AUT Journal of Modeling and Simulation

AUT J. Model. Simul., 49(2)(2017)239-263
DOI: 10.22060/miscj.2016.874

Review Article

A Survey of Dynamic Replication Strategies for Improving Response Time in Data Grid Environment

N. Mansouri* and M. M. Javidi

Computer Science Department, Shahid Bahonar University of Kerman, Kerman, Iran

ABSTRACT: Large-scale data management is a critical problem in a distributed system such as cloud, P2P system, World Wide Web (WWW), and Data Grid. One of the effective solutions is data replication technique, which efficiently reduces the cost of communication and improves the data reliability and response time. Various replication methods can be proposed depending on when, where, and how replicas are generated and removed. In this paper, different replication algorithms are investigated to determine which attributes are assumed in a given algorithm and which are declined. We provide a tabular representation of important factors to facilitate the future comparison of data replication algorithms. This paper also presents some interesting discussions about future works in data replication by proposing some open research challenges.

Review History:

Received: 8 August 2016
Revised: 6 December 2016
Accepted: 6 December 2016
Available Online: 21 December 2016

Keywords:
Data Grid
Dynamic Replication
Data Availability
Simulation

1- Introduction

This paper covers the issues regarding with data management in Data Grid environment with a special focus on the problem of the replication process. The idea of Data Grids and their constituent components are explained in section 2. The definition of a Data Grid used throughout the rest of the paper is provided in section 3. Section 4 presents some examples by motivating Data Grid use. Section 5 explains data replication process and then discusses some general issues and challenges in replication for Data Grids. The taxonomy of replication processes is shown in section 6. The taxonomy of replication validation methods is described in section 7. Section 8 introduces the different architectures for Data Grids. Section 9 gives an overview of previous works on data replication. Section 10 shows the simulation results using OptorSim, which is proposed by European Data Grid Project. Finally, section 11 concludes our paper and gives proposals for future works.

2- Overview of the Idea of Grids

The definition of a Grid has now moved on from the simple analogy with the electrical power Grid. In [1], the idea of virtual organizations (VOs) was presented. These include "dynamic collections of individuals, institutions, and resources" for which the Grid provides an infrastructure to solve their challenges. A VO is described by a set of rules and conditions by which they share the Grid resources and examples, including an international team collaborating on the design of a new aircraft, a group that uses satellite imaging to investigate climate modeling, or members of an experiment in high energy physics. With such a wide variety of VOs, it is important to determine the set of standards that is acceptable to all and that enables interoperability of the Grid between each of the VOs sharing its resources. A main three-point checklist that describes a Grid is given in [2]. To be truly called a Grid, a system must:

1. coordinate resources that are not subject to a centralized controller. The resources on a Grid are owned and controlled by various institutes but the Grid must manage access to them and deal with the issues of security, payment, and agreements with local strategies.
2. Use standard, open, common-purpose protocols and interfaces. It is important that these protocols and interfaces be standard and open.
3. Deliver non-trivial qualities of service. The Grid must guarantee the main level of resource availability, security, throughput and response time for the users such that the utility of the combined system is considerably higher than that of the sum of its parts.

3- The Data Grid

In 1999, [3- 4] proposed a specialization and extension of the classic grid as a data grid system. They considered a multilayer structure (Fig. 1) to handle different requirements similar to the one explained in [4]. There are two main levels. The lower layer contains important services and the higher layer services are built on the top. The lower level services present abstract heterogeneous storage elements to enable common operations such as read, remove, generate, and modify data files. In addition, a metadata service that can control information about the data, as well as replicated data, is another core service. One of the obvious differences between classical grids and data grids is the replica system and its subsystems: components of replication selection and replica management.

The corresponding author; Email: n.mansouri@uk.ac.ir

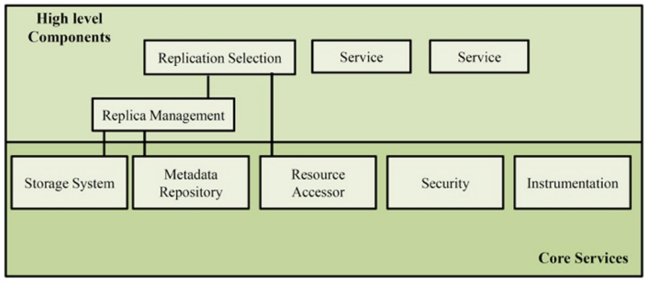

Fig. 1. Data Grid structure [4].

4- Motivating Examples For Data Grid

The scale of scientific experiments has grown so fast, thus, traditional computational procedures used to answer the questions are inadequate.

The Information Power Grid: NASA is establishing the Information Power Grid (IPG) [5] to enhance their ability to find a solution for their big data and computationally intensive problems. They are developing an infrastructure to combine the resources at geographically distributed NASA centers to present a shared resource, while still allowing local management of these resources.

NEES grid: The Network for Earthquake Engineering Simulation Grid (NEESgrid) [6] is being developed to make a link between earthquake engineers and their distributed instruments and computational resources. NEES grid presents a common infrastructure to help scientists with building more complex and perfect models and a simulation of the results for earthquakes.

GridPP: GridPP [7] is the UK Grid for Particle Physics. It is aimed to generate a Grid of resources at UK institutes for studying physics data from experiments in the LHC at CERN and others. GridPP is helping to provide applications for physics studies using the Grid, contributing to middleware development, and presenting the hardware infrastructure at the various organizations.

LCG: Its fundamental task is the deployment of existing Grid middlewares in a consistent package, the constructing of a Grid for the study of data from the LHC experiments [8]. It uses the Globus Toolkit and Condor-G along with services provided by the European Data Grid.

EDG: The European Data Grid (EDG) [9] project was supported by the European Union to enable access to geographically distributed resource elements for three important data-intensive applications: high energy physics, biological and medical image processing, and the Earth observation science.

5- Data Replication

This section provides an introduction to data replication and then discusses some general issues and challenges in replication for Data Grids. One of the practical techniques to enhance the efficiency of data sharing in Data Grid is data replication. In addition, load balancing, fault tolerance, reliability, and the quality of service can be improved with the help of data replication strategy [10- 12].

When the data are placed at a single data server, that server can be a bottleneck if too many requests need to be served at the same time. Consequently, the whole system slows down. In other word, access time in terms of requests per second is increased. By storing multiple replicas at different sites, requests can be served in parallel with each replica providing data access to a smaller community of users. If several users access data file over a network from geographically distant locations, data access will be slower than in a small local-area network given that LANs have lower network latencies than WANs. By providing data as close to the user as possible (data locality), the smaller distances over the network can also contribute to a better performance and lower response times. Moreover, if a single data file is only placed at a single server, this data file cannot be used if the server crashes or does not respond. In contrast, if a replica of the data file is stored on multiple servers, this additional server can provide the data file in case of a server or network failure. Thus, the availability of data can be improved even in the event of natural disasters like earthquakes.

However, despite the several advantages, the justifications of using copies are largely bounded by their storage and communication overheads. The following fundamental issues are identified.

a) Replica placement method is necessary to improve the overall performance according to the goals of applications.

b) The degree of replications should be set to create a minimum number of replicas without wasting the resources.

c) Replica selection should choose the replica that best matches the user's quality of service (QoS) requirements.

d) Replica consistency management should guarantee that the several replicas of a given file are kept consistent in the presence of concurrent updates.

e) The impact of data replication on job scheduling performance must also be considered.

Fig. 2 shows a visual taxonomy of these issues, which will be used in the following subsections.

5- 1- Replica Placement

Although data replication is one of the key optimization techniques for enhancing high data availability, reliability, and scalability, the problem of replica placement has not been well investigated for large-scale distributed environments [13- 14].

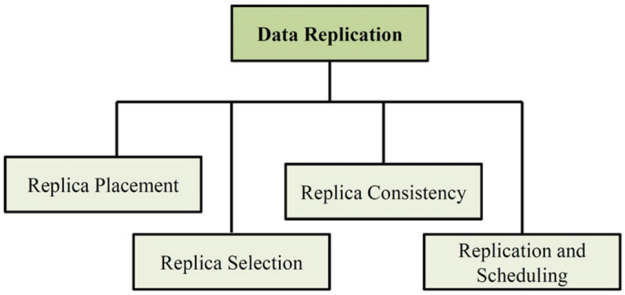

Fig. 2. Taxonomy of issues in data replication.

To obtain the maximum possible benefits from file replication, dynamic replica placement strategy in the Grid environment is necessary. Replica placement strategy determines where in the system new replica should be stored. In fact, various replication algorithms can be designed depending on when, where, and how replicas are generated and removed.

5- 2- Replica Selection

A system that includes replicas also needs a strategy for choosing and locating them based on the file access time. Selecting and accessing suitable replicas are very critical to minimize the usage of Grid resources. A replica selection

strategy determines the available replicas and chooses the "best" replica given the user's location and, possibly, the quality of service (QoS) requirements [15-16].
Typical QoS requirements when doing replica selection might include access time, location, security, computation power, cost, and other constraints. Network performance can play an important role when choosing a replica. High access time decreases the efficiency of data transfer regardless of client and server implementation. Correspondingly, one of the primary strategies to choose the best replica from different sites is the checking of the available (or predicted) bandwidth between the requester and provider sites. The best site, in this case, would be the one that has the minimum predicted retrieval time required to move the data files to the requester site. Although network bandwidth plays an important role in choosing the appropriate replica, other parameters, including additional features for transferring data- most notably, latency- replica host load, and disk I/O performance are significant, as well.

5- 3- Replica Consistency

One of the important problems is the consistency of replicas in Data Grid environments that is not well investigated in the existing studies with files often being regarded as being read-only. However, as Grid solutions are increasingly used by a range of application types, requirements will arise for strategies that keep the consistency of replicated data that can change over time. The replica consistency issue deals with concurrent updates made to several replicas of a file. When the content of one file is changed, all other replicas then have to have the same data and thus present a consistent view. Replica consistency is a traditional problem in distributed systems, but it introduces new challenges in Data Grid environments [17].
Traditional consistency maintenance approaches such as invalidation protocols [18], distributed locking mechanisms [19], atomic operations [20] and two-phase commit protocols [21] are not necessarily appropriate for Data Grid systems because of the long delays introduced by the use of a wide-area network and the high level of autonomy of data Grid resources [22]. For example, in a Data Grid, the replicas of a file may be distributed over various countries. Thus, if one site, which keeps a replica, is not available when the update operation is underway, the whole update process could fail.

5- 4- The Impact Of Data Replication On Job Scheduling Performance

Dealing with a large number of data files that are geographically distributed causes many problems in a Data Grid. One that is not regularly considered is the scheduling of jobs to take data location into account when specifying job scheduling. The locations of data required by a job obviously impact Grid scheduling decisions and performance [23];therefore, it is essential to pick a suitable job execution site. Traditional job schedulers for Grid environments are responsible for allocating user jobs to sites in such a way that some popular parameters are met, such as the reduction of the average access time, the maximization of throughput, and processor utilization. Therefore, the appropriate job scheduling considers not only the abundance of computational resources but also data locations. A node with the high number of processors may not be the best candidate for computation if it does not have the needed data nearby. Similarly, a site with local replicas of the needed file is not an optimal site to compute if it does not have adequate computation capability. A successful scheduling strategy is required that will allow the fastest possible access to the needed data, hence, reducing the data access time. Since generating several replicas can obviously reduce the data access cost, a tighter integration of job scheduling, and dynamic data replication could bring substantial improvement in job execution performance.

6- Taxonomy Of Replication Methods

Fig. 3 indicates the taxonomy for different replication management. Generally, replication management is classified as centralized or decentralized. The centralized method determines the replica placement using a single entity (i.e., job scheduler or replica catalog) [23]. But decentralized method places the replicas by different entities (i.e., sites or users) [24].
Replication type can be static or dynamic. Static methods generate replicas and will exist in the same location till users remove them explicitly. The main drawback for the static replication method is that it cannot adapt to the user's behavior. But static replication has some benefits such as low overhead in comparison to the dynamic replication [25, 26]. On the other hand, the dynamic method generates and replaces replicas based on the changes of the system, i.e. data access pattern [27,31]. Dynamic data replication leads to the better overall performance. Due to the dynamic nature of Grid environment, the requirements of users are variable during the time [32]. It is necessary to note that large-scale data transfer that is a consequence of dynamic replication can waste the resource of the network. Therefore, we must avoid the inessential replication during job execution. A dynamic replication may be considered centralized or in a distributed structure. These structures also have some problems, e.g. the centralized structure has a high overload if the system nodes enter and leave frequently. Decentralized manner needs further synchronization procedure in its decision.
Another feature of replication process that belongs to the middleware or user's level is replication actor. For instance, in [33] storage element determines which file should be replicated based on its profit. In general, the middleware implements the replica placement methods that are more efficient. It guarantees a uniform consistent process across the infrastructure. We can classify the user space as dedicated managers and applications. In the primary case, the application decides on replica placement, allowing more specific optimizations. In [32], [34] and [35], replication methods are implemented as middleware services.

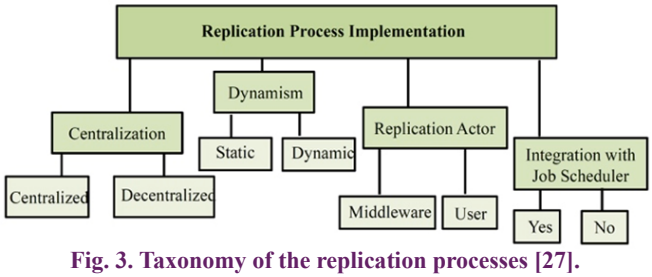

Fig. 3. Taxonomy of the replication processes [27].

In the sequel, there is a superiority between data replication methods that can cooperate with the job broker and those that cannot. The authors in [34, 36] showed that we could reduce

the retrieval time and makespan with the interaction between data replication and job scheduling. For instance, high-energy physics experiments schedule their data-intensive jobs on the sites where the most required data are located.

7- Different Replication Validation Methods

Fig. 4 presents the different methods for replication validation. There are various Grid simulators such as MicroGrid [37] Bricks [38] SimGrid [39], GridSim [40] and OptorSim [41]. In some cases such as [42] and [43], evolution results are reported based on a real environment. In addition, theoretical validation (i.e., mathematical modeling or formal proofs) is applied for the performance evaluation [44].

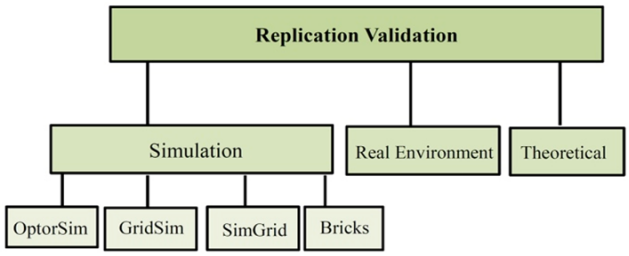

Fig. 4. Taxonomy of replication validation.

8- Architectures Of Data Grids

Grid architecture has a great impact on our data replication and job scheduling performance. There are various architectures for the grid implementation such as multi-tier architecture, tree, graph, P2P, and hybrid topology. Hierarchal topology, i.e., multi-tier, which is used by the GriPhyN project, is the most common structure. Ranganathan et al. discussed six data replication methods based to the GriPhyN project architecture [45]. The later researchers used this hierarchal topology as a basis and modified it. For example, sibling tree is a modification to hierarchal topology that the sibling nodes are also connected. Most replication works considered the hierarchal architecture and extended their study to general graphs.

9- Dynamic Replication Strategies

In this section, first, feature comparison of data replication strategies is given. Second, we categorize the dynamic replication strategies into various groups based on their nature and architecture.

9- 1- Feature Comparison And Its Tabular Representation

To evaluate different approaches theoretically, we focus on the comparison of different features:

Scalability: This is the ability of an algorithm to place replicas for a system with any finite number of sites. This parameter should be improved.

Usability: This shows that a user can achieve goals with effectiveness of, efficiency of, and satisfaction with this product.

Popularity: This option determines whether the replication strategy considers data popularity, i.e. replica frequency, in replica creation or not.

Availability: This option determines whether the strategy provides a predefined data availability level or not.

Fault tolerance: If the replication strategy detects a failure in the system and tolerates it, we call the method "fault tolerant."

Bandwidth consumption: This option indicates whether replication strategy tries to decrease the bandwidth consumption by storing replicas as close to the user as possible.

An optimal number of replicas: Some replication strategies determine the number of replicas in the system based on the cost of keeping them.

Replica placement in wise manner: The replica placement step is another critical process because it affects the performance of system significantly. For instance, if the new copy is placed in the best site, it optimizes the workload of various servers.

File access pattern: File access pattern specifies the order of file requests by jobs. There are five different access patterns in the Grid environment. Five important access patterns are described as follows: sequential pattern (data files are requested based on the job configuration file), random pattern (successive file requests are exactly one element away from the previous file request and direction is random), unitary random pattern (data file requests are one element away from prior file request but the orientation is random), Gaussian random walk pattern (data files are accessed according to the Gaussian distribution), and Random Zipf pattern (given by $P_i = K/ i^s$, where P_i shows the ith-ranked item frequency, K indicates the most frequently accessed data popularity, and s is the distribution shape).

Storage assumption: The traditional replication strategies consider unlimited storage space in their replication process. But new replication methods assume a limited amount of storage capacity.

Table 1a- 1h show the results of the comparative study on different data replication strategies cited in the article. The notation + shows that the option is considered. The notation – shows that the replication strategy does not consider that parameter. Finally, NR indicates that the option is not reported in the article. In the sequel, we discuss various data replication algorithms in four subsections.

9- 2- Techniques For Peer-to-peer Architecture

Dafei et al. [46] proposed a new replication strategy based on the Peer-to-Peer Geospatial Data Grid prototype. Their replica generation uses the feature P2P, spatial content and DHT directory, then, it put forward the replica selection and the maintenance issue [47]. In this strategy, the user propose an initial threshold, e.g. three, and then adapts it based on the replica popularity. The proposed strategy logs the history of user's access to each file. It replicates file with a high popularity (hotspot) in different sites to balance the load of servers. Therefore, it uses a searching process, i.e. DHT, based replica dissemination to determine suitable site in the P2P system. After the proposed strategy determines some appropriate nodes as destinations, then it saves the information of replica, i.e. replication directory, when placing the replica on them. In the first step, the spatial information of the local nodes is created. Next, it gets information of other sites and updates its own history. At the same time, it also sends its own catalog to other nodes. After several iterations, the local replication directory is converted to the global consistent resource metadata. In the sequel, it uses the message queue to deliver the message, broadcast and collect heartbeat for replica maintenance.

Abdullah et al. [48] presented a P2P model that improves different performance metrics such as, reliability, data availability, and scalability. They assumed that all works are placed in different groups independently. All peers of a group perform a predefined set of services. At any time, peers can leave or join a group and a peer can be assigned to more than one group. After a peer affiliates to a group, it should share its

data with other peers and access to the data of others without knowing from which peer they are obtaining the data. When a data request is sent to the nodes based on the information of routing table, data discovery process is triggered. If the requested data is not available locally, it forwards the request to the other nodes until its time-to-live and also it saves a track of hop count. The authors investigated four replication strategies as follows: ‘‘requester node placement strategy’’, ‘‘path node placement strategy’’, ‘‘path and requester node placement strategy’’, and ‘‘N-hop distance node placement’’ strategy. ‘‘requestor node placement strategy’’ replicates file in requester site when it gets the requested file. But ‘‘path node placement strategy’’ stores replica of the file in all nodes on the path from the provider to the requester site. ‘‘N -hop distance node placement’’ stores the replica of the file in the neighbors of provider site within an N-hop distance. They designed a Grid simulator that tests the proposed strategies in terms of execution time and bandwidth usage. The experiments indicated the proposed strategy has better success rates and execution time in comparison with the others.

9- 3- Qos Aware Data Replication

Andronikou et al. [49] designed a complete replication strategy that consists of replica generation, replica placement, replica replacement and retirement based on the QoS parameters such as replica cost, workload balancing, network bandwidth, and data importance value. In the first step, it assigns ‘Importance’’ value to each data file in terms of which is directly related to the profit (money, kind, reputation, etc.). One of the important problems in Grid is the profit maximization based on the QoS requirements of different users. Therefore, the authors considered two factors as Requested Access Latency (TL) and Requested Availability (RA). TL shows how fast the data can be made available and depends on the features of the site, e.g. CPU, memory, hard disk access delays, and features of the network such as bandwidth and distance. RA shows the probability that this request will be fulfilled. Higher the importance of data, more copies are to be created and stored in different locations. It specifies an appropriate number of replicas based on the free storage space and the Importance value of the file.

Then, it must determine where to place replicas based on the network availability. A site is a suitable candidate if its connection with the frequent requester site has high bandwidth. In addition, it constructs a distance metric based on the network structure, administrative barriers, and bandwidth. Consequently, replica relocation strategy identifies the best location for available replicas in the system based on the access pattern of current users. It is obvious that a replica retirement step is necessary since data replication is the resource-consuming approach. It deletes replicas that are no longer requested. But it always keeps at least one copy of each file in the system to provide reliability and availability for the data files.

Shorfuzzaman et al. [50] presented a decentralized replication strategy based on the hierarchical structure. In the first step, it determines the number of replicas to provide predefined quality requirements. In the second step, it determines the location of popular replicas to decrease the replication cost such as read and update cost and improves access time based on the traffic pattern. In addition, they used dynamic programming to overcome the disadvantages of centralized algorithms such as reliability and performance bottlenecks. The proposed strategy is triggered at different time periods to determine the location of new replicas according to the file popularity value. When sufficient storage is not available, the oldest replicas must be deleted and replaced with new replicas. It sets time period based on the request rate. If the arrival rate is high, it considers a short period. This leads to a higher computational cost but adapts more accurately to dynamic access patterns. Experiments demonstrated that the suggested strategy reduces the mean job execution time and has relatively low bandwidth usage. In addition, when the QoS parameters become more stringent, the improvement of the performance of this strategy is more visible. Simulation results are based on the different storage configurations and access patterns in constant and fluctuating arrival rates. In the sequel, the proposed replica placement shows a good performance with high quality assurance.

Jaradat et al. [51] presented a new data replication called Balanced QoS Replica Selection Strategy (BQSS). BQSS is according to the mathematical model, the balanced QoS time, availability, and security. In the first stage, BQSS specifies the best location of the replica with a high quality, high capacity, timely, and consistent balanced rates of QoS factors. In the second stage, BQSS computes the value of QoS factors (i.e. time, security and availability) in one value. The range of each factor is between 0% and 100% based on the site capabilities. They rated the time based on the equations introduced by E. Husni et al. [52], which considers replica requests waiting in the queue for storage. BQSS defines the site availability according to the relation between the operation time of the provider to present certain VOs and the time of file transfer from the same provider in replica selection progress. They considered the computing trust factor (TF) proposed by V. Vijayakumar et al. [53] in their security model. Trust factor includes the Self-Protection Capability (SPC) and reputation weightage. The SPC of a site shows the ability to detect the viruses and unauthorized access to provide a secured files storage. Reputation procedure builds a reliable way through social control based on the community-based feedback about the past experiences of elements. Simulation results demonstrated that the value of standard deviation for the three QoS factors were improved in most scenarios.

Cheng et al. [54] presented a new dynamic replication strategy based on the general graphs and a more realistic model. It considers storage cost, update cost, replica access cost, and server load in replica decision. Therefore, each data request can be quickly answered without wasting the limited capacity of servers. The authors presented two heuristic methods for efficient replica placement based on the storage cost, update cost, and satisfaction of user’s requirements. They defined storage cost for the replication as an aggregation of all storage costs for replica servers during the replication process. In the final phase, it performs consistency procedure. For this purpose, the origin server R sends the update messages to every replica servers. The number of update messages that are sent by R in each interval is μ. The authors assumed that update distribution tree T that links all the servers in the system is available. When server V needs to get data from replica server U, it must corporate with U. The access cost for replication is defined as an aggregation of the communication costs of servers to get the required data. This strategy is suitable in the real environment since it can find the near-optimal solutions. For the evaluation step, they used the Waxman model [55], which has N nodes in an s-by-s square. Analytical results indicated that Greedy-Remove and Greedy-Add found near-optimal situations in all cases.

Table 1a. Features of replication algorithms.

Strategy	Dafei et al. [46]	Abdullah et al. [48]	Andronikou et al. [49]	Shorfuzzaman et al. [50]	Jaradat et al. [51]	Cheng et al. [54]
Year	2007	2008	2012	2011	2011	2009
Replication type	Dynamic	Dynamic	Dynamic	Dynamic	Dynamic	Dynamic
Replication management	Decentralized	Decentralized	Centralized	Decentralized	Centralized	Decentralized
Replication actor	Middleware service	Middleware service	Middleware service	Middleware service	Middleware service	Middleware service
Grid topology	P2P	P2P	NR	Multi-tier	NR	Waxman Model (graph)
Availability	+	+	+	+	+	+
Popularity	+	-	+	+	-	-
Response time	NR	+	-	+	+	+
Fault tolerance	-	-	-	-	+	-
Bandwidth consumption	NR	Average	Low	Average	Low	NR
Consistency	+	-	-	-	+	+
Storage assumption	Limited	Limited	Limited	Limited	Limited	Limited
File access pattern	NR	Random	Random	Sequential, random, unitary random, Gaussian random walk	NR	NR
Optimal number of replicas	+	-	+	+	-	+
Place of replica	+	+	+	+	-	+
Replica condition	Cost model	When requested file is not in site	Importance factor	Invoked at regular intervals	NR	When requested file is not in site
Integration with job scheduler	-	-	-	-	-	-
Validation method	Theoretical	Simulation	Simulation	Simulation	Simulation	Simulation and Theoretical
Main additional features	Spatial content	Number of hops	Retirement mechanism	Replication cost	Security rating	Replication cost

Nong et al. [56] proposed a new replication strategy in three steps by considering the QoS parameter and TP-GABMAC matrix. For replica consistency step, they considered a replica ring process with update conflicting procedure. In the first step, it solves a local problem and an issue by a new technique. Therefore, the main problem is divided into several small problems by a landmark-based clustering method. In the second step, it solves the small problems based on the matrix-based technique with the name of GABMAC. In the sequel, it integrates all small problems into the original replica process.

Previous works mostly considered a replica-updating tree (i.e., primary replica method) [57-58] while in this work [56], the authors introduced two replica types. An original data file is named primary and the others are secondary. In the proposed tree topology, the primary performs updating operations and secondary replicas are synchronized with the primary. It is obvious that the primary files are bottleneck. For ratification of requirements and enhancing the consistency efficiency, replicas in the network are linked one by one, and the first replica links to the last. Therefore, the replica ring (RR) structure is yielded.

RR topology can solve the bottleneck issue in the primary-secondary replica procedure. Also every node in RR structure can perform the read/write operation concurrently. For consistency problem, the proposed strategy [56] locally performs the data file writing operations and it adheres to the global negotiation. Then, only the winner's writing operation is run and all others are dropped. In this case, there is not simultaneous updating process since only the winner can do the updating process in one-time interval. The results showed that the suggested method is stable and scalable and has a good performance in various configurations and access patterns.

9- 4- Strategies For Multi-tier Architecture

Dan-wei et al. [59] proposed a new replica placement strategy using the degree of distribution for large-scale networks. The authors presented two candidates replica nodes: a degree-based candidate pool and a frequency- based candidate pool. It sets a threshold for replica creation. For example, setting λ= 20% means that when the cost is reduced to 20%, it creates the second replica. Considering the environment, they copied data in the node with a minimum local cost. Measures of complex networks consist of degree distribution, average path length, clustering coefficient, and other metrics. Degree distribution is a key metric, in which ki is the degree of node i, which represents the number of edges connected to node i. Degree distribution p(k) means that the probability of randomly choosing a node is k. According to the law of complex networks, if a site has a higher degree, then it is more significant. Moreover, other nodes are more likely to connect to such a node. Therefore, Data Grid replicas should be stored in nodes having high degrees so that all Grid nodes can easily access replicas. Further, they presented and proved a replica creation theorem. For the implementation of the new model, they presented a dynamic multi-replicas creation strategy (DMRC). Simulation results demonstrated that the proposed strategy could improve the performance in terms of makespan and storage usage.

Wang et al. [60] presented a novel data replication as well as replica deletion based on the data access history known as Closest Access Greatest Weight with Proactive Deletion (CAGW_PD). The main goal of CAGW_PD is the minimization of total data transfer cost (DTC) based on the read cost and the write cost. Therefore, the number of messages that are transfered among nodes of the system must be small. It is obvious that the cost of the write operation from the ordinary server to the primary one is inevitable. Consequently, CAGW_PD should minimize the read cost and update the cost of a primary server to reduce the DTC. When the number of replicas is increased, the read cost decreases significantly. In addition, the cost of updating data is high and the DTC might be minimized. Therefore the number of replicas must be controlled and unnecessary replications must be avoided. In the first step, the authors applied the popularity concept for each file. If the popularity of file is higher than the mean popularity of all the files, then CAGW_PD replicates it. In addition, they assigned weight to each file based on the access time since some files were popular a long time ago but are unpopular now. A file with a very recent access has a higher value of weight. CAGW_PD uses a proactive deletion method to remove replica when free storage space of the server is not enough for storing new replica. CAGW_PD deletes files in which the benefit of reducing read cost is less than the detriment of increasing update overhead.

Almuttairi et al. [61] proposed a high-level brokering service based on the replica location information for Data Grid environment. Broker chooses an appropriate provider among storage replica to ensure reliable bulk data transfer in the limited access time. Therefore, a server with stable status and the closest match to the particular user is a suitable selection to this problem. They designed a two-level broker that finds the best providers based on the transfer time. Two-phased Service Oriented Broker (2SOB) acts according to the data mining methods such as associated file discovery. Association rules technique extracts a group of site providers that have low latency and packet drops [62]. 2SOB transfers data files in a parallel way. In step one, it considers Coarse-grain selection metrics as sifting replica sites with uncongested links. Next, it assumes fine-grain selection metrics. In other words, 2SOB selects a site that has the lowest cost and satisfies the QoS requirements. Analytical results demonstrated that it could have a reasonable access time in comparison with current algorithms and reduce the mean job execution time.

Yang et al. [63] proposed a new maintenance strategy known as Dynamic Maintenance Service (DMS) and a novel consistency strategy as Replica Consistency Service (ORCS). DMS considers data file access frequency, free space of storage, and condition of the network for replica placement phase. In ORCS, the authors designed an asynchronous replication method with consistency ability to improve storage usage for new temporary data generated by simulations. It can find the best replica provider for transferring the needed data set and increase the storage device usage. In addition, DMS stores replica in appropriate locations based on the number of requests, thus the number of remote access is decreased significantly. On the other hand, ORCS method keeps the content of replicas in the system consistent, thus the access performance is increased. One of the important parameters that DMS and ORCS strategies consider is the available storage capacity, thus, the probability of applications crashing or having to resubmit jobs to other locations are decreased. The simulation results demonstrated that DMS and ORCS could improve the overall performance and storage usage.

Choi et al. [64] proposed a new dynamic hybrid protocol that efficiently improves the current protocols such as Grid protocol and Tree Quorum protocol. Some protocols use all replicas for read and write operations. For instance, the Quorum consensus protocol has 16 replicas, thus sum read quorum and write quorum needs to be higher than 16.

Assigning a logical structure to the replicas and combining Grid topology can solve this problem. But tree protocol has a major problem that the number of replicas quickly grows as the level of tree grows. On the other hand, when the number of failures increases, the value of read cost grows. In addition, Grid protocol has the main disadvantage that has the identical operation cost whether a faulty node exists or not. However it provides a higher availability than the tree structure. It is obvious that the combination of Grid and Tree topologies yields the low operating cost of Tree Quorum protocol and high availability of Grid protocol. This paper merges the previous protocols to achieve the benefit of the low operating cost if there exists no failure and the number of replicas is small. The suggested structure can be flexibly adapted based on the three configuration parameters as the tree's height, the number of descendants, and the Grid depth. If we want to increase the availability, then the tree's height and the number of descendants must be decreased and the depth of topology must be increased. However, to increase the write availability, tree's height and the depth should be decreased and the number of descendants should be increased. The simulation results showed that a combined topology could reduce the communication overhead and he cost of operation. It also allowed significantly smaller response time.

Taheri et al. [65] proposed a new heuristic approach for Grid environment, called JDS-HNN. JDS-HNN assigns tasks to appropriate locations and replicates data files in multiple sites to reduce the mean job execution time and the total delivery time

Table 1b. Features of replication algorithms.

Strategy	Nong et al. [56]	Dan-wei et al. [59]	Wang et al. [60]	Almuttairi et al. [61]	Yang et al. [63]	Choi et al. [64]
Year	2010	2010	2011	2013	2010	2012
Replication type	Dynamic	Dynamic	Dynamic	Dynamic	Dynamic	Dynamic
Replication management	Decentralized	Centralized	Centralized	Decentralized	Decentralized	Decentralized
Replication actor	Middleware service	Middleware service	Middleware service	Middleware service	Middleware service	Middleware service
Grid topology	General	General	Multi-tier, Random graph	General	Multi-tier	Tree structure
Availability	+	+	+	+	+	+
Popularity	-	+	+	-	+	+
Response time	+	+	+	+	+	+
Fault tolerance	-	-	-	+	+	+
Bandwidth consumption	Average	Low	Low	Average	Low	Low
Consistency	+	-	-	-	+	+
Storage assumption	Limited	Limited	Limited	Limited	Limited	Limited
File access pattern	NR	NR	Random	Random, Sequential	Random	NR
Optimal number of replicas	-	-	+	-	-	+
Place of replica	+	+	+	-	-	-
Replica condition	NR	Set one threshold	Set one threshold	NR	Set one threshold	NR
Integration with job scheduler	-	-	-	-	-	-
Validation method	Simulation	Simulation	Simulation	Simulation	Simulation	Simulation
Main additional features	Update eliminating mechanism	Complex network theory	Cost model	Associated replicas discovery	Cost model	Node search algorithm

for requested files by dependent jobs. A natural distribution of a variety of stones among different jars inspired JDS-HNN. In addition, it applies a Hopfield Neural Network in the optimization steps to minimize the makespan. The matchmaking procedure in schedulers was designed and its main idea was a holistic scheduling strategy to reduce the makespan of all tasks and the total data transfer time. JDS-HNN uses important information in their scheduling and replication decision such as characteristics of computing/storage nodes in a system, the dependency of tasks and requirements, and the bandwidth between the provider and requester sites. The authors applied different benchmarks in medium- to very-large-sized systems to evaluate the suggested strategy. Experiments indicated that JDS-HNN improved execution time by replicating data and job scheduling in an effective manner.

Ma et al. [66] proposed a new replica placement strategy based on the Quantum Evolutionary Algorithm (QEA) for Data Grid. In addition, the authors provided the Computing Intelligent Algorithm (CIA) to optimize the strategy. QEA-based replication reduces the decision-making cost and improves the resource utility rate by pre-creating replica according to bandwidth, storage usage, and data access frequency. In other words, QEA-based strategy stores the required replicas before job execution. For improving the response time and bandwidth usage, it merges the optimization techniques. Broker selects the site for job execution according to the estimated time for providing the required files of the job and providing all files for all jobs in the queue at that site. It assigns a high priority to the site provider that has the lowest response time. QEA-based replication strategy contains three stages for parameter initialization as single replica generation, multi replica generation, and comprehensive optimization. The experiment results with OptorSim indicated that QEA-based strategy reduced the mean job time and bandwidth

usage in comparison with Genetic Algorithms (GAs), Ant Colony Optimization (ACO), Particle Swarm Optimization (PSO) methods.

Zhang et al. [67] decreased the data transfer time to improve the job scheduling performance. For this goal, the authors presented a parallel downloading scenario that replicates partitions of data and downloads them in a parallel way to reduce data transfer time. In a Grid environment, the size of storage is restricted, thus it is not possible to replicates all the complete files across the system. Therefore, they divided the file into N (N∈{2,3,4,...}) segments and placed it on N different locations. If one server needs a particular file, the fragments of it are transferred from different servers simultaneously and resembled there. They stored popular replicas on the best server based on the merit value [9]. The merit value for servers is given by

$$Merit_s = \sum_{i \in CS} \frac{C_i}{B_{i,s}} \quad (1)$$

where C_i shows the computing power for site i, CS is the set of all the computing sites, and Bi,s denotes the bandwidth between requester site i and provider site s. It is obvious that parallel downloading and the completeness are preserved concurrently that can improve retrieval time. The authors compared the parallel downloading approach with non-parallel downloading based on three scheduling strategies that were Shortest Turnaround Time (STT), Least Relative Load (LRL) and Data Present (DP). The experiment results presented that the proposed parallel downloading strategy reduced the mean job turnaround time in all three scheduling cases. In addition, when a distributed system has a low network bandwidth and comparatively high computing power server, the benefit of the parallel system is more obvious.

Chang et al. [68] presented a new downloading strategy based on the dynamic status of all servers. The proposed strategy reevaluates the current servers; if they unsatisfactorily perform, then it replaces them with others that provide the reusable performance. All previous downloading methods transferred the load of the ill-performing servers to the more powerful servers. However, intuitively if there are idle servers near that and have the required data, why not apply them? Thus, the suggested downloading method checks all servers that have the requested files even if a server is not available in the downloading process initially. Therefore the load is transferred from busy servers to nonworking servers instead of just moving to another working server. In addition, the authors used the available bandwidth, processor utilization, and memory usage in the determination of server suitability. This method is tested in a real Grid environment. The proposed strategy could reduce the completion time about 1.63%- 13.45% lower than the recursive co-allocation approach in a real Grid system. In addition, it could reduce the completion time about 6.28%- 30.56% in the Grid system with other injected load. The suggested downloading method is suitable in a dynamic environment. This is because it adapts to the variation of the environment and decreases the access time effectively.

Taheri et al. [69] proposed Job Scheduling using Bee Colony (JDS-BC) optimization technique. In the first step, JDS-BC assigns the jobs to the site with minimum load. In the second step, it replicates the required data file among sites to reduce the makespan and file transmission time in the heterogeneous distributed system. They modified the BCO procedure accurately to match with the important requirement of replication and scheduling problems. In this case, bees act in more than one role since the number of computing nodes are limited. Moreover, every bee searches its neighborhood as a scout and then pays attention to dancing bees on the dance floor as a follower to find its collecting nectar source. Thereby, it provides advantages from its selection after it selects food and its nectar. It is possible that one of the dancing bees is replaced when it collects more benefit than it. The authors divided the overall replicating process into two independent sub-processes that will be described in the following. It has been demonstrated to be more effective [70]. Besides, it can decrease the problem complexity, thus convergence time is reduced accordingly. For these purposes, it uses the BCO-based strategy to assign jobs to computing nodes and, then, replicates files to reduce the access files in dependent jobs. The following procedure describes such processes: (1) For each data file, D_x. (2) Determine the total upload time of D_x to all its dependent jobs if it is copied on each storage node; keep computed uploading times in an array called ArrUpTimes. (3) Sort ArrUpTimes in ascending order. For k=1 to MaxNumReplicas.

$\frac{ArrUpTimes(k)}{MinUpTime(D_x)} < 2$ If then, replicate Dx onto SNx.

They evaluated the JDS-BC in three different tests with varying configurations from small to large. Experiments demonstrated that JDS-BC is suitable for data-intensive and computation intensive systems as well as other systems that are data and compute intensive. JDS-BC can automatically adapt to the environment based on its optimization process. In addition, it decides in a balanced way, where it sometimes relaxes one of its goals (e.g. transfer time) to gain more from optimizing the other one (e.g. makespan).

Jaradat et al. [71] presented a new replica selection strategy based on the availability parameter and data transmission time. They can accurately estimate the response time using site availability. It is obvious that if it selects an unavailable site or site with insufficient time, then it leads to disconnection and the wasting of resources since they must transfer data/job to another site for resuming the download process or starting the download from scratch based on the fault tolerance strategy. The authors defined site availability as a proportion of the required time for downloading a replica and the remaining time obtained by the provider site. The remaining overtime to provide service for the user is considered as the remaining time of a site. The time passed for data transmission from one site to another is defined as the response time. Resource broker must find the requested physical file names and their locations with the help of the replica location service. All necessary information can be achieved from GRIS, e.g. Network Weather Service (NWS) [72], Meta-computing Directory Service (MDS) [73] and Grid File Transfer Protocol (GridFTP) [73]. Then, it selects the best replica provider (i.e., a site with the minimum response time and the minimum probability of disruption) for user's job. The experiment results indicated that the proposed strategy significantly decreased the mean execution time.

Saadat et al. [74] presented Pre-fetching based Dynamic Data Replication Algorithm in Data Grids (PDDRA) to improve the overall performance. The authors tried to pre-replicate necessary files before requests are triggered, assuming that

Table 1c. Features of replication algorithms.

Strategy	Taheri et al. [65]	Ma et al. [66]	Zhang et al. [67]	Chang et al. [68]	Taheri et al. [69]	Jaradat et al. [71]
Year	2013	2013	2011	2010	2011	2013
Replication type	Dynamic	Dynamic	Dynamic	Dynamic	Dynamic	Dynamic
Replication management	Decentralized	Centralized	Centralized	Centralized	Centralized	Centralized
Replication actor	Middleware service	Middleware service	Middleware service	Middleware service	Middleware service	Middleware service
Grid topology	General	General	General	General	General	General
Availability	+	+	+	+	+	+
Popularity	-	-	+	-	+	-
Response time	+	+	+	+	+	+
Fault tolerance	-	-	-	-	-	+
Bandwidth consumption	Low	Low	Low	Low	Average	Low
Consistency	-	-	-	-	-	-
Storage assumption	Limited	Limited	Limited	Limited	Limited	Limited
File access pattern	Random	NR	Unitary random	NR	NR	Sequential
Optimal number of replicas	-	+	-	-	+	-
Place of replica	+	+	Minimum location merit	-	-	-
Replica condition	NR	Population size	File with a larger number of accesses	When requested file is not in site	ArrUpTimes(k)/ MinUpTime (Dx)<2	NR
Integration with job scheduler	+	-	+	-	+	-
Validation method	Simulation	Simulation	Simulation	Real environment	Simulation	Simulation
Main additional features	Hopfield neural network	Quantum evolutionary	Parallel downloading	Parallel downloading	Bee Colony based optimization	Site availability

users have a similar interest in files. They predicated future file requirements based on the history of file accesses in the system, thus most of the time each site can get necessary files locally. Data mining techniques can predicate files that will be requested in the near future. PDDRA includes three main steps. In the first step, it creates a global database that contains the file access sequences. In the second step, when a site needs a file and it doesn't have in local storage, replication is triggered. It must check whether replication provides significant benefits or not. If yes, then PDDRA pre-fetches adjacent files. In the third step, if enough space is not available for the new replicas, it replaces some old replicas. The experimental results showed that PDDRA could reduce job execution time and enhance effective network usage, and decrease the number of replications. But the main drawback of PDDRA is that it does not mention the replica selection.

Lei et al. [75] focused on the maximization data availability issue in the Grid environment. The authors considered two important factors as System File Missing Rate SMFR and System Byte Missing Rate SBMR to address the system reliability. In addition, they assigned a higher value to the hot data file than the cold data file for the replacement step. The data file frequently requested is defined as a hot data. It is obvious that total system availability is more critical than single file availability and correctness of available data is necessary too. The relation between the number of files that are not available and the total number of requested files by all jobs is indicated as SMFR factor. The relation between the bytes that are not available and the total number of bytes requested by all jobs is indicated as SBMR factor. MinDmr optimization strategy has four steps. In the first step, it controls whether the requested file is available on the local site or not, if it is present, then replication does not trigger. In the second step, if the required file is not available locally, then MinDmr

strategy checks the free space, if there is sufficient storage, the requested file is stored. In the third step, if the available storage is not sufficient, MinDmr strategy candidates files based on their weights for replacement. In the fourth step, MinDmr strategy removes candidate files if the replication gain is more than replacement loss. The experimental results demonstrated that MinDmr strategy could increase data availability in different file access patterns and job schedulers.
Horri et al. [76] proposed an advanced methodology to determine the time of file transfer for various sites. Accordingly, the lowest calculated transfer time is considered as a criterion to fetch replication. Transfer time is estimated by the employment of common sub-routes between the paths and the bandwidth of each part of the network. They indicated each path by vector V. The length of the path is determined by the number of edges in the network graph. Si (site i) is situated in LPT array, in which LPT[j] is defined as the transfer time between Sj to Si, and is estimated using weighting average time of three paths in Si‘s catalog with the highest similarity to the current path. This algorithm can be employed as common traffic factors calculator, e.g., round trip time (RTT) for every network. They investigated the ability of cosine similarity replication model, regression methodology, and neural network model for four various file access patterns. The simulation output of OptorSim revealed that compared to the multi-regression and neural network methods, this strategy has a higher performance. Moreover, in the case of available storage that is not enough for replication, the proposed algorithm is able to copy the files that are not available in the closest site. Accordingly, compared to LRU strategy, this strategy shows a lower response time. In the case of enough available storage for replication as well as bandwidth between the source and destination, LRU and the proposed strategy have the same behavior.
Bellounar et al. [77] presented a strategy with considering placement of replicas and redistribution of replicas in Data Grid. Their strategy is able to determine the destination of replicas by employing the cost model. They used four main criteria for the construction of replication model as (1) service of creating replicas (2) service for suppression replicas (3) service for fault management (4) maintenance service. Accordingly, replication is done when one of the following cases is carried out:
If a node needs a data, which is not available in the local cluster, the leader searches into the other clusters, beginning with the nearest cluster in terms of low bandwidth, and generates a replica of the given data on the node with the lowest load.
If the cost of access to the data exceeds the cost of its replication, the leader generates a replica of the data in the node with the lowest load.
In the case of insufficient space to sort out a new replica, removing the least popular replica is inevitable. The proposed strategy employed the message aya(•) (Are You Alive) to maintain the alivesystem. Periodically, the message aya (•) is sent to the node of the cluster and if no ack (•) is received after the predefined deadline, another message is sent again. In the case of any response to the third message, the leader realizes that the node is failed. According to simulation results, the presented strategy is able to decrease bandwidth consumption, minimize data access costs, and improve data availability. Moreover, it is able to reduce the number of the inter-cluster message with considering intra-cluster interactions.
Yi et al. [78] illustrated a decentralized architecture for integration of task scheduling by employing game theory for replica placement. They divided the task scheduling into online mode task scheduling and batch-mode processes [79]. In the case of batch mode task, the task completion time is accurately calculated and the proposed strategy uses it in scheduling decision. It was necessary to note that during the fast variation of bandwidth in Data Grid, the estimation of data transmission time is relatively hard. Moreover, the responsibility of task scheduling is the only task assignment to virtual organizations. As a consequence, it can be considered that the number of task scheduling requests received by a task scheduler at a time is small. Online-mode task scheduling strategy is able to assign every new arrival task immediately by current grid performance and locations of data in a way that the Data Grid performance was satisfied. In other words, the online-mode task is executed as quickly as possible. In this study, the summation of task execution time and task waiting time in computing resources are considered as completion time. In the proposed strategy after broadcasting the data request by data manager in the computing resource, the time of requests at predefined time interval is determined based one file frequency and hot file, i.e., files with the highest frequency are considered for replication. Moreover, the replica placement process would be started when the candidate file for the replication does not exist in storage resource and the frequency of candidate files is higher than the predefined threshold simultaneously.
Application of Game theory [80] to solve such problem with various competitor demands and resources is popular. In fact, the decentralization of replica placement can be considered as a game in which all the storage resources engaged in competition for replication of one specific hot file. Finally, with consideration of average job completion time and average network load, four compositions of task scheduling and replica placement strategies were compared with each other. The simulation results demonstrated that the centralized integration strategy has acceptable performance, especially in the case of storage resource and disk space restrictions.
Meroufel et al. [81] presented a hierarchical replication strategy in the Grid that employs the crash failure parameter. It considers the availability and the popularity of the data in replication decisions. The availability must be maximized for every portion of data and estimated from its access history in the past, since each node has its own stability probability. It was possible to depict the replica availability as the stability of the node on which was placed. Access information of data is saved in the local history table. If the number of total accesses for the data surpasses a particular value, replication is performed in the best site, i.e. the node with the highest number of access to the specific portion of data. In the case of requesting for a new replica, the availability or the popularity criterion must be checked again. Then, the nodes check the possibility of enough space to store new replica. All existing files arranged by a predefined program on the base of access frequency and the file items in the sorted list are removed until enough space is produced for the new replica. The simulation results showed that this strategy has a reasonable performance compared to the dynamic strategies with considering the response time and the data availability in replication decision.

Table 1d. Features of replication algorithms.

Strategy	Saadat et al. [74]	Lei et al. [75]	Horri et al. [76]	Bellounar et al. [77]	Yi et al. [78]	Meroufel et al. [81]
Year	2011	2008	2011	2012	2010	2012
Replication type	Dynamic	Dynamic	Dynamic	Dynamic	Dynamic	Dynamic
Replication management	Decentralized	Decentralized	Decentralized	Decentralized	Decentralized	Semi-centralized
Replication actor	Middleware service	Middleware service	Middleware service	Middleware service	Middleware service	Middleware service
Grid topology	General	Graph	General	Multi-tier	Multi-tier	Multi-tier
Availability	+	+	+	+	+	+
Popularity	+	+	-	+	+	+
Response time	+	+	+	+	+	+
Fault tolerance	-	-	-	+	-	+
Bandwidth consumption	Low	+	Low	Low	Low	Low
Consistency	-	-	-	-	-	-
Storage assumption	Limited	Limited	Limited	Limited	Limited	Limited
File access pattern	Sequential, random, unitary random, Gaussian random walk, random Zipf	Sequential, random, Gaussian random walk, random Zipf	Sequential, random, Gaussian random walk, random Zipf	NR	NR	NR
Optimal number of replicas	-	-	-	+	-	-
Place of replica	+	-	-	-	+	+
Replica condition	When requested file is not in site	When requested file is not in site	Set a threshold	Cost of access, cost of replication	Invoked at regular intervals	Set a threshold
Integration with job scheduler	-	+	-	-	+	-
Validation method	Simulation	Simulation	Simulation	Simulation	Simulation	Simulation
Main additional features	Predicts future need, Pre-fetch files	Minimize data missed rate	Cosine similarity predictor function	Cost model	Game theory	Crash failures in the system

JeyaSheeli et al. [82] proposed an efficient centralized data replication algorithm in the Grid system. The authors discussed the limitations of current replication algorithms and then presented a new greedy centralized replication algorithm to store replicas at a specific site in a way to improve the benefits of replication. The different users of the Grid execute tasks with different resource requirements.

The scheduling of jobs is carried out in FIFO order. It stores new replicas in different locations to maximize the total access cost reduction in the system. Different parameters such as bandwidth consumption of the file during transmission, the number of file requests, and the number of hops between requesters and provider sites can affect the access cost value. The algorithm ends when the replication consumed the free storage space or further access cost reduction is impossible (Eq. 2).

$$\alpha(G,R)=\sum_{i=1}^{m}\sum_{j=1}^{n}\text{minhops}_{ij}\times\text{needcount}_{ij}\times s_j/\beta \quad (2)$$

where, $\alpha(G,R)$ is defined as the total access cost of n data files across m sites. R represents the set $\{w_1, w_2, \ldots, w_n\}$ where each member consists of a set of sites where data file D_j, $1 < j < n$, is replicated. The parameter minhopsij

shows the minimum number of hops between the site i and site j. needcountij indicates the number of times S_i requests file D_j. S_j shows the size of file D_j. β indicates bandwidth/ rate of transmission. The main advantage of this strategy can be depicted as the extensive area of the data usage and applications.

Tu et al. [83] considered data partitioning approach as well as dynamic replication to improve the security and data access performance in Data Grid. They tried to optimize the allocation for the segments of important data based on ensuring coding scheme and secret sharing scheme. They used Grid topology in two layers. The upper layer consists of various clusters to indicate the network topology. In addition, the topology of each cluster is tree graph. The proposed strategy is able to divide the shared replica allocation issue into the two forms. The first form is Optimal Inter-cluster Resident Set Problem (OIRSP), which determines clusters that request common files. The second form is Optimal Intra-cluster Share Allocation Problem (OISAP), which determines locations of common files with the appropriate number of replicas.

To solve subproblems, two heuristic strategies are employed. Comparison of simulation results between OIRSP and randomized K-replication presented good performances on reducing the communication cost. Moreover, the effect of graph size that is equal to the cluster number in system, the graphs of a cluster, and the update/read ratio, i.e. the ratio of the total number of update requests to the average number of read requests from the single cluster are investigated. The results confirmed that in the case of larger graph size, the OIRSP heuristic strategy showed a better performance in comparison to no replication method. In larger graph size, the clusters number that requests the replicas enhances dramatically. In this condition, allocation share replica schema is able to decrease communication cost, and as a consequence, the heuristic algorithm presents a lower job execution time compared to no replication method.

Shorfuzzaman et al. [84] illustrated a Popularity-Based Replica Placement (PBRP) for hierarchical Data Grids. In their proposed strategy, they considered the access rate of a file by the client and the higher tendency of the recently popular file for the near future requests as criteria for determination of popularity value. PBRP is invoked at predefined time periods. It checks the access histories to find the most popular file as a new replica. PBRP tries to store a replica of popular files close to users for reducing the network and storage resources. It was necessary to note that the performance of PBRP strategy is effectively proportional to the threshold value for popularity files. Also, they proposed an advanced version of PBRP with the ability of dynamically determination of threshold by employing data request arrival rates and available storage capacities at servers of the replica as criteria. PBRP is able to enhance the performances of Aggregate Bottom-Up (ABU) strategy by replica generation at the site nearest to the clients with high access counts. Simulation results confirmed that the proposed strategy could decrease job execution time as well as bandwidth consumption in a distributed system.

Chang et al. [85] presented Latest Access Largest Weight (LALW) in three steps. Firstly, by employing access frequency as a criterion, the popularity of each file is determined. Then, the number of needed replica that must be created is estimated, and finally, the location of new replica storing is determined. If T is defined as a system parameter, then LALW is operated at the end of each interval of T seconds. In this strategy, a dynamic replication policymaker manages the replication process. If the cluster is defined as several Grid sites, then a cluster header is employed to manage the information of a cluster. The information of accessed files is sent by headers to the Policymaker. Also, the details of file access counting are stored in each cluster headers. Therefore they can share their information with each other. In this approach, a cluster may be contributed with one or more replicas and the cluster header is responsible to determine the location of the necessary replica. In the beginning of replicating, Policymaker determines the number of pre-existed replicas through cluster header. If this value is less than the value set by Policymaker, the replication is carried out. Accordingly, LALW strategy provides higher weights to the recent requested file and guides the replication to do only cluster levels rather than the site level.

Sashi and Thanamani [86] proposed different versions of Latest Access Largest Weight (LALW) strategy. This algorithm decreases the average job execution time by employing the number of file requests as well as response time as criteria in replica placement step.

Mansouri [87] presented a Modified Latest Access Largest Weight (MLALW). Like LALW, MLALW, at first, determines the needed replica number for each region and then specifies the best site (BS) at the region level. BS selection in MLALW is done with consideration of the highest number of replica access in the future. Prediction of the next number of file access was carried out using exponential decay. For replica deletion, MLALW considers the least frequently used replicas, the least recently used replicas, and the size of the replica as criteria. The simulation results confirmed that MLALW strategy shows a higher performance in terms of execution time, storage usage, and effective network usage with respect to the other replication strategies. One of the most common algorithms for patterns of random requests is Fast Spread (FS) [88]. In this algorithm, replication of the requested file is carried out in all nodes between the source node and the client node. In the case of insufficient storage in nodes, one or more previously stored replicas are replaced with new replicas. There are two traditional replacement algorithms for LRU and LFU. The combination of FS with LRU deletes the least recently replica first while a combination of FS with LFU deletes the least frequency replica first. It is necessary to note that some issues arise when the existing replicas have a higher value with respect to the new replica.

This problem has been discussed by Bsoul et al. [89] that introduce Enhanced Fast Spread (EFS) strategy. EFS strategy considered frequency and the number of requests, the replica size, and the last time in which the replica was requested in replica value assignment. Simulation results confirmed the higher performance of EFS in comparison with the strategies of Fast Spread, FS-LFU, and FS-LRU combination.

Table 1e. Features of replication algorithms.

Strategy	JeyaSheeli et al. [82]	Thuraisingham et al. [83]	Shorfuzzaman et al. [84]	Sashi et al. [86]	Mansouri [87]	Bsoul et al. [89]
Year	2012	2010	2010		2012	2010
Replication type	Dynamic	Dynamic	Dynamic	Dynamic	Dynamic	Dynamic
Replication management	Centralized	Decentralized	Centralized	Centralized	Centralized	Centralized
Replication actor	Middleware service	Middleware service	Middleware service	Middleware service	Middleware service	Middleware service
Grid topology	General	Graph	Multi-tier	Multi-tier	Multi-tier	Graph
Availability	+	+	+	+	+	+
Popularity	+	+	+	+	+	+
Response time	+	+	+	+	+	+
Fault tolerance	-	+	-	-	-	-
Bandwidth consumption	Low	Low	+	Low	Low	Low
Consistency	-	-	-	-	-	-
Storage assumption	Limited	Limited	Limited	Limited	Limited	Limited
File access pattern	NR	Random	Sequential, random, unitary random, Gaussian random walk, random Zipf	Random Zipf	Sequential, random, unitary random, Gaussian random walk,	NR
Optimal number of replicas	-	-	-	+	+	+
Place of replica	+	+	+	+	+	-
Replica condition	NR	NR	Set a threshold	Invoked at regular Intervals	Invoked at regular intervals	When requested file is not in site
Integration with job scheduler	-	-	-	-	-	-
Validation method	Simulation	Simulation	Simulation	Simulation	Simulation	Simulation
Main additional features	Computing access cost	Secure storage mechanism	Adaptive popularity-driven replica placement	Set different weight for data	Concept of exponential decay	Keep the important replica

Khanli et al. [90] proposed a replication strategy named PHFS methodology to determine the relationship between files for the next time interval. It is employed to diminish the latency of data access. It estimates the future usage of files and pre-replicates the candidate files in hierarchal Date Grid between the sources and requester sites. Accordingly, at first, the access information of all files called file access log is created and then, PHFS by employing data mining techniques (e.g., association rules) clusters the file access logs. PHFS groups files that have a high probability of accessing each other.. Then, it forms the most frequent files based on the sequential access pattern and these files have logical spatial locality. PHFS uses a predictive working set (PWS) when one request is triggered and then it replicates all items of PWS and requested file on all sites in the path. PHFS is appropriate in a situation that the user works in the same context for a long time and his requests are not random. In summary, PHFS has three phases. (1) Monitoring step gets the file access reports from all sites and creates a global log file. (2) Analyzing step extracts a pattern of files based on the data mining technique. (3) Replication step creates replicas of PWS and manages replicas in the system.

Park et al. [91] proposed a Bandwidth Hierarchy based Replication (BHR) with the ability to decrease data access time by enhancing network-level locality as well as forbidden network congestions. This algorithm partitions the sites into

various regions in a way that the bandwidth within the regions is higher than the bandwidth among the regions. The main advantage of BHR is its ability to copy files, which are likely to be needed frequently in the near future. BHR considers the region as well as cooperation of sites in replication decision. If the necessary files are not available on the local site and there is enough free space, BHR transfers it form the provider site and replicates it. But if there is insufficient storage, the replacement strategy is triggered. Firstly, the presence of new replica on another site in the local region was checked and removed if it exists. Otherwise, another region that has the requested replica is determined. Then, to provide enough free space, any unpopular file must be deleted. All files are arranged with considering the access frequency. BHR deletes files from the sorted list until sufficient space is available. The simulation results confirmed the better performance of BHR especially in the case of a small capacity of storage elements is available. Modified BHR [92] is an advanced version of MBHR that considers access frequency and the possibility of needed files in the future as criteria for replica placement.

A Hierarchical Cluster Scheduling strategy (HCS) and Hierarchical Replication Strategy (HRS) are proposed by Chang et al. [93] to improve the accessibility of data in Grid. The former strategy (HCS) assumed hierarchical scheduling as well as cluster information to diminish the time of search for the proper computing node. HRS method employed the concept of network locality similar to BHR strategy. The replica creation is carried out in the case of the presence of sufficient free storage space. If enough storage is not available and the candidate replicas for the deletion belongs to the site of the same cluster, it is situated to be deleted and replaced with a new replica. In the case of the replica belonging to the other cluster, some files must be deleted in the following consequence. Firstly, the replicas that exist in other sites of the same cluster must be removed. If the storage of the free space is still not enough, the least frequently used replicas must be deleted in the same trend. Accordingly, the main advantage of HRS can be expressed as the consideration of inter-cluster file transference, the possibility of checking site providers for best replica, and employment of popularity of replicas at the site level compared to HRS that employed popularity of replica at the cluster level. A combination of HCS scheduling with HRS replication strategy reduced the data access time as well as the amount of inter-cluster communication compared to the different scheduling algorithms and replication strategies. However, deletion and selection of replica in HRS are carried out only based on the bandwidth as criterion.

Pérez et al. [94] illustrated a replication algorithm as Branch Replication Scheme (BRS) with three characteristics with respect to the present algorithm that is optimization of storage usage by preparation of sub replicas; enhancing the data access performance using parallel I/O methodologies, and preparing the possibility of replica modification based on consistency technique during updating file. In this strategy, the employment of naming arrangement strategy is carried out on the base of RNS standards. The main characteristics of BRS are given in the following.

- Root replica: a site that contains the original file is named root.
- Parallel replication: to create a new replica, n nodes were candidates to save the sub-replicas. BRS is able to break the primary replica into chunks and generates sub-replicas by copying the chucks in a parallel manner to target nodes based on the GridFTP. In the current approach, it was possible to decrease the replication time in comparison to the time necessary to generate the complete replica of the single storage node.
- Partial replication. It copies file fragments with consideration of popularity or geographic distribution metrics based on the parallel strategy.
- Parallel data access. Parallel I/O is applied as GRIDFTP and parallel file system to access data of different sites.
- Better resource usage. BRS consumes low storage space for storing fragmentations of the file. Therefore, it is suitable for a network with a high storage restriction.

This method is able to check the writing, the reading or the updating of available replicas. The authors modeled and compared hierarchical (HRS), server-directed (SDRS), and branched replication (BRS). The simulation results confirmed that BRS is able to reduce data access time for various file sizes during the reading and the writing operations.

In [95], a 3-level hierarchical strategy is used to propose a new replication algorithm. In the proposed structure, regions show the first level and linked by low bandwidth (i.e., Internet). The second level is the local LANs within the region, which have a higher bandwidth with respect to the first level. The third level is determined by the computer within each local LAN with a very high bandwidth within the interconnections. According to this strategy, replication feasibility is checked at first. If the requested file size is higher than storage size, the file will be used from a remote location. Replica selection of proposed strategy selects provider site that has the highest bandwidth from the requester site. The same trend is repeated during the deletion of file. This trend is able to enhance the performance compared to LRU replacement strategy. Moreover, to provide an efficient scheduling, their algorithm is able to select the best region, LAN, and site. For example, the region with the most essential files is known as the best region.

A Dynamic Hierarchical Replication (DHR) is proposed by Mansouri et al. [96]. This strategy enables us to store a new replica in appropriate location with the highest number of requests. Similar to Horri et al. [95], a 3-level topology is considered three levels as regions, LANs and nodes within each LAN. Moreover, the proposed strategy decreases the access latency with determination of the best replica in the case of the presence of several replicas in various sites. It takes into account the number of requests waiting in the queue and data transfer time in replica selection step. The proposed algorithm could decrease the job execution time in system with storage restriction.

Mansouri et al. [97] illustrated a Combined Scheduling Strategy (CSS) that employs the number of jobs waiting in the queue, the location of required data for the job, and the computational capability as criteria for making a decision. The proposed strategy is able to improve file access time by a good estimation of the responsible time (RT) that is given by

$$RT = T1 + T2 \tag{3}$$

where T1 is the storage access latency and T2 is the waiting time in the request queue. The simulation results showed that the proposed algorithm could decrease the job execution time and the number of replication compared to the other strategies, especially, in storage restriction. Moreover, the results confirmed that by increasing file size as well as the number of jobs, the performance is enhanced, significantly.

Table 1f. Features of replication algorithms.

Strategy	Khanli et al. [90]	Park et al. [91]	Sashi et al. [92]	Chang et al. [93]	Pérez et al. [94]	Horri et al. [95]
Year	2011	2004	2010	2007	2010	2008
Replication type	Dynamic	Dynamic	Dynamic	Dynamic	Dynamic	Dynamic
Replication management	Decentralized	Decentralized	Decentralized	Decentralized	Centralized	Decentralized
Replication actor	Middleware service	Middleware service	Middleware service	Middleware service	Middleware service	Middleware service
Grid topology	Multi-tier	Multi-tier	Multi-tier	Multi-tier	Multi-tier	Multi-tier
Availability	+	+	+	+	+	+
Popularity	+	+	+	+	+	-
Response time	+	+	+	+	+	+
Fault tolerance	-	-	-	-	+	-
Bandwidth consumption	Low	Low	Low	+	+	+
Consistency	-	-	-	-	+	-
Storage assumption	Limited	Limited	Limited	Limited	Limited	Limited
File access pattern	Random	Sequential	Zipf access, Sequential	Sequential	NR	Sequential
Optimal number of replicas	-	-	-	-	+	-
Place of replica	+	-	+	-	+	-
Replica condition	When requested file is not in site	When requested file is not in site	When requested file is not in site	Invoked at regular intervals	NR	When requested file is not in site
Integration with job scheduler	-	-	-	+	-	+
Validation method	Simulation	Simulation	Simulation	Simulation, Real environment	Simulation	Simulation
Main additional features	Predict future need	Network-level locality	Store replica in a particular site	Inter-cluster communications cost	Branch replication scheme	The differences between intra-LAN and inter-LAN

According to the literature, in spite of DHR's advantages, i.e. some improvements in performance metrics like mean job time; it suffers from some definitions due to low efficiency during replica selection and the replica replacement. Therefore, Modified Dynamic Hierarchical Replication Algorithm (MDHRA) that improves DHR strategy is proposed by Mansouri et al. [98]. MDHRA has two steps for storing a new replica in the case of insufficient space. Firstly, the files with the lowest transferring time are deleted and then if the free space is still insufficient, it removes replicas based on the last time that replica requested, access number and replica file size. When various sites hold the same replica, replication strategy improves the access latency by selection of the best replica provider based on the data transfer time, the storage access latency, the replica requests waiting in the queue and the distance between nodes. They also proposed a novel scheduling strategy that finds the most suitable region, i.e. the region with the most requested files. This trend is able to decrease the total transfer time, and, as a consequence, decrease the traffic of the network. The number of jobs waiting in the queue, the location of required data for the job, the degree of parallelism and computational capability are other parameters that are considered by this strategy.

Mansouri et al. [99] proposed a new replication algorithm, named Enhanced Dynamic Hierarchical Replication (EDHR) that modifies the Dynamic Hierarchical Replication (DHR) strategy [31]. In the first step, EDHR determines the appropriate location in a local region for storing new replica based on the frequency of requests for the replica and the last time the replica was requested. EDHR could reduce access time significantly since the two former parameters imply the probability of requesting the file in the near future. In addition, the authors presented an economic model based on the future value of a data file in the replacement step for providing sufficient space. The experiments

demonstrated that the proposed strategy reduced the storage usage and the number of replications in a hierarchal structure.

Parallel Job Scheduling (PJS) and Threshold-based Dynamic Data Replication (TDDR) algorithms are proposed by Mansouri et al. [100]. The former (PJS) decreased the time of the search in a parallel manner by employing hierarchical scheduling structure. PJS considers network characteristics, a number of jobs waiting in the queue, file locations, and disk read speed of storage drive at data sources as criteria to improving the performance.

The new version strategy, i.e. TDDR, works based on data request arrival rates and available storage capacities in the determination of the dynamic threshold. To compensate the storage space limitation in every node, an efficient replica replacement strategy is proposed in two steps. Firstly, the files with the lowest transferring time are removed, and secondly, in the case of insufficient storage space, the last time the replica was requested, number of access, size of replica and file transfer time must be considered for the further replica removing. Simulation results confirmed that the proposed strategy has a higher performance compared to the other algorithms in such metrics as Mean Job Time, Number of Inter-communications, Number of Replications, Computing Resource Usage, and Effective Network Usage.

Another Hierarchical Job Scheduling Strategy (HJSS) and Advanced Dynamic Hierarchical Replication Strategy (ADHRS) are proposed by Mansouri [101]. HJSS employs network characteristics, that are the number of jobs waiting in the queue, location of the file, and disk read speed of storage drive at replica providers, as criteria for replica selection. While the ADHRS considers file transfer time as in the following relation,

if ($B_{ji} < DiskSpeed_i$)

$$TransferTime_{fi} = PropagationDelay_{ij} + (|f_i| * 8) / B_{ji}$$

else

$$TransferTime_{fi} = PropagationDelay_{ij} + |f_i| / Disk\ Speed_i \quad (4)$$

where B_{ji} is the bandwidth from site S_i to the site S_j that f_i resides. $DiskSpeed_i$ is data transfer rate (in bytes per second) of storage drives of the resource in S_i. PropagationDelayij is propagation delay/network latency (in seconds) from site S_j to site S_i.

Mansouri et al. [102] designed a QoS Data Replication (QDR) strategy. To guarantee the effectiveness and to ensure the contentment of the grid users, the proposed strategy replicates required files in a timely way and in a secure technique. QDR considers the replica selection issue as the main goal. The main advantages of QDR can be depicted as the employment of response time estimation and security parameter during replica selection process. Moreover, QDR uses storage access latency, the distance between nodes, IDS capabilities, firewall capabilities, authentication mechanism, secured file storage capabilities, interoperability in selecting the best replica provider. Due to the restriction of storage capacity for each site, the employment of efficient replica replacement is inevitable. To solve this issue, the authors used the availability of the file, the last time the replica was requested, the number of access, and the size of the replica as criteria for data replica replacement step. With considering the strategy's mean job execution time, Effective Network Usage, SE usage, Replication frequency, and Hit ratio were used as the performance evaluation metrics. The simulation results confirmed a better overall performance of QDR than the reported algorithms' in Data Grid.

Bsoul et al. [103] designed a round-based data replication strategy with the name of Improved Popular File Replicate First (IPFRF). IPFRF method is able to select the most appropriate file at the final of every round using the number that was requested in the last round and the file size as criteria. This strategy is the advanced version of IPFRF [104] in which the replica storing in the most appropriate cluster node is carried out with consideration of the number of requests, free storage space, and node centrality. In the case of insufficient free storage space, using the popularity threshold it removes the lowest popularity files to provide enough space. To compare IPFRF with PFRF two different conditions are considered. Firstly, requesting a file by cluster nodes has a uniform distribution. In this case, requesting for any file has the same probability. Secondly, requesting a file by cluster nodes has Zipf distribution. The simulation results confirmed that IPFRF could decrease the average file delay per request up to 18.00% and 55.84% in the same condition. Moreover, IPFRF strategy could improve the file found percentage up to 46.69% and 217.81 % in the first and second conditions, respectively.

A new schema with the name of Efficient Replica Consistency Model (ERCM) is proposed by Guroob et al. [105]. The main advantage of ERCM is its ability to decrease job execution time as well as enhancing the replica consistency by updating propagation. The first goal is satisfied by optimal allocation of replicas to minimize the retrieval time. ERCM uses the local hash table list for finding the needed files. If necessary files are not available on the local site, the proposed strategy must transfer the necessary files from the master server to local site and replicate it for the future execution. The second goal is satisfied by optimal allocation of replica consistency and updating propagation in the writing process. ERCM contains an asynchronous replica consistency inter-replica site to guarantee replica consistency for changes that frequently happen by users, and then the updates are distributed simultaneously, where the master server performs the propagation operations to all sites that have a similar replica that was just modified. The experiments showed that the ERCM has a reasonable execution time for reading and writing operations with a high availability. Unfortunately, they did not check the update propagation methods under various factors settings.

Mostafa et al. [106] presented a new replication strategy based on the neural network to reduce the response time for satisfying the user's data requirements. The presented model predicates the location of necessary files based on the predictive component. It finds the location of the file either in the cache, local resources, or remote ones with the help artificial neural network and the previous history of file accesses. An artificial neural network determines the location of the file after training features taken from GridSim. The artificial neural network is a computing system that is trained to determine the patterns of its inputs. In this case, the job is to associate different combinations of job requirements with file locations. The overhead of prediction depends on the type of application. For example, the overhead and the delay of prediction process in small applications are more advisable than applications with higher execution times. Their results showed that the delay for this estimation process is significantly lower than that associated with the stored replica catalogue search algorithm. The main drawback of their work is that they considered the state-of-the-art search and prediction procedure used in closely related disciplines to improve accuracy and overhead.

Table 1g. Features of replication algorithms.

Strategy	Mansouri et al. [96]	Mansouri et al. [97]	Mansouri et al. [98]	Mansouri et al. [99]	Mansouri [100]	Mansouri [101]
Year	2012	2012	2012	2013	2013	2013
Replication type	Dynamic	Dynamic	Dynamic	Dynamic	Dynamic	Dynamic
Replication management	Decentralized	Decentralized	Decentralized	Decentralized	Decentralized	Decentralized
Replication actor	Middleware service	Middleware service	Middleware service	Middleware service	Middleware service	Middleware service
Grid topology	Multi-tier	Multi-tier	Multi-tier	Multi-tier	Multi-tier	Multi-tier
Availability	+	+	+	+	+	+
Popularity	+	+	+	+	+	+
Response time	+	+	+	+	+	+
Fault tolerance	-	-	-	-	-	-
Bandwidth consumption	Low	Low	Low	Low	Low	Low
Consistency	-	-	-	-	-	-
Storage assumption	Limited	Limited	Limited	Limited	Limited	Limited
File access pattern	Sequential	Sequential	Sequential	Sequential	Sequential, random, unitary random, Gaussian random walk, random Zipf	Sequential, random, unitary random, Gaussian random walk, random Zipf
Optimal number of replicas	-	-	-	-	+	-
Place of replica	+	+	+	+	+	+
Replica condition	When requested file is not in site	When requested file is not in site	When requested file is not in site	When requested file is not in site	Set a threshold	When requested file is not in site
Integration with job scheduler	-	+	+	+	+	+
Validation method	Simulation	Simulation	Simulation	Simulation	Simulation	Simulation
Main additional features	The replica requests that waiting in the storage	Storage Speed	Distance between nodes	Economic model for file deletion	Dynamic threshold	File transfer time

Hamrouni et al. [107] presented an acronym of Replication Strategy based on Correlated Patterns (RSCP) with consideration of granularity as a set of correlation files. By employing data mining technique, the maximal frequent correlated strategy determines the groups of related files. The strongpoint of RSCP is the strengthening of the support measure and the ability to discover the most correlated files. The authors investigated the impact of important parameters such as the time interval and the support value on the execution time. The simulation results revealed that RSCP could decrease the execution time with respect to the other reported strategies. The weakness of this strategy is the consideration of read-only data and the algorithm ignoring the consistency issues.

Rahmani et al. [108] proposed a new replica placement strategy to balance the load of the system. In the first step, it determines the hot spot and under-utilized servers based on their workload. In the second step, it replicates files on the best server, i.e. a server that has the lowest access load. In the third step, it specifies the optimized distance between the requester site and the replica provider site. Therefore, the suggested strategy could decrease data transfer time significantly. In the fourth step, they implemented a tree topology to show the Grid environment. In the sequel, the authors assigned a label to each node based on the Dewey Encoding that is generally applied in XML databases operations. The simulation results indicated that the proposed replication algorithm provides a good load balancing in the system and enhances mean response time.

Table 1h. Features of replication algorithms.

Strategy	Mansouri et al. [102]	Bsoul et al. [103]	Guroob et al. [105]	Mostafa et al. [106]	Hamrouni et al. [107]	Rahmani et al. [108]
Year	2016	2016	2016	2015	2015	2015
Replication type	Dynamic	Dynamic	Dynamic	Dynamic	Dynamic	Dynamic
Replication management	Decentralized	Decentralized	Decentralized	Centralized	Decentralized	Decentralized
Replication actor	Middleware service	Middleware service	Middleware service	Middleware service	Middleware service	Middleware service
Grid topology	Multi-tier	Multi-tier	Multi-tier, Random graph	General	General	Tree structure
Availability	-	+	+	-	+	+
Popularity	-	+	-	-	+	+
Response time	+	+	+	+	+	+
Fault tolerance	+	-	-		-	-
Bandwidth consumption	Average	Low	Average	Average	Low	Low
Consistency	-	-	+	-	-	+
Storage assumption	Limited	Limited	Limited	Limited	Limited	Limited
File access pattern	Sequential, random, unitary random, Gaussian random walk	Uniform, random Zipf	Random Zipf	NR	Sequential, random, Random Zipf, Gaussian random walk	random, Random Zipf
Optimal number of replicas	-	+	-	-	-	-
Place of replica	-	+	+	+	+	+
Replica condition	NR	-	-	-	-	-
Integration with job scheduler	-	-	-	-	-	-
Validation method	Simulation	Simulation	Simulation	Simulation	Simulation	Simulation
Main additional features	Security factors	Round based strategy	Update propagation	ANN prediction mechanism	Set of correlated files	New labeling scheme

10- Simulation Results And Discussion

We evaluated new replication strategies as HRS, LALW, BHR, EFS, DHRA, Modified BHR, MDHRA, TDDR, and QDR with OptorSim simulator. Figure 5 indicates the network structure in our simulation. In addition, main parameters that are set in OptorSim are presented in Table 2. We set 10 GFLOPs for the speed of CPU and 300 for the maximum queue size of computing element. We generate 50 different job types that each one on average needs 15 files with 2 GB for completion. Simulator randomly chooses jobs according to the job probability and sends them to the resource broker at regular intervals.

We can see that some types of jobs would be selected frequently, thus some replicas are required repeatedly. We set Queue Access Cost scheduler, which assigns the job to the site with lowest access cost for the job and all jobs in the queue, in our replication strategies evaluation. Figure 6 presents the mean job time is determined as the average time needed to run a job starting from the time it is scheduled to the site until it complete processing all of the needed files. This is a very common measure in evaluation of replication strategy in the Grid environment.

In BHR strategy, replicas that have a high probability of being needed again are placed in a location with a high bandwidth. In other words, it stores different replicas in a region to provide the benefit from the network-level locality. In sequential access pattern, HRS strategy has 6% lower mean job time in comparison to the BHR. This is due to the

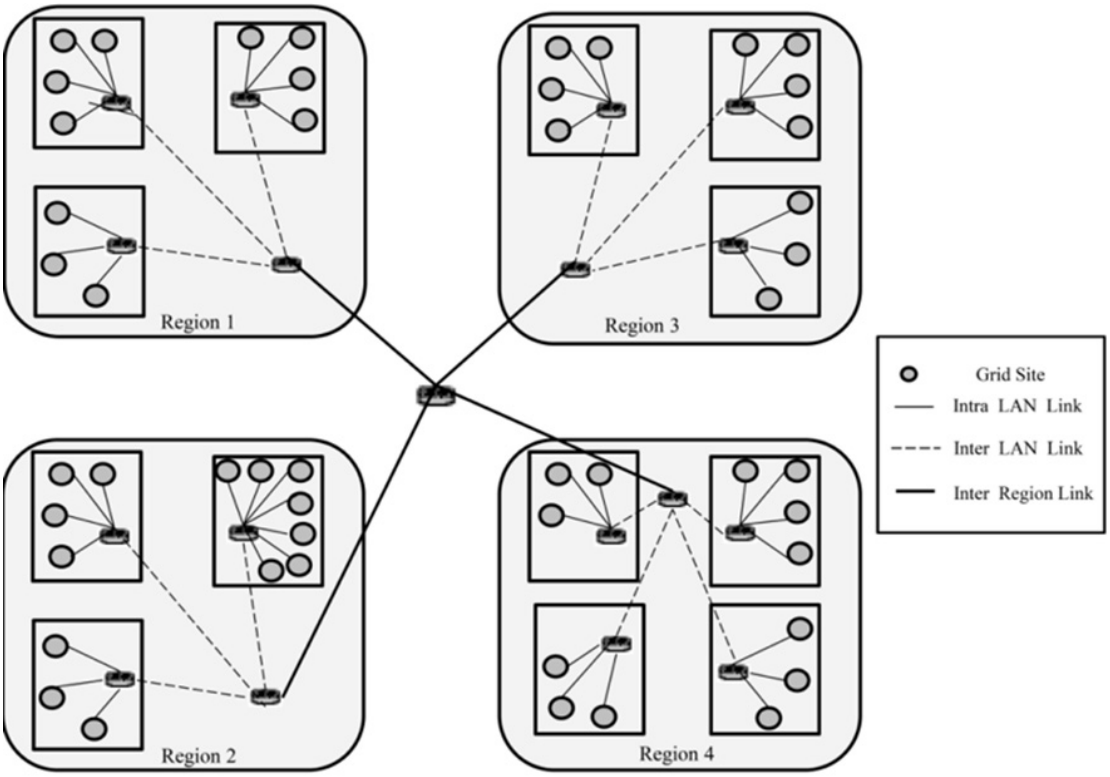

Fig. 5. Grid topology in the simulation

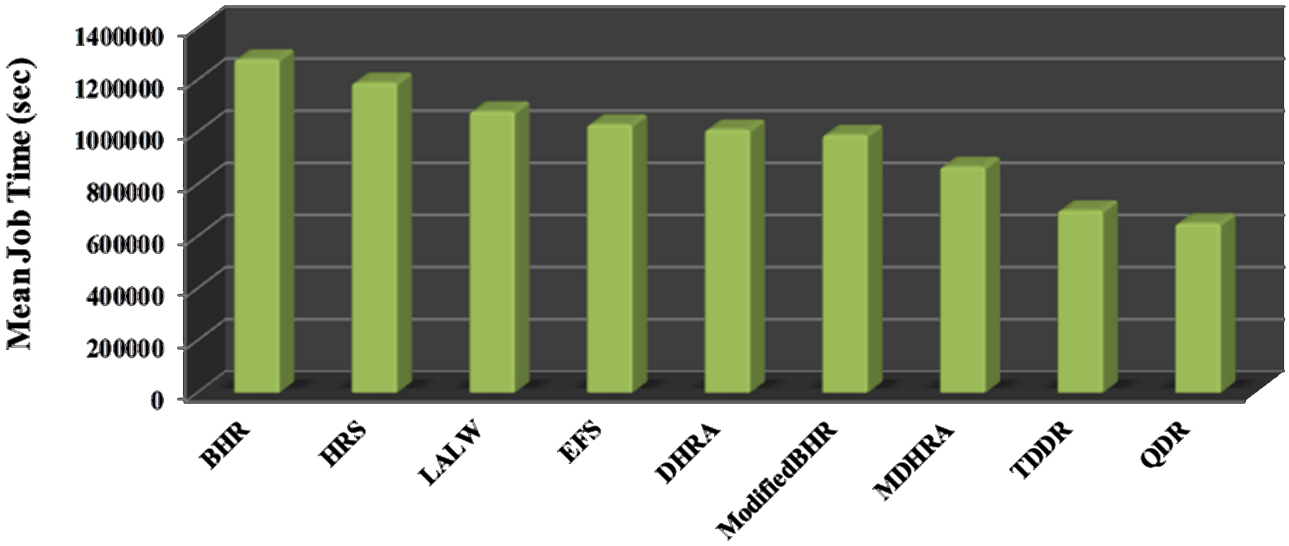

Fig. 6. Mean Job time for different replication strategies.

fact that HRS assigns a high priority to the required replica of the local cluster while BHR algorithm searches all locations for replica selection.

EFS strategy shows that the mean job time is lower (about 7%) than the one in LALW strategy. The main reason is that when free space is insufficient, EFS removes a set of candidate replicas only if the value of new replica is more than the value for a set of candidate replica. LALW strategy executed faster (on average 15%) compared to BHR algorithm. The main advantage of LALW is that it assigns an appropriate weight to the data file based on the time of access. Then, it replaces only the most important file. In Zipf access pattern, TDDR has the mean job time 50% lower than LFU strategy, and 22% lower than ModifiedBHR strategy. Figure 6 indicates that QDR strategy executes all jobs at the lowest time in all different distributions. Especially in random distribution, a particular file has a high probability to be requested again, thus the most of the required files have been stored before. Consequently, QDR methods and also all the other replication algorithms are preferable for random file access patterns.

11- Conclusion And Future Research

In this work, we reviewed the different data replication algorithms on the basis of different parameters. Also, the

Table 2. Parameter setting in the simulation.

Topology parameter	Value
Region number	4
Size of storage (GB)	50
Bandwidth of Inter LAN	1000
Bandwidth of Intra LAN	100
Bandwidth of Intra Region	10
Job configuration	Value
Access pattern	RandomZipf
Job number	1500
Number of jobs types	50
Number of file access per jobs	16
File size (GB)	2
Job delay (msec)	10000

different replication strategies were described and also their simulation results for various parameters are discussed. A comparison of the different strategies with respect to various parameters such as availability, reliability, fault tolerance, response time, etc. were presented. From this review paper, it can be observed that there are still a lot of investigations to be done in the area of data replication for Data Grid system. Some open research issues are presented below.

We see that there is no particular architecture with proper properties for a Data Grid system. Most of the related works considered a hierarchal structure but actually, a general graph is a more realistic topology. The main reason for this choose is that the hierarchal structure reduces the message transfer time of communications in each tier. In addition, it decreases the number of messages transferred. Several studies modified the hierarchal architecture to show the real Grid structure.

Most recent existing works investigated the effect of access patterns on the different algorithms. It is obvious that data access patterns change during jobs execution, thus dynamic replication strategy must monitor data access distribution to determine the number and the location of the replica in a suitable way. Different investigations demonstrated that Gaussian and Zipf distributions commonly model the behavior of many applications, which apply Data Grid as their data and computing system, and are widely applied in statistics and many statistical evaluations. It is interesting that the Sequential access pattern is appropriate for some applications such as high-energy physics experiments.

In a distributed system, data files can be stored in various locations. It makes the redirection of requests for that data file to the most suitable site possible. Different critical decisions need to be made, for example, the number of replicas to be stored, replica replacement, and the location of new replicas in order to improve performance. In other words, this target is expressed as the QoS requirements such as high-reliability level, and low infrastructure cost from user and service provider side. Different studies showed the minimization of replication cost is an NP-hard problem. Indeed, there are many powerful techniques for replica placement. Few studies have focused on the QoS factors. Most of them considered several performance parameters such as access latency in replication decisions. Although these factors of performance are critical, they neither guarantee any high-performance level nor provide the diverse QoS requirements of individuals adequately.

Another important problem that is completely ignored by most researchers is consistency. Users may modify data files and present a critical issue of maintaining data consistency among replicas stored in various Grid sites. For that reason, how to maintain the consistency of those replicas is the main question. Therefore, adaptive replica consistency will be the best solution. Adaptive consistency is based on such a fact that if the demand of an application is satisfied, the consistency level can be relaxed wisely, and the update of replicas causes a slight delay.

Most of the studies about Grid environment used the simulator to test the performance of algorithms. As the next phase of study, these replication techniques must be evaluated and implemented in a real Grid environment. Then we have very realistic results of the suppositions that have been made for those methods.

Several papers tried to estimate the future file requirements based on the past access sequences. We can see that data mining tasks would be a good idea in the prediction of file access patterns. Finally, fault tolerance and scalability of local and global databases can be further research issues in Data Grid environment.

REFERENCES

[1] M. Chetty, R. Buyya, Weaving computational grids: How analogous are they with electrical grids?, Computing in Science and Engineering, 4(4) (2002) 61.

[2] I. Foster, What is the Grid? - a three point checklist, citeulike, 1(6) (2002) 1-4.

[3] C.H. Schulbach, Nasa's Information Power Grid Project, Computational Aerosciences in the 21st Century, (2000) 11.

[4] A. Chervenak, I. Foster, C. Kesselman, C. Salisbury, 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(3) (2000) 187-200.

[5] G. Zhou, F. Lian, G. Li, Influence of alloy elements on magnetic properties of Fe-based amorphous alloys, JOURNAL OF MATERIALS SCIENCE AND TECHNOLOGY-SHENYANG-, 16(2) (2000) 157-158.

[6] N. Chau, N.H. Luong, N.X. Chien, P.Q. Thanh, L.V. Vu, Influence of P substitution for B on the structure and properties of nanocrystalline Fe73.5Si15.5Nb3Cu1B7−xPx alloys, Physica B: Condensed Matter, 327(2–4) (2003) 241-243.

[7] M. Müller, N. Mattern, The influence of refractory element additions on the magnetic properties and on the crystallization behaviour of nanocrystalline soft magnetic Fe-B-Si-Cu alloys, Journal of Magnetism and Magnetic Materials, 136(1–2) (1994) 79-87.

[8] W. Liu, J. Tang, Y. Du, Nanocrystalline soft magnetic ribbon with α″-Fe16N2 nanocrystallites embedded in amorphous matrix, Journal of Magnetism and Magnetic Materials, 320(21) (2008) 2752-2754.

[9] I. Maťko, E. Illeková, P.Š. Sr, P. Švec, D. Janičkovič, V. Vodárek, Microstructural study of the crystallization of amorphous Fe–Sn–B ribbons, Journal of Alloys and

Compounds, 615, Supplement 1(0) (2014) S462-S466.

[10] R.M. Rahman, R. Alhajj, K. Barker, Replica selection strategies in data grid, Journal of Parallel and Distributed Computing, 68(12) (2008) 1561-1574.

[11] D.T. Nukarapu, B. Tang, L. Wang, S. Lu, Data replication in data intensive scientific applications with performance guarantee, IEEE Transactions on Parallel and Distributed Systems, 22(8) (2011) 1299-1306.

[12] C.Z. X. Meng, An ant colony model based replica consistency maintenance strategy in unstructured P2P networks, Computer Networks, 62 (2014) 11.

[13] R.M. Rahman, K. Barker, R. Alhajj, Replica placement strategies in data grid, Journal of Grid Computing, 6(1) (2008) 103-123.

[14] A. Benoit, V. Rehn-Sonigo, Y. Robert, Replica placement and access policies in tree networks, IEEE Transactions on Parallel and Distributed Systems, 19(12) (2008) 1614-1627.

[15] H.H.E. Al-Mistarihi, C.H. Yong, On fairness, optimizing replica selection in data grids, IEEE Transactions on Parallel and Distributed Systems, 20(8) (2009) 1102-1111.

[16] K. Skakowski, R. Sota, D. Król, J. Kitowski, QoS-based storage resources provisioning for grid applications, Future Generation Computer Systems, 29(3) (2013) 713-727.

[17] Y.S. G. Belalem, A Consistency Protocol Multi-Layer for Replicas Management in Large Scale Systems, World Academy of Science Engineering and Technology, 16 (2008) 6.

[18] P.C. D. Li, M. Dahlin, WCIP: Web Cache Invalidation Protocol, in: IETF Internet Draft, 2002.

[19] C.T. Wilkes, R.J. LeBlanc Jr, Distributed locking: A mechanism for constructing highly available objects, in: Proceedings - Symposium on Reliability in Distributed Software and Database Systems, 1988, pp. 194-203.

[20] A. Devulapalli, D. Dalessandro, P. Wyckoff, Data structure consistency using atomic operations in storage devices, in: Proceedings - 5th IEEE International Workshop on Storage Network Architecture and Parallel I/Os, SNAPI 2008, 2008, pp. 65-73.

[21] G.P. S. Ceri, Databases- Principles and Systems, McGraw-Hill, 1985.

[22] A. Domenici, F. Donno, G. Pucciani, H. Stockinger, K. Stockinger, Replica consistency in a Data Grid, Nuclear Instruments and Methods in Physics Research, Section A: Accelerators, Spectrometers, Detectors and Associated Equipment, 534(1-2) (2004) 24-28.

[23] M. Tang, B.S. Lee, X. Tang, C.K. Yeo, The impact of data replication on job scheduling performance in the Data Grid, Future Generation Computer Systems, 22(3) (2006) 254-268.

[24] M.M. Deris, J.H. Abawajy, H.M. Suzuri, An efficient replicated data access approach for large-scale distributed systems, in: 2004 IEEE International Symposium on Cluster Computing and the Grid, CCGrid 2004, 2004, pp. 588-594.

[25] O. Tatebe, Y. Morita, S. Matsuoka, N. Soda, S. Sekiguchi, Grid datafarm architecture for petascale data intensive computing, in: 2nd IEEE/ACM International Symposium on Cluster Computing and the Grid, CCGrid 2002, 2002.

[26] P.K. Kesselman, M. Ripeanu, B. Schwartzkopf, H. Stockinger, K. Stockinger, B. Tierney, A Framework for Constructing Scalable Replica Location Services, in: Supercomputing, ACM/IEEE Conference, 2002, pp. 1-7.

[27] J. Ma, W. Liu, T. Glatard, A classification of file placement and replication methods on grids, Future Generation Computer Systems, 29(6) (2013) 1395-1406.

[28] U. Čibej, B. Slivnik, B. Robič, The complexity of static data replication in data grids, Parallel Computing, 31(8-9) (2005) 900-912.

[29] Y. Yuan, Y. Wu, G. Yang, F. Yu, Dynamic data replication based on local optimization principle in data grid, in: Proceedings of the 6th International Conference on Grid and Cooperative Computing, GCC 2007, 2007, pp. 815-822.

[30] H. Lamehamedi, Z. Shentu, B. Szymanski, E. Deelman, Simulation of dynamic data replication strategies in Data Grids, in: Proceedings - International Parallel and Distributed Processing Symposium, IPDPS 2003, 2003.

[31] K. Ranganathan, I. Foster, Identifying dynamic replication strategies for a high-performance data grid, in: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 2001, pp. 75-86.

[32] R.S. A.H. Elghirani, A.Y. Zomaya, A proactive Noncooperative Game-Theoretic Framework for Data Replication in Data Grids, in: Eighth IEEE International Symposium on Cluster Computing and the Grid (CCGRID), 2008, pp. 433-440.

[33] R.S. A. Elghirani, A.Y. Zomaya, Intelligent Scheduling and Replication in Data Grids: a Synergistic Approach, in: IEEE International Symposium on Cluster Computing and the Grid, 2007, pp. 179–182.

[34] M. Carman, F. Zini, L. Serafini, K. Stockinger, Towards an economy-based optimisation of file access and replication on a data grid, in: 2nd IEEE/ACM International Symposium on Cluster Computing and the Grid, CCGrid 2002, 2002.

[35] R.S. A. Elghirani, A.Y. Zomaya, Intelligent Scheduling and Replication in Data Grids: a Synergistic Approach, in, 2007, pp. 179–182.

[36] W.H. Bell, D.G. Cameron, R. Carvajal-Schiaffino, A.P. Millar, K. Stockinger, F. Zini, Evaluation of an economy-based file replication strategy for a data grid, in: Proceedings - CCGrid 2003: 3rd IEEE/ACM International Symposium on Cluster Computing and the Grid, 2003, pp. 661-668.

[37] H.J. Song, X. Liu, D. Jakobsen, R. Bhagwan, X. Zhang, K. Taura, A. Chien, The MicroGrid: A scientific tool for modeling Computational Grids, Scientific Programming, 8(3) (2000) 127-141.

[38] A. Takefusa, S. Matsuoka, H. Nakada, K. Aida, U. Nagashima, Overview of a performance evaluation system for global computing scheduling algorithms, in: IEEE International Symposium on High Performance Distributed Computing, Proceedings, 1999, pp. 97-104.

[39] H. Casanova, Simgrid: A toolkit for the simulation of application scheduling, in: Proceedings - 1st IEEE/ACM

International Symposium on Cluster Computing and the Grid, CCGrid 2001, 2001, pp. 430-437.

[40] R. Buyya, M. Murshed, GridSim: A toolkit for the modeling and simulation of distributed resource management and scheduling for grid computing, Concurrency Computation Practice and Experience, 14(13-15) (2002) 1175-1220.

[41] W.H. Bell, D.G. Cameron, L. Capozza, A.P. Millar, K. Stockinger, F. Zini, OptorSim: A grid simulator for studying dynamic data replication strategies, International Journal of High Performance Computing Applications, 17(4) (2003) 403-416.

[42] H. Sato, S. Matsuoka, T. Endo, N. Maruyama, Access-pattern and bandwidth aware file replication algorithm in a grid environment, in: Proceedings - IEEE/ACM International Workshop on Grid Computing, 2008, pp. 250-257.

[43] I.F. K. Ranganathan, Design and Evaluation of Dynamic Replication Strategies for a High Performance Data Grid, in: International Conference on Computing in High Energy and Nuclear Physics, 2001.

[44] C.B. Y. Dafei, H. Zhou, L. Xin, Z. Ke, F. Yu, Replication Strategy in Peer-to-Peer Geospatial Data Grid, in: Geoscience and Remote Sensing Symposium, 2007, pp. 5013-5016.

[45] P. Kněževíc, A. Wombacher, T. Risse, DHT-Based self-adapting replication protocol for achieving high data availability, in: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 2009, pp. 201-210.

[46] A. Abdullah, M. Othman, H. Ibrahim, M.N. Sulaiman, A.T. Othman, Decentralized replication strategies for P2P based scientific data grid, in: Proceedings - International Symposium on Information Technology 2008, ITSim, 2008.

[47] V. Andronikou, K. Mamouras, K. Tserpes, D. Kyriazis, T. Varvarigou, Dynamic QoS-aware data replication in grid environments based on data "importance", Future Generation Computer Systems, 28(3) (2012) 544-553.

[48] M. Shorfuzzaman, P. Graham, R. Eskicioglu, QoS-aware distributed replica placement in hierarchical data grids, in: Proceedings - International Conference on Advanced Information Networking and Applications, AINA, 2011, pp. 291-299.

[49] A. Jaradat, A. Patel, M.N. Zakaria, M.A.H. Amina, Accessibility algorithm based on site availability to enhance replica selection in a data grid environment, Computer Science and Information Systems, 10(1) (2013) 105-132.

[50] H.C. A. Husni, Response Time Optimization for Replica Selection Service in Data Grids, Journal of Computer Science, 4 (2008) 487-493.

[51] R.W. V. Vijayakumar, Security for Resource Selection in Grid Computing Based on Trust and Reputation Responsiveness, IJCSNS, 8 (2008) 12.

[52] C.W. Cheng, J.J. Wu, P. Liu, QoS-aware, access-efficient, and storage-efficient replica placement in grid environments, Journal of Supercomputing, 49(1) (2009) 42-63.

[53] B.M. Waxman, Routing of Multipoint Connections, in: IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATION, 1991, pp. 347–352.

[54] C.B. Y. Dafei, H. Zhou, L. Xin, Z. Ke, F. Yu, Replication Strategy in Peer-to-Peer Geospatial Data Grid, in: Geoscience and Remote Sensing Symposium, 2007, pp. 5013–5016.

[55] A.H.M.A. A. Jaradat, M.N. Zakaria, Balanced QoS Replica Selection Strategy to Enhance Data Grid, in: 2nd International Conference on Networking and Information Technology 2011, pp. 356-364.

[56] F.W. X. Nong, L. XiCheng, QoS-awared Replica Placement Techniques in Data Grid Applications, Science China Information Sciences, 53 (2010) 1487–1496.

[57] I.G. W. Jeon, K. Nahrstedt, QoS-aware Object Replication in Overlay Networks, in: Proceeding of Global Telecommunications, San Francisco, 2005, pp. 1–5.

[58] H. Wang, P. Liu, J.J. Wu, A QoS-aware heuristic algorithm for replica placement, in: Proceedings - IEEE/ACM International Workshop on Grid Computing, 2006, pp. 96-103.

[59] Z.S.-t. C. Dan-wei, R. Xun-yi, K. Qiang, Method for Replica Creation in Data Grids based on Complex Networks, The Journal of China Universities of Posts and Telecommunications, 7 (2010) 110–115.

[60] Z. Wang, T. Li, N. Xiong, Y. Pan, A novel dynamic network data replication scheme based on historical access record and proactive deletion, Journal of Supercomputing, 62(1) (2012) 227-250.

[61] R.M. Almuttairi, R. Wankar, A. Negi, C.R. Rao, A. Agarwal, R. Buyya, A two phased service oriented Broker for replica selection in data grids, Future Generation Computer Systems, 29(4) (2013) 953-972.

[62] K.W.R. J.F. Kurose, Computer Networking a Top–Down Approach Featuring the Internet, ADDISON-WESLEY, 2005.

[63] C.T. Yang, C.P. Fu, C.H. Hsu, File replication, maintenance, and consistency management services in data grids, Journal of Supercomputing, 53(3) (2010) 411-439.

[64] S.C. Choi, H.Y. Youn, Dynamic hybrid replication effectively combining tree and grid topology, Journal of Supercomputing, 59(3) (2012) 1289-1311.

[65] J. Taheri, A.Y. Zomaya, P. Bouvry, S.U. Khan, Hopfield neural network for simultaneous job scheduling and data replication in grids, Future Generation Computer Systems, 29(8) (2013) 1885-1900.

[66] T. Ma, Q. Yan, W. Tian, D. Guan, S. Lee, Replica creation strategy based on quantum evolutionary algorithm in data gird, Knowledge-Based Systems, 42 (2013) 85-96.

[67] J. Zhang, B.S. Lee, X. Tang, C.K. Yeo, Improving job scheduling performance with parallel access to replicas in Data Grid environment, Journal of Supercomputing, 56(3) (2011) 245-269.

[68] R.S. Chang, C.F. Lin, S.C. Hsi, Accessing data from many servers simultaneously and adaptively in data grids, Future Generation Computer Systems, 26(1) (2010) 63-71.

[69] I.F. K. Ranganathan, Computation and Data Scheduling in Distributed Data-Intensive Applications, in: In:

Proceedings of 11th IEEE inter- national symposium on high performance distributed computing (HPDC'02), 2002, pp. 352–358.

[70] Q.Y. T. Ma, W. Tian, D. Guan, S. Lee, Replica Creation Strategy based on Quantum Evolutionary Algorithm in Data Gird, Knowledge-Based Systems, 42 (2013) 85–96.

[71] I.F. S. Fitzgerald, C. Kesselman, G. Von Laszewski, W. Smith, S. Tuecke, A Directory Service for Configuring High-Performance Distributed Computations, in: The Sixth IEEE International Symposium on High Performance Distributed Computing, 1997, pp. 365-375.

[72] A.M.R. N. Saadat, PDDRA: A New Pre-Fetching based Dynamic Data Replication Algorithm in Data Grids, Future Generation Computer Systems, 28 (2012) 666-681.

[73] M. Lei, S.V. Vrbsky, X. Hong, An on-line replication strategy to increase availability in Data Grids, Future Generation Computer Systems, 24(2) (2008) 85-98.

[74] A. Horri, R. Sepahvand, G. Dastghaibyfard, A novel replication method in data grid, in: 2011 1st International eConference on Computer and Knowledge Engineering, ICCKE 2011, 2011, pp. 291-296.

[75] A.K. F.Z Bellounar, B.Yagoubi, Dynamic Data Grid Replication with Storage Constraint based on Cost Model, in: 2nd International Symposium on Modelling and Implementation of Complex Systems Constantine, 2012, pp. 11-16.

[76] K. Yi, H. Wang, F. Ding, Decentralized integration of task scheduling with replica placement, in: Proceedings - 9th International Symposium on Distributed Computing and Applications to Business, Engineering and Science, DCABES 2010, 2010, pp. 332-336.

[77] S.A. M. Maheswaram, H.J. Siegel, D. Hengsen, R. Freund, Dynamic Matching and Scheduling of a Class of Independent Tasks onto Heterogeneous Computing Systems, in: 8th Heterogeneous Computing Workshop (HCW'99), 1999.

[78] a.A.R. M. Osborne, A Course in Game Theory, MIT Press, 1994

[79] B. Meroufel, G. Belalem, Dynamic replication based on availability and popularity in the presence of failures, Journal of Information Processing Systems, 8(2) (2012) 263-278.

[80] a.M.A. P.G. JeyaSheeli, Efficient Centralized Data Replication Algorithm for Data Grids, in: International Conference on Computing, Electronics and Electrical Technologies, 2012, pp. 900-904.

[81] P.L. M. Tu, I.L. Yen, Secure Data Objects Replication in Data Grid, in: IEEE Transactions on Dependable and Secure Computing, pp. 50-64.

[82] M. Shorfuzzaman, P. Graham, R. Eskicioglu, Adaptive popularity-driven replica placement in hierarchical data grids, Journal of Supercomputing, 51(3) (2010) 374-392.

[83] R.S. Chang, H.P. Chang, Y.T. Wang, A dynamic weighted data replication strategy in data grids, in: AICCSA 08 - 6th IEEE/ACS International Conference on Computer Systems and Applications, 2008, pp. 414-421.

[84] K. Sashi, A.S. Thanamani, Dynamic replication in a data grid using a Modified BHR Region Based Algorithm, Future Generation Computer Systems, 27(2) (2011) 202-210.

[85] N. Mansouri, An effective weighted data replication strategy for data Grid, Australian Journal of Basic and Applied Sciences, 6(10) (2012) 336-346.

[86] P.L. M. Tu, I.L. Yen, Secure Data Objects Replication in Data Grid, IEEE Transactions on Dependable and Secure Computing, 7 (2007) 50-64.

[87] M. Bsoul, A. Al-Khasawneh, E.E. Abdallah, Y. Kilani, Enhanced fast spread replication strategy for data grid, Journal of Network and Computer Applications, 34(2) (2011) 575-580.

[88] L.M. Khanli, A. Isazadeh, T.N. Shishavan, PHFS: A dynamic replication method, to decrease access latency in the multi-tier data grid, Future Generation Computer Systems, 27(3) (2011) 233-244.

[89] S.M. Park, J.H. Kim, Y.B. Ko, W.S. Yoon, Dynamic data grid replication strategy based on internet hierarchy, in: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 2004, pp. 838-846.

[90] R.S. Chang, J.S. Chang, S.Y. Lin, Job scheduling and data replication on data grids, Future Generation Computer Systems, 23(7) (2007) 846-860.

[91] J.M. Pérez, F. García-Carballeira, J. Carretero, A. Calderón, J. Fernández, Branch replication scheme: A new model for data replication in large scale data grids, Future Generation Computer Systems, 26(1) (2010) 12-20.

[92] N. Mansouri, G.H. Dastghaibyfard, A dynamic replica management strategy in data grid, Journal of Network and Computer Applications, 35(4) (2012) 1297-1303.

[93] N. Mansouri, G.H. Dastghaibyfard, Job scheduling and dynamic data replication in data grid environment, Journal of Supercomputing, 64(1) (2013) 204-225.

[94] N. Mansouria, G.H. Dastghaibyfard, Enhanced dynamic hierarchical replication and weighted scheduling strategy in data grid, Journal of Parallel and Distributed Computing, 73(4) (2013) 534-543.

[95] B.D. Lee, J.B. Weissman, Dynamic replica management in the service grid, in: IEEE International Symposium on High Performance Distributed Computing, Proceedings, 2001, pp. 433-434.

[96] N. Mansouri, A Threshold-based Dynamic Data Replication and Parallel Job Scheduling strategy to enhance Data Grid, Cluster Computing, 17(3) (2014) 957-977.

[97] N. Mansouri, G.H. Dastghaibyfard, E. Mansouri, Combination of data replication and scheduling algorithm for improving data availability in Data Grids, Journal of Network and Computer Applications, 36(2) (2013) 711-722.

[98] N. Mansouri, Network and data location aware approach for simultaneous job scheduling and data replication in large-scale data grid environments, Frontiers of Computer Science, 8(3) (2014) 391-408.

[99] N. Mansouri, QDR: a QoS-aware data replication algorithm for Data Grids considering security factors, Cluster Computing, 19(3) (2016) 1071-1087.

[100] M. Bsoul, A.E. Abdallah, K. Almakadmeh, N. Tahat,

A Round-based Data Replication Strategy, IEEE Transactions on Parallel and Distributed Systems, 27(1) (2016) 31-39.

[101] M.C. Lee, F.Y. Leu, Y.P. Chen, PFRF: An adaptive data replication algorithm based on star-topology data grids, Future Generation Computer Systems, 28(7) (2012) 1045-1057.

[102] D.H.M. A.H Guroob, Efficient replica consistency model (ERCM) for update propagation in data grid environment, International Conference On Information Communication And Embedded System, (2016).

[103] I.A.R. N. Mostafa, A. Hamza, An Intelligent Dynamic Replica Selection Model within Grid Systems, in: Proceedings of the 8th IEEE GCC Conference and Exhibition, Muscat, 2015.

[104] T. Hamrouni, S. Slimani, F. Ben Charrada, A data mining correlated patterns-based periodic decentralized replication strategy for data grids, Journal of Systems and Software, 110 (2015) 10-27.

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

[106] N. Mostafa, I. Al Ridhawi, A. Hamza, An intelligent dynamic replica selection model within grid systems, in: 2015 IEEE 8th GCC Conference and Exhibition, GCCCE 2015, 2015.

Please cite this article using:

N. Mansouri and M. M. Javidi, A Survey of Dynamic Replication Strategies for Improving Response Time in Data Grid Environment, *AUT J. Model. Simul., 49(2)(2017)239-263*.

DOI: 10.22060/miscj.2016.874