Replica Maintenance Strategy for Data Grid

Mohammed K. Madi[1], Yuhanis Yusof[2], Hatim Mohamed Tahir[2], Khuzairi Mohd Zaini[2], Suhaidi Hassan[2]
[1]*Faculty of Engineering, Hasan Kalyoncu Üniversitesi, Gaziantep, Turkey.*
[2]*School of Computing, Universiti Utara Malaysia, 06010 Kedah, Malaysia.*
mohammed.uum@gmail.com

***Abstract*—Data Grid is an infrastructure that manages huge amount of data files, and provides intensive computational resources across geographically distributed collaboration. Increasing the performance of such system can be achieved by improving the overall resource usage, which includes network and storage resources. Improving network resource usage is achieved by good utilization of network bandwidth that is considered as an important factor affecting job execution time. Meanwhile, improving storage resource usage is achieved by good utilization of storage space usage. Data replication is one of the methods used to improve the performance of data access in distributed systems by replicating multiple copies of data files in the distributed sites. Having distributed the replicas to various locations, they need to be monitored. As a result of dynamic changes in the data grid environment, some of the replicas need to be relocated. In this paper we proposed a maintenance replica placement strategy termed as Unwanted Replica Deletion Strategy (URDS) as a part of Replica maintenance service. The main purpose of the proposed strategy is to find the placement of unwanted replicas to be deleted. OptorSim is used to evaluate the performance of the proposed strategy. The simulation results show that URDS requires less execution time and consumes less network usage and has a best utilization of storage space usage compared to existing approaches**.

***Index Terms*—Data Grid; Replica Deletion; Storage Usage.**

I. INTRODUCTION

A Data Grid [1, 2] is an infrastructure that deals with huge amounts of data to enable grid applications to share data files in a coordinated manner. Such an approach is seen to provide fast, reliable and transparent data access. Nevertheless, Data Grid creates a challenging problem in a grid environment because the volume of data to be shared is large despite the limited storage space and network bandwidth [3, 4]. Furthermore, resources involved are heterogeneous as they belong to different administrative domains in a distributed environment. It is unfeasible for all users to access a single instance of data (e.g. a data file) from one single organization (e.g. site). This would lead to the increase of data access latency. Furthermore, one single organization may not be able to handle such a huge volume of data by itself.

Motivated by these considerations, a common strategy is used in Data Grids as well as in distributed systems, and this strategy is known as replication.

Replication vouches efficient access without large bandwidth consumption and access latency [5-11]. The replication technique is one of the major factors affecting the performance of Data Grids [12]. Creating replicas can reroute client requests to certain replica sites and offer higher access speeds. Hence, well-defined replication strategies will smooth data access, and reduce job execution cost [13].

Dynamic replication is a long-term optimization technique which aims at reducing average job execution time in a Data Grid [14]. Data replication has two direct improvements on the performance of the Data Grid. One is to speed up data access, which leads to a shorter execution time of grid jobs; and the other one is to save bandwidth between sites, which can avoid network congestion with the sudden frequently required data. However, replication is also bounded by two factors: the size of storage available at different sites within the Data Grid and the bandwidth between these sites [15]. Furthermore, the files in a Data Grid are mostly large [16, 17]; so, replication to every site and hosting unlimited number of replicas would be unfeasible.

Due to the dynamic nature of data grids, the candidate site that holds replicas may currently not be the best sites to fetch replicas in subsequent periods [18]. Therefore, replica maintenance is needed to delete or relocate replicas to different sites if the performance metric degrades. Replica maintenance service comprises two functions: firstly, moving replicas to the appropriate location based on the information collected relating to some effect factors [19, 20]. Secondly, deleting unwanted replicas that got low demand and not used by users [21]. Replicas should be adjusted to the appropriate locations that are closer to the computing devices in order to adapt the current network environment to reduce time when the computing device accesses the data, as well as to maintain optimal performance of the network environment.

On the other hand, the network environment is changeable, which makes the same replica sites not always being the best choice to download data while reducing transmission time. Increasing storage space availability leads to decrease number of times the system invokes replacement strategy and reduce the processing time and improve the performance of other replication strategies. In this work, we propose a maintenance replica placement strategy termed as Unwanted Replica Deletion Strategy (URDS) as a part of Replica maintenance service. The main purpose of the proposed strategy is to find the placement of unwanted replicas to be deleted.

Current works [20, 21, 24, 25] interested in most valuable files (hot files), i.e. identifying placement of replicas to be created. However, the unwanted files are out of the considerations, i.e. identifying placement of replicas to be deleted. As a result, there will be an insufficient utilizing of storage resource space, which in turn will lead to less storage availability. According to [22, 23] less storage availability would lead to longer job execution time and larger network usage because only fewer replicas can be accommodated in the Data Grid, and most files will be read remotely. Moreover, the proliferations of data lead to think more in saving storage space because the storage media acts as an extra hardware and thus increase the total system cost.

The rest of this paper is structured as follows. Section 2 provides a brief description on existing work in dynamic replication strategies. We include the details of our proposed strategy in Section 3 and the performance evaluation is presented in Section 4. Finally, we summarize some conclusions in Section 5.

II. Related Works

The popularity of the file or the file value is used in two directions: the first direction is to trigger replica creation/deletion strategy. The second direction is to trigger replica replacement strategy, as the less valuable file is replaced by the most valuable file. The difference between replica deletion and replica replacement is that replica deletion is invoked before the replica replacement strategy where the files that have the minimum values are deleted.

The work in [26] suggested a model that helps to determine number of replicas needed to maintain the desired availability in P2P communities so that each site within the Data Grid is authorized to create replicas for the files. The availability of a file depends on the failure rate of peers in the network. A function has been developed to calculate the number of replicas needed for a certain availability threshold. However this model has disadvantages: firstly, the exact number of replicas is not determined; rather it depends on the location service accuracy which depends on the existing number of replicas. The accuracy of the replica location service determines the percentage of accessible files, and thus if the location service is ineffective, more replicas are created to ensure data availability. Secondly, the replica deletion mechanism is not considered, thus the storage cost may be increased.

In a different approach, the authors of [20] proposed a dynamic maintenance strategy called Dynamic Maintenance Service (DMS) to improve the performance of the grid environment. DMS decides where to place the replicas based on two main parameters: request frequency and free storage space. However, the replica deletion mechanism is not considered; rather the system does not locate the replica at a site unless there is enough space even if it brings benefit to system performance.

The authors in [27, 28] proposed a placement algorithm so that the workload of user requests among the replicas is balanced. The workload is defined as number of requests that a server satisfies. Given the data usage and maximum workload allowed for each replica server, they suggested algorithm can efficiently determine the minimum number of replicas required. On the other hand, the authors in [26] suggested an algorithm that provides a function that evaluates the placement of replica. The objective of this function is to maximize the difference between the replication benefits and replication cost (storage cost and transfer time). The benefit is the reduction in transfer time to the potential users, the storage cost is the storage cost at the remote site, and the transfer time is the duration from the current location to the new location. Yet again, the replica deletion mechanism was still not considered, thus the storage space cost may be increased.

III. Unwanted Replica Placement Strategy

In our previous work [29], we proposed a replica creation model that evaluates the files based on the exponential and dependency level of files in grid system. Each file in the system is evaluated and given a File Value (FV). The main goal of our previous works [29] was to decide on which replica to be created (wanted files) or deleted (unwanted files) and how many copies. Details on such approach can be seen in [29]. In this work, we are pursuing to identify sites that best to delete the replicas from. Thus we assume that the unwanted files already determined and we use their values in this work.

The Unwanted Replica Deletion Strategy (URDS) finds location sites to delete the less valuable replica or the redundant replicas, such that the total Read Cost (RC) is minimized, which is defined as [30] the cost of transferring data file from the underlying site to the remote sites. The best locations to delete the replica from are the sites that host the redundant replicas. In this context the redundant replica is the one that cost the maximum read cost. Hence, choosing the best location sites depends on four parameters: 1) File Transfer Time, 2) Read cost, 3) Sites' Workload and 4) Replication Sites.

i. File Transfer Time (FTT): FTT is the data transmission time, and depends on the size of the file and the current network bandwidth of the link between the underlying two sites. FTT is computed as following Equation [11]:

$$FTT = \frac{File\ Size}{Bandwidth\ h} \quad (1)$$

ii. Read Cost (RC): RC is the cost of transferring data file from the underlying site to the remote sites [30], and can be computed as:

$$RC = \frac{\sum_1^n FV_{si} * FTT}{m} \quad (2)$$

where:
n: The total number of the sites in the grid.
m: Number of sites that request the replica from the underlying site.
FV_{si}: The file value with respect to the specific site, which could be computed as:

$$FV_{si} = \frac{NOR_{si}}{File\ Value} \quad (3)$$

where:
NOR_{si}: Number of request for a file from a specific site.

The proposed strategy, namely URDS, combines the five parameters together in order to make the decision on the placement of redundant replicas, according to the following steps shown in Figure 1:

```
/* Unwanted Replica Deletion Strategy */
for each file in Unwanted_List
    List all sites that contain file_i
    for each site in list
        Calculate FV_si ← File Value / NOR_si
        Calculate FTT ← File Size / Bandwidt h
        Calculate RC ← (Σ_1^n FV_si × FTT) / m
        DescendSort [site_j]
    while (ENoR(file_i) ≠ 0)
    Delete file_i from site_j
        ENoR(file_i) + +
    Break;
```

Figure 1: Pseudo code of URDS

In order to understand the mechanism, consider the following example: suppose that we have eight sites and the bandwidth of the links between sites is shown in Figure 2. Suppose that new replicas of File1, File2, and File3 with size = 1000 MB, 400 MB, and 700 MB respectively need to be created by the system. File1 requires one replica, File2 two replicas, and File3 one replica. After computing the FTT of each file by applying equation 3, the graph become as shown in Figure 3, Figure 4, and Figure 5.

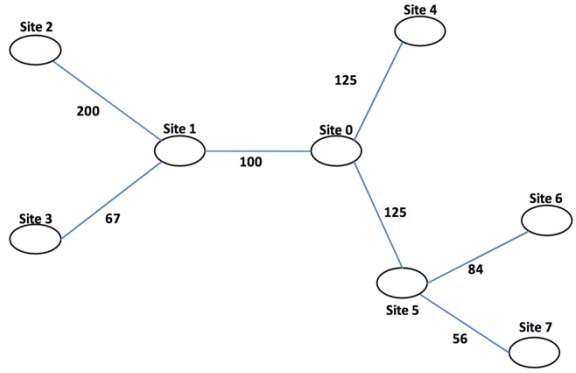

Figure 2: A grid network consists of eight sites and their links that represent the network bandwidth

Number of access from each site to the file is shown between the brackets. Suppose that site5 stores one replica of File1, site6 stores one replica of File2, and site1 stores one replica of File3.

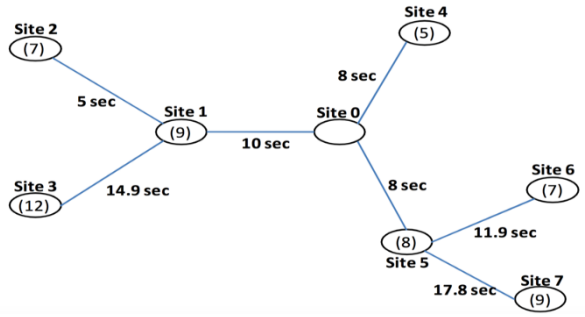

Figure 3: A grid network consists of eight sites and their links that represent the transfer time of File1

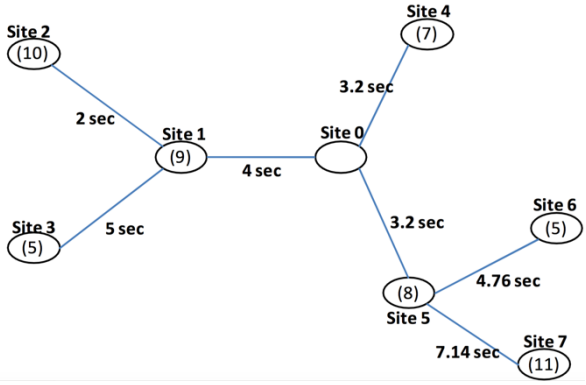

Figure 4: A grid network consists of eight sites and their links that represent the transfer time of File2

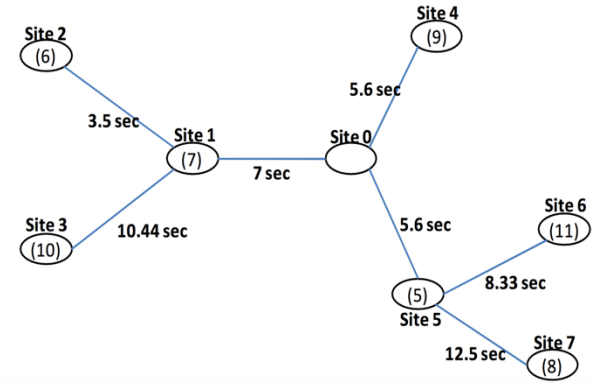

Figure 5: A grid network consists of eight sites and their links that represent the transfer time of File3

The RC for the sites that hosting File1 are computed by applying equation 2, as shown below:

$$\text{RC for site2(File1)} = \frac{9\times5 + 12\times19.9}{2} = 136.7$$

$$\text{RC for site1(File1)} = \frac{79\times14.9 + 7\times19.9}{2} = 141.41$$

$$\text{RC for site3(File1)} = \frac{7\times5 + 12\times19.9}{2} = 106.9$$

The RC for the hosting sites for File2 are computed by applying equation 2, as shown below:

$$\text{RC for site2(File2)} = \frac{9\times2 + 5\times7 + 7\times9.2}{3} = 31.8$$

$$\text{RC for site1(File2)} = \frac{10\times2 + 5\times5 + 7.2\times7}{3} = 39.1$$

$$\text{RC for site3(File2)} = \frac{9\times5 + 10\times7 + 12.2\times7}{3} = 63.4$$

$$\text{RC for site4(File2)} = \frac{10\times9.2 + 9\times12.2 + 9\times7.2}{3} = 72.6\,3$$

The RC for the candidate sites for File3 are computed by applying equation 2, as shown below:

$$\text{RC for site5(File3)} = \frac{9\times11.2 + 11\times8.33 + (8\times12.5)}{3} = 97.4$$

$$\text{RC for site6(File3)} = \frac{5\times8.33 + 8\times20.83}{2} = 104.1\,2$$

$$\text{RC for site7(File3)} = \frac{5\times12.5 + 11\times20.83}{3} = 145.8$$

Thus, the RC for each site that hosting the redundant replica can be tabulated, as shown in Table 1. The intersection of the Sites in the rows and the Files in the column is the RC for the site of a certain file.

Table 1
RC of three unwanted files for eight sites

	File1	File2	File3
$Site_0$			
$Site_1$	136.7	31.8	
$Site_2$	141.41	39.1	
$Site_3$	106.9	63.4	
$Site_4$		72.6	
$Site_5$			97.4
$Site_6$			104.1
$Site_7$			145.8

According to the information in Table 1, the implemented mechanism will decide to place File1 in Site3, as it has the minimum RC, two copies of File2 in Site1 and Site2, and one copy of File3 in Site5.

IV. Performance Evaluation

A Java-based data grid simulator called OptorSim was developed by the European Data Grid Project (EDG project) [31]. OptorSim provides a framework to simulate real-world data grids by considering an array of parameters and scenarios found in reality conclusion.

A. Simulation Setup

The study of RPS was carried out using the model of LALW DataGrid32 sites and their associated network geometry shown in Figure 6. It comprises of 12 grid sites, each site has a storage capacity of 50 GB, while Site8 has 100 GB to hold all the original files. This configuration has four clusters and each one has three sites. One site has the most capacity in order to hold all the master files at the beginning of the simulation. The others have a uniform size of 50GB. The network bandwidth is set as 100 Mbit/sec, while the connection bandwidth is 100 Mbps. We ran the simulation with 500 jobs.

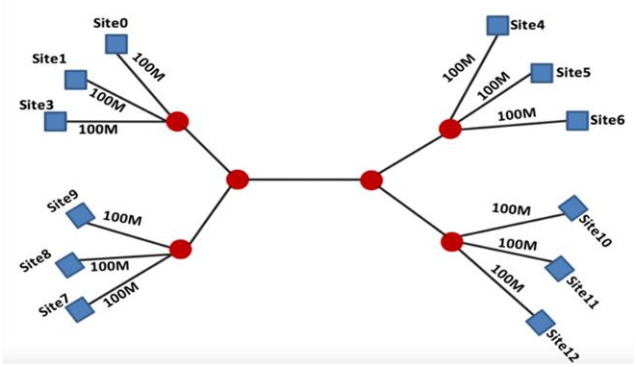

Figure 6: LALW Test bed Sites and their associated network geometry

A job is submitted to Resource Broker every 25 second. Resource Broker then submits to Computing Element according to an QAC scheduling algorithm. There are 6 job types, and each job type requires specific files for execution. The order of files accessed in a job is sequential and is set in the job configuration file as an input to the simulation. The number of files in our simulation is 150, and a file size is 1 GB.

B. Evaluation Metrics

The performance metrics we chose to evaluate the proposed system are: Mean Job Execution Time (MJET), Efficient Network Usage (ENU), and Average Storage Usage (ASU). MJET is the average time a job takes to execute; from the moment it is scheduled to Computing Element to the moment when it has finished processing all the required files. ENU is defined as a measure of how well the replication strategy uses the network [3, 4], A lower value indicates that the utilization of network bandwidth is more efficient. ASU is the percentage of capacity reserved by files according to the total capacity for the underlying storage.

V. Results and Discussion

The proposed model (URDS) is compared against LALW algorithm [32] and other existing algorithms that are employed in OptorSim (LFU, LRU, and Economy algorithm) [33, 34] that have been mentioned in details in Section 2 of this paper. The results of our simulation are shown in Table 2

Table 2
Simulation results of URDS and other existing mechanisms

# Jobs	Metrics	LRU	LFU	Economy	LALW	URDS
500	MJET	4358	4154	4814	4013	3448
	ENU	47.13	47.41	36.72	33.11	31.95
	ASU	34.52	36.63	36.78	31.91	29.52

A better algorithm is the algorithm that has less MJET. As shown in Figure 7 and Figure 8, URDS performs the best among the compared existing algorithms. URDS consumes 16.05% less MJET compared to LALW,30.02% over Economy, 18.89% over LFU, and 23.16% over LRU. This is due to the replication decision that has been made by URDS, in which decides to replicate a group of valuable files at the same time (i.e. in one decision). As a result, replicas of popular files are spread in grid and increase the availability. On the other hand, LALW that replicates only one popular file at one decision. In addition, URDS offers the decision includes the deletion of unwanted replicas, which helps in providing a free space in the storage element of sites. As a result, the need for invoking the replacement strategy will be decreased (save the time spent in determining the victim file). Thus, the replication process will be hastened, and spread the replicas as fast as possible.

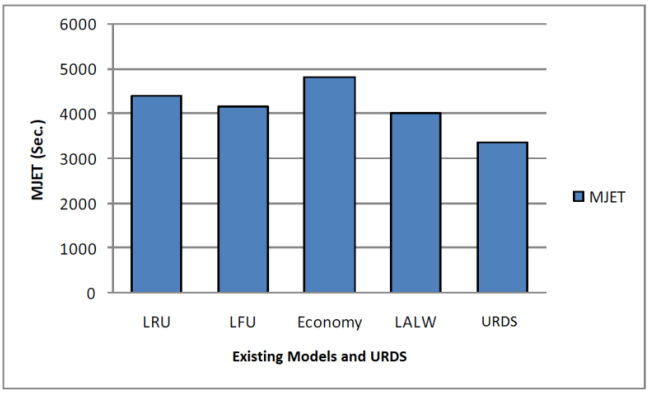

Figure 7: The MJET of URDS and existing Algorithms

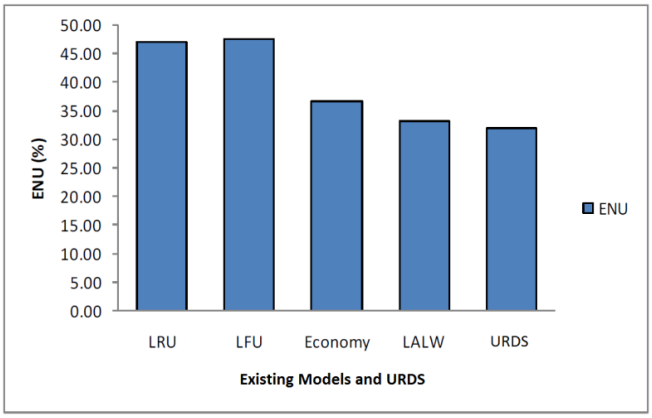

Figure 8: The ENU of URDS and existing Algorithms

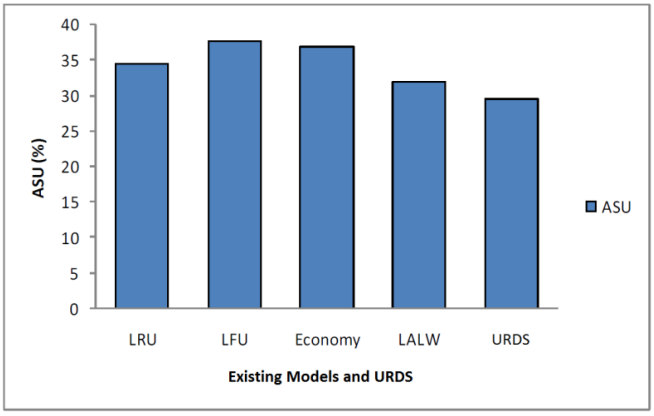

Figure 9: The ASU of URDS and existing Algorithms

The effect of replica deletion process on ASU is quite pronounced, as shown in Figure 9, URDS uses the least amount of storage usage (ASU) by outperforming LALW by 7.49%.

VI. CONCLUSIONS

This study describes the replica deletion services as a part of replication management in Data Grid. In this context, replica deletion service includes determining the place of redundant replicas to be deleted. The key advantage of the proposed strategy is that it introduces a deletion function that removes unwanted replicas from the system. Thus, increase the storage space availability. Increasing storage space availability leads to decrease number of times the system invokes replacement strategy and reduce the processing time and improve the performance of other replication strategies.

As a future work, this work could be further improved and extended in several aspects, such that relocating the replicas to location sites that provide better services in the context of the current situation and network conditions. Furthermore, the throughput and system performance could be exposed by running simulation in different scenarios.

REFERENCES

[1] A. Chervenak, E. Deelman, C. Kesselman, B. Allcock, I. Foster, V. Nefedova, J. Lee, A. Sim, A. Shoshani, and B. Drach, "High-performance remote access to climate simulation data: A challenge problem for data grid technologies," in Super Computing, 2003, 1335-1356.

[2] I. Foster, E. Alpert, A. Chervenak, B. Drach, C. Kesselman, V. Nefedova, D. Middleton, A. Shoshani, A. Sim, and D. Williams, "The Earth System Grid II: Turning climate datasets into community resources," in Annual Meeting of the American Meteorological Society, 2002.

[3] B. Wilkinson, Grid computing: techniques and applications: Chapman & Hall/CRC, 2009.

[4] C. Nicholson, D. G. Cameron, A. T. Doyle, A. P. Millar, and K. Stockinger, "Dynamic data replication in lcg 2008," Concurrency and Computation: Practice and Experience, 20, 1259-1271, 2008.

[5] A. Chervenak, E. Deelman, I. Foster, W. Hoschek, A. Iamnitchi, C. Kesselman, M. Ripeanu, B. Schwartzkopf, H. Stockinger, and B. Tierney, "Giggle: A framework for constructing scalable replica location services," in International IEEE Supercomputing Conference (SC 2002) Baltimore, USA, 2002, 1-17.

[6] A. Chervenak, I. Foster, C. Kesselman, C. Salisbury, and S. Tuecke., "The Data Grid: Towards an Architecture for the Distributed Management and Analysis of Large Scientific Datasets," Journal of Network and Computer Applications, 23, 2001.

[7] L. Guy, P. Kunszt, E. Laure, H. Stockinger, and K. Stockinger, "Replica management in data grids," in Global Grid Forum. 5, 2002.

[8] H. Lamehamedi, Z. Shentu, B. Szymanski, and E. Deelman, "Simulation of dynamic data replication strategies in data grids," in Proceedings of 12th Heterogeneous Computing Workshop (HCW2003), Nice, France, 2003.

[9] H. Lamehamedi, B. Szymanski, Z. Shentu, and E. Deelman, "Data Replication Strategies in Grid Environments," in Fifth International Conference on Algorithms and Architectures for Parallel Processing, 2002, 378.

[10] E. Otoo, F. Olken, and A. Shoshani, "Disk cache replacement algorithm for storage resource managers in data grids," in 2002 ACM/IEEE conference on Supercomputing, Baltimore, Maryland 2002, 1-15.

[11] K. Ranganathan and I. Foster, "Identifying Dynamic Replication Strategies for a High-Performance Data Grid," International Grid Computing Workshop, 75-86, 2001.

[12] X. You, G. Chang, X. Chen, C. Tian, and C. Zhu, "Utility-Based Replication Strategies in Data Grids," in Fifth International Conference on Grid and Cooperative Computing, 2006, 500-507.

[13] M. Tang, B. S. Lee, X. Tang, and C. K. Yeo, "The impact of data replication on job scheduling performance in the Data Grid," Future Generation Computer Systems, 22, 254-268, 2006.

[14] S. M. Park, J. H. Kim, Y. B. Ko, and W. S. Yoon, "Dynamic data grid replication strategy based on Internet hierarchy," International Workshop on Grid and Cooperative Computing, 1001, 1324–1331, 2004.

[15] S. Venugopal, R. Buyya, and K. Ramamohanarao, "A taxonomy of data grids for distributed data sharing, management, and processing," ACM Computing Surveys (CSUR), 38, 2006.

[16] R. M. Rahman, K. Barker, and R. Alhajj, "Replica placement strategies in data grid," Journal of Grid Computing, 6, 103-123, 2008.

[17] R. M. Rahman, K. Barker, and R. Alhajj, "Performance evaluation of different replica placement algorithms," International Journal of Grid and Utility Computing, 1, 121-133, 2009.

[18] M. R. Rahman, "Replica placement and selection strategies in data grids," in Department of Computer Science. vol. PhD. thesis Alberta: University of Calgary, 2007.

[19] C. T. Yang, C. J. Huang, and T. C. Hsiao, "A Data Grid File Replication Maintenance Strategy Using Bayesian Networks," in Intelligent Systems Design and Applications, 2008. ISDA'08, 2008.

[20] C. T. Yang, C. P. Fu, and C. J. Huang, "A dynamic file replication strategy in data grids," in TENCON 2007-2007 IEEE Region 10 Conference, 2007, 1-5.

[21] Saleh, A., Javidan, R., and FatehiKhajeh, M. T., "A four-phase data replication algorithm for data grid", Journal of Advanced Computer Science & Technology, 4(1), 163-174, 2015.

[22] Tos, U., Mokadem, R., Hameurlain, A., Ayav, T., and Bora, S. "Dynamic replication strategies in data grid systems: a survey" The Journal of Supercomputing, 71(11), 2015, 4116-4140.

[23] David G. Cameron, "Replica management and optimisation for data grids," PhD. Thesis, University of Glasgow, 2005.

[24] Rahmani, A.M., Fadaie, Z. and Chronopoulos, A.T., "Data placement using Dewey Encoding in a hierarchical data grid", Journal of Network and Computer Applications, 49, 2015. 88-98.

[25] Grace, R.K. and Manimegalai, R., "HGASA: An Efficient Hybrid Technique for Optimizing Data Access in Dynamic Data Grid". In Distributed Computing and Internet Technology, 132-136, 2016. Springer International Publishing.

[26] K. Ranganathan, A. Iamnitchi, and I. Foster, "Improving data availability through dynamic model-driven replication in large peer-to-peer communities," in Global and Peer-to-Peer Computing on Large Scale Distributed Systems Workshop, 2002, 376–381.

[27] L. Yi-Fang, L. Pangfeng, and W. Jan-Jan, "Optimal placement of replicas in data grid environments with locality assurance," in Parallel and Distributed Systems, 2006. ICPADS 2006. 12th International Conference on, 2006,

[28] Y. F. Lin, J. J. Wu, and P. Liu, "A List-Based Strategy for Optimal Replica Placement in Data Grid Systems," in *Proceedings of Parallel Processing*, 2008. ICPP'08. 37th *International Conference on*, 2008, 198-205.

[29] M.K. Madi, H.M. Tahir, Y. Yusof, and S. Hassan, S., "A novel dynamic replica creation mechanism for Data Grids". In Game Physics and Mechanics International Conference (GAMEPEC), 2015, 1-5. IEEE.

[30] Y. Mansouri, M. Garmehi, M. Sargolzaei, and M. Shadi, "Optimal Number of Replicas in Data Grid Environment," in First International Conference on Distributed Framework and Applications, 2008., 96-101.

[31] The European Data Grid Project. http://eudatagrid.web.cern.ch/eu-datagrid

[32] C. Ruay-Shiung, C. Hui-Ping, and W. Yun-Ting, "A dynamic weighted data replication strategy in data grids," in AICCSA 2008: Proceedings of IEEE/ACS International Conference on computer systems and applications, 2008, 414-421.

[33] D. G. Cameron, R. Carvajal-Schiaffino, A. P. Millar, C. Nicholson, K. Stockinger, and F. Zini, "Evaluating scheduling and replica optimisation strategies in OptorSim," Journal of Grid Computing, 57-69, March 2004.

[34] W. H. Bell, D. G. Cameron, L. Capozza, P. Millar, K. Stockinger, and F. Zini, "Simulation of Dynamic Grid Replication Strategies in OptorSim," Journal of High Performance Computing Applications, 17, 2003.