          Delete LRU file from the
          list,
          L, in order to create space $f_i$ in $BN_i$
        }
           Place file $f_i$ in $BN_i$
      }
}
```

Table 1. Terminology used.

freq(f_i)	Access frequency of a file *i*
$freq_{avg}$	Average access frequency of all files
BN_i	Best Node *i*
Size(f_i)	Storage space required by file *i*
Nc_f	Communication cost of highest rank node in L_1
Nc_d	Communication cost of highest rank node in L_2

4. Experiments and Results

4.1. Simulation Tool

OptorSim [6] is a simulator used to evaluate the performance of the proposed replica placement strategy. It was developed by European Union (EU) data grid project team. The OptorSim architecture [4] is shown in Figure 1. It mainly consists of four components: Computing Elements (CEs), Storage Elements (SEs), Resource Broker (RB) and RM. The jobs submitted to the grid are scheduled to the computing elements by resource broker based on the policies of the selected scheduling algorithms such as random scheduling, access cost scheduling, queue access cost scheduling and shortest queue scheduling. The data files are stored in the SEs. The RM which is available in each site manages the data flow between sites and act as an interface between computing elements and SEs. It has a replica catalog to maintain replica location information. An optimizer which is available in the RM creates, selects and deletes replicas.

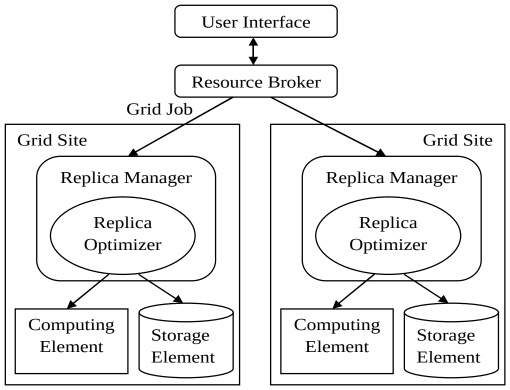

Figure 1. OptorSim architecture.

4.2. Configuration Files

The following four files are available in OptorSim: Parameter file, grid configuration file, job configuration file and bandwidth configuration file. Parameter file consists of parameters such as number of jobs, scheduling algorithm, queue length, etc., which are needed for simulation. Grid configuration file consists of grid topology and the contents of each site. Figure 2 shows the data grid model and the bandwidth configuration used in the proposed algorithm. The data grid model used in the simulation has 20 sites. Each site consists of 1000MB storage and a CE with a worker node. The simulation configuration parameters are shown in Table 2.

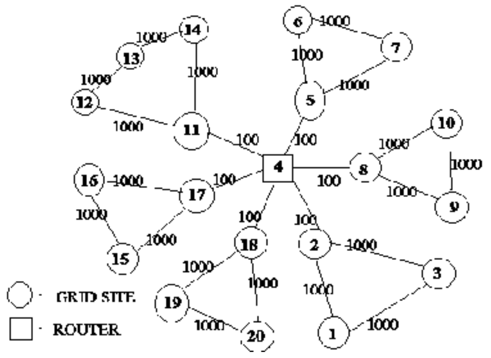

Figure 2. Data grid model.

Table 2. General configuration parameters.

Parameters	Value
Number of Sites	20
Number of Storage Elements (SEs)	18
Storage Capacity at each Site (MB)	1000
Number of Jobs (Maximum)	500
Number of Job types	6
Size of a Single File (MB)	100
Job Delay (ms)	2500
Maximum Queue size	100
Access Pattern	Sequential

4.3. Experimental Results

This section presents experimental results achieved using the proposed algorithm, namely, RPLB, RPLB$_{frequency}$ and RPLB$_{degree}$. The experimental results achieved using the proposed algorithm are also compared with that of the existing algorithms such as Always Replicate, Eco-zipf Optimizer, No Replication (Simple), BHR [30] and MBHR [35]. Figure 3 shows the storage utilization by various data replication algorithms when sequential access pattern is employed during job selection. The storage consumption is very less when no replication algorithm is used because the data files are stored in only one site. When compared to Always Replicate, Eco-zipf Optimizer and No Replication algorithms, the proposed RPLB algorithm performs better in terms of storage consumption. It should be noted that the variants of the proposed algorithms, namely, RPLB$_{frequency}$ and RPLB$_{degree}$ consume more memory when the number of jobs is 200 and 400. The results obtained for the parameter make span is presented in Figure 4. No replication strategy has the largest make span and the proposed RPLB algorithm has minimum make span. The make span is low using the proposed algorithm because; the replicas are placed in the local region and are selected from the best (nearest) site possible. Interestingly, the variants, RPLB$_{frquency}$ and RPLB$_{degree}$ also have less make span most of the time.

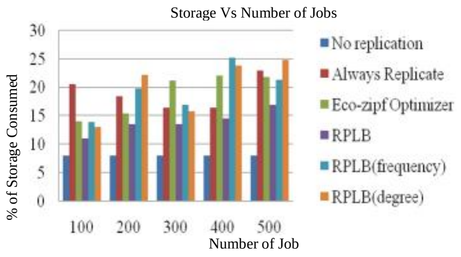

Figure 3. Storage utilization by various data replication algorithms.

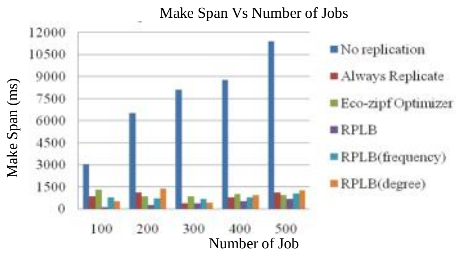

Figure 4. Make span using various data replication algorithms.

Figure 5 shows the number of replicas created by various data replication algorithms during job execution. The proposed RPLB algorithm has generated less number of replicas than the other algorithms. As the always replicate strategy, creates a new replica whenever a request is made, it is not suitable when the grid has limited storage constraint. But the proposed RPLB algorithm creates a new replica only when the file is not available in the local region.

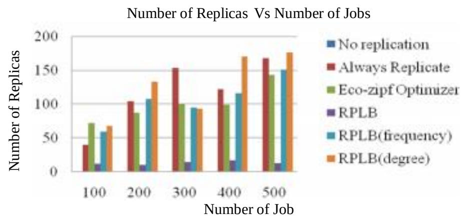

Figure 5. Number of replicas generated by various data replication algorithms.

We know that file replication takes extra time and little network bandwidth. The Effective Network Usage (ENU) ranges from 0 to 1. It is calculated using Equation 2 in [7].

$$ENU = \frac{N_{remote\_file\_accesses} + N_{file\_replication}}{N_{remote\_file\_accesses} + N_{local\_file\_replication}} \quad (2)$$

Where $N_{remot\_file\_accesses}$: Is the number of times the CE reads a file from a SE on a different site, $N_{file\_replication}$: Is the total number of file replications done during the job execution and $N_{local\_file\_replication}$: Is the number of times a CE reads a file from a SE on the same site.

The ENU is better in $RPLB_{frequency}$ and $RPLB_{degree}$ because the replica is present in the job running site. The proposed RPLB algorithm has less ENU compared to BHR algorithm. The ENU of different data replication algorithms is shown in Figure 6. Figure 7, clearly shows that the proposed RPLB algorithm takes less makespan compares to $RPLB_{frquency}$, $RPLB_{degree}$, BHR and MBHR. The storage space used for different data replication algorithms is depicted in Figure 8. When compared to the $RPLB_{frequency}$, $RPLB_{degree}$, BHR and MBHR algorithms, RPLB algorithm performs better. Configuration files are generated using a specific GUI designed, which is shown in Figure 9. The simulation results for 100 and 500 jobs are presented in Table 3.

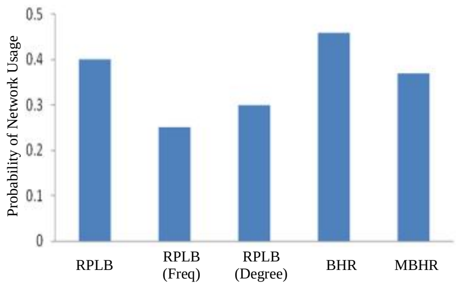

Figure 6. ENU for executing 100 jobs.

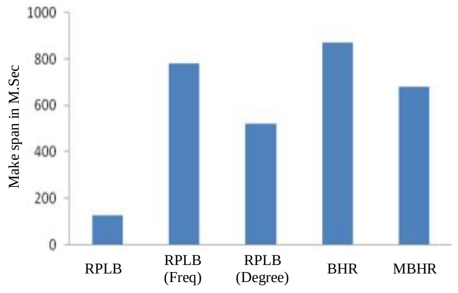

Figure 7. Make span for executing 100 jobs.

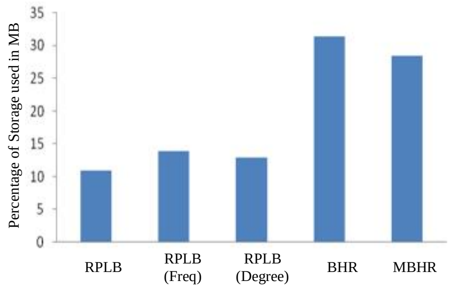

Figure 8. Storage used for executing 100 jobs.

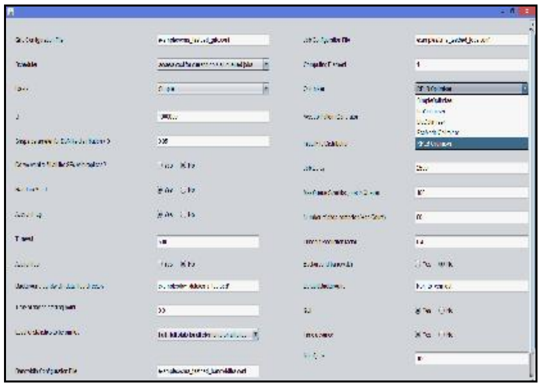
Figure 9. GUI for generating parameters configuration file.

Table 3. OptorSim results for 100 and 500 jobs.

Sl. No.	Replication Algorithms	Make Span		Percentage of Storage Used		No. of Replicas		ENU	
		100 jobs	500 jobs	100 jobs	500 jobs	100 jobs	500 jobs	100 jobs	500 jobs
1.	No Replication	3046	11418	8	8	0	0	0.946	0.774
2.	Always Replicate	834	1112	20.54	23	40	168	0.086	0.025
3.	Eco-Zipf Optimizer	1279	931	14.09	21.81	72	143	0.059	0.022
4.	RPLB	127	663	11.00	16.99	12	13	0.449	0.315
5.	$RPLB_{frequency}$	781	1023	13.90	21.36	59	151	0.054	0.024
6.	$RPLB_{degree}$	521	1269	13	24.81	68	176	0.043	0.038

5. Conclusions

Data replication is an efficient technique to increase data availability in data grid. Since, grid is dynamic in nature, the user behavior and network may change in time. The replica must be managed in terms of its creation, placement and deletion. In this paper, an efficient data replication algorithm RPLB and its two variants, namely, $RPLB_{frequency}$ and $RPLB_{degree}$ are proposed. Instead of storing replicas in many sites, they are stored in the best site to reduce storage consumption. The proposed RPLB algorithm generates less number of replicas and consumes less make span for executing grid jobs. The proposed data replication algorithm in this work can be combined with scheduling algorithm to achieve better performance in terms of number of replicas created and makespan.

Acknowledgement

The authors would like to thank the Management and Principal of Sri Ramakrishna Engineering College, and the Head of the Department of Computer Science and Engineering, for their support in completing this work.

References

[1] Abawajy J., "Placement of File Replicas in Data Grid Environments," in *Proceedings of the 4th International Conference on Cognitive Systems*, New Delhi, pp. 66-73, 2004.

[2] Allcock W., Bester J., Bresnahan, J., Chervenak A., Foster I., Kesselman C., Meder S., Nefedova

V., Quesna D., and Tuecke S., "Data Management and Transfer in High Performance Computational Grid Environments," *Parallel Computing Journal*, vol. 28, no. 3, pp. 749-771, 2002.

[3] Allcock W., Bester J., Bresnahan J., Chervenak A., Foster I., Kesselman C., Meder S., Nefedova V. Quesnel D., and Tuecke S., "Secure, Efficient Data Transport and Replica Management for High-Performance Data-Intensive Computing," available at: https://arxiv.org/ftp/cs/papers/0103/0103022.pdf, last visited 2001.

[4] Amjad T., Sher M., and Daud A., "A Survey of Dynamic Replication Strategies for Improving Data Availability in Data Grids," *Future Generation Computer Systems*, vol. 28, no. 2, pp. 337-349, 2012.

[5] Articulation Points Detection Algorithm., available at: http://www.ibluemojo.com/school/articul_algorithm.html, last visited 2012.

[6] Bell W., Cameron D., Capozza L., Millar A., Stockinger K., and Zini F., "OptorSim-a Grid Simulator for Studying Dynamic Data Replication Strategies," available at: https://dcameron.web.cern.ch/dcameron/talks/OptorSimIJHPCA2003.pdf, last visited 2003.

[7] Cameron D., Millar A., and Nicholson C., "OptorSim: A Simulation Tool for Scheduling and Replica Optimization in Data Grids," available at: http://vis.lbl.gov/~kurts/research/OptorSimCHEP2004.pdf, last visited 2004.

[8] Cibej U., Slivnik B., and Robic B., "The Complexity of Static Data Replication in Data Grids," *Parallel Computing*, vol. 31, no. 8, pp. 900-912, 2005.

[9] Chang R. and Chang H., "A Dynamic Data Replication Strategy Using Access-Weight in Data Grids," *Journal of Supercomputing*, vol. 45, no. 3, pp. 277-295, 2008.

[10] Chen D., Zhou S., Ren X., and Kong Q., "Methods for Replica Creation in Data Grids using Complex Network," *the Journal of China Universities of Posts and Telecommunications*, vol. 17, no. 4, pp. 110-115, 2010.

[11] Chervenak A., Deelman E., Foster I., Guy L., Hoschek W., Iamnitchi A., Kesselman C., Kunst P., Ripeanu M., Schwartzkopf B., Stockinger H., Stockinger B., and Tierney B., "Giggle: A Framework for Constructing Scalable Replica Location Services," *in Proceedings of ACM/IEEE 2002 Conference on Supercomputing*, pp. 58, 2002.

[12] Chervenak A., Foster I., Kesselman C., Salisbury C., and Tuecke S., "The data Grid: Towards Architecture for the Distributed Management and Analysis of Large Scientific Datasets," *Journal of Network and Computer Applications*, vol. 23, no. 3, pp. 187-200, 2001.

[13] Chervenak A., Schuler R., Kesselman C., Koranda S., and Moe B., "Wide Area Data Replication for Scientific Collaboration," *in Proceedings of the 6th IEEE/ACM International Workshop on Grid Computing*, Seattle, 2005.

[14] Chervenak A., Schuler R., Ripeanu M., Amer M. A, Bharathi S., Foster I., and Kesselman C., "The Globus Replica Location Service: Design and Experience," *IEEE Transaction on Parallel and Distributed Systems*, vol. 20, no. 9, pp. 1260-1272, 2009.

[15] Foster I. and Kesselman C., *The Grid: Blueprint for a New Computing Infrastructure*, Morgan Kaufmann, 1999.

[16] Garmehi M. and Mansouri Y., "Optimal Placement Replication on Data Grid Environments," *in Proceedings of the 10th International Conference on Information Technology*, pp. 190-195, 2007.

[17] Hanandeh F., Khazaaleh M., Ibrahim H., and Latip R., "CFS: A New Dynamic Replication Strategy for Data Grids," *The International Arab Journal of Information Technology*, vol. 9, no. 1, pp. 94-99, 2012.

[18] Hong L., Xue-dong Q., Xia L., Zhen L., and Wen-xing W., "Fast Cascading Replication Strategy for Data Grid," *in Proceedings of International Conference on Computer Science and Software Engineering*, Wuhan, pp. 186-189, 2008.

[19] Horri A., Sepahvand R., and Dastghaibyfard H., "A Hierarchical Scheduling and Replication Strategy," *International Journal of Computer Science and Network Security*, vol. 8, no. 8, pp. 30-35, 2008.

[20] Human Genome Project., available at: http://www.nhgri.nih.gov/, last visited 2013.

[21] Human Brain Project., available at: http://www-hbp.scripps.edu, last visited 2013.

[22] Kingsy G. and Manimegalai R., "Dynamic Replica Placement and Selection Strategies in Data Grids-A comprehensive Survey," *Journal of Parallel and Distributed Computing*, vol. 74, no. 2, pp. 2099-2108, 2014.

[23] Lamehamedi H., Shentu Z., Szymanski B. and Deelman E., "Simulation of Dynamic Data Replication Strategies in Data Grids," *in Proceedings of the 17th International Parallel and Distributed Symposium*, pp.100-102, 2003.

[24] Lee M., Leu F., and Chen Y., "PFRF: An Adaptive Data Replication Algorithm Based on Star Topology Data Grids," *Future Generation Computer Systems*, vol. 28, no. 7, pp. 1045-1057. 2011.

[25] Mansouri N. and Dastghaibyfard G., "A Dynamic Replica Management strategy in Data Grid," *Journal of Network and Computer Applications*, vol. 35, no. 4, pp. 1297-1303, 2012.

[26] Mansouri N., Dastghaibyfard G., and Mansouri E., "Combination of Data Replication and Scheduling Algorithm for Improving Data Availability in Data Grids," *Journal of Network and Computer Applications*, vol. 36, no. 2, pp. 711-722, 2013.

[27] Mansouri N., Dastghaibyfard G., and Mansouri E., "Enhanced Dynamic Hierarchical Replication and Weighted Scheduling Strategy in Data Grid," *Journal of Parallel Distributed Computing*, vol. 73, no. 4, pp. 534-543, 2013.

[28] Meroufel B. and Belalem G., "Managing Data Replication and Placement Based on Availability," *AASRI Conference on Parallel and Distributed Computing Systems, AASRI Procedia*, pp. 147-155, 2013.

[29] Nukarapu D., Tang B., Wang L., and Lu S., "Data Replication in Data Intensive Scientific Applications with Performance Guarantee," *IEEE Transactions on Parallel and Distributed Systems*, vol. 22, no. 8, pp. 1299-1306, 2011.

[30] Park S., Kim J., Ko Y., and Yoon W., "Dynamic Data Replication Strategy Based on Internet Hierarchy BHR," available at: https://pdfs.semanticscholar.org/6b32/05fc21cba 61f5aec0a0321f4095b85e420c0.pdf, last visited 2004.

[31] Rahman M., Barker K., and Alhajj R., "Replica Placement Strategies in Data Grid," *Journal of Grid Computing*, vol. 6, pp. 103-123, 2008.

[32] Ranganathan K. and Foster I., "Identifying Dynamic Replication Strategies for a High Performance of Data Grids," *in Proceedings of the 2nd international workshop on Grid Computing*, Berlin, pp. 75-86, 2005.

[33] Rehn J. Barrass T., Bonacorsi D., and Wu Y., "Phedex: High-Throughput Data Transfer Management System," *in Proceedings of Computing in High Energy and Nuclear Physics*, 2006.

[34] Rasool Q., Li J., Oresu G., and Munir E., "Fair-Share Replication in Data Grid," *Information Technology Journal*, vol. 7, no. 5, pp. 776-782, 2008.

[35] Sashi K. and Thanamani A., "Dynamic Replication in a Data Grid using Modified BHR Region Based Algorithm," *Future Generation Computer Systems*, vol. 27, no. 2, pp. 202-210, 2011.

[36] Shorfuzzaman M., Graham P. and Eskicioglu R., "Distributed Popularity Based Replica Placement in Data Grid Environments," *in Proceedings of International Conference on Parallel and Distributed Computing, Applications and Technologies*, Wuhan, pp. 66-77, 2010.

[37] Tang M., Lee B., and Yeo C., "Dynamic Replication Algorithm for the Multi-tier Data Grid," *Future Generation computer systems*, vol. 21, no. 5, pp. 775-790, 2005.

[38] Tatebe O., Morita Y., Matsuoka S., Soda N. and Sekiguchi S., "Grid Datafarm Architecture for Petascale Data Intensive Computing," *in Proceedings of 2nd IEEE/ACM International Symposium on Cluster Computing and the Grid*, pp. 102, 2002.

[39] The Large Hadron Collider (LHC)., available at: http://public.web.cern.ch/Public/en/LHC/LHC-en.html, last visited 2012.

[40] The Earth System Grid Project., available at: http://www.earthsystemgrid.org/, last visited 2013.

[41] Thomas H., Charles E., Leiserson, Ronald L. Rivest, Clifford Stein, *Introduction to Algorithms*, MIT Press Cambridge, 2001.

[42] Thuy N., Anh T., Thanh D., Tung D., Kien N., and Giang T., "Construction of a Data Grid for Meteorology in Vietnam," *in Proceedings of International Conference on Grid Computing and Applications*, pp. 186-191, 2007.

[43] Viger F. and Latapy M., "Efficient and Simple Generation of Random Simple Connected Graphs with Prescribed Degree Sequence," *in proceedings of the 11th Conference of Computing and Combinatoric*, pp 440-449, 2005..

[44] Xiong F., Xin-xin Z., Jing-yu H., and Ru-chuan W., "QoS-aware Replica Placement for Data Intensive Applications," *the Journal of China Universities of Posts and Telecommunications*, vol. 20, no. 3, pp. 43-47, 2013.

[45] Yuan Y., Wu Y., Yang G., and Yu F., "Dynamic Data Replication Based on Local Optimization Principle in Data Grid," *In Proceedings of GCC*, pp. 815-22, 2007.

[46] Zaman S. and Grosu D., "A Distributed Algorithm for the Replica Placement Problem," *IEEE Transactions on Parallel and Distributed Systems*, vol. 22, no. 9, pp. 1455-1468, 2011.

Kingsy Rajaretnam has graduated in Computer Science and Engineering from Noorul Islam College of Engineering, India, in 2003 and completed her M.E. Computer Science and Engineering in 2005 from Karunya Institute of Technology, India. Currently she is pursuing her Ph.D at Anna University, Chennai, India. Her areas of interest include Grid Computing and Cloud Computing. Her research focus is on Dynamic replica placement and selection strategies in data grid. She has about 10 years of teaching experience. She is currently working as an Assistant Professor in CSE, Sri Ramakrishna Engineering College, Coimbatore, India. She is life member of ISTE.

Manimegalai Rajkumar has graduated from PSG College of Technology in Computer Science and Engineering. She is also an alumnus of College of Engineering Guindy, Anna University and IIT Madras where she has done her Master's and Doctorate respectively. She has more than twenty years of experience in teaching, research and industry put-together. Currently she is working as a Professor and Research Director with Park College of Engineering and Technology, Coimbatore, India. She holds life membership in CSI, IE (India) and ISTE. She is also a member of IEEE and VLSI society of India. Her areas of interest include Reconfigurable Computing, VLSI/FPGA Algorithms, Distributed Systems and Cloud Computing. She has widely published in journals and conferences and is guiding several PhD research scholars.

Ranjith Venkatesan has completed his BE degree in Computer Science and Engineering at Sri Ramakrishna Engineering College, Coimbatore in 2013. Currently, he is working as Member Technical Staff in Zoho Corporation, Chennai, India. His areas of interests include grid computing, cloud computing and data base management systems.