Future Generation Computer Systems 28 (2012) 337–349

Contents lists available at SciVerse ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

A survey of dynamic replication strategies for improving data availability in data grids

Tehmina Amjad *, Muhammad Sher, Ali Daud

Department of Computer Science and Software Engineering, International Islamic University, Islamabad, Pakistan

ARTICLE INFO

Article history:
Received 11 April 2011
Received in revised form
12 June 2011
Accepted 30 June 2011
Available online 23 July 2011

Keywords:
Data grid
Data replication
Dynamic replication techniques
Replication strategies

ABSTRACT

Data grid is a distributed collection of storage and computational resources that are not bounded within a geophysical location. It is a fast growing area of research and providing efficient data access and maximum data availability is a challenging task. To achieve this task, data is replicated to different sites. A number of data replication techniques have been presented for data grids. All replication techniques address some attributes like fault tolerance, scalability, improved bandwidth consumption, performance, storage consumption, data access time etc. In this paper, different issues involved in data replication are identified and different replication techniques are studied to find out which attributes are addressed in a given technique and which are ignored. A tabular representation of all those parameters is presented to facilitate the future comparison of dynamic replication techniques. The paper also includes some discussion about future work in this direction by identifying some open research problems.

1. Introduction

Nowadays, in various scientific disciplines, large data sets are becoming an important part of shared recourses. In many fields, which are totally diverse in nature, like high energy physics, bioinformatics, earth observations, global climate changes, image processing, and data mining; the volume of data of interest is measured in terabytes and some time in petabytes. Such an enormous amount of data is accessed by the communities of researchers and scientists by using sophisticated computational devices. Both the researcher's communities and the computing and storage devices are geographically distributed around the globe.

This huge volume of data and the calculations involved create new problems regarding the data access, processing and distribution. With the large amount of data, different geographical locations and complex computations involved, it becomes very difficult to meet the challenges of data management infrastructure. Data grid is a solution of all problems mentioned above. It is an architecture for the distributed management and analysis of large scientific datasets [1]. Data grid is an extension of the concept "Grid" which is an infrastructure for the integration of distributed computing components.

Grid

In 1998 Kesselman and Foster introduced us to a definition of a grid as follows

"A computational grid is a hardware and software infrastructure that provides dependable, consistent, pervasive, and inexpensive access to high-end computational capabilities" [2].

Later in 2002 they improved the earlier definition to address the social and policy issues in an article [3]. Finally in 2002 they further improved the definition in [3] as follows

"A system that coordinates resources that are not subject to centralized control, using standard, open, general purpose protocols and interfaces to deliver non-trivial qualities of services".

When we have to perform some complex computational experiments which require high computational resources, we do not need to install that computing infrastructure, rather we can simply become the part of a grid with high computational powers. The idea of sharing the computing powers of the available resources across the grid environment to perform some experiment, without having to install additional computational resources is called the Computational Grid.

On the other hand data grid is a type of grid which provides services and infrastructure to assist the widely distributed data intensive applications which require the access of huge amounts of data. The basic services provided by data gird architecture are storage systems, data access, and metadata services [4].

2. Motivation

In highly distributed environment of a grid, availability of data, response time, access cost, bandwidth consumption, reliability, scalability are some very important metrics to be considered.

* Corresponding author.
E-mail addresses: tehminaamjad@iiu.edu.pk (T. Amjad), m.sher@iiu.edu.pk, hdcs@iiu.edu.pk (M. Sher), ali.daud@iiu.edu.pk (A. Daud).

0167-739X/$ – see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.future.2011.06.009

The motivation of this survey is to explore the existing dynamic replication strategies so that the researchers can include all the necessary metrics in their works in this domain and the limitations of the existing ones can be overcome.

3. Availability of data in data grids and replication

In a data grid system the computers are distributed across several geographical locations. The issue is to provide maximum availability of data to the users which are normally scientists from different universities and research laboratories. The size of data that needs to be accessed is in terabytes or petabytes. The efficient access of such huge data which is widely distributed, is slowed down due to network latencies and bandwidth problems. With the growing size of a grid, the complexity of this infrastructure is increasing. High availability of data is a major challenge in the grid environment.

The computational applications have a tremendous amount of data. Maintaining a local copy of data is very expensive and impractical. Therefore coping with high latencies of networks and limitations of storage capacities at different sites for the provision of high availability of data is a difficult challenge. To meet the challenge of high availability, data replication is considered to be the major technique. It promotes high data availability, low bandwidth consumption, increased fault tolerance and improved scalability and response time [5–11]. When data is replicated, copies of data files are created at many different places in the data grid. Replication can save storage resources as compared to the storage occupancy of data present at each site. It also saves a large amount of bandwidth as compared to the condition that data is present at only one site. Hence, for the provision of speedy data access all the time, data replication is an excellent tradeoff between storage availability and network bandwidth availability [12]. The main idea is to keep the data close to the user in order to make the access efficient and fast. The data replication algorithm has to answer critical questions such as which data must be replicated and where the replica must be placed. The dynamic behavior of grid users makes it difficult to make decisions regarding the data replications to attain the target of maximum availability [13].

4. Classification of existing data replication techniques

Replication techniques can be classified into two main streams, static replication and dynamic replication. In a static replication strategy, the number of replicas and the host node is chosen statically at the start of the life cycle, no more replicas are created or migrated after that [14,15]. On the other hand, dynamic strategies adapt to changes in user request pattern, storage capacity and bandwidth and can create replicas on new nodes and can delete replicas that are no longer required depending upon the global information of the data grid [6,12,16,17]. The dynamic strategies are better than the static ones because they can make intelligent decisions about the placement of data depending upon the information of the grid environment. Simultaneously, there are drawbacks as well; a replication decision center is required in a data grid which needs to collect the runtime information of all the nodes in a complex grid infrastructure. The overload of this central decision center further increases if the nodes in a data grid enter and quit frequently. In case of the decentralized approach, further synchronization is involved making the task difficult. Hence, the focus of this paper is only the dynamic replication strategies keeping in mind that the static replication strategies, though very simple to implement, are less useful.

5. Issues involved in data replication

Dynamic replication is an optimization technique which aims to reduce the average job execution time. It ensures high availability of data, and improved usage of network bandwidth available. There are certain issues which a data replication technique must address during replication according to the constraints of a specific situation:

Dynamic nature: The nature of the grid is very dynamic and users can join and leave a grid at any time. So the number of participants present in a grid at any given time can vary. The data replication algorithm must be adaptive to the changing size of the grid in order to provide better results.

Grid architecture: The replication technique is highly dependent upon architecture of the grid. A data grid can be supported by many different architectures. It can be a multi-tier architecture; a tree like structure in which the nodes are arranged in a tree like hierarchy. For example, the data grid of the GriPhyN project in which tier 0 is the main data source (CERN), tier 1 contains the national centers, tier 2 the regional centers, tier 3 the work groups and finally at tier 4 are the desktops. Alternatively it can be a graph like topology, in which any node can be connected to any other node without any restrictions of tree topology. It can be a peer to peer topology, or it can be any hybrid model. A replication technique is designed according to the architecture in question.

Decision making: Data replication involves a very critical decision, i.e. when to replicate data, which files should be replicated, and where the replica should be placed. Depending on the answers, different replication strategies have been evolved.

Available storage space: Although the storage devices have now become very cheap, the replication strategies must still keep into account the amount of available storage space before creating a replica. In case the available storage is not sufficient enough to store a replica, a replacement strategy is adopted.

Cost of replication: The replication strategy must ensure that the benefit of the replication is higher than the cost of replication.

6. Dynamic replication techniques

We have divided the dynamic replication techniques into several different categories according to their nature and architecture. Work done in all those categories is discussed individually in this section.

6.1. Techniques for peer to peer architecture and decentralized decision making

Ranganathan et al. [8] have presented a Dynamic Model-Driven Replication strategy which proposes that data availability can be improved in large peer to peer communities. In this approach the peers can automatically produce the replicas in a decentralized fashion whenever it is required to improve the availability of data. So a threshold level of availability is always maintained. In this model all the peers are independent to take replication decision and they can create replicas of files they contain. Each peer has a set of tools by which it can find out the state of the system to take the replication decision. As the replication decision is taken in a decentralized fashion so the state of information is partial not accurate and there is a chance that two peers can create replicas of same file simultaneously. The benefit of decentralized decision is that there is no single point of failure. The proposed replication strategy considers the following parameters: (1) Average probability of a node being up. (2) The transfer time to replicate data from one node to another node. (3) The storage cost of file F at a given node. (4) The accuracy of replica location method. The algorithm works in following steps:

(1) Collect the above mentioned parameters.
(2) Using those parameters it finds out a required number of replicas (r) for a file.
(3) Using a replica locator service finds out existing number of replicas (M).
(4) If $M > r$ wait and check again.
(5) If $M < r$ use a resource discovery procedure to find a host for replica.
(6) Send replica to the selected remote host.

The target of this approach is to find an optimal number of replicas and determine the best host for a new replica. The best client is the one which maximizes the difference between cost and benefit.

In the decentralized model for dynamic creation of replicas there is always a chance that extra replicas may get created which is waste of storage. The limitation of the model is that an unlimited amount of storage is assumed to be available, which is possible in simulation but not in real life. Another overhead is to invoke the replica location service again and again to check for replication.

Abdullah [18] presented a P2P model in 2008 for higher availability, reliability, and scalability. They have developed their own data grid simulator to test the proposed replication strategy, taking response time, number of hops and average bandwidth consumption as basic parameters for evaluation. In proposed models all the peers operate independently within a peer group. All peers working in a group agrees upon a common set of services. Peers can join or leave a group at any time. When a peer joins a group it can share the data of other peers, and its own data is shared by others. A peer can be a member of more than one group at a time. Peers can share the data sets with each other without knowing from which peer they are getting the data. The process of data discovery starts when a request is forwarded to all the neighbors depending upon information stored in the routing table. The nodes check the requested file and if not present forward it to other nodes until its time-to live, and the simulator will keep track of hop count. In this research they are studying four replication strategies, out of which two are existing strategies: ''requestor node placement strategy'' and ''path node placement strategy'', and two are newly proposed in this research: ''path and requestor node placement strategy'', and ''N-hop distance node placement'' strategy. In the ''requestor node placement strategy'' a required file when found is replicated to the requestor node only. In the ''path node placement strategy'' the file is replicated to all the nodes on the path from requestor node to provider node. The newly proposed strategy ''path and requestor node placement strategy'' is a combination of the first two strategies. In ''N-hop distance node placement'' a file is replicated to all neighbors of provider nodes within an n hop distance. The results of a simulation show that new strategies have shown better performance than existing ones in terms of performance, success rates and response time. However, the proposed strategies use more bandwidth than the existing strategies.

6.2. Techniques for multi-tier architecture and decentralized decision making

The multitier topology provides a very economical and efficient way to share the storage, computational and the network resources. It allows hundreds and thousands of users to share the common resources efficiently. Ranganathan et al. [6] have proposed six different replication techniques for three different access patterns in 2001. The main aims are reduced access latency and bandwidth consumption. The three access patterns considered are (1) random, (2) some temporal locality and (3) some temporal and geographical locality. They have presented and tested six different replication strategies which are as follows:

1. No Replication: in this case only the root node contains the replicas.
2. Best Client: a replica is created for the client who accesses the file the most.
3. Cascading: a replica is created on the path of the best client.
4. Plain Caching: a local copy is stored upon initial request.
5. Caching plus Cascading: combines plain caching and cascading strategies.
6. Fast Spread: file copies are stored at each node on the path to the best client.

Using a simulation they have evaluated two parameters, the response time and the bandwidth consumption. The results show that the fast spread shows consistent performance for all the access patterns. Cascading shows good results when there is locality in the access patterns. Although it is a client server type architecture, the replica creation decision is not made at one central location. For example in case of best client each node keeps a detailed record of all the files it hosts, including the information of many times a file has been requested and from which site the request has been made. In such a way when a number of requests for a particular file from a particular site increases, a threshold value from that site is identified to be the best client and the node replicates the file to the best client.

The virtuosity of the work done by Ranganathan et al. is that the proposed six strategies have provided the base for the new replication strategies until today. This work has provided a very sound base for the future researchers. Many of the latter researchers have compared the effectiveness of their proposed techniques with the results of six strategies of Ranganathan et al..

On the other hand there are some limitations as well. Due to the dynamic nature of grid, the client that accesses a file for most of the time, may not always keep on accessing the same file, i.e. the best client may not always be the best client. In case of plain caching a replica is created on initial request, and the initial request may be the only request so there is no need for replication in certain cases. In case of fast spread an unlimited amount of storage is required. All these strategies are simulated under ideal circumstances which is not possible in practice. The last limitation of the work is that read only databases are considered.

In 2007 Yuan et al. [12] proposed a dynamic data replication strategy based on the local optimization principle. They considered the bottleneck of data grid storage capacity of different nodes and bandwidth available between these nodes. The proposed data replication strategy is based upon two important factors (1) the storage capacity available at different nodes and (2) the bandwidth available between different nodes. The idea is to achieve the global data access optimization, first by achieving the local data access optimization. In the model each node can perform the replication locally, ensuring local optimization in the area it belongs to. This strategy is a threshold based strategy because the replication trigger condition is again a predefined value of data request frequency. We call it a decentralized decision because first in a decentralized manner local optimization is achieved in all local areas and then through interaction of local optimizations, global optimization is achieved. The problem has been addressed in two parts.

(1) Basic Local Optimization (BLO): the users and developers of the data grid are more concerned with the efficient data access speed than the preservation of bandwidth, because it can reduce the average job execution time therefore increasing the job throughput. It improves the utilization of computational resources, which is why BLO targets to speed up the average data access by data migration. But in some cases BLO can increase the bandwidth consumption which is against the aims of a data replication algorithm.

(2) Reformative Local Optimization (RLO): in RLO they have proposed to create a moderate number of replicas only on the most important nodes in such a way that access latency is minimized. This is done by considering both the storage capacity of nodes and available bandwidth between nodes.

Their model provides reliability by identifying critical replicas. Critical replicas are the ones which cannot be deleted because they are the only copy available in whole grid. Such reliability is very important in cases where there is a decentralized replication decision environment.

Yuan's model is again a tree structure because of its simplicity. Tree structures are not very suitable in real grid environments because their infrastructures are very dynamic in nature, and nodes in grid can be added and deleted at any time.

Khanli et al. [19] proposed PHFS (predictive hierarchal fast spread), which is a replication technique designed to decrease the access latency of data access. This is an extension of fast spread presented by Ranganthan et al. [6]. PHFS uses predictive techniques to predict the future usage of files and then pre-replicates them in hierarchal data grid on a path from source to client. It works in two phases, in phase one it makes the file access log files by collecting the access information from all over the system. In the next phase it applies data mining techniques like clustering and association rule mining to find useful knowledge like clusters of files that are accessed together or most frequent sequential access patterns. In this way PHFS finds the relationship between the files for future predictions. The relationship of files is assigned a value α which is between 0 and 1, 0 representing no relationship between two files and 1, representing that the two files are completely related. In this way the files are arranged according to value of α, and this arrangement is called the PWS (predictive working set). Therefore whenever a client requests a file, PHFS finds the PWS of that file and replicates all members of PWS along with the requested file on the path from source to client. In this way PHFS tries to increase the locality in access by predicting the user's succeeding file demands and pre-replicates them in the hierarchal method in advance and achieves higher availability with optimized usage of storage resources.

It is noticed that the PHFS method is more suitable for the applications in which the clients keep on working in the same context for a long time period, and requests of clients are not random. So it is more suitable for scientific applications in which researchers are working on a project.

6.3. Techniques for multi-tier architecture, centralized decision making and limited amount of storage

Tang et al. [20] in 2005 have presented two dynamic replication algorithms, *Simple Bottom Up* (SBU) and *Aggregate Bottom Up* (ABU) to reduce the average response time of data access. They have developed a simulator called DRepSim for the evaluation of the proposed algorithms. In the proposed architecture each node at any middle tier provides resources to the lower tier nodes as a server. To do so each node has a *Local Replica Manger* running on it to manage replicas stored at local site. A replication decision is made only at the *Dynamic Replication Scheduler* which keeps information about the data access history and client access pattern. *Replica Catalog* contains information about replicas including their logical and physical file names for mapping. The *Replica Selector* is available to choose one replica in case of multiple replicas available. The job of SBU is to create a replica as close to the client as possible, only for the client for which a request for a certain file increases a threshold value. It only processes the records individually in the access history and does not know their relationship. While the ABU's job is to aggregate the history records to the next upper tiers one by one till it reaches the root node. The results of the simulation show that these two algorithms reduce the data access time significantly when compared to the static replication strategies. If ABU is compared with the fast Spread strategy, ABU performs better as well. When SBU is compared with fast Spread, fast spread's response time is better but its replication frequency is too high for real time applications. Fast spread assumes an unlimited amount of storage space is available at all nodes which is not the case in SBU and ABU. The possible problem with the two algorithms is that Least Recently Used (LRU) is used as a file replacement strategy which may not be an optimal file replacement strategy in some cases.

Shorfuzzman et al. [21] presented a dynamic replica placement algorithm in 2008 for hierarchal data grid based on popularity of a file, as strategic placement of replica is very important to improve the availability of data and speed up the access. Their solution is based upon the multi-tier hierarchal architecture presented by Ranganathan [6]. They proposed the Popularity Based Replica Placement (PBRP) algorithm which improves the access time by dynamically creating the replicas of popular data. The popularity of data is measured by keeping the record of access of that data. It is assumed that popular files have more chances of access in the future and the main emphasis is on finding the popular files. The algorithm triggers at regular intervals, the history logs are cleared at the start of each replication to capture the dynamics of access patterns. As storage space is limited, they have used a modified form of the LRU replacement strategy to ensure that the replicas created in the current interval may not be deleted. This replication process consists of two steps: one is "bottom up access aggregation" in which they find out the access count of records of all nodes, starting from leaves and aggregating towards the roots along with the addition of all accesses, and the second is "top down replica placement" which traverses down the hierarchy to find the access aggregate. If it's more than a threshold value, replication is done. The performance metrics evaluated are job execution time, average bandwidth usage, and storage utilization. They have compared the efficiency of their presented algorithm with the replication policies given by Ranganathan [6], and have shown that PBRP performs well. The job execution time of PBRP is better than caching and best client. The bandwidth cost of caching is better than PBRP. The storage cost charged by fast spread and caching are too high, while PBRP is medium in terms of storage cost. However best client and cascading are lower than PBRP.

Work done by Shorfuzzaman produces good results keeping in account the job execution time, average bandwidth usage and storage available. But still the disadvantages of simulating and testing the proposed algorithm in a hierarchal network are present here as well.

A very similar work was done by Chang et al. [22,23] in 2008, in which they presented a dynamic replication algorithm using access weights. Shorfuzzaman has associated a popularity measure with each file and Chang associates the access weight of each file to keep track of the access history of all files. They have exploited the concept of temporal locality. Temporal locality means that the files which were popular in that past are more likely to be accessed in the future as well. They proposed the *"Latest Access Largest Weight"* (LALW) algorithm which finds out a popular file for replication, the number of replicas and their locations. In the proposed hierarchical architecture they consider each grid site as a cluster. The cluster header is used to manage the information within a cluster. All the cluster headers can communicate with each other by sharing the information they have. In this way they can find out which file needs to be replicated and where to place the new replica. At a regular time interval, one cluster header [22] gets file access information from all other cluster heads. In [23] a central location the *Dynamic Replication Policy*

maker performs this job. The information received at different times has different weights to differentiate their importance. The concept of half life is used to represent the weight of records. The weight of records reduces to half of its previous value after a fixed amount of time. LALW works in three steps: (1) find the most popular file, (2) find how many replicas are required, (3) choose host for new replica. It replicates the data at regular time intervals by maintaining data access history which includes file name, number of requests for that file, and the node requesting that file according to their access weights are assigned to files and popular files are selected for replication. The proposed strategy has been tested in a simulator. They have compared their LALW algorithm for its efficiency with LFU. Both give similar results when total job execution time is measured. LALW performs better in terms of effective network usage and storage usage, because LALW replicates at regular intervals while LFU always replicates.

Perez et al. in 2010 [24] have presented a new model for data replication in data grid environment. They have presented a Branch Replication Scheme (BRS) which caters for the following two metrics:

(1) optimal usage of the storage with the help of creating a sub-replica and
(2) improved data access performance with the help of parallel I/O techniques.

BRS maintains the consistency of the replicas while updating. The updating feature for the replicas is ignored by most of the replication strategies as they assume that data is read only. BRS provides improved scalability, performance and fault tolerance. In their model each replica is composed of disjoint sub-replicas which are organized in a hierarchal manner. This approach guarantees optimal storage consumption because only that part of file is replicated to the site which is required and hence storage is not occupied unnecessarily and parallel access to those sub-replicas is made possible. They have evaluated the performance of their model by running a simulation of 50 sites where each site has several processors and 10 storage nodes which are connected with a gigabit Ethernet. The BRS is compared with the Hierarchal Replication Scheme (HRS) and BRS performs better than HRS for all file sizes for both read and write operations.

In 2011 Sashi et al. [25] have presented a modified form of Bandwidth Hierarchy Replication (BHR) [26] to overcome its limitations. (BHR was presented in 2004 by Park et al. and we have discussed it in Section 5.10.) In the modified BHR model a network region is defined as a network topological space where sites are located closely. Whenever the required replica is present in the same region, the job completion will be fast. The storage locations for popular files are determined by considering the temporal and geographical localities. If the required file is not present locally than the Replica Optimizer algorithm looks for it in the nearby sites of same region and proceeds to execute the job. Then it sorts files in all storage elements (SE) in Most Frequently Accessed order to find the SE which accesses the file for most of the time. If the selected SE has enough space to hold this file, the file is replicated to it. Otherwise it looks for a duplication of this replica in other sites within the same region; if such duplications are present the replica optimizer will be terminated. If no duplications are present the replica optimizer will find the LRU file and checks that this file is duplicated on any other site in the same region or not. If it is present within same region and its access frequency is less than the access frequency of new replica, then it is deleted from the selected SE to make room for the replication of the new file. In this way the Modified BHR replicates the file within the region with the condition that the replica is present in the site where it is accessed for most of the time.

They have compared Modified BHR with No Replication, LRU, LFU, and BHR with two different access patterns using optorsim. Results show that access cost is reduced with improved availability and optimized use of storage space by using Modified BHR. They have used LFU for the replica replacement policy and other replica replacement policies can also be used and investigated to make further improvements.

6.4. Techniques for hybrid architecture, decentralized decision making and limited amount of storage

Lamehamedi et al. [7] studied the replication problem in 2002 and presented a set of replica management services and protocols to offer high data availability, low bandwidth consumption, improved fault tolerance, and scalability of the system. In their approach replication is considered to balance the load of data requests within the system to improve reliability. The replication decision is based on access cost and replication gains and is done by the replica management system. The hybrid architecture combines the ring topology and the tree structure to get the benefits of both models. The multi-tier architecture increases the availability of data and the ring topology between the root nodes of multiple hierarchal structures improves scalability. All the entities in the data grid keep a replica set, which is empty initially. Then the replication process starts and creates replicas at the nodes which receive a larger number of requests. In the hierarchical topology, each node maintains a list of its parent and child locations, while in the flat topology; each replica keeps a list of the locations of its neighbors. A replica is removed from a site when the user chooses to do so. Otherwise, the system can delete a replica in following three scenarios:

(1) If a replica is not required locally any longer or,
(2) If there are no requests to access it remotely from other sites after a certain amount of time or,
(3) When space is required to make room for more frequently used data.

For their experiments they have introduced additional traffic on the grid which utilizes bandwidth and introduces extra delays so that the results are close to the original scenarios. The simulation results show that the performance gains increase with size of data.

To generate their results they have assumed that requests are made sequentially and each user accesses only one file. This assumption is far away from an original scenario. Although their results show potential, they are based on synthetic workloads and simplified grid scenarios so the introduced services and protocols must be tried and tested on original scenarios.

6.5. Techniques for multi-tier sibling tree architecture, centralized decision making and limited amount of storage

In 2009 Rasool et al. [27] proposed a two way replication strategy. The multi-tier sibling tree architecture is used which is a mixture of the architectures presented by Ranganathan [6] and Lin [28]. It's a hierarchical model in which all the siblings are connected to each other as well. In this two way replication (TWR) scheme the most popular data is identified and placed to its proper host in a bottom up manner in this way they are closer to the clients. In top down manner the less popular files are identified and are placed to one tier below the root node, in this way they are close to the roots. Replication is done in a centralized way by the grid Replication Scheduler (GRS). GRS has communication with the replica catalog which administers all the information about the replicas. GRS also communicates with Replica Selector in order to choose the best replica server. The request path is defined as follows. A client requests a file not present in its cache from its parent, if it is present on the parent it is transferred to the client. If the file is not present at parent node, it is searched for in the sibling ring. The siblings are connected just like P2P topology. If

the requested file is present in the sibling ring it is transferred to the client, otherwise it is forwarded to the parent and so on. The GRS finds out the average access frequency of the file with the help of history records. These files having a greater frequency than this average value are called More Frequent Files MFF, and files with less are called Less Frequent Files LFF. The replication of MFF and LFF is done concurrently. In both cases the replicas are placed on the path of the best client. Replica selection is done by using the closest policy which tries to provide the data from the nearest site. A replica replacement policy is required, as a limited amount of storage is assumed. The function evacuate is used for replica replacement. It finds the replicas that have not been used in the current session and removes them until there is enough space for a new replica. They have tested the proposed strategies using a simulator and compared the results with the Fast Spread replication technique [6]. Results show that TWR is better than the fast spread in terms of number of replicas and better utilization of storage elements, while its response time and availability of data is comparable to fast spread. The drawback of the research is that it's only considering the homogeneous data grid nodes and cannot be applied to heterogeneous nodes while the nodes in a data grid are normally heterogeneous.

6.6. Techniques for multi-tier sibling tree architecture, decentralized decision making and limited amount of storage

Lamehamedi et al. [16] attempted to meet the challenges of huge data distribution and network latencies by using dynamic replication in 2003. They presented an approach where a replica creation decision is based on the estimation of cost, data access gains, replica creation cost and replica maintenance cost. These costs are calculated from the cost of read/write operations, bandwidth, and network latency. Scalability is provided in the proposed approach by the use of distribution topologies that can support replica placement. In this way, services can be provided to a large number of users. Keeping in view the dynamic nature of the grid, the authors have mainly addressed two challenging features; (1) scalability, (2) adaptability. In the proposed approach the replication component calculates the improvement in data access gained by replication and compares it to the cost of replica creation and maintenance at run time to decide whether replication must be performed or not. To provide scalability multi-tier sibling tree architecture is used. In this way the hierarchy is provided by tiers of the hierarchal model and all the siblings are connected via ring topology to provide even more connectivity and scalability. A simulation is developed to evaluate the proposed model. It is a decentralized approach as the replica management entity decides to create a local replica. Parameters which are considered before creating and placing a replica are the access patterns, free storage available at a given node, and estimated cost. So the strategy finds out an optimal solution by comparing the costs and benefits of replication. For experimentation they have done simulation for three different scenarios. The results show around 60% of improvement in response time and data transfer time. Data transfer costs and bandwidth consumption are reduced as well. It is noticed that the performance of high work load scenarios increases when high bandwidth is provided.

The results of Lamehamedi et al. are very promising but the problem is that the results are compared to the case when no replication was performed. The proposed model is not compared with any other existing replication technique, which is a problem because we cannot conclude how good or bad this replication strategy is.

6.7. Techniques for multi-tier architecture to ensure a balanced workload

In 2006 Liu et al. [29] have presented efficient algorithms keeping in view some very important parameters. They tried to maintain a balanced workload on all the servers by optimal placement of replicas. For ensuring efficient data access as well as cheaper maintenance of replicas they have proposed an optimal number of replicas. They have emphasized the fact that strategic data placement is very important in order to get maximum benefits from replication. Optimizing the access cost and reducing the replication costs are two opposites and it's difficult to find a balance between them. To achieve this task a hierarchal model is presented where the request initiated by a user is routed upward as long as the required data is found. Assume that

- T is the tree representing data grid and r is the root.
- L is set of all leaves and l is a leaf node, then $w(l)$ is the amount of data requests from l.
- N is a set of nodes in T.

Then the workload of a node n can recursively be defined as

$$fR(n) = \begin{cases} w(n) & \text{if } n \text{ is a leaf} \\ \sum cfR(c) & \text{where } c \text{ is a child of } n \text{ and } c \\ & \text{does not belongs to } R. \end{cases}$$

Now the problem is divided into two parts *MinMaxLoad* and *FindR*. In *MinMaxLoad* the number of replicas is given and we have to find R in such a way that maximum workload is minimized. In *FindR* problem the amount of data, a replica or a hub can provide is given as input to the problem and we have to find R so that the workload is not more than the given workload. So the input parameters are an estimated amount of data usage from each user site and maximum workload allowed for each replica server. The time complexity to compute workload is $O(n)$ when no replica is place. Cost of updating the workload of ancestor is $\Omega(kn)$. As cost of updates is high and not prohibitive so they have proposed an idea of *Lazy Update* in their baseline algorithm called *feasible*. The idea of *Lazy update* is to use a deduction value on an internal node n to keep track of amount of workload that should be removed from n and all of its ancestors. It is a depth first traversal. Using *Lazy update* they find an optimal replica set for *FindR* in time $O(n \log n)$ where n is the number of nodes in the data grid. *BinSearch* finds optimal replica set for *MinMaxLoad*in cost $O(n \log n)$. So in this way both algorithms work in tile $O(n \log n)$.

They have presented the algorithms but have not tested them in a real environment or in a simulator. The limitation of the strategy is that it can only work well with the tree topology and cannot give results if applied to a general graph topology.

Later in 2008 Wu et al. [30] extended the work of Liu [29]. Wu involves the issue of quality assurance in the previous work; now in their model each request must be given a quality of service guarantee. Their new algorithms ensure both workload balance and quality of service. They presented two models, The Unconstrained Model and Constrained Model. The algorithms of Unconstrained model are same as their previous work [29], and the Constrained model involves a range limit with each request for locality assurance. That means that the request must be served by a replica or hub within a fixed number of hops towards root.

In both proposals by Liu and Wu [29,30] they have assumed the hierarchical structures for their models, so the solution cannot work for the general graphs which are more close to a real grid environment. The problem of network congestion is not considered as all the requests if not served by intermediate nodes are finally directed towards the hub. In their later work the constraint of locality had not only made the work more difficult but had also decreased the availability.

6.8. Techniques for multi-tier sibling tree architecture to ensure a balanced workload

In 2008 Lin et al. [28] have addressed the problem of placement of a new replica in a proper place by considering a priority list. Their

work is done in the prospective that the user who is requesting data may have different levels or types of authorization for accessing the resources. Majorly they have addressed two issues (1) some users may have a limited or no access to some data resources, so their requests must be prohibited from accessing such data, and (2) with some special requirements for example quality of services, a static policy may not satisfy a data request. The proposed replica placement algorithm which finds a suitable host for the replica in such a way that workload among the replicas is balanced. They also proposed an algorithm which finds out the minimum number of replicas when the maximum workload capacity of each replica is given. The hierarchal model is different from other related works done considering hierarchal model because they assume a logical connection between all siblings of a parent and a request can be served from a node present in sibling ring. If requested data is not present in sibling ring than parent ring is searched. This architecture is called a Sibling Tree model, which is an extension of a normal tree structure. In the proposed priority list model, each node in the data grid can request data, and each request uses a priority list which indicates the sites from which the request can ask for the required data. The replica placement problem is been divided into two parts:

(1) *LoadBalance*: with given k number of replica find a feasible server set S containing k servers in such a way that maximum workload among all the servers is minimized. The complexity of this algorithm is $O(N^3(\log M))$, where N is number of nodes in data grid and M is the number of requests.
(2) *PlaceR*: with given W maximum workload of each server, find the minimum number of servers in server set S. The complexity of this algorithm is $O(N^3)$ where N is the number of nodes in the data grid.

The presented hierarchal model assumes a logical connection between the siblings and actually all connections from one sibling to another physically involves the parent i.e. at most two hops. This means the actual time taken to serve a request is infect more than it is presented, as this logical connection is assumed physical and already the time complexity is too high. The problem of network congestion is not considered as all the requests if not served by intermediate nodes are finally directed towards the root.

6.9. Techniques for single-tier architecture to minimize data missed

Lei et al. [31,32] proposed a dynamic data replication strategy to handle the problem of maximizing the availability of data in data grid. They have presented two new metrics, *System File Missing Rate* SMFR and *System Byte Missing Rate* SBMR, assuming limited replica storage space for maximizing data availability by minimizing the data missed rate. They presented a greedy algorithm that treats hot data differently than cold data for assigning of weight-age to files for replacement. Hot data is the data that is being used more frequently. The emphasis is that availability of the whole system is of more importance than the availability of a single file and correctness of available data is of high importance too. SMFR is the ratio of number of file that may not be available to the total number of requested files by all jobs. Likewise SBMR is the ratio of the bytes that may not be available to the total number of bytes requested by all jobs. The MinDmr optimizer takes four steps to perform replication. It first checks that requested file is present in Storage element or if, if it is present then of course no replication is performed. If requested file is not present then optimizer checks free storage space available, if its large enough requested file is replicated. Thirdly if there is not enough space available optimizer has to select file or files to be removed to make enough room for new file depending upon their weights. Finally, it has to guarantee that replication gain is more than replacement loss. The proposed strategy is for single tier architecture and thus it can be used in a single tier of multi-tier grid architecture.

The strategy adopted by Lei is performing well, but it is based upon an assumption that all the file sizes in the system are same and that's why SFMR and SBMR are same. There must be some enhancement in the algorithm so that system files missing rate and system bytes missing rate in the grid can be differentiated when the file size is not unique. Furthermore, the smaller sized files must be given preference above larger files when file sizes vary.

6.10. Techniques for general grid topology and limited amount of storage

In 2004 [26] Park et al. proposed an internet hierarchy based replication strategy called BHR to reduce the data access time by avoiding network congestions. To achieve this, they are exploiting the concept of network level locality. In proposed model there can be different network regions combined with each other. If a required file is present within a region there will be less number of routers in path, but if the file has to be fetched across the region from another region, there will be more number of routes in the path. With in a region there will be broad bandwidth available. Network level locality means that if the required file is fetched from the site having broad bandwidth, it will reduce the response time significantly. BHR tries to improve the network level locality by replicating the files within the region. The region optimizer keeps the count of files that are accessed within the region to find the regional popularity of files. If the required file is not present within the site, it is fetched from any other site and after processing it is decided whether to replicate this file or not. If the local storage element has enough space the file is stored. If the available space is not enough the decision of removing existing files to make room for new files is taken in two steps. First it is checked that required new replica is duplicated on another site within region, file is not stored. Otherwise Optimizer will find out such files on local storage which are present on other sites within region. Any of such files will be deleted to make enough room for new file. Secondly all files in the storage element are sorted in least frequently used order. All files having access frequency less than access frequency of new file are deleted until there is enough space for new replica, and finally the new replica is stored. First step ensures the duplication is avoided, and second step considers the popularity of a file on regional level. This increases the network locality. The results of simulation of BHR are compared with LRU delete strategy and Delete Oldest strategy. It is shown that BHR takes the shorter total job time specially when small storage is available and the bandwidth hierarchy is clear.

BHR takes advantage of network level locality, i.e. the required file is present in a site that has a large amount of bandwidth available between it and the site where job is executed. However in data grid environment the files may not be present in the nearby locations with high bandwidth. So less time will be consumed in fetching the required file only if file is present in same region.

Ding et al. proposed Data Placement algorithm and Self tuning data replication algorithm in 2009 for general grid topology in [33] for improved load balancing, reduced response time and conserved network bandwidth. In proposed model grid is composed of clusters, with each cluster having different storage and computational capabilities. The system model has following main components: (1) Global Data Scheduler to replicate and distribute new data objects, (2) Global computational Scheduler to dispatch jobs to different cluster sites, (3) Local computational Scheduler to dispatch jobs locally and (4) Local Data Scheduler to delete, create, and transfer replicas. The Data placement algorithm is proposed for Global data scheduler, and data replication algorithm is proposed for local data scheduler. When new data

objects are created they are placed in data grid and their replicas are allocated to selected sites by Data placement algorithm. This is done by keeping the workload balanced and considering the size of storage available. As the resources in the cluster sites and data access patterns keeps on changing so a self tuning data replication algorithm is proposed to automatically adjust such changes. The Local data scheduler finds out the heavily loaded site and pairs it with a lightly loaded side using pair establish protocol. After this pairing some of the replicas are shifted to lightly loaded site and load is balanced again. The simulation results have shown that new data placement algorithm shoes better results than the general data placement algorithms. The new replication algorithm outperforms the general threshold based algorithms in terms of efficiency and load balancing.

6.11. Techniques for general graph architecture, decentralized decision making

Most of the work done in field of replication in data grid environment is considering the multi-tier (hierarchical) grid architecture. In literature we have found comparatively very less amount of work which is considering a general or planner graph as grid architecture.

In 2005 Rehman et al. [9] presented six dynamic replication strategies based on utility and risk for two different kinds of access patterns. Before placing a replica at a site they considered both utility and risk index for each site according to the current network load and user requests. A site with optimized utility and risk index is than chosen for replication. They have used Graph as architecture to represent a data grid, which is closer to the real life grid scenario than a typical hierarchal architecture. Their model uses average response time a basis for comparison among various replication strategies. The best replication strategy has lower response time. The algorithms proposed based on utility select a replica site assuming that the future requests and current loads and user requests. Algorithms proposed based on risk index expose sites far from all other sites and assume a worst case whereby future requests will primarily originate from here. The replication algorithm selects one site per iteration to host a replica. The replication strategies are based on distributed and decentralized model and as they are dynamic so they can adapt changes to both user and network behavior.

In 2007 Rehman et al. [34] took their work further and argued that due to the dynamic nature of grid environment the candidate sites that hold data may not always be the best site to hold that replica. So they proposed a replica maintenance algorithm which can relocate the replica if performance degrades. They have considered both network state and file requests before placing a replica to a site. They have extended their previous work by locating more than one candidate sites at the same time rather than one site per iteration.

The results in [9,34] are very promising and representing the real grid scenario. Some assumptions considered during simulation are not like a real grid environment. The number of sites is fixed in the simulation and the connection between sites does not drop throughout the simulation. Whereas actually nature of grid is dynamic, sites can join and quit frequently. Thus, the replica placement algorithm must be capable of taking care of the drop connection between sites.

Bsoul et al. [35] in 2010 have presented an Enhanced Fast Spread replication technique for data grid. EFS consider the number and frequency of requests, size of replica and last time the replica was requested while making the replication decision. They have considered a network topology (which is a complete graph) in which there is one server node and many clients. The server node has the main storage with all the data and the clients have less storage space available as compared to the server. Whenever a file is required it is first searched locally, and is used if found. If it is not available then the client traverses the shortest path until it finds the required file. The fast spread strategy replicates the file on all nodes along the path. If storage on any node is not enough if remove some file(s) to make room for new replica using LRU or LFU. The EFS, on the other hand considers the value of the group of files to be removed with the value of new replica. It replicates the new replica only if value of new replica is greater than value of group to be removed. The replica value (RV) and group value (GV) are calculated from the factors like number of requests, size of replicas and last time it was requested. During the simulation EFS has been compared with Fast Spread with LRU and Fast Spread with LFU under three different scenarios mentioned as under.

- Probability of requesting any of the replicas is same.
- Probability of replicas in most wanted group (MWG) is 30% and rest of replicas is 70%.
- Probability of both, the MWG and rest of replicas is 50%.

The parameters considered for evaluation of performance are the total response time and total bandwidth consumption. The simulation results show that EFS performs better than the original fast spread.

Another Dynamic Replication Algorithm (DRA) [36] was proposed by Sashi et al. in 2010 for European DataGrid. It considers a network topology in which different clusters are present. Within a cluster the sites are located closely. DRA improves the availability of a file by replicating it within a cluster. The data is initially produced in cluster master and it is than distributed to all cluster heads. Access frequency of all the files is determined and most popular files are replicated to the site where it is requested for most of the times, considering the geographical and temporal localities. The metrices used for evaluation of performance of DRA are Mean Job Execution time (MJET) and Average Storage Used. During simulation they have compared DRA with No Replication, LRU and LFU and it is observed that performance of DRA is better.

In both EFS [35] and DRA [36] the comparison is been made with vary basic strategies like LFU and LRU. We can see from this survey that a number of advanced strategies for dynamic replication have been emerged, but none of them is used for comparison. Experimentation is required to test those new strategies with other known good ones instead of simple LRU and LFU.

6.12. QoS aware data replication

Andronikou et al. [37] have presented a QoS aware data replication mechanism containing set of algorithms for the complete replication life cycle, including replica creation, placement, relocation and retirement. These algorithms consider the infrastructural constraints like locality, cost, network and bandwidth etc. and the importance of data as well. The focus of the solutions is on determining the number and locations of the replicas to be created or deleted, considering the workload balancing on all nodes, QoS satisfaction and improved usage of available network bandwidth.

First of all the QoS determination mechanism of replica management identifies the important data. Higher the importance of data, more replicas are to be created. The number of replicas to be created is then identified according to importance and storage available. Once the number of replicas is identified, replica management has to identify where to place them. The process involves selection of best storage node. Replica relocation mechanism is involved to cope with the dynamic environment of data grid with great flux of retrieval requests. Relocation algorithm find the best location for existing replicas according to the current user request patterns. As replication is very capacity consuming technique, so replica retirement mechanism is also presented. The replicas that are no longer are in used are deleted, ensuring that at least one copy always remains in the system, hence providing reliability and securing the storage space for more important data.

7. Feature comparison and its tabular representation

Now we are going to summarize all the techniques discussed in Section 5 in such a way that we can identify the following:

1. What are different promises of a replication strategy?
2. What are different parameters a replication technique may consider to meet the promises?
3. What different models or architectures are available, and pros and cons?
4. What different assumptions are made?
5. What different research methods are used?

7.1. Benefits of data replication strategies

Availability: All the replication strategies aim to provide maximum availability. Rather, it would be better to say that replication is the only way to improve availability of data: generally in all distributed database environments and specifically in data grids.

Reliability: When replication increases the availability, the reliability is improved as well. The more the number of replicas more is the chance that user's request will be serviced properly, and hence systems is more reliable.

Scalability: It is another important metric that must be considered by a replication algorithm. The extent to which scalability can be provided depends upon the architecture chosen for the data grid. Different architectural models support different levels of scalability. That means, scalability is more dependent on model than replication algorithm.

Adaptability: This is a very important parameter which must be provided by a replication strategy. The nature of the grid is very dynamic. Nodes keep on entering and leaving the grid very frequently. The replication algorithm must be adaptive to provide support to all nodes present in a data grid at any given time.

Performance: As the availability of data increases the performance of the data grid environment increases.

7.2. Different parameters and their importance

To gain the above mentioned benefits there is a set of parameters. All replication strategies use any subset of these parameters.

- Reduced access latency.
- Reduced bandwidth consumption.
- Balanced workload.
- Less maintenance cost.
- Strategic replica placement.
- Job execution time.
- Increased fault tolerance.
- Quality assurance.

Almost all the replications strategies try to reduce the access latency thus reducing the job response time and hence increase the performance of the data grids. Similarly almost all the replication strategies try to reduce the bandwidth consumption to improve the availability of data and performance of the system. The target is to keep the data as close to user as possible, so that data can be accessed efficiently. Some of the replication strategies explicitly target to provide a balanced workload on all the data servers. This helps in increasing the performance of the system and provides better response time. With more number of replicas in a system the cost of maintaining them becomes an overhead for the system. Some of the strategies aim to make only an optimal number of replicas in the data grid. This ensures that the storage is utilized in an optimal way and the cost of replica maintenance is minimized. Some strategies target the strategic placement of the replicas along with an optimal number of replicas. The strategic placement of replicas is a very important factor because it is integrated with few other very important factors. For example, if the replicas are placed on the optimal locations it helps to optimize the workload of different servers. It is also related with the maintenance of the cost. If a strategy goes on replicating a popular file blindly, it will created too many replicas thus increasing burden for the system as replica maintenance costs will become too high.

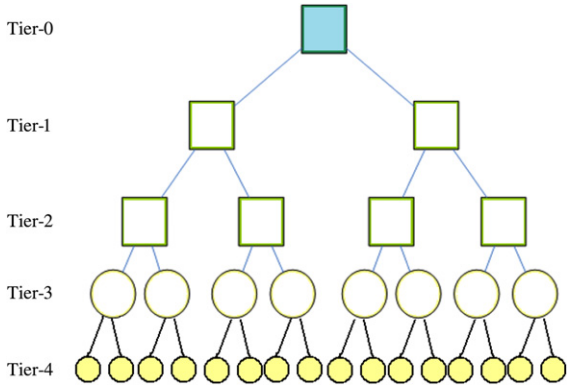

Fig. 1. Multi-tier architecture.

Job execution time is another very important parameter. Some replication strategies target to minimize the job execution time with optimal replica placement. Idea is to place the replicas closer to the users in order to minimize the response time, and thus job execution time. This will increase the throughput of the system.

Only a few replication strategies have considered replication as an option to provide fault tolerance and quality assurance.

7.3. Different architectures for data grids

The performance of replication strategies is highly dependent upon the underlying architecture of data grid. Different architectural models have been proposed. The very basic model is the hierarchal data model, also known as multi-tier, as envisioned by the GriPhyN project. It is a five tier hierarchal structure (shown in Fig. 1). Tier 0 is CERN where all the data is produced; at tier 1 there are the national centers; at tier 2 the regional centers are present; at tier 3 the work groups are present and finally at tier 4 are the desktops. In 2001 Ranganathan and Foster [6] presented six replication techniques for this architecture of the GriPhyN project. This is a form of client–server architecture and is easier to implement because of its simplicity. The problem with this type of architecture is the strict rules of a tree structure; there is only one path available from a leaf to the root. Child nodes can communicate only to their direct parent and cannot communicate with any other node. This type of model is efficient only for the grids which are designed from scratch. If we are randomly adding nodes to the grid then this type of architecture fails to represent the grid.

Latter researchers have used this work as a basis and proposed many replication strategies for this hierarchal model, and for certain modifications of this model as well. The sibling tree model shown in Fig. 2 is a modification of this hierarchal model in which the sibling nodes are also connected. This improves some of the limitations of the hierarchal grid.

A true representation of a grid is a general graph in which there is no central node designated as a root node, and any node can be connected with any number of nodes. In literature we can see the work done on a graph as the grid architecture is much less. Most researchers have worked on hierarchal structure and have mentioned extending their work to general graphs in the future.

7.4. Different assumptions

Few earlier replication strategies have assumed that there is an unlimited amount of storage space available for placing new replicas. Recent work assumes that storage space, though very

Table 1a
Features of replication techniques studied in the survey.

Reference #	[8]	[18]	[6]	[12]	[20]	[21]	[22,23]	[7]
Year	2002	2008	2001	2007	2005	2008	2008	2002
Replication decision	Decentralized	Decentralized	Decentralized	Decentralized	Centralized	Centralized	Centralized	Decentralized
Architecture	P2P	P2P	Multi-tier	Multi-tier	Multi-tier	Multi-tier	Multi-tier	Hybrid
Improved availability	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Reduced response time	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Scalability	No	Yes	No	No	No	No	No	Yes
Reliability	No	Yes	Yes	Yes	No	No	No	No
Bandwidth consumption consideration	No	Yes	Yes	Yes	Yes	No	Yes	Yes
Load balancing consideration	No	No	Yes	No	No	No	No	No
Fault tolerance consideration	No	No	No	No	No	No	No	Yes
Storage assumption	Unlimited storage	Limited	Limited	Limited	Limited	Limited	Limited	Limited
Storage utilization	Increased due to duplication	Increased	Increased	Improved	Improved	Improved	Improved	Increased
Reduced access cost	No	No	No	No	No	Yes	No	Yes
Simulation	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes
Complexity					$O(n^2)$	$O(n^2)$		
Threshold based	Yes	No	Yes	Yes	Yes	Yes	Yes	Yes
Optimal number of replicas	Yes	No	No	Yes	No	No	No	No
Replication cost consideration	Yes	No	No	No	Yes	Yes	No	Yes
Any additional features	No	Number of hops	3 access patterns	Local optimization	Different access patterns	Strategic placement & different access patterns	No	Size of replica

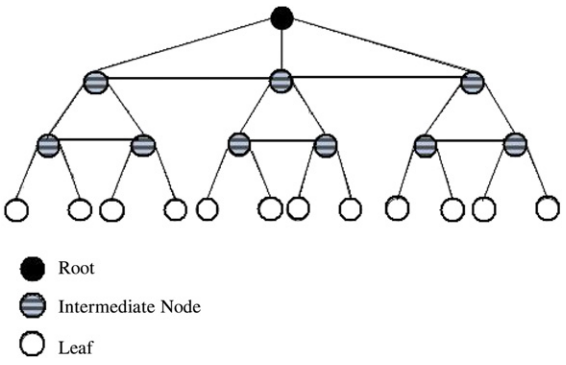

Fig. 2. The sibling tree architecture.

cheap, is limited. Some strategies assume that all the files are of same size, some assume that all sites have equal amount of storage space and equal computational powers. Another assumption made by some strategies is that the network connection between different nodes does not drop throughout the simulation. All those assumptions have a potential impact on the working of the grid environment. The more unrealistic the assumption, the less efficient the strategy in real life scenarios.

7.5. Research methods used

Almost all the work done in the field of replication is validated by the simulation. OptorSim was developed as part of the EDG [38–41] project. It is the most commonly used simulator for the evaluation of proposed replication strategies. Network Simulator NS has also been used. Some strategies have written their own simulators like DRepSim is created in [20] and GridNet in [16] which is also used by [27].

7.6. Features of replication strategies

The Tables 1a–1c show a summarized form of features of all research strategies studied above. In this section we compare the reviewed replication techniques according to their features.

The grid architecture for which the replication technique is developed makes the strategies different from one another. Four main different architectures are found, Peer-to-Peer, Multi-tier, Sibling Tree, and General Graph. The replication decision making authority is an important feature. Decision making can be centralized or distributed. In centralized decision making there is a chance of bottleneck in case of more than an average load on the network. In decentralized decision making there is a chance of of duplication and unnecessary replication. Almost all the replication strategies are found to improve the availability of data and reduce the response time. There are no differences in these two features. Scalability of a system is mostly dependent upon the architecture on which the system is based. Some architectures provide more flexibility and some are less flexible. The reliability of the system involves the correctness of available data. Reliability is sometimes measured with time when the system is up, i.e. reducing the downtime of the system improves the reliability. There is an optimal consumption of bandwidth by some of the replication strategies, although it is not considered by many replication strategies because in data replication the main target is to improve the availability of data. This improved availability can sometimes be at the cost of more bandwidth consumption than average. Load balancing and fault tolerance are supported by only a few replication strategies and are not considered as an important feature in most of the work done. The amount of storage space assumed by a replication strategy is a very important feature. Few earlier replication strategies have assumed that there is an unlimited amount of storage available, and as many replicas can be created as required. Although the storage cost is becoming reasonably low these days, to ensure realism a strategy must still assume a fixed amount of storage. Such strategies also use a file

Table 1b
Features of replication techniques studied in the survey.

Reference #	[27]	[16]	[29]	[30]	[28]	[31,32]	[26]	[33]
Year	2009	2003	2006	2008	2008	2006, 2008	2004	2009
Replication decision	Centralized	Decentralized	Centralized	Centralized	Centralized	Decentralized	Decentralized	Centralized
Architecture	Sibling tree	Sibling tree	Multi-tier	Multi-tier	Sibling tree	Graph	General	General
Improved availability	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Reduced response time	Yes	Yes	No	No	No	Yes	Yes	Yes
Scalability	No	Yes	No	No	No	No	No	No
Reliability	No	No	No	No	No	No	No	No
Bandwidth consumption consideration	No	Yes	No	No	No	Yes	Yes	Yes
Load balancing consideration	No	No	Yes	Yes	Yes	No	No	Yes
Fault tolerance consideration	No	No	No	No	No	No	No	No
Storage assumption	Limited	Limited	Limited	Limited	Limited	Limited	Limited	Limited
Storage utilization	Improved	Improved	Improved	Improved	Improved	Improved	Improved	Improved
Reduced access cost	No	Yes	Yes	Yes	Yes	Yes	Yes	No
Simulation	Yes	Yes	No	Yes	No	Yes	Yes	Yes
Complexity			$O(n \log n)$	$O(LN \log^2 N)$	$O(N^3)$, $O(N^3(\log M))$	$O(n \log n)$		
Threshold based	Yes	Yes	n/a	n/a	n/a	Yes	Yes	Yes
Optimal number of replicas	Yes	No	Yes	Yes	Yes	No	No	No
Replication cost consideration	No	Yes	Yes	Yes	Yes	No	Yes	Yes
Any additional features	Relocation of replicas	Strategic placement and replica size	Strategic placement of replica	Quality assurance and locality assurance	Strategic placement of replicas	Minimize data missed rate	Reduced network congestion	Self tuning data replication

Table 1c
Features of replication techniques studied in the survey.

Reference #	[9]	[34]	[24]	[25]	[35]	[36]	[19]	[37]
Year	2005	2008	2010	2011	2010	2010	2011	2011
Replication decision	Decentralized	Decentralized	Centralized	Centralized	Centralized	Decentralized	Decentralized	Centralized
Architecture	Graph	Graph	Multi-tier	Multi-tier	Graph	Graph	Multi-tier	Not mentioned
Improved availability	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Reduced response time	Yes	Yes	Yes	No	Yes	Yes	Yes	No
Scalability	Yes	Yes	Yes	No	No	No	No	No
Reliability	Yes	Yes	No	No	No	No	No	Yes
Bandwidth consumption consideration	Yes	Yes	No	Yes	Yes	Yes	Yes	Yes
Load balancing consideration	Yes	Yes	No	No	No	No	No	Yes
Fault tolerance consideration	Yes	Yes	Yes	No	No	No	No	No
Storage assumption	Unlimited	Unlimited	Limited	Limited	Limited	Limited	Limited	Limited
Storage utilization	Average	Improved	Optimal	Optimal	Optimal	Optimal	Average	Optimal
Reduced access cost	Yes	Yes	No	Yes	No	No	Yes	Yes
Simulation	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Complexity								$O(k \log k)$
Threshold based	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Optimal number of replicas	No	No	Yes	Yes	Yes	Yes	No	Yes
Replication cost consideration	No	No	Yes	Yes	Yes	Yes	No	Yes
Any additional features	Two different access patterns	Relocation of replicas	Parallel I/O techniques	Different access patterns	Three different scenarios considered	Mean job execution time	Reduced access latency	Importance of data set

replacement mechanism. Selection of a suitable file replacement algorithm affects the overall system performance. The optimal use of storage space is ensured by some strategies by making an optimal number of replicas, avoiding the unnecessary duplication which is of course beneficial. The cost of replication must be considered to ensure better output from a system. Simulation is the only research methodology used by almost all research strategies to evaluate, and validate the proposed strategies, and to compare them with existing strategies.

8. Conclusion and future research

We have presented a survey and classification of dynamic replication strategies for a data grid environment. It can be seen

that different strategies have presented their own terms for the evaluation of their proposed methods. The issues that have been considered for developing these strategies, and how those issues have been resolved by them are also discussed. From this survey it can be seen that there is still a lot of work to be done in the field of data replication in a data grid environment. Some open research problems are discussed below.

It has been observed that there exists no standard architecture for a data grid environment. Most of the work done follows a hierarchal architecture but actually a general graph is a more realistic architecture. Different modifications of the hierarchal architecture have been proposed to make it closer to the real grid environment.

It has been observed that there is no single strategy that addresses all issues involved in data replication. For example some strategies consider providing reliability, scalability, fault tolerance and load balancing while some totally ignore these issues. Some strategies consider that conserving the network bandwidth is important, while some strategies have used more bandwidth than average.

Most of the techniques included in this survey have used simulation to evaluate and test the algorithms. As a next step these techniques must be prototyped and tested in real world scenarios. It will provide a very realistic evaluation of the assumptions that have been made for those strategies.

It is also observed from this survey that most of the strategies compare the results with the very basic strategies like LFU and LRU and they do not compare the proposed ones with the existing strategies which are already far better than the basic LFE and LRU. So a lot of experimentation is required to test the known effective ones.

Another open research question is totally ignored by most researchers. Most of the work done considers that data in the grid environment is read only and hence replication is easy. In reality the data is not always read only; rather it is updateable. The replication strategies are unable to cope with consistency of data when it is not read only.

References

[1] 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.
[2] I. Foster, C. Kesselman, The Grid: Blueprint for a New Computing Infrastructure, Morgan Kaufmann, 2004.
[3] I. Foster, C. Kesselman, S. Tuecke, The anatomy of the grid, 2001.
[4] J. Zhang, B.S. Lee, X. Tang, C.K. Yeo, A model to predict the optimal performance of the hierarchical data grid, Future Generation Computer Systems 26 (1) (2010) 1–11.
[5] I. Foster, The grid: A new infrastructure for 21st century science, 2002.
[6] K. Ranganathan, I. Foster, Design and evaluation of dynamic replication strategies for a high performance data grid, in: International Conference on Computing in High Energy and Nuclear Physics, vol. 2001, 2001.
[7] H. Lamehamedi, B. Szymanski, Data replication strategies in grid environments, in: ICA3PP, 2002, p. 0378.
[8] K. Ranganathan, A. Iamnitchi, I. Foster, Improving data availability through dynamic model-driven replication in large peer-to-peer communities, in: CCGrid, 2002, p. 376.
[9] R.M. Rahman, K. Barker, R. Alhajj, Replica placement in data grid: Considering utility and risk, 2005.
[10] S. Vazhkudai, S. Tuecke, I. Foster, Replica selection in the globus data grid, in: CCGrid, 2001, p. 106.
[11] H. Stockinger, A. Samar, K. Holtman, B. Allcock, I. Foster, B. Tierney, File and object replication in data grids, Cluster Computing 5 (3) (2002) 305–314.
[12] Y. Yuan, Y. Wu, G. Yang, F. Yu, Dynamic data replication based on local optimization principle in data grid, 2007.
[13] F. Schintke, A. Reinefeld, Modeling replica availability in large data grids, Journal of Grid Computing 1 (2) (2003) 219–227.
[14] O. Tatebe, Y. Morita, S. Matsuoka, N. Soda, S. Sekiguchi, Grid datafarm architecture for petascale data intensive computing, in: CCGrid, 2002, p. 102.
[15] A. Chervenak, et al. Giggle: A framework for constructing scalable replica location services, 2002.
[16] H. Lamehamedi, Z. Shentu, B. Szymanski, E. Deelman, Simulation of dynamic data replication strategies in data grids, 2003.
[17] B.D. Lee, J.B. Weissman, Dynamic replica management in the service grid, in: High Performance Distributed Computing, 2001, Proceedings, 10th IEEE International Symposium on, 2001, pp. 433–434.
[18] A. Abdullah, M. Othman, H. Ibrahim, M.N. Sulaiman, A.T. Othman, Decentralized replication strategies for P2P based scientific data grid, in: Information Technology, 2008, ITSim 2008, International Symposium on, vol. 3, pp. 1–8.
[19] L.M. Khanli, A. Isazadeh, T.N. Shishavanc, PHFS: A dynamic replication method, to decrease access latency in multi-tier data grid, Future Generation Computer Systems (2010).
[20] M. Tang, B.S. Lee, C.K. Yeo, X. Tang, Dynamic replication algorithms for the multi-tier data grid, Future Generation Computer Systems 21 (5) (2005) 775–790.
[21] M. Shorfuzzaman, P. Graham, R. Eskicioglu, Popularity-driven dynamic replica placement in hierarchical data grids, in: 2008 Ninth International Conference on Parallel and Distributed Computing, Applications and Technologies, 2008, pp. 524–531.
[22] R.S. Chang, H.P. Chang, A dynamic data replication strategy using access-weights in data grids, The Journal of Supercomputing 45 (3) (2008) 277–295.
[23] R.S. Chang, H.P. Chang, Y.T. Wang, A dynamic weighted data replication strategy in data grids, in: Computer Systems and Applications, 2008, AICCSA 2008, IEEE/ACS International Conference on, 2008, pp. 414–421.
[24] J.M. Pérez, F. Garcıa-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.
[25] 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.
[26] S.M. Park, J.H. Kim, Y.B. Ko, W.S. Yoon, Dynamic data grid replication strategy based on Internet hierarchy, in: Grid and Cooperative Computing, 2004, pp. 838–846.
[27] Q. Rasool, J. Li, S. Zhang, Replica placement in multi-tier data grid, in: 2009 Eighth IEEE International Conference on Dependable, Autonomic and Secure Computing, 2009, pp. 103–108.
[28] Y.F. Lin, J.J. Wu, P. Liu, A list-based strategy for optimal replica placement in data grid systems, in: 37th International Conference on Parallel Processing, 2008, pp. 198–205.
[29] P. Liu, J.J. Wu, Optimal replica placement strategy for hierarchical data grid systems, 2006.
[30] J.J. Wu, Y.F. Lin, P. Liu, Optimal replica placement in hierarchical data grids with locality assurance, Journal of Parallel and Distributed Computing 68 (12) (2008) 1517–1538.
[31] M. Lei, S.V. Vrbsky, X. Hong, A dynamic data grid replication strategy to minimize the data missed, in: Broadband Communications, Networks and Systems, 2006, BROADNETS 2006, 3rd International Conference on, pp. 1–10.
[32] 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.
[33] Y. Ding, Y. Lu, Automatic data placement and replication in grids, in: High Performance Computing, HiPC, 2009 International Conference on, pp. 30–39.
[34] R.M. Rahman, K. Barker, R. Alhajj, Replica placement strategies in data grid, Journal of Grid Computing 6 (1) (2008) 103–123.
[35] M. Bsoul, A. Al-Khasawneh, E. Eddien Abdallah, Y. Kilani, Enhanced fast spread replication strategy for data grid, Journal of Network and Computer Applications (2010).
[36] K. Sashi, A.S. Thanamani, A new dynamic replication algorithm for European data grid, in: Proceedings of the Third Annual ACM Bangalore Conference, 2010, p. 17.
[37] 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, Corrected proof, in press (doi:10.1016/j.future.2011.02.003).
[38] W.P.O. Team, OptorSim, a Replica Optimiser Simulator.
[39] D.G. Cameron, R. Carvajal-Schiaffino, A.P. Millar, C. Nicholson, K. Stockinger, F. Zini, OptorSim: a grid simulator for replica optimisation, in: UK e-Science all Hands Conference, vol. 31, 2004.
[40] W.H. Bell, D.G. Cameron, A.P. Millar, L. Capozza, 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.
[41] D.G. Cameron, R. Carvajal-Schiaffino, A.P. Millar, C. Nicholson, K. Stockinger, F. Zini, Evaluating scheduling and replica optimisation strategies in OptorSim, in: Proceedings of the 4th International Workshop on Grid Computing, 2003, p. 52.

Tehmina Amjad is a Ph.D. Computer Science scholar at the Computer Science Department of the International Islamic University, Islamabad, Pakistan. She received her MSCS degree in 2004 from the same institute. Currently she is working as a lecturer in the Department of Computer Science, International Islamic University, Islamabad. Her research interests are distributed computing, data grids and database management.

Muhammad Sher is Chairman of the Department of Computer Science and an Assistant Professor. He has done his Ph.D. in Computer Science from TU Berlin, Germany, and his M.Sc. in Computer Science from Quaid-e-Azam University, Islamabad, Pakistan. His research interests are in next generation networks and data grids.

Ali Daud is an Assistant Professor at the Computer Science Department of the International Islamic University Islamabad, Pakistan. He received his Ph.D. Computer Science degree from Tsinghua University, Beijing, China in 2010. His areas of interest are Text Mining, Information Retrieval, Information Extraction, Social Network Analysis, Data Grid, and Machine Learning.