

# A GA-Based Replica Placement Mechanism for Data Grid

Omar Almomani

Department of Network and Computer Information System  
Faculty of Informtion Technology  
The World Islamic Sciences & Education University, Jordan

Mohammad Madi

School of Computing  
College of Arts and Sciences  
Universiti Utara Malaysia, 06010 Sintok, Kedah

**Abstract**—Data Grid is an infrastructure that manages huge amount of data files, and provides intensive computational resources across geographically distributed collaboration. To increase resource availability and to ease resource sharing in such environment, there is a need for replication services. 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. Replica placement mechanism is the process of identifying where to place copies of replicated data files in a Grid system. Choosing the best location is not an easy task. Current works find the best location based on number of requests and read cost of a certain file. As a result, a large bandwidth is consumed and increases the computational time. Authors proposed a GA-Based Replica Placement Mechanism (DBRPM) that finds the best locations to store replicas based on five criteria, namely, 1) Read Cost, 2) Storage Cost, 3) Sites' Workload, and 4) Replication Site.

**Keywords**—Data Grid; Data replication; distributed systems; Replica placement mechanism; GA-Based Replica Placement Mechanism

## I. INTRODUCTION

Data Grids [1, 2] is an infrastructure that deals with huge amount 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, the approach is considered as a challenging problem in grid environment because the volume of data to be shared is large despite of limited storage space and network bandwidth. Furthermore, resources involved are heterogeneous as they belong to different administrative domains in a distributed environment.

However, 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 is known as replication. Replication vouches the efficient access without large bandwidth consumption and access latency [3-9]. Replication technique is one of the major factors affecting the performance of data grids [10]. Creating replicas can reroute a client requests to certain replica sites and offer a higher access speed [11].

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 [12]. Furthermore, the files in Data Grid are mostly large [13, 14]; so, replication to every site is infeasible. Therefore deciding on the optimal locations to host a certain popular files is needed, in order to reduce the bandwidth consumption of the network. In this paper a GA-Based Replica Placement Mechanism (GARPM) propose by which the process of placing files in grid sites can be done in optimal or near-optimal manner. Authors present an adaptive genetic algorithm that solves the replica placement problem in data grid. The proposed mechanism considered as a long-term optimization technique that has two direct improvements on the performance of data grid. One is to optimize data access which leads to shorter execution time by considering the read cost of files; and the other one is to optimize the network bandwidth, which can avoid network congestion with the sudden frequently required data by considering workload of grid sites and distribution of current replicas.

The GARPM addresses the problems of current replication mechanisms which could be epitomized in two points:

A large amount of network bandwidth is consumed resulting from a bad utilization of the network by the existing systems [11, 15-22]. As a result of bad utilization of network bandwidth will lead to increasing of the job execution time [17, 23-27]. The proposed work is expected to minimize network bandwidth consumption and reduce job execution time. The rest of this paper is structured as follows. Section 2 provides a brief description on existing work in replica placement mechanisms. Authors include details of our proposed replication mechanism in Section 3 and provide a numerical example that explains how the proposed mechanism works in Section 4. Finally, conclude the paper in Section 5.

## II. RELATED WORKS

There are many studies in the literature that concern replica placements issues. Chin-Min Wan et al. [19] proposed a replica placement scheme that tries to overcome the bottleneck caused by increasing the downlinks, which are occurring at the same time. The proposed strategy chooses the best site to host the replica according to the evaluation result based on the number of user request and transmission cost.

The purpose of the strategy is to replicate the file to a site that provides minimum average transmission cost. Transmission cost is defined to be inversely proportional to bandwidth, and the site that provides the minimum average transmission cost is selected.

Following the bandwidth aspect, [28] proposed a dynamic replication strategy, called Bandwidth Hierarchy based Replication (BHR) to reduce access time by avoiding network congestion. BHR reduces the time taken to access and transfer the file. It places a replica at a high bandwidth location. However, such an approach only considers transmission cost and does not guarantee to minimize the overall cost.

A load balancing replication strategy has been proposed by [21], where the most frequently accessed file is placed closest to the users and the decision of replica placement is made based on the access load and the storage load of the candidate replica servers and their sibling nodes. In relation to this, [29] discussed various replication strategies namely; MinimizeExpectedUtil, MaximizeTimeDiffUtil, MinimizeMaxRisk, and MinimizeMaxAvgRisk while considering the utility and risk indexes, and making the replica placement decision by optimizing the average response time. They concluded that considering both current network state and file requests are better than considering the file requests alone.

Meanwhile, the work on dynamic replication algorithm by [22] had resulted in a Popularity Based Replica Placement (PBRP) algorithm for hierarchical Data Grids. The idea behind PBRP is to place replicas as close as possible to those clients that frequently request data files. Further work by [30] presented a dynamic replica placement in multi-tier Data Grid that categorized the files based on their access frequency into two groups: 1) Most Frequent Files (MFF) that are replicated and placed at the parent node of their respective best clients, where the best client for a file is a client which generates the maximum request for that file, and 2) Least Frequent Files (LFF) that are placed at one tier below the root of the Data Grid along the path of their best client. In [31], a dynamic placement algorithm was proposed that takes into account the dynamicity of sites in the Data Grid, since a site can at any time leave the grid and possibly join again later. Thus, two parameters were investigated: the request number for each file by each site, and utility of each site that involves the number of times the site did not answer to a file request due to its absence from the grid.

On the other hand, the authors in [23] suggested a model 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, site workload is not considered, thus the system will not guarantee to perform well with increasing of running jobs.

Ruay-Shing et al. [17] proposed a dynamic replication mechanism that replicates a popular file to suitable site according to the access frequencies for each file that has been requested. Access frequency is an essential parameter that should be taken into account when determining replica placement. However, some important parameters such as overall cost (i.e. storage cost and read cost), distance and availability should not be neglected; otherwise the overall system performance is degraded.

### III. REPLICA PLACEMENT STRATEGY

In previous work [32], authors 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 work [32] was to identify file that need to be replicated (also known as popular files). Details on such approach can be seen in [32]. In this work, we are pursuing to identify sites that best to host the newly created replicas. Thus assume that the popular file already determined and authors use their values in this work

The GA-Based Replica Placement Mechanism (GARPM) finds location sites to place the newly created replicas, such that the total Read Cost (RC) is minimized, which is defined as [26] the cost of transferring data file from the underlying site to the remote sites. The best locations are the sites that provide the best service to all other sites and users in the grid system. In users' perspective, the best sites are located as close as possible to the sites that most potentially request the underlying replicas. This improves the geographical locality of the sites, which consider files that requested by the sites are likely to be requested by nearby sites [33]. However, in sites' perspective the best sites are located as far as possible from the replication sites that never request the underlying replicas. Hence, choosing the best location sites depends on four parameters: 1) Storage cost, 2) Read cost, 3) Sites' Workload, and 4) Replication Sites.

1) *Storage Cost (SC)*: RC is the cost of storing a file at a certain site [23-26, 34]. The storage cost might reflect the size of the file, the throughput of the site, or the fact that a copy of the file is residing at a specific site. In this context the storage cost is the storage space used to store data, and can be computed as following equation [33]:

$$SC = \frac{\text{File Size}}{\text{Free Space}} \quad (1)$$

Where,

Free Space: is the current available space of the underlying storage site

2) *Read Cost (RC)*: RC is the cost of transferring data file from the underlying site to the remote sites [26], and can be computed as:

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

Where,

$n$ : The total number of the sites in the grid.

$m$ : Number of sites that request the replica from the underlying site.

$FV_{s_i}$ : The file value with respect to the specific site  $s_i$ , which could be computed as:

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

Where,

$NOR_{s_i}$ : Number of request for a file from a specific site  $s_i$

$FTT$ : is the data transmission time, and depends on the size of the file and the current network bandwidth of the link

between the two underlying sites. FTT is computed as in the following equation [26]:

$$FTT = \frac{\text{File Size}}{\text{Bandwidth}} \quad (4)$$

3) *Sites' Workload*: The workload of the site is defined as the number of request that can be satisfied by the underlying site [24, 35]. The candidate site should not exceed a specific amount of workload that is assigned to it.

4) *Replication Sites*: Replication site is the site that is hosting the replica of the underlying file. Replication site influence the candidate sites. The candidate site should be located as far as possible from the replication sites, because of two main reasons: 1) the replication sites itself never request a replica that is already stored on it, 2) the load need to be distributed.

The proposed strategy, namely GARPM, combines the four parameters together in order to make the decision on the placement of replicas, according to the following steps:

1) Calculate the storage cost of the popular file by applying equation 1;

2) Calculate the transfer time of the popular file by applying equation 4;

3) Identify the sites that could be excluded from being candidates sites to hold the replicas, and those sites have the following characteristics:

a) already stored the replicas in their storage elements (Replication Sites),

b) already exceeded their maximum workload, and

c) have a direct connection to replication sites;

4) Calculate the RC of each candidate site by applying equation 2;

5) Up to this step, we are given the number of copies to be created of a popular file, and a set of candidate sites with associated read cost. Our goal then to fine the best sites to host the certain number of copies, so as to optimize the total read cost.

#### IV. GA-BASED ALGORITHM

Genetic algorithms (GA) are an evolutionary optimization approach which is an alternative to traditional optimization methods [36]. The effectiveness or quality of a GA (for a particular problem) can be judged by its performance against other known techniques – in terms of solutions found, and time and resources used to find the solutions [37]. moreover, GA has shown itself to be extremely effective in problems ranging from optimizations to machine learning [38]. An important advantage of GA is that they search for the optimal solution by examining only the overall all valuation of a solution; they require no specific problem related information for their search. i.e. it is a blind search [39].

In general GA search strategy consists of the following steps:

1) *Generate initial population (Initialization)*: generate random population of  $n$  chromosomes

2) *Evaluate fitness*: evaluate the fitness of each chromosome in population

3) *Create new population*: create a new population by repeating the following steps until the new population is complete:

a) *Select two parent chromosomes from the population according to their fitness ( the better fitness the bigger chance to be selected)*

b) *Crossover the parents to form a new offspring (children)*

c) *Mutate new offspring at each locus*

d) *Place the new offspring in the new population*

4) *Replace*: use the new generated population for further run of algorithm

5) *Test*: if the end condition is satisfied, stop and return the best solution in the current population

6) *Loop*: go to step 2.

GA begins with an initial population represented by chromosomes. Chromosome is a set of solutions from one population. It can be taken and In general when apply the GA replica placement problem, the algorithm will works as following: at the first we start with a random initial population  $P_0$ .  $P_0 = [k_1, k_2, k_3, \dots, k_n]$

The size of initial population is  $n$  chromosomes. Each chromosome  $s_i$  of this population consists of  $n$  binary bits or (sites).

$$k_i = [s_1, s_2, s_3, \dots, s_n] \text{ where } s_i \in \{0,1\}$$

Therefore each bit (site) of a chromosome can be either included ( $s_i = 1$ ) or excluded ( $s_i = 0$ ) from being a candidate to host one replica. Number of bits in each chromosome has to be same as number of sites in the grid system, as each bit represent one site. Moreover, number of ones in each chromosome must be equals to number of copies that are created of the popular file. Example of possible initial population is as follows.

$$\begin{bmatrix} k_1 = [1 \ 1 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 1 \ 1 \ 0 \ 0 \ 0 \ 0 \ 1] \\ k_2 = [0 \ 0 \ 0 \ 1 \ 0 \ 1 \ 0 \ 0 \ 0 \ 0 \ 1 \ 0 \ 1 \ 1 \ 0] \\ k_3 = [1 \ 0 \ 1 \ 1 \ 0 \ 0 \ 1 \ 1 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0] \\ \vdots \\ k_n = [0 \ 0 \ 0 \ 1 \ 0 \ 0 \ 0 \ 1 \ 1 \ 1 \ 0 \ 1 \ 0 \ 0 \ 0] \end{bmatrix}$$

From the above example, by looking at the chromosomes it clearly seen that the total number of sites is 15, and number of copies to be hosted is five copies. For instance, the first chromosome ( $k_1$ ) indicates that

$site_1, site_2, site_9, site_{10},$  and  $site_{15}$

have been selected to host the five replicas of the popular file.

After the initial population is generated randomly, the fitness value of each chromosome is evaluated by using objective function or cost function. In our case the cost function represented by the Overall Cost (OC) of sites, therefore the objective is to minimize the total OC. So, the lower the total OC, the fitter the solution represented by that chromosome is.

The value of fitness function is given by the following equation:

$$\sum_{i=1}^n RC(site_i) + SC(site_i) \quad (5)$$

Where,  $n$  is the total number of sites.

For example, the fitness value of the first chromosome could be calculated by summing the total OC of candidate sites that represented by 1 in the chromosome. In other words,  $fitness(k_1) = OC(s_1) + OC(s_2) + OC(s_9) + OC(s_{10}) + OC(s_{15})$

Assume that OC of  $site_1, site_2, site_9, site_{10},$  and  $site_{15}$  are 20, 50, 44, 32, and 60 respectively, so the  $fitness(k_1) = 20 + 50 + 44 + 32 + 60 = 206$ . The same goes for the rest of chromosomes.

Having calculated the fitness value of the population, the next generation can be determined. Select chromosomes for reproduction, more fit chromosomes are more likely to be selected for reproduction. For selection, the Roulette Wheel selection used, where fitness level is used to associate a probability of selection with each chromosome. The roulette wheel selection scheme can be implemented as follows:

- Evaluate the fitness,  $fitness(k_i)$ , of each chromosome in population
- Compute the probability,  $(P_i)$ , of selection each member of the population:  $P_i = \frac{fitness(k_i)}{\sum_{j=1}^n fitness(k_j)}$ , where  $n$  is the population size
- Calculate the cumulative probability,  $(q_i)$ , for each chromosome:  $q_i = \sum_{j=1}^n P_j$
- Generate a random number,  $r \in (0, 1]$ .
- If  $r < q_1$  then select the first chromosome,  $x_1$ , else select the chromosome  $x_i$  such that  $q_{i-1} < r \leq q_i$ .
- Repeat steps 4-5  $n$  times.

Having selected the parents for reproduction, crossover is performed by taking two parts of two chromosomes to create new chromosomes. Crossover process is illustrated in the example below as shown in Figure 1. Suppose that there two parents namely  $P_1$  and  $P_2$ , to create the children let say  $Ch_1$  and  $Ch_2$  do the following steps:

- Go through  $P_1$  from the left side and take the first  $n/2$  number of ones, then write them down in the same position in  $Ch_1$ .
- Go through right side of  $P_2$  and take the first  $(n - \frac{n}{2})$  number of ones, then write them down in the same position as  $P_2$  in  $Ch_1$ .
- Fill in the rest of positions of  $Ch_1$  by zeros.
- To create  $Ch_2$  follow the steps above by replacing  $P_1$  with  $P_2$ .

P <sub>1</sub>	0	0	0	0	1	0	1	1	0	0	1	0	1	0	1
P <sub>2</sub>	1	0	1	0	0	1	0	0	0	1	1	0	0	1	0

1. Write down the first 6/2 left ones from the first parent in the same position

				1		1	1								
--	--	--	--	---	--	---	---	--	--	--	--	--	--	--	--

2. Write down the first 6 - (6/2) right ones from the second parent in the same position

				1		1	1		1	1				1	
--	--	--	--	---	--	---	---	--	---	---	--	--	--	---	--

3. Fill in the rest of positions

Ch <sub>1</sub>	0	0	0	0	1	0	1	1	0	1	1	0	0	1	0
Ch <sub>2</sub>	1	0	1	0	0	1	0	0	0	0	1	0	1	0	1

Fig. 1. Example of crossover process between two parents

Mutation performed by a little modifying a chromosome. In this case it can be achieved by randomly picking a one attribute of a chromosome and convert it. Figure 2 below lists an example in which the bit (site) number two and five of a chromosome mutated and converted from 0 to 1 and from 1 to 0 respectively.

0	0	0	0	1	0	1	1	0	1	1	0	0	1	0
↓														
0	1	0	0	0	0	1	1	0	1	1	0	0	1	0

Fig. 2. Example of mutation process

Parents have been selected and children chromosomes created via crossover and an occasional mutation. After that, it is the time to insert the newly created children in to the population and begin the selection, crossover, and mutation process again until some stopping criterion is met. three criteria used as stopping conditions. (1) The evolution stops if the total number of iterations reaches a predefined number of iterations, (2) if the fittest chromosome of each generation has not changed much, that is, the difference is less than 10-3 over a predefined number, or (3) if all chromosomes have the same fitness values, i.e., when the algorithm has converged. below shows the algorithm described above.

- 1: **Begin**
- 2: Initialize the population,  $P$
- 3: Evaluate  $P$
- 4: **While** stopping conditions not true **do**
- 5: Apply Roulette Wheel Selection for Reproduction (create  $P_{mating}$ )
- 6: Crossover  $P_{mating}$
- 7: Mutate  $P_{mating}$
- 8: Replace  $P$  with  $P_{mating}$
- 9: Evaluate  $P$
- 10: **End**

GA-based Algorithm

## V. CONCLUSION AND FUTURE WORK

This study describes the replica placement services as a part of replication management in Data Grid. The GA-Based Replica Placement Mechanism (GARPM) finds the best location sites to place the newly created replicas. From the users' perspective, the best sites are located as close as possible to the sites that most potentially will request the underlying replicas to improve the geographical locality of the sites, while considering that the files that are requested by the sites are likely to be requested by nearby sites [33]. However, from the sites' perspective, the best sites are the ones that are located the farthest from the replication sites that never request the underlying replicas. The proposed strategy can make good decision on which replicas each site should store, such that comply with users' satisfaction and resource's satisfaction.

As a future work, it is our intention to implement the presented replication mechanism in a grid environment, for example by using OptorSim, a grid simulator. Furthermore, the strategy can be tested on a larger of number of sites and of different topologies.

### 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, pp. 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] 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, pp. 1-17.
- [4] 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*, vol. 23, 2001.
- [5] L. Guy, P. Kunszt, E. Laure, H. Stockinger, and K. Stockinger, "Replica management in data grids," in *Global Grid Forum*. vol. 5, 2002.
- [6] H. Lamahemedi, 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.
- [7] H. Lamahemedi, 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, p. p.378.
- [8] 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, pp. 1-15.
- [9] K. Ranganathan and I. Foster, "Identifying Dynamic Replication Strategies for a High-Performance Data Grid," *International Grid Computing Workshop*, pp. 75-86, 2001.
- [10] 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, pp. 500-507.
- [11] 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, pp. 775-790, 2005.
- [12] S. Venugopal, R. Buyya, and K. Ramamohanarao, "A taxonomy of data grids for distributed data sharing, management, and processing," *ACM Computing Surveys (CSUR)*, vol. 38, p. 3, 2006.
- [13] R. M. Rahman, K. Barker, and R. Alhaji, "Replica placement strategies in data grid," *Journal of Grid Computing*, vol. 6, pp. 103-123, 2008.
- [14] R. M. Rahman, K. Barker, and R. Alhaji, "Performance evaluation of different replica placement algorithms," *International Journal of Grid and Utility Computing*, vol. 1, pp. 121-133, 2009.
- [15] M. Tang, B. Lee, X. Tang, and C. Yeo, "Combining data replication algorithms and job scheduling heuristics in the data grid," *Lecture notes in computer science*, vol. 3648, p. 381, 2005.
- [16] 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*, vol. 22, pp. 254-268, 2006.
- [17] 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, pp. 414-421.
- [18] H. P. Chang, "A Dynamic Data Replication Strategy Using Access-Weights in Data Grids," 2006.
- [19] C. Wang, C. Yang, and M. Chiang, "A Fair Replica Placement for Parallel Download on Cluster Grid," *Lecture Notes in Computer Science*, vol. 4658, p. 268, 2007.
- [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, pp. 1-5.
- [21] Q. Rasool, L. Jianzhong, G. S. Oreyu, Z. Shuo, and Y. Donghua, "A load balancing replica placement strategy in Data Grid," in *Proceedings of Third International Conference on Digital Information Management, ICDIM*, London, UK, 2008, pp. 751-756.
- [22] M. Shorfuzzaman, P. Graham, and R. Eskicioglu, "Popularity-Driven Dynamic Replica Placement in Hierarchical Data Grids," in *Parallel and Distributed Computing, Applications and Technologies*, 2008. PDCAT 2008, 2008, pp. 524-531.
- [23] 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, pp. 376-381.
- [24] 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, p. 8.
- [25] L. Pangfeng and W. Jan-Jan, "Optimal replica placement strategy for hierarchical data grid systems," in *Cluster Computing and the Grid*, 2006. CCGRID 06. Sixth IEEE International Symposium on, 2006, p. 4 pp.
- [26] 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. DFmA 2008. , 2008, pp. 96-101.
- [27] K. Ranganathan and I. Foster, "Design and Evaluation of Dynamic Replication Strategies for a High Performance Data Grid," in *International Conference on Computing in High Energy and Nuclear Physics*, Beijing, 2001.
- [28] 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*, vol. 1001, pp. 1324-1331, 2004.
- [29] R. M. Rahman, K. Barker, and R. Alhaji, "Replica placement in data grid: considering utility and risk," in *Proceedings of Information Technology: Coding and Computing*, 2005. ITCC 2005. International Conference on, 2005.
- [30] Q. Rasool, J. Li, and S. Zhang, "Replica Placement in Multi-tier Data Grid," in *Proceedings of 2009 Eighth IEEE International Conference on Dependable, Autonomic and Secure Computing*, 2009, pp. 103-108.
- [31] F. Ben Charrada, H. Ounelli, and H. Chettaoui, "An Efficient Replication Strategy for Dynamic Data Grids," in *Proceedings of*

- International Conference on P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC), 2010, pp. 50-54.
- [32] Mohammed Madi, Yuhani Yusof, and Suhaidi Hassan, " A Dynamic Replica Creation: Which File to Replicate?," in the Proceedings of the 3rd International Conference on Computing and Informatics (ICOCI 2011), Bandung, Indonesia., 8-9 June 2011.
- [33] K. Ranganathan and I. Foster, "Identifying dynamic replication strategies for a high-performance data grid," *Grid Computing—GRID 2001*, pp. 75-86, 2001.
- [34] H. H. E. Al Mistarihi and C. H. Yong, "Replica management in data grid," *International Journal of Computer Science and Network Security IJCSNS*, vol. 8, p. 22, 2008.
- [35] 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, pp. 198-205.
- [36] A. Elghirani, R. Subrata, A. Y. Zomaya, and A. Al Mazari, "Performance Enhancement through Hybrid Replication and Genetic Algorithm Co-Scheduling in Data Grids," in *Computer Systems and Applications, 2008. AICCSA 2008. IEEE/ACS International Conference on*, 2008, pp. 436-443.
- [37] S. N. Sivanandam and S. N. Deepa, *Introduction to genetic algorithms*: Springer Verlag, 2007.
- [38] D. E. Goldberg, *Genetic Algorithms in Search , Optimization and Machine Learning*: Addison-wesley, 1989.
- [39] T. Wright, "A genetic algorithm approach to scheduling resources for a space power system," in *Electrical Engineering and Applied Physics*. vol. Ph.D.: Case Western Reserve University, 1994.