International Journal of Computer Applications Technology and Research
Volume 4– Issue 7, 491 - 506, 2015, ISSN:- 2319–8656

A Survey of File Replication Techniques In Grid Systems

Moslem kaviani
Department of Computer Engineering
Iauksh University
Kermanshah, Iran

Faezeh Pournaghdali
Department of Computer Engineering
Razi University
Kermanshah, Iran

Mohmood Ahmadi
Department of Computer Engineering
Razi University
Kermanshah, Iran

Abstract: Grid is a type of parallel and distributed systems that is designed to provide reliable access to data and computational resources in wide area networks. These resources are distributed in different geographical locations. Efficient data sharing in global networks is complicated by erratic node failure, unreliable network connectivity and limited bandwidth. Replication is a technique used in grid systems to improve the applications' response time and to reduce the bandwidth consumption. In this paper, we present a survey on basic and new replication techniques that have been proposed by other researchers. After that, we have a full comparative study on these replication strategies.

Keywords: Grid environment, Data Grid, Replication, Replication Strategy, Replication Techniques.

1. Introduction

Nowadays, the management of the huge distributed and shared data resources efficiently around the wide area networks 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 Which need to be shared and analyzed[1].

Storing such amount of data in the same location is difficult, even impossible. Moreover, an application may need data produced by another geographically remote application. For this reason, a grid is a large scale resource sharing and problem solving mechanism in virtual organizations and is suitable for the above situation [2]. In addition, users can access important data that is available only in several locations, without the overheads of replicating them locally. These services are provided by an integrated grid service platform so that the user can access the resource transparently and effectively [3].Managing this data in a centralized location increases the data access time and hence much time is taken to execute the job. So to reduce the data access time, "Replication" is used [4]. When a user generates a request for a file, large amounts of bandwidth could be consumed to transfer the file from the server to the client. Furthermore the latency involved could be significant considering the size of the files involved. The main aims of using replication are to reduce access latency and bandwidth consumption. Replication can also help in load balancing and can improve reliability by creating multiple copies of the same data [5].

The replication is the process of creation and placement of the copies of entities software. The phase of creation consists in reproducing the structure and the state of the replicated entities, whereas the phase of placement consists in choosing the suitable slot of this new duplication, according to the objectives of the replication. So, replication strategy can shorten the time of fetching the files by creating many replicas stored in appropriate locations [6], [7]. By storing the data at more than one site, if a data site fails, a system can operate using replicated data, thus increasing availability and fault tolerance. At the same time, as the data is stored at multiple sites, the request can find the data close to the site where the request originated, thus increasing the performance of the system. But the benefits of replication, of course, do not come without overheads of creating, maintaining and updating the replicas [8].

There is a fair amount of work on data replication in grid environments. Most of the existing work focused on mechanisms for create, decision and delete replicas. The purpose of this document is to review various replication techniques and compare these techniques which have been presented by other researches in different distributed architectures and topologies.

The rest of this paper is organized as follows. In the third section, we present an overview of grid systems, types of grids and topologies that exist for grid systems. The third section describes replication scenario, challenges and parameters of evaluating replication techniques. Section four takes a closer look on basic and new existing data replication strategies in grid environment. In section five, we present a comparative study on the replication techniques that were discussed in the previous Section. Finally, section six will be reserved for the conclusion.

1. 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.

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.

2. Grid Systems

When a user generates a request for a file, large amounts of bandwidth could be consumed to transfer the file from the server to the client. Furthermore the latency involved could be significant considering the size of the files involved. Our study investigates the usefulness of creating replicas to distribute these huge data sets among the various scientists in the grid. The main aims of using replication are to reduce access latency and bandwidth consumption. The other advantages of replication are that it helps in load balancing and improves reliability by creating multiple copies of the same data [9]. In [10] Foster introduced us to a definition of a grid as follows “coordinated resource sharing and problem solving in dynamic, multi-institutional virtual organizations”. There are different types and topologies of Grid developed to emphasize special functions that will be defined in the two next sections.

2.1. Types of Grid

Grid computing can be used in a variety of ways to address various kinds of application requirements and it has three primary types. Of course, there are no hard boundaries between these grid types and often grids may be a combination of two or more of these [11]. Types of grids are summarized below:

- Computational grid: Computational grid is focused on setting aside resources specifically for computing power. Such as most of the machines are high-performance servers [11].
- Scavenging grid: Scavenging grid is most commonly used with large numbers of desktop machines that are scavenged for available CPU cycles and other resources. Owners of the desktop machines are usually given control over when their resources are available to participate in the grid [11].
- Data grid: A data grid is a collection of geographically distributed computer resources. These resources may be located in different parts of a country or even in different countries. A grid connects all these locations and enables them to share data and other resources [9]. For example, you may have two universities doing life science research, each with unique data. A grid connects all these locations and enables them to share their data, manage the data, and manage security issues such as who has access to which data [12].

2.2. Grid Topologies

In this section we present an overview of major grid topologies. The performance of replication strategies is highly dependent on the underlying architecture of grid [13].Graph and tree models are used where there is a single source for data and the data has to be distributed among collaborations worldwide [13]. The Figure 1 and Figure 2, shows the Graph and tree models respectively.

A tree topology also has shortcomings. The tree structure of the grid means that there are specific paths to the messages and files can travel to get to the destination. Furthermore, data transference is not possible among sibling nodes or nodes situated on the same tier [13].

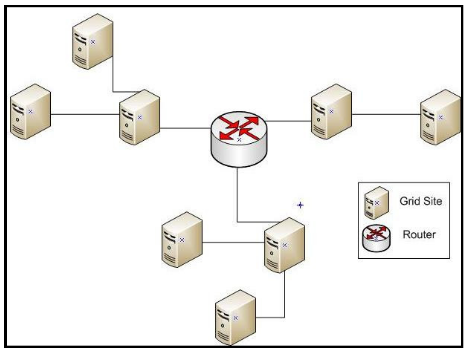

Figure 1.An example of Graph topology.

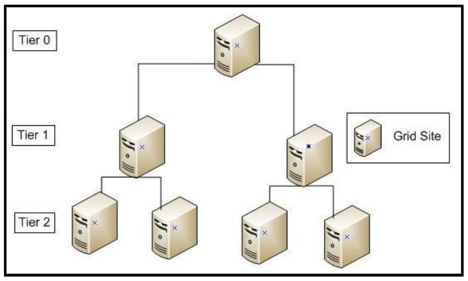

Figure 2.An example of Tree topology.

Peer to Peer (P2P) systems overcome these limitations and offer flexibility in communication among components. A P2P system is characterized by the applications that employ distributed resources to perform functions in a decentralized manner. From the viewpoint of resource sharing, a P2P system overlaps a grid system. The key characteristic that distinguishes a P2P system from other resource sharing systems is its symmetric communication model between peers, each of which acts as both a server and a client [13]. The Figure 3 shows an example of the P2P structure.

Hybrid Topology is simply a configuration that contains an architecture consisting of any combination of the previous mentioned topologies. It is used mostly in situations where researches working on projects want to share their results to further research by making it readily available for collaboration [13]. A hybrid model of a graph grid with peer linkages at the edges is shown in Figure 4.

A hybrid topology can carry features of both tree and P2P architectures and thus can be used for better performance of a replication strategy [12].

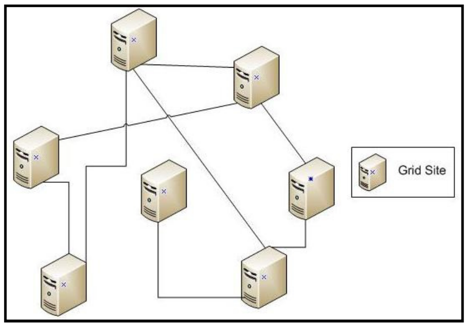

Figure 3.An example of Peer to Peer topology.

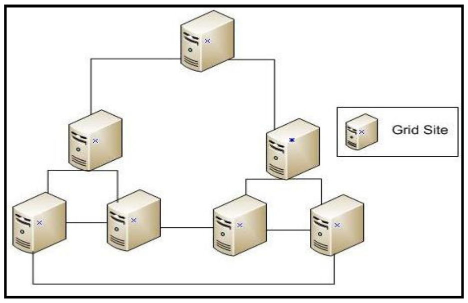

Figure 4.An example of Hybrid topology.

2.3. Classification of existing data replication techniques

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.

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]. 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 [9]. 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.

2.4. Data Replication challenges

Using replication strategies in grid environment may cause some challenges. The four important challenges in replicated environments areas follow [8]:

- Time of creation of a new replica: If strict data consistency is to be maintained, performance is severely affected if a new replica is to be created. As sites will not be able to fulfill request due to consistency requirements.
- Data Consistency: Maintaining data integrity and consistency in a replicated environment is of prime importance. High precision applications may require strict consistency of the updates made by transactions.
- Lower write performance: Performance of write operations can be dramatically lowered in applications requiring high updates in replicated environment, because the transaction may need to update multiple copies.
- Overhead of maintenance: If the files are replicated at more than one site, it occupies storage space and it has to be administered. Thus, there are overheads in storing multiple files.

2.5. Data Replication evaluation

All replication strategies use subset of these parameters [15]:

- Access latency: Almost all the replications strategies try to reduce the access latency thus reducing the job response time and hence increase the performance of the grids.
- Bandwidth consumption: 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 the user as possible, so that data can be accessed efficiently.
- Balanced workload: 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.
- Maintenance cost: 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 maintenance cost of replica is minimized.
- Strategic placement: 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 cost of the maintenance.
- Job execution time: another very important parameter is Job execution time. Some replication strategies target to minimize the job execution time with optimal replica placement. The idea is to place the replicas closer to the users in order to minimize the response time, and thus the job execution time. This will increase the throughput of the system.
- Fault tolerance, quality assurance: Only a few replication strategies have considered replication as an option to provide fault tolerance and quality assurance.

3. Replication Techniques

The role of a replication strategy is to identify when a replica should be created, where to place replicas, when to remove replicas and how to locate the best replica. Several replication replacement strategies have been proposed in the past and they are the basics of other replication algorithms.

3.1. Techniques for peer to peer architecture

Ranganathan et al. [16] 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. Each peer has a set of tools by which it can find out the state of the system to take the replication decision.

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.

Abdullah [17] presented a P2P model in 2008 for higher availability, reliability, and scalability. 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. 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.

Cashing plus Cascading combines cascading and plain cashing strategies. The client caches file locally, and the server periodically identifies the popular files and propagates them down the hierarchy. Note that the clients are always located at the leaves of the tree but any node in the hierarchy can be a server. Specifically, a Client can act as a Server to its siblings. Siblings are nodes that have the same parent [12].

3.2. Techniques for multi-tier tree architecture

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. [9] have proposed six different replication techniques for three different access patterns in 2001. The main aims are reduced access latency and bandwidth consumption.

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.

- NO Replication strategy will not create replica and therefore, the files are always accessed remotely. One example of the implemented strategy is the Simple Optimizer algorithm, which never performs replication; rather it reads the required replica remotely. Simple Optimizer algorithm is simple to implement and performs the best relative to other algorithms in terms of the storage space usage, but performs the worst in terms of job execution time and network usage [12].

- Best client creates replica at the client that has generated the most requests for a file, this client is called the best client. At a given time interval, each node checks to see if the number of requests for any of its file has exceeded a threshold, then the best client for that file is identified [12].
- Cascading Replication supports tree architecture the data files generated in the top level and once the number of accesses for the file exceeds the threshold, then a replica is created at the next level, but on the path to the best client, and so on for all levels, until it reaches to the best client itself [12].
- Plain Cashing: The client that requests a file stores a copy locally. If these files are large and a client has enough space to store only one file at a time, then files get replaced quickly [12].
- Fast Spread: In this method a replica of the file is stored at each node along its path to the client. When a client requests a file, a copy is stored at each tier on the way. This leads to a faster spread of data. When a node does not have enough space for a new replica it deletes the least popular file that had come in the earliest [12].
- Simple Bottom-Up (SBU) and Aggregate Bottom-Up (ABU) are two dynamic replication mechanisms that are proposed in the multi-tier architecture for data grids. The SBU algorithm replicates the data file that exceeds a pre-defined threshold for clients. The main shortcoming of SBU is the lack of consideration to the relationship with historical access records. For the sake of addressing the problem, ABU is designed to aggregate the historical records to the upper tier until it reaches the root. The results shown improvements against Fast Spread strategy. The values for interval checking and threshold were based on data access arrival rate, data access distribution and capacity of the replica servers [18].
- Proportional Share Replica (PSR) policy is an improvement in Cascading technique. The method is a heuristic one that places replicas on the optimal locations by assuming that the numbers of sites and the total replicas to be distributed are already known. Firstly an ideal load distribution is calculated and then replicas are placed on candidate sites that can service replica requests slightly greater than or equal to that ideal load [19].
- Multi-objective approach is a method exploiting operations research techniques that is proposed for replica placement. In this method, replica placement decision is made considering both the current network status and data request pattern. The problem is formulated in p-median and p-center models to find the p replica placement sites. The p-center problem targets to minimize the max response time between user site and replica server whereas the p-median model focuses on minimizing the total response time between the requesting sites and the replication sites [20].
- Latest Access Largest Weight (LALW) is a dynamic data replication mechanism. LALW selects a popular file for replication and calculates a suitable number of copies and grid sites for replication. By associating a different weight to each historical data access record, the importance of each record is differentiated. A more recent data access record has a larger weight. It indicates that the record is more pertinent to the current situation of data access [21].
- Adaptive Popularity Based Replica Placement (APBRP) is a dynamic replica placement algorithm, for hierarchical data grids which is guided by “file popularity”. The goal of this strategy is to place replicas close to clients to reduce data access time while still using network and storage resources efficiently. The effectiveness of APBRP depends on the selection of a threshold value related to file popularity. APBRP determines this threshold dynamically based on data request arrival rates [22].

- Predictive hierarchical fast spread (PHFS) is a dynamic replication method in multi-tier data grid environments which is an improve version of common fast spread. The PHFS tries to forecast future needs and pre-replicates the min hierarchal manner to increase locality in accesses and improve performance that consider spatial locality. This method is able to optimize the usage of storage resources, which not only replicates data objects hierarchically in different layers of the multi-tier data grid for obtaining more localities in accesses. It is a method intended for read intensive data grids. The PHFS method use priority mechanism and replication configuration change component to adapt the replication configuration dynamically with the obtainable condition. Besides that, it is developed on the basis of the concept that users who work on the same context will request some files with high probability [23].

- Dynamic Hierarchical Replication (DHR) is a dynamic replication algorithm for hierarchical structure that places replicas in appropriate sites. Best site has the highest number of access for that particular replica. This algorithm minimizes access latency by selecting the best replica when various sites hold replicas. The replica selection strategy of DHR algorithm, selects the best replica location for the users running jobs by considering the replica requests that waiting in the queue and data transfer time. It stores the replica in the best site where the file has been accessed most, instead of storing files in many sites [24].

- Modified Latest Access Largest Weight (MLALW) is a dynamic data replication strategy. This strategy is an enhanced version of Latest Access Largest Weight strategy. MLALW deletes files by considering three important factors:

  1. Least frequently used replicas
  2. Least recently used replicas
  3. The size of the replica

  MLALW stores each replica in an appropriate site in the region that has the highest number of access in future for that particular replica. The experiment results show that MLALW strategy gives a better performance compared to the other algorithms and prevents unnecessary creation of replica which leads to efficient storage usage [25].

- The combination of Modified DHR Algorithm (MDHRA) and Combined Scheduling Strategy. MDHRA considers three factors for replacing replicas: (1) the last time the replica was requested; (2) number of accesses and (3) size of replica. It computes the response time based on the data transfer time, storage access latency, the unprocessed replica requests and the distance between nodes. The computed response time is used to select the best replica location [26].

3.3. Techniques for general grid topology

In 2004 [27] Park et al. proposed an internet hierarchy based replication strategy called BHR to reduce the data access time by avoiding network congestions. 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. Within 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.
Bandwidth Hierarchy Replication (BHR) is a novel dynamic replication strategy which reduces data access time by avoiding network congestions in a data grid network. With BHR strategy, we can take benefits from "network-level locality" which represents that required file is located in the site which has broad bandwidth to the site of job execution. BHR strategy was evaluated by implementing in OptorSim simulator and the results show that BHR strategy can outperform other optimization techniques in terms of data access time when hierarchy of bandwidth appears in Internet. BHR extends current site-level replica optimization study to the network-level [27].

3.4. Techniques for general graph architecture

In 2005 Rehman et al. [28] 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.

Bsoul et al. [29] 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.

- Agent-based replica placement algorithm is proposed to determine the candidate site for the placement of replica. For each site that holds the master copies of the shared data files will deploy an agent. The main objective of an agent is to select a candidate site for the placement of a replica that reduces the access cost, network traffic and aggregated response time for the applications. Furthermore, in creating the replica an agent prioritizes the resources in the grid based on the resource configuration, bandwidth in the network and insists for the replica at their sites and then creates a replica at suitable resource locations.
- Least Frequently Used (LFU) strategy always replicates files to local storage systems. If the local storage space is full, the replica that has been accessed the fewest times is removed and then releases the space for new replica. Thus, LFU deletes the replica which has less demand (less popularity) from the local storage even if the replica is newly stored [30].
- Least Recently Used (LRU) strategy always replicates files to local storage system. In LRU strategy, the requested site caches the required replicas, and if the local storage is full, the oldest replica in the local storage is deleted in order to free the storage. However, if the oldest replica size is less than the new replica, the second oldest file is deleted and so on [30].
- Weight-based dynamic replica replacement strategy calculates the weight of replica based on the access time in the future time window on the last access history. After that, calculate the access cost which embodies the number of replicas and the current bandwidth of the network. The replicas with high weight will be helpful to improve the efficiency of data access, so they should be retained and then the replica with low weight will not make sense to the rise of data access efficiency, and therefore, should be deleted. The access history defines based on the zipf-like distribution [31].
- Efficient Replication strategy is a replication strategy for dynamic data grids, which take into account the dynamic of sites. This strategy can increase the file availability, improved the response time and can reduce the bandwidth consumption. Moreover, it exploits the replicas placement and file requests in order to converge towards a global balancing of the grid load. This strategy will focus on read-only-access as most grids have very few dynamic updates because they tend to use a "load" rather than "update" strategy.

There are three steps provided by this algorithm, which are:

1) Selection of the best candidate files for replication; Selected based on requests number and copies number of each files.
2) Determination of the best sites for files placement which are selected in the previous step; Selected based on requests number and utility of each site regarding to the grid.
3) Selection of the best replica; Taking account the bandwidth and the utility of each site [32].

- Value-based replication strategy (VBRS) is proposed to decrease the network latency and meanwhile to improve the performance of the whole system. In VBRS, threshold was made to decide whether to copy the requested file, and then solve the replica replacement problem. VBRS has two steps; At the first steps, the threshold will be calculated to decide whether the requested file should be copied in the local storage site. Then at the second stage, the replacement algorithm will be triggered when the requested file needs to be copied at the local storage site does not have enough space. The replica replacement policy is developed by considering the replica's value which is based on the file's access frequency and access time. The experiment results show that the effectiveness of VBRS algorithm can reduce network latency [33].
- Enhance Fast Spread (EFS) is an enhanced version of Fast Spread for replication strategy in the data grid. This strategy was proposed to improve the total of response time and total bandwidth consumption. Its takes into account some criteria such as the number and frequency of requests, the size of the replica and the last time the replica was requested. EFS strategy keeps only the important replicas while the other less important replicas are replaced with more important replicas. This is achieved by using a dynamic threshold that determines if the requested replica should be stored at each node along its path to the requester [34].

4. Comparative Study

In this section, we present a full comparative study on the replication techniques that were discussed in the previous section.Benefits of data replication strategies are:

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.

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.

These replication strategies that discussed in the previous section are compared in the Table 1, Table 2, and Table 3.

Table 1: Replication Technique for Tree topology

Replication Technique	Year	Performance Metric	Method	Additional Feature
Best client	2001	Response time , Bandwidth conservation	Replicates file to site that generates maximum number of requests	Need to compute number of request for each file
Cascading	2001	Response time , Bandwidth conservation	If number of requests exceeds threshold then replica trickles down to lower tier	Need to define a threshold for number of requests
Plain Cashing	2001	Response time , Bandwidth conservation	A requesting client receives the file and stores a replica of it locally	
Fast Spread	2001	Response time , Bandwidth conservation	If a client requests a file then a replica of file stores at each node along the path toward the client	Need to storing request history to avoid of double replicating
Simple Bottom-Up (SBU)	2005	Replication frequency, Bandwidth cost, Response time	Creates replicas as close as possible to the clients that request the data files with high rates exceeding the pre-defined threshold	Need to process records in the access history individually
Aggregate Bottom-Up (ABU)	2005	Replication frequency, Bandwidth cost, Response time	Aggregates the history records to the upper tier step by step till it reaches the root	Need to access history
Proportional Share Replica (PSR)	2004	Mean of response time	Calculates an ideal workload and distributes replicas	Need to define ideal workload
Multi-objective	2006	Average response time	Reallocates replicas to new candidate sites if a performance metric degrades significantly over best k-time periods	Need to calculate replica relocation cost
Latest Access Largest Weight (LALW)	2008	Network usage, Mean job execution time	Selects a popular file for replication and calculates a suitable number of copies and grid sites for replication	Need to find out a popular file and suitable site
Adaptive Popularity Based Replica Placement (APBRP)	2010	Storage cost, Average bandwidth cost, Job execution time	Selects a threshold value related to file popularity and places replicas close to clients to reduce data access time while still using network and storage resource efficiency	Need to determines threshold value dynamically, based on data request arrival rates
Predictive hierarchical fast spread (PHFS)	2011	Average access latency	Tries to forecast future needs and pre-replicates the min hierarchical manner. Uses the hierarchical replication to optimize the utilization of resources	Need to considering spatial locality and using predictive methods
Dynamic Hierarchical	2012	Mean job execution time	Selects best replica when various sites hold replicas.	Need to access history

Replication (DHR)			Places replicas in appropriate sites that has the highest number of access for that particular replica	
`Modified Latest Access Largest Weight (MLALW)	2012	Effective network usage, Mean job execution time	Stores each replica in an appropriate site. Deletes files by considering least frequently used replicas, least recently used replicas and the size of the replica factors	Need to LRU lists of replicas, LFU lists of replicas and access history
Modified Dynamic Hierarchical Replication Algorithm (MDHRA)	2013	minimizes the bandwidth consumption thus reducing the network traffic	replaces replicas based on the last time the replica was requested, number of access, and file size of replica	Need to the last time the replica was requested, number of accesses and size of replica

Table 2: Replication Technique for peer to peer & general grid topology

Replication Technique	Year	Performance Metric	Method	Additional Feature
Cashing plus cascading	2001	Response time , Bandwidth conservation	Joining two replication techniques: Cashing and cascading techniques	Need to define a threshold for number of requests
Bandwidth Hierarchy Replication (BHR)	2004	Total job execution time	Replicates files which are likely to be used frequently within the region in near future	Need to define network-level locality and regions

5. Conclusion

Replication enables faster access to files, decreases bandwidth consumption, and distributes server load. In contrast to static replication, dynamic replication automatically creates and deletes replicas according to changing access patterns, and thus ensures that the benefits of replication continue even if user behavior changes.

In this paper, a review and a comparative study has been done on basic and new replication techniques that have been implemented in grids. After a brief introduction, an overview of grid systems, types of grids and grid topologies were presented in Section 3. In Section 3, replication scenario, challenges and ways of evaluating replication techniques were described. In Section 4, we have presented a classification of dynamic replication strategies for grid environment. In Section 5, a full comparative study was presented on the replication techniques that were discussed in Section 4. And finally, in this section, Table 4 is presented that shows the results of discussed replication techniques.

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 grid environment.

Table 3: Replication Technique for Graph topology

Replication Technique	**Year**	**Performance Metric**	**Method**	**Additional Feature**
Agent-based replica	2009	Execution time test, Data availability test	By an agent for each site that holding the master copies, select a candidate site for the placement of replica that exceeds the conditions	Need to define agents
Least Frequently Used(LFU)	2003	Job execution time	Always replicates files to local storage , if no space : delete least accessed files	Need to files access history
Least Recently Used (LRU)	2003	Job execution time	Always replicates files to local storage , if no space : delete oldest file in the storage	Need to files access history
Weight-based dynamic replica	2008	Effective network usage, Mean job execution time	Calculates the weight of replica based on the access time in the future time window, based on the last access history	Need to access history that define based on zip-like distribution
Efficient Replication	2010	Response time, Effective Network Usage	Takes into account the dynamic of sites. Exploits the replicas placement and file request in order to converge towards a global balancing of grid load	Need to considering dynamicity of sites
Value-based replication strategy (VBRS)	2010	Mean job time, Effective Network Usage	Calculates the ideal threshold to decide whether the file should be copied or not. Chooses the replica that should be replaced based on the values of the local replicas	Need to define threshold
Enhance Fast Spread (EFS)	2011	Total response time, Total bandwidth consumption	Uses a dynamic threshold that determines if the requested replica should be stored at each node along its path to the requester. Keeps only the important replicas while other less important replicas are replaced with more important replicas	Need to frequency of requests, the size of the replica and the last time that the replica was requested

Table 4 : SUMMARIZES THE MAJOR RESULTS OF REPLICATION TECHNIQUES IN GRIDS

Replication Technique	Result	proper strategies
Cascading[12]	Has an small degree of locality	Tree
Cashing[12]	High response time	Tree
Cascading plus Cashing[12]	Better performance than cascading Better performance than cashing	Peer to Peer
Fast Spread[12]	High I/O and CPU load High storage request	Tree
Proportional Share Replication (PSR)[19]	Better results over cascading technique	Tree
Bandwidth Hierarchy Replication (BHR)[27]	Better total job times than LRU and LFU	Peer to Peer
Simple Bottom-Up (SBU)[18] Aggregate Bottom-Up (ABU)[18]	Better results over Fast Spread technique	Tree
Multi-objective approach[20]	Good performance in dynamic environments	Tree
Weight-based replication[31]	Better performance than LRU and LFU	Graph
Least Access Largest Weight (LALW)[21]	Better job execution time and effective network usage than LRU, LFU and BHR	Tree
Adaptive Popularity Based Replica Placement (APBRP)[22]	Better performance than Best client, Cascading, Fast Spread, ABU and LRU	Tree
Efficient replication strategy[32]	Improves the response time Increases data availability Reduces bandwidth consumption	Graph
Value Based Replication Strategy (VBRS)[33]	Decreases network latency Improves performance of the hole system	Graph
Enhanced Fast Spread (EFS)[34]	Improves total of response time Improves total bandwidth consumption Enhanced version of Fast Spread for replication strategy in data grid	Graph
Predictive Hierarchical Fast Spread (PHFS)[23]	Optimizes the utilization of resources Decreases access latency in multi-tier data grids Improved version of common Fast Spread Lower latency and better performance compared with common Fast Spread	Tree
Dynamic Hierarchical Replication (DHR)[24]	Prevents unnecessary creation of replica Efficient storage usage Minimizes access latency	Tree
Modified Least Access Largest Weight (MLALW)[25]	Modified version of LALW strategy Better performance than LRU, LFU, BHR, LALW and DHR	Tree
Modified Dynamic Hierarchical Replication Algorithm (MDHRA)[26]	Modified DHR Algorithm is compared with LFU, LRU, BHR, Modified BHR, 3LHA and DHR and was proved the best	Tree

6. References

[1] 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.

[2] N. Mohd. Zin, A. Noraziah, A. Che Fauzi, and T. Herawan, Replication Techniques in Data Grid Environments, in *Intelligent Information and Database Systems*, vol. 7197, Eds. Springer Berlin, Heidelberg, pp. 549–559, 2012.

[3] F.B. Charrada, H. Ounelli, and H. Chettaoui, Dynamic period vs static period in data grid replication, in International Conference on P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC), pp. 565–568, 2010.

[4] N. Mohd. Zin, A. Noraziah, A. Che Fauzi, and T. Herawan, Replication Techniques in Data Grid Environments, in Intelligent Information and Database Systems, vol. 7197, Eds. Springer Berlin, Heidelberg, pp. 549–559, 2012.

[5] K. Ranganathan and I. Foster, Identifying dynamic replication strategies for a high-performance data grid, Grid Computing, pp. 75–86, 2001.

[6] S. Venugopal, R. Buyya, and K. Ramamohanarao, A taxonomy of data grids for distributed data sharing, management, and processing, in Acm Computing Surveys, vol. 38, no. 1, p. 3, 2006.

[7] K. Ranganathan, A. Iamnitchi, and I. Foster. Improving data availability through dynamic model-driven replication in large peer-to-peer communities, in 2nd IEEE/ACM International Symposium on Cluster Computing and the Grid, pp. 376–376, 2002.

[8] S. Goel and R. Buyya, Data replication strategies in wide area distributed systems, in Enterprise Service Computing: From Concept to Deployment, vol. 17, 2006.

[9] 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.

[10] I. Foster, C. Kesselman, and S. Tuecke, The Anatomy of the Grid: Enabling Scalable Virtual organizations, in International Journal of High Performance Computing Applications, vol. 15, no. 3, pp. 200–222, 2001.

[11] B. Jacob, Grid computing: what are the key components, IBM Developer Works, 2003.

[12] K. Ranganathan and I. Foster, Identifying dynamic replication strategies for a high-performance data grid, Grid Computing, pp. 75–86, 2001.

[13] Q. Rasool, J. Li, G. Oreku, E. Munir, and D. Yang, A Comparative Study of Replica Placement Strategies in Data Grids, in Advances in Web and Network Technologies, and Information Management, vol. 4537, Springer Berlin, Heidelberg, pp. 135–143, 2007.

[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] T. Amjad, M. Sher, and A. Daud, A survey of dynamic replication strategies for improving data availability in data grids, in Future Generation Computer Systems, vol. 28, no. 2, pp. 337–349, 2012.

[16] 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.

[17] 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.

[18] M. Tang, B.S. Lee, C.K. Yeo, and X. Tang, Dynamic replication algorithms for the multi-tier Data Grid, in Future Generation Computer Systems, vol. 21, no. 5, pp. 775–790, 2005.

[19] J. Abawajy, Placement of File Replicas in Data Grid Environments, in Computational Science (ICCS), vol. 3038, Springer Berlin, Heidelberg, pp. 66–73, 2004.

[20] R.M. Rahman, K. Barker, and R. Alhajj, Replica placement designs with static optimality and dynamic maintainability, in Sixth IEEE International Symposium on Cluster Computing and the Grid (CCGRID), vol. 1, pp.4, 2006.

[21] R.S. Chang and H.P. Chang, A dynamic data replication strategy using access-weights in data grids, in The Journal of Supercomputing, vol. 45, no. 3, pp. 277–295, 2008.

[22] M. Shorfuzzaman, P. Graham, and R. Eskicioglu, Adaptive popularity-driven replica placement in hierarchical data grids, in The Journal of Supercomputing, vol. 51, no. 3, pp. 374–392, 2010.

[23] L.M. Khanli, A. Isazadeh, and T.N. Shishavan, PHFS: A dynamic replication method, to decrease access latency in the multi-tier data grid, in Future Generation Computer Systems, vol. 27, no. 3, pp. 233–244, 2011.

[24] N. Mansouri and G.H. Dastghaibyfard, A dynamic replica management strategy in data grid, in Journal of Network and Computer Applications, vol. 35, no. 4, pp. 1297–1303, 2012.

[25] N. Mansouri, An Effective Weighted Data Replication Strategy for Data Grid, in Australian Journal of Basic and Applied Sciences, vol. 6, no. 10, pp. 336–346, 2012.

[26] N. Mansouri , Gh. Dastghaibyfard, Combination of data replication and scheduling algorithm for improving data availability in Data Grids, in Journal of Network and Computer Applications, pp. 711–722, 2013.

[27] 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

[28] R.M. Rahman, K. Barker, and R. Alhajj, Replica placement in data grid: Considering utility and risk, 2005.

[29] 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).

[30] W.H. Bell, D.G. Cameron, A. P. Millar, L. Capozza, K. Stock-inger, and F. Zini, Optorsim: A grid simulator for studying dynamic data replication strategies, in International Journal of High performance Computing Applications, vol. 17, no. 4, pp. 403–416, 2003.

[31] W. Zhao, X. Xu, N. Xiong, and Z. Wang, A weight-based dynamic replica replacement strategy in data grids, in IEEE Asia-Pacific Services Computing Conference,(APSCC), pp. 1544–1549, 2008.

[32] F.B. Charrada, H. Ounelli, and H. Chettaoui, An efficient replication strategy for dynamic data grids, in International Conference on P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC), pp. 50–54, 2010.

[33] W. Zhao, X. Xu, Z. Wang, Y. Zhang, and S. He, Improve the performance of data grids by value-based replication strategy, in Sixth International Conference on Semantics Knowledge and Grid (SKG), pp. 313–316, 2010.

[34] M. Bsoul, A. Al-Khasawneh, E.E. Abdallah, and Y. Kilani, Enhanced fast spread replication strategy for data grid, in Journal of Network and Computer Applications , vol. 34, no. 2, pp. 575–580, 2011.