"Vasile Alecsandri" University of Bacău
Faculty of Sciences
Scientific Studies and Research
Series Mathematics and Informatics
Vol. 24(2014), No. 2, 153-192

REPLICATION METHODS IN DATA GRID

ZAHRA MIROEI AND MARJAN KUCHAKI RAFSANJANI

Abstract. Today, scientific research and commercial applications generate large amounts of data. In such circumstances, storage and management of these data volume centrally is difficult or even impossible. Data Grid is a good solution for data management, storage resource management and execution of applications. In applications related to data, required data transmission time is the main cause of delay in task execution. In order to decrease it, one method is to put the required data in places near task execution. On the other hand, one of the tasks of data grid is to share data and increase data availability and reliability. In order to reach these goals, one suitable solution is to replicate data in different parts of the system. In this method, not only required data for task execution are replicated near it and file transmission time and task execution time are decreased, also data availability is increased, system reliability is increased, fault tolerance is increased, system efficiency and scalability are increased. Benefits of data replication have led to many researchers to study in this field and propose many algorithms. In this paper the researches done in this field are studied and compared.

Keywords and phrases: Data grid, Data replication, popular file, requested file, requesting site.
(2010)Mathematics Subject Classification: 47H10, 54H25

1. Introduction and preliminaries

Today, so much information is collected, that keeping and managing these amount of data centrally is impossible. Many researchers have been found solutions to store these data in different places in a distributed form and to share them between different users throughout the world and also to solve the problem of heterogeneous environment and requirements of users to high and inexpensive computational power for processing these information.

1.1. **Definitions.** The grid system was firstly introduced by Foster and Kesselman [18] as a suitable solution for sharing many data and also solving the problem of heterogeneous environment, *"grid is a software and hardware infrastructure which provides reliable, stable, broad and inexpensive availability to the shared resources"*.

They modified this definition in 2002 as follows [19]: *"A system that coordinates resources that are not subject to centralized control, using standard, open, general purpose protocols and interfaces to deliver nontrivial qualities of services"*.

Of different kinds of grid systems, computing grid and data grid can be referred to

Computing grid is a distributed computing model, and works with duties which need high computing power, so it is designed for sharing computing recourses

Data grid is a system which has been designed and configured to share high data resources between users and execution of their data-based operation. Different methods have been proposed to manage data in data grid effectively and optimally. One of these methods is data replication (copying data). Data replication is a method in which the same copies of specific data are produced in different parts of the system. But we should mention that just popular files, files that are requested by the system frequently, are copied

Data replication have two disadvantages, increases memory consumption and also needs to update different copies simultaneously. It is clear that making different copies in the system has some benefits that justify additional costs for this deed.

1.2. **Advantages of data replication.**
The advantages of data replication include:

- *Increased availability*: existence of different copies causes that if one sample of copied resources could not respond to the user request, the other source does so.

- *Increased reliability*: Multiple copies of a resource reduce the probability that all copies do not work simultaneously and consequently the probability of entire system failure is reduced.
- *Increased efficiency*: Different copies of data in the system prevent overload on a server by suitable distribution of data request of users between different servers and therefore, the response time of the server to user requests is decreased and the efficiency of the whole system is increased.
- *Increased fault tolerance*: This advantage has a direct relation with the advantage of "*increased reliability*" so when there is more than one of a special data in the system, losing or deleting that data causes no fault in the system because the system meets its needs from other copies of the data.
- *Increased scalability*: When there are many copies of a special data in different parts of the system, users who need this data, can be added to the system more easily.

1.3. **General view toward data replication algorithm in the grid environment.** Since huge data transfers between places with large distances by Internet with limited bandwidth have caused increased consumption of the network bandwidth and also increased file transmission time, one of the solution for decreasing consumption bandwidth an decreasing the delay of the network is that the task is done near places where required files are available for that task and these methods are characterized by task scheduling algorithms[32]. But another method is to put files copies of required task in places near to places of task execution so that these methods are determined by replication data algorithms. Generally it can be said that replication data algorithms try to copy data required for task execution in places near to the place of task execution in order to save the network bandwidth and decrease the network delay but some questions usually are arisen and different replication data algorithms try to answer them as follows:

- When a site needs a file and does not have that file in its storage element, it should catch that file from the site which has a copy of it but since many copies may exist in the system, from which sites will the requesting site catch file copies ?
- What file or files are replicated?
- When the file is replicated?
- Where copied file should be saved?

- If there is no enough space where the file copy should be stored, how is this space provided?
- How many optimized copies of a file in the system are there?
- Are there any methods to predict files required for a task before they are needed and store in appropriate locations?

In this article some replication data algorithms are classified and discussed according to answers to the above questions and it is determined how each algorithm answers to the above questions.

2. Replication data algorithms

From one point of view, replication data algorithms are divided into two groups:

- *Static data replication*: In these methods the place of copies is specified while designing the system and these copies are placed in specified locations while configuring and their places are stable until the last activity of the system. These algorithms are suitable for cases in which users' s access pattern is the same and if it changes, the system should be redesigned and reconfigured because the present configuration may be the worst one for new pattern which is the disadvantage of this method. In [12] the writers showed that putting optimally file copies in different sites in a static form is a problem in Non-deterministic Polynomial-time hard (NP-hard) type and it is non-approximable. Considering parameters of file size and distances among sites and capacity of each site, they have formulated a linear planning problem in a method that system cost is minimized in addition to correct system limitations but this problem was NP-hard and non-approximable. In order to solve this problem, they forced to simplify the conditions and limitations of the system and changed the real system into a simplified system. Then, they solved problem of inserting file copies in different places of the system in order to minimize system costs. The main advantage of the static replication is its easiness.
- *Dynamic data replication*: In these methods, configuration of copies is specified dynamically based on the current needs of users and these methods controls continuously users' request

pattern and if any changes, process of arrangement and configuration of copies will be changed in a method that optimal system configuration is maintained [1,2,3,5,6,11,14,16,17,20-26,30-36,38,40-52].

From another view, data replication algorithms are divided into two categories:

- *Centralized Decision*: In these methods type, there is only one central decision maker for file replication in the whole grid system. This node makes decision with information collected from all nodes and considering designed algorithm, it decides about what file(s), when, in which node(s) and how many should be replicated. The advantage of this type of decision is that it prevents from proliferation of unnecessary data, but the disadvantage is that the system decision node becomes the bottleneck of the system and can cause many traffic in the system and also workload of this node is high and causes imbalanced workload in the system. Also the uniqueness of the decision maker node causes reliability and fault tolerance of the system to reduce because by damaging this point the system will not work and transferring information from every node in the system to decision maker node will cause over consumption of network bandwidth[2,6,8,23,28,31,36,37,40,43,47].
- *Distributed Decision*: In this kind of decision, every node decides separately about what file, when and where should be replicated. The advantage of this method is that there is no bottleneck node in the system and its disadvantages is the proliferation of unnecessary data from each node decides separately. [22,26,35,47,48,49-52].

2.1. **A general view toward the architectures of grid environment.** The replication data methods in grid environment have a close relationship with the architecture of that environment; it means that before proposing every algorithm, the architecture is considered and services present on that architecture are defined, then the replication data algorithm is represented using special characteristics of that architecture and its defined services

One of the main architecture of grid environment is Multi-Tier that many of the replication data algorithms are defined according to it

Multi-Tier data grid is a grid with tree structure which was firstly introduced in model of networked analysis at regional center (MONARC)

[53] project for distributed computing modeling the large hadron collider (LHC) [54] in CERN [55]. This architecture which is used for sharing the produced data in CERN is supported by European Union and is called EU Data Grid [56]. In this model raw data is stored in Tier 0 of CERN which is the root. These raw data are used and analyzed by regional centers located in the next Tier (Tier1). These regional centers may consist of one or more countries which are near each other' s geographically. Many research, industrial and university centers are in Tier 2. Finally, these data are used by hundred or thousand users located in Tier 3. In this architecture leaves are computational sites that can either process tasks or store data, middle nodes only can store the data. On first data are only placed in root node then they are replicated in middle nodes and leaves. The requests move from leaves toward the root and firstly storage element of every node is searched in this course in order to find the requested file, if the file is found, the requesting node will be responded by that node, otherwise the request is sent to the parent node in a higher level

But the limitations existed in this architecture are actually the limitations of tree architecture: there is one unique path from every leaf to the root and every child just relates with its parent directly and children do not have any direct relation with each other

Multi-Tier architecture has many differences with real architecture of grid environment. One another architecture was proposed which is called sibling in order to improve this structure, in this architecture which also has tree structure, co-ancestor nodes not only relate with parents and children but also relate with their sibling. Like tree architecture, in this architecture file requests are just transmitted from the leaves and middle nodes just can store data. In this architecture when there is a request from a leaf to a parent node in a higher level, if the requested file is not in the parent node, firstly file request will reach the sibling of parent node and is searched in those nodes and if it is not found in them, the request will be transmitted to one higher level

Another architecture which is seen in many articles is Graph architecture. This architecture is closer to real architecture of grid environment so that there is not any central node and data is stored in different nodes and all nodes can process and store them and these nodes are interconnected. Some researchers in general graph theory have divided the nodes into two or three levels and in order that nodes in which there is a communication with higher bandwidth are placed

in one group and those with low bandwidth communication are placed in different groups. In this kind of architecture, replication data algorithms try to copy files in the sites in a way that file transfer is done by high bandwidth communication in order to decrease network delay

In some architecture, combination of two or more topologies is used for example combination topology of tree with topology of circle. Also some defined frameworks try to use general graph architecture and simplify it as a tree topology

2.2. **The architecture used in researches under study.** In [20, 26, 33, 34, 45] tree topology has been used. Additionally, a weight is attributed to each node of the tree in [34] which is defined according to the usefulness of the node; the defined weight for middle nodes is higher than the final nodes

Sibling architecture has been used in [40]

In architecture [38] firstly it has been supposed that all main files are in site zero and this site does not accept any task and other sites have been expressed by two features, one is the number of that site requests in the present period for a special file and the other is the distance of that site to the zero site and this distance has been introduced in seconds. If one site connects to zero site directly its distance to zero site will be the number expressed in the link otherwise the distance will be sum of distances from zero site to the specified site

Two-level hierarchical architecture has been used in [37,42,43]. In the first level, sites which are close to each other geographically and have high bandwidth communication are located in the same region and at the second level; the regions with low bandwidth communication are connected together

The architecture used in [2,23,30,31] is three-level hierarchical structure. In the first level, sites connected by internal local network LAN with high bandwidth constitutes sub regions and in the second level sub regions are connected together by internal network LAN with intermediate bandwidth and regions are formed and in the third level, the regions are interconnected by internet with low bandwidth

Architectural structure in [47] is the collection of domains and every domain contains one copy server and some computational sites. The duty of every copy server is to store files copies and to maintain the information of files accesses by computational sites present in the domain in the present session. The copy server that is co-domain with a computational site is called primary copy server and the other copy servers are called secondary copy servers

In [21] authors combine tree topology of grid architecture with multi-ring topology of peer to peer architecture to model replica network topology

Graph architecture has been used in [17] in which the nodes are divided into two kinds. One of them is consuming center which requests files and the other is delivery center where file copies are placed

The architecture used in [13] has a complicated network structure

In architecture suggested in [28], every cluster contains some sites and one local replication controller (LRC) which are connected together by LAN. LRC keeps one local replication table (LRT) which contains the file name, the file place, accessing to the file, its weight, and master file for recording the information of file availability. The weight of the file shows it popularity. Count of file accessibility shows access number of sites related to a cluster to a file. All master files are distributed in different cluster sites. Global replica controller (GRC) is a central server which is connected to all clusters through Internet. This server records sum of file accessibility by all sites in order to determine what file should be replicated in what cluster. GRC maintains a general replication table (GRT) which is a collection of information recorded in LRTs

In [6], the architectural structure contains a server node and some user nodes that can connect together. The server node has the main storage place such that all data are stored there. In this structure one path which is the shortest one between user node and server node is considered as the main path. When a user node is requested for one file and there is no local access to that file, the request moves across the shortest path to find one copy of that file

The architecture used in [41] is a tree architecture in which grid sites are in the lowest level, some grid sites shape a virtual organization. There is one local server for every virtual organization in which Replica Catalogue is placed. It should be noted that the accessible bandwidth among sites in a virtual organization is higher than that of different virtual organizations. In the higher layer there is one regional server which contains one or more virtual organizations. Regional servers are connected by Internet with low bandwidth. In addition, there is one replica catalogue table in every regional server that is a directory of all stored files on that region

2.3. **Basic definitions in dynamic data replication.**

- *Popular files*: these are files that are requested highly in the system

- *Temporal locality*: files accessed recently will be requested again
- *Geographical locality*: files requested by a user will be requested by users who are geographically close to him
- *Space locality*: files related to the requested file will be requested later
- *Temporal network- level*: a site without its requested file tries to get its favorable file from a site that has as high bandwidth as that site.

It should be noted that in most of proposed algorithms, the system has been assumed in form of read only for simplicity [8, 11, 14, 16, 17, 20, 24, 25, 33-38,40,41,43,46,47,50,51]. But in few algorithms, a Read-Write system has been considered [30, 31, 44, 45] because keeping file copies updated simultaneously makes the system complicated based on this assumption

As mentioned before, the dynamic data replication algorithms try to answer the questions including from where is the required file caught? What and when is file replicated? Where is the copy? How many optimal replications are there in the system? Here we try to review some answers dynamic data replication algorithms have given these questions

2.4. **Choosing a suitable site for fetching files.** When a site requires a file for a task execution firstly it will search it on its storage element (SE). If it is found, it will access it locally otherwise it tries to identify the sites containing that file and choose the best site among them and fetch the specified file. According to the specified architecture and represented data replication algorithm, each method has chosen the best site as follows:

In [8,37,42,47] in which two-level hieratical architecture has been used, at first the region related to requested site searches for sites containing requested file in order to find the most proper site for file fetching. If such sites are found, site that have as high bandwidth as the site requesting file, will be selected and the file is fetched from that site but if in region related to requesting site, there is no site containing that file, sites containing that file will be searched for in other regions and again among them, the site that has as high bandwidth as requesting site will be selected

In [23] in which three-level hieratical architecture has been used, firstly sites containing file that are co-sub region with requesting site will be searched for. If there is no site, site containing file will be searched among sites that are in the co-region with requesting site. If

it is not found, other regions will be searched for the site. In other words, this method tries to fetch the file from site which has as high bandwidth as requesting site

In [30,31], again three-level hieratical architecture is considered and the method of searching for site containing the file is like the method mentioned in [23] but the difference is that in these methods the system is Read-Write and the bandwidth has been considered the same in each level. According to these two characteristics, in order to fetch file, a site with the shortest file request queue is chosen among sites containing file in each level

In [46] two parameters are considered in order to select the suitable site

If M shows sites containing the specified file, for every site $i \in M$, if H_i shows the least distance of site to file requesting site and also B_i shows the highest bandwidth accessible from i site to file requesting site, then a site will be suitable for file fetching if $\min_{i=1}^{|M|}(\frac{H_i}{B_i})$ is satisfied.

In [50], the file is fetched from a site containing requested a file that is closest to the requesting site

In [3], a site with the least file access cost and the least response time to that request was chosen among sites containing file replica. In order to determine this site, file requesting site sends a bid to all sites containing file copy. After receiving bid, each site sends a bid back to the requesting site and after examining all bid backs, the requesting site will choose a site the least access cost and the shortest waiting time in queue

In [11] two parameters are considered to determine the most suitable site.

One of them is "*bandwidth*" parameter because it is better to fetch file from the site which has as high bandwidth as the requesting site. The second parameter is "*system reliability*", it is better to fetch the file copy from the site which often connects to the system So among the sites containing file copy, a site will be chosen which has the highest amount of $\max(BW_{s_i,s_j} * utility\ S_j)$where s_i is the requesting site and s_j is the site containing file and BW_{s_i,s_j} is bandwidth between s_i and s_j and a site that is always connected to grid, has "*utility*" better than another site that is sometimes connected and sometimes disconnected.

In [28], file is fetched from the nearest cluster.

In [52],in order to obtain the most suitable site the parameters of "*processing power cpu*(W_c)", "*network transfer power* (W_n)" and "*I/O disk power* (W_{dio})" are considered for every site and these parameters

are obtained as follows:

$$W_c = frequency * (1 - U_{cpu}) \tag{1}$$

$$W_n = band * (1 - U_{band}) \tag{2}$$

$$W_{dio} = diskband/comm \tag{3}$$

where "*frequency*" is frequency of cpu,
"U_{cpu}" is the consumed amount of cpu, "*band*" is the network bandwidth, "U_{band}" is network consumed band width, "*diskband*" is the disks' bandwidth, "*Comm*" is the number of communications. Now by using these parameters, the following metric can be obtained.

$$lpc = \alpha * W_c + \beta * W_c + \gamma * W_{dio} \tag{4}$$

where α, β, γ are three constant values. This metric is obtained for all the sites containing the file then one of nodes that its *lpc* is more than threshold is selected and the file will be fetched. If all the sites have *lpc* less than threshold a site with the highest *lpc* will be chosen and the specified file will be fetched from there.

2.5. **Files suitable for replication.** Another question is that what file or files are good candidates for replication?

In order to answer this question in [3, 6, 8, 22, 23, 26, 31, 33, 37, 42, 45, 48, 49] , a file is introduced for suitable replication when a site needs that file for doing a task and cannot access to it locally, so after finding that file, it will be stored in a suitable place. But a problem is that the file is not a popular file for requesting site. It means that requesting site needs the file rarely so file replication will lose storage space in the system.

In [47], deciding about data replication has been discussed in two centralized and distributed forms. In centralized form, there is a copy master in which at beginning of every session, copy server sends collected information to the master and the master collects the information and keeps sum of file accesses in table H. Then, dynamic replication algorithm obtains mean access to the files and deletes the files that their access number is less than average from the table and replicates others. In distributed form, each copy server keeps table H for itself. When a site requests to access a file, primary copy server stores related information record in its background table and also distributes this information in other copy servers. Therefore, every copy server obtains the sum of file accesses throughout the system. Data replication algorithm determines the mean access numbers to files in the whole system and every copy server deletes file copies from table H

that their access numbers are less than average and tries to replicate others.

In [46], in order to obtain files that should be replicated, one calculates an average amount of data accessed in a period τ. For each file F let $|F|$ be the file size, let $M_\tau(F)$ be the number of times file F is accessed in a period τ and also let $N_\tau(F)$ be the number of replicas (file copies) of F in period τ. Denote by T_f the total number of single files in the data grid system during τ . The average amount of data accessed in period τ is calculated as follows:

$$D_{avg}(\tau) = \frac{\sum_{i=1}^{T_f} (M_\tau(F_i) * |F_i|)}{T_f} \tag{5}$$

The amount of data from F is $D_F(\tau) = M_\tau(F) * |F|$ is obtained. If $\frac{D_{avg}(\tau)}{D_F(\tau)} > N_\tau(F)$ then another copy of the file F should be made.

In [50], after fetching a file, the threshold amount related to that file will be calculated. If the number of file replication is less than its threshold, the file should be replicated, otherwise there is no need to replicate the file.

In order to obtain threshold amount related to every file f, if C_{sum} is sum of storage capacities of nodes in whole system and F_f is file access frequency, then "*access frequency(f)*"is calculated as follows:

$$AF_f = \frac{F_f}{\sum_{i=1}^{n} F_{f_i}} \tag{6}$$

Then the threshold is obtained from the following relation.

$$N_f = \frac{AF_f * C_{sum}}{S_f} \tag{7}$$

Where S_f is file size.

In [11], in order to find the best files for replication, two parameters have been used: "*the number of file requests*" and "*the number of file copies' access*" and mean file is obtained from dividing the number of file requests by the number of file copies' access. The mean file shows average number of the requests for a copy of that file. Then from dividing sum of requests for all system files by sum of current replica in the system, mean of the whole system is obtained. Now among the candidate files for replication, files with the mean greater than that of the whole system are chosen for replication

In many methods, the only criterion for determining the popularity of a file is the number of access to that file. A problem that arises here is that a file has been accessed more frequently in the past but recently

there is no need to it. This file is still considered and replicated as a popular file because of being accessed frequently in the past while in fact this file is not a popular file anymore.

In order to solve this problem in some methods like [7], intervals are considered and the frequency of files accesses are counted and stored. The main idea in this algorithm is that by giving different weights to access numbers in different time intervals, it is tried to control the role of access number in different time intervals in determining the popularity of the files. In this algorithm, the highest weights are given to access numbers which occurred in time intervals close to the present and the more the time interval of access frequencies from the present time, the less its role in determining the popularity of the file with its decreasing weight

In [43], a method has been proposed in which initially information is collected by region master from all region headers in all regions in constant intervals and a weight is assigned to information related to each interval. The number of times the file is accessed and file's weight are used to calculate access frequency. Among the candidate files for replication, a file with the highest access frequency is chosen for replication.

In other words if N_T is considered as the number of intervals and a_{ij} shows the number of access to i_{th} file in j_{th} interval and F is the collection of requested files, then "*access frequency for file* f, $(AF(f))$" is obtained by the following relation.

$$AF(f) = \sum_{i=1}^{N_T} (a_{ft} * 2^{-(N_t - t)}) \tag{8}$$

for all $f \in F$. The file which has the highest AF is introduced as a popular file.

In [29] the authors said that a weight-based method cannot be an exact method for determining the popularity of the files. They claimed that for determining the popularity of the files, the rate of increasing or decreasing of access to files is more important factor than the number of accesses. According to their idea, if the number of access to a file increases in recent intervals, even if the number of access to that file is low, this file will be more popular than the file that its previous access is high but the number of access is decreasing in recent intervals.

The rate of increasing or decreasing access number in two successive intervals is calculated by the following relation:

$$\alpha = \ln \frac{a_f^t}{a_f^{t-1}} \tag{9}$$

where a_f^t is access number to file f in interval t, and a_f^{t-1} is access number to file f in interval $t-1$.

If α is positive, it means that the number of accesses in present time interval has been increased compared to previous interval. Conversely, If α is negative, it means that the number of accesses has been decreased.

In [51] by using two parameters of "*sum of capacity of all nodes of the system*", "*sum of size of all files in the system*" and dividing the first parameter by the second one, a value is obtained that is called threshold limit for file replication so, if the number of file copies in the system is more than this threshold, there is no need to replicate that file otherwise, that file should be copied.

In [52], it was supposed that just main files are stored in some sites firstly. Now, in order to determine which file should be copied, a recorder of the number of access has been placed in each site where main files have been stored. Also, in this method a collection of threshold limits has been considered. Now, if the number of access to a file exceeds threshold limit, data replication will be started.

In [1], when a file is requested by a site that is not present in that site and other co-cluster sites, replication is started.

2.6. **Finding a suitable place for storing the file replica.** In some methods, the concept of temporal locality has been used to find a proper place for storing file replica. Therefore, a file requested by a site will be required by that site in the future, so these methods try to store the file copy requested in storage element (SE) of requesting site and if necessary, the site will access it locally in future. Thereby, bandwidth and implementation time of all tasks are decreased [1,3,6,8,23,26,28,37,46,48,49,50,51].

But the disadvantage of these methods is that they just have used the concept of temporal locality without considering the file access pattern during the time. In other words it is possible that the requested file is needed just one time by that site so replication of that file wastes the storage space, but in some methods, in addition to using the concept of temporal locality, geographical locality and file access pattern during the time have been used. They have stored requested file copy in a site that is as same region or cluster as the requesting

site (also as high bandwidth as requesting file) and also that site has the highest request for that file [31,42]. In this case, according to the concept of temporal locality if file requesting site needs that file in future, it can be fetched by a site which is as high bandwidth as the requesting site. On the other hand, a site that accesses many times to a file will need that file probably so it can access to that file locally.

Other researchers have considered other parameters to find suitable places for file storage. In [34] authors have considered two limitations from the user side, the first one is that the expense of whole system should be maximally equal to budget determined by user and the second is time spent for accessing to data replications is equal to the time specified by the user. Authors represented a nonlinear planning problem using by parameters of file size, the expense of file storage in site, the expense of file transfer between two sites and the weight of the nodes. The aim of this problem is to minimize total cost function which is sum of cost of file storage in the site and file transfer cost between two sites. In addition, limitations existed in the problem should be considered. For example, the cost of making copies should be maximally equal to the budget specified by the user and the number of file copies should be equal to the specified number [39], the sum of weight of nodes should be one (it should be noted that in architecture of this article, tree topology has been used and a weight has been attributed to each node based on its importance for the system), the capacity of site storage is limited, the transmission time of file between two sites is equal to the time specified by the user side but the complexity of this problem is NP-complete therefore, by ignoring details and simplifying the problem, the authors has changed it into a linier planning problem and then they have solves it, so they have found places with the least cost to store file copies.

In [24] a method has been presented that is very similar to the previous method [34], with this difference that in this method, a weight is not considered for the sites and a zero-one linear planning problem has been proposed by using other parameters mentioned above and solve it and in this case they have found the places of files copies where the cost becomes as low as possible.

In [44], three methods have been stated to find the most suitable place for file copy as follows:

- The first method, by combing two methods of Minimize ExpectUtil and best client will overcome problems mentioned above. In this case, among sites with the Lowest expect utile, the site

that is closest to best client site will be chosen for file replication. In addition, it should have the lowest distance to other sites.

Another problem of best client is that when the number of requests for a file is more than the threshold, copy of that file will be stored in that site without considering the related storage cost and file transfer cost, because it is possible file transfer cost to the requesting site is less than storage cost in that site, so in this case it is cost-effective to fetch the file only to that site.

- The second method solves the problem mentioned above, the site which has the most requests for file should be considered. This method calculates remote access cost from the requesting site to the file using parameters of write rate, read rate, bandwidth between the requesting site and site containing that file, physical distance between these two sites and the size of the file. If this expense is more than the threshold, it will be clear that storing a file copy on the requesting site is cost effective and a copy will be stored on that site, otherwise it will refer to the next site which has the most requests for that file.
- In the third method which is called Topology based replication strategy, the site is chosen for insertion of file copy based on topology network, meaning it tries to store file in a site which has the most connection with other sites.

But in this method, there are two problems; firstly, in order to find the best site regardless of the connection expense of two sites, just the direct connection between two sites are considered, consider this scenario: suppose that there are two sites s_1, s_2 in the network where site s_1 has a direct connection with all other sites and this connection has high cost. Site s_2 has low direct connection with other sites but sum of cost of direct and indirect connections of this site with other sites is less than the sum of direct connection cost of site s_1 with other sites. According to Topology based replication strategy, a suitable site s_1 is chosen for insertion of file copies but it is clear that site s_2 is more suitable. The second problem of this method is that connection pattern is not considered during the time, it means that maybe a site has the highest connection with other sites during this period but it is separated from the system in the next period. It is better to use the mean number of site connections during recent several periods.

In simulation, these three methods have been compared with three methods of No replication, Best client, Fast spread and the results

showed that three methods suggested in this article has low access expense.

As previous method, in [13] a method has been represented which tries to store data replication in a site which has the most connection with other sites but the topology used in this method is also a complicated network. The probability of storing data replication in every site is proportional to the number of connections of that site with other sites. In this method, in order to choose appropriate site for saving file copy, among the sites which have the most number of connections with other sites, a site will be chosen which has the lowest expense including the storage cost and file transfer cost to that site.

Each of the methods proposed in [44] just find one place for putting the data copy, and if we want to get several places for putting some copies of one file, these methods should be repeated several times.

But in [17] a method was proposed that is called Facility Location (FL) Problem and if this method is done one time, several places will be specified for putting some copies of one file. By using two parameters of storage cost of file copy and file copy transfer cost between two sites and calculation of sum of these two costs, in this method, it is tried to get the places of file copies in a way that the lowest cost is obtained.

In this research, there has been another method in which by formulating a linear planning problem it tries to decrease bandwidth in addition to the access expense to files.

In simulation of FL and comparing this method with other methods stated in [38], it has been shown that bandwidth used by proposed methods in [38] are 1.7 times of bandwidth used by FL and also the time delay in methods of [38] is 1.73 times of time delay present in FL method.

In [47], two methods have been proposed to find the best place for file copy insertion. Before discussing these two methods, we note that the architectural structure used in this paper is composed of several domains and each domain consists of a copy server which keeps the file copies and a number of computing sites and all sites present in a domain have the same bandwidth, with a particular copy server. These two methods are mentioned below:

- The first method which is called response-time oriented replica placement assumes that the more processing capacity of sites related to a domain, that domain is able to do more tasks, so

more files are required for doing those tasks. Therefore considering parameters of each domain processing capacity, the probability of request of each file and bandwidth of each domain node to copy server containing the file, metric of average response time to all requests is obtained for a particular file and a site is selected for storing a file copy in a way that the minimum value of this metric is obtained.

- The second method which is called Server merit oriented replica placement obtains a metric location using two parameters of processing capacity of domain related to that copy server and bandwidth between that server copy and other sites. Then among copy servers, the one that has the lowest metric location is chosen to put file copy.

In [14] a method has been proposed which tries to put related file copies close to each other in a small region and close to a site where task is done. In this method by using the file size parameters, the bandwidth between sites, a metric is defined by which closeness to placement of files required to a task is calculated.

In [20,33] there is a method in which it is assumed that the number of different files existed in the system is M, three parameters are considered, One of them is the rate of file reading by a node which shows the number of file accesses in a specified time period through that node, the second refers to the cost related to the link between two nodes and the third is the cost of storing copy of file in the node. Now suppose that sites where a copy of file is saved are known. Storage costs of file copies in the sites are obtained. Then, service costs to the requests from all system nodes for reading the file are calculated. Total cost is obtained from sum of these costs. Then the same operation is done for M file copies in the system and total system cost is obtained from sum of costs. The place of M file copies is regulated in the system in a way that cost of the whole system is minimized.

The complexity of this method is in the worst state of $O(nhk)$ in which n is the number of nodes in the system, h is the height of the tree (tree topology is used in the architecture of this article) and k is the number of files copies in the system.

In [35] in order to find the most suitable sites for putting the file copies, a software agent is placed in every site. This agent obtains a rank for each site according to the following parameters. Then among sites, the one which has the highest rank is chosen for copying file.

The parameters include "*band-rate*" which shows the structure of connection channel of specified site, this factor will influence data transmission time in a way that the more the band-rate of a site, the less file transmission time from that site to the requested site. "*Load*" shows the existed load on the cpu of a site. "*Demand Load*" shows the number of site requests for the file. "*cpu rating*" shows the cpu processing capacity. "*Storage capacity*" shows the storage capacity of the site.

In [11] a suitable site will be considered for copying file if it requests a file frequently and connects to grid system several times.

In this method, a metric called "*estimating the number of requests for file f by site*S_i" is obtained by using two parameters of "*the number of request for file f by site* S_i" and "*site usefulness*S_i" and their multiplication. Then the mean request numbers for file f is obtained by n site of grid environment, and then a site is considered suitable for putting f file copy in which the number of responded requests for f file by that site is more than the mean request number for file f by n site of grid environment. It should be noted that since grid environment is dynamic and in every moment, a site can leave or bind the grid environment so the usefulness of every site is variable. Usefulness has a direct relation with request numbers responded by the site (request for a file which exists on that site) and has a reverse relation with the request numbers for which the site is not able to respond.

In [43], in order to determine where the popular file should be stored, at first region master receives from all region headers information related to popular file in specified time intervals and then $AF(f)$ is obtained for every region from relation(8) and then regions are arranged in descending order based on $AF(f)$ and the first region of this ordered list is the best candidate for copying file f.

In [52] a method was proposed in which sites are chosen to store file copies in a way that the system load is balanced. In this method by using parameters "*cpu processing power*", "*the empty space of disk*", "*free memory*", "*I/O speed*", "*the consumed bandwidth*" and specifying one weight for each of them, a metric called "*load processing power*" is defined for every site. When a copy of file is required to be stored, "*load processing power*" is obtained for all sites and a site which has the highest load processing power will be chosen for storing file copy.

In [45] which have been used tree topology, in order to obtain the most suitable places for putting file copy, at first a cost function has

been defined. Then by using this cost function for every site from leaves to the root, first the local storage cost of file copy and then the file transfer cost to that node will be calculated and this information will be transferred to the parent node. This process is continued until the information reaches the tree root and then data replication storage or transfer in the site is decided from root to the leaves. It should be noted that in this method, in order to obtain cost of data replication storage in the site, cost of file reading and cost of file updating will be used.

In the algorithm stated in [36], at first it is supposed that there is not any copy of the files in the sites and just the main files are stored in the sites, in each step of algorithm, a file is copied in the site in which the total access cost of data grid has most reduction. This algorithm continues until all files are replicated in the sites or the total access cost is not decreased in grid anymore.

In [25], like the previous method, first it is supposed that no copy of files exists in any site and just the main files are stored in sites. In the first step, the access cost to all files is obtained in the grid system by using bandwidth parameters, "*the size of every file*", "*the least number of connections between sites*", "*the number of every file required by every site*". In the second step, benefit resulted from copying every file in every site is obtained. In order to obtain the benefit of file replication in a site, it is supposed that the file is stored in the site and then again access cost to files in the system is obtained. Then, obtained value is subtracted from value obtained in the first step. Obtained differential value is the benefit resulted from copying the file in the site. Finally among sites, a site where replication has the highest interest in the system will be chosen as an appropriate site for file replication.

2.7. **Creating the space required for file storage.** Another question outlined here is that when a suitable site is chosen for file replication and that site does not have necessary space for copying how should this space be prepared? Two basic methods are "*Least Recently Used*" (LRU) and "*Least Frequently Used*" (LFU), these two methods usually are used for comparing efficiency of substituted algorithms.

In [37,13] if there is not enough space for storage, at first it is checked if the requested file exists in a site as same region as the requested site, storage of file copy in storage element of requested site (SE) will not be needed, otherwise at first the files existed on SE are ordered in an ascending form according to the access number to them and from the

beginning of arranged list the files existed in sites that are as same region as requesting site will be omitted to provide necessary space. If all these files are omitted but the required space is not provided, then from the beginning of the ordered list the files which access number is lower than the access number of requested file will be omitted to provide the necessary space. If the required space is reached, the requested file will be replicated there, otherwise, replication of requested file will not be done.

In [8] if there is another copy from the requested file in one of co-cluster sites with SE, file replication will not be required, otherwise, all files existed on SE are omitted which another copy of them exists on the co-cluster sites with SE. If the required space is not provided, according to the algorithm LFU, the files on SE will be omitted to provide necessary space for storing file copy.

In [23] it is checked whether the copy of requested file exists in co-sub region sites with SE or not? If so, file storage will not be required in SE, otherwise all the files on SE for which there is another copy on co- sub region sites with SE, are arranged according to LRU and then from the beginning of organized list, omission is done until necessary space is provided for file replication or all the files are omitted. If the necessary space is not achieved, all the files on SE for which there are other copies in co-region sites with SE are arranged according to LRU and then omission is done from the beginning of the ordered list until necessary space is provided for file replication or all the files are omitted. Again if necessary space is not provided from the beginning of the list, all the files are omitted until the necessary space for storing the requested file is provided.

In [49] a method by using the popular file parameters, estimating the most and the least amounts of file popularity, the distance between the requesting site to site containing file and the bandwidth between these two sites, a profit model is defined. By using this model, the profit of requested file replication is calculated in the requesting site and also the profit of all files in the requesting site is reached. Now the files that their profit is lower than that of requested file replication profit will be omitted and the requested file substitutes it.

In [40] among file replications presented on the site, those which have been produced in the remote time than the current time and have not been accessible in the current time session are selected as deletion candidates and are transformed to one higher level in parent node

and this action will continue until the space required for replication is provided.

In [3], files existed on SE are arranged according to the number of access in an ascending order and from the beginning list files that their access numbers are lower than request numbers for favorite file, will be omitted to get the necessary space for storing the requested file.

In [43], like previous method the files existed on SE are arranged according to their access numbers in an ascending order and from the beginning of the list, files will be omitted to get the necessary space for storing the requested file.

In [52] in order to replace files, this fact has been used that among the files existed on SE, if files with larger sizes are omitted, then freer space will be produced and the number of substitutes become lower. In this method, a metric called "*file value*" is reached by using parameters of file production time, the number of access to it and the size of it and their summation. So in order to determine which files should be deleted to provide the necessary space, for all files existed on SE, metric of file value will be calculated and then files are arranged according to their values in an ascending order and from the beginning of the list less valuable files will be deleted to provide the required space.

In [6] if there is not necessary space and it is essential to delete a group of existed file copies on that node in order to reach enough space, the deletion of that group and storing new copy will be done if the value of new file copy is more than the value of group of file copies. In this method, authors have considered four important parameters in order to calculate the value of copy groups and new file copy: the number of requests, frequency of requests, the size of file copy and the last time that a file copy has been requested.

The value of selected file groups for deletion is obtained by the following relation that is shown by GV

$$GV = \frac{\sum_{i=1}^{n} NOR_i}{\sum_{i=1}^{n} S_i} + \frac{\sum_{i=1}^{n} NORFSTI_i}{FSTI} + \frac{1}{CT - \frac{\sum_{i=1}^{n} LRT_i}{n}} \tag{10}$$

where n shows the number of copies in the group, NOR_i shows the number of times in which i_{th} file copy is requested, S_i shows the size of i_{th}file copy, $FSTI$ is a specific interval for replication, $NORFSTI_i$ is the number of times that i_{th}file copy is requested in $FSTI$ time interval, CT shows the current time of the system and LRT shows the last requested time for i_{th} file group copy.

The value of new file copy which is shown by RV is reached from the

following relation:

$$RV = \frac{NOR}{S} + \frac{NORFSTI}{FSTI} - \frac{1}{CT - LRT} \tag{11}$$

Where NOR shows the number of requests for new file copy, S the size of this file copy, $FSTI$ is a replication special time interval, $NORFSTI$ shows the number of requests for new data copy in $FSTI$, CT shows the current time system and LRT specify the last time requested for new data copy.

In [31] at first from files existed in both requesting site and local LAN, a list has been arranged from old to new files and they will be deleted from the beginning of the list to provide the necessary space. If the necessary space is not reached, according to LRU, an organized list will be chosen from files that a copy of them exist in the current site and another copy of them in local region, and files will be deleted from the beginning of the list and again if the necessary space is not reached, the remaining files will be placed in an ordered list according to LRU and then files will be deleted from the beginning of the list to provide the necessary space.

In [1], at first among files existed on SE for which there is another copy on co-region site with SE, some files will be chosen and deleted based on LRU to get the necessary space, if the necessary space is not obtained, this algorithm will omit files based on LRU for which there are copies in other sites that are not as same region as SE to get the necessary space.

2.8. **Calculate the optimum number of copies of each file in the system.** In [14] by using parameters of the size of files,the number of access to every file $NOA(f)$ and the number of all different files in system N, the following metric is reached which shows the mean data access in the system.

$$\bar{\varphi} = \frac{\sum NOA(f) * |f|}{N} \tag{12}$$

The access amount of every file is reached through relation $\varphi_f = NOA(f) * |f|$ then the optimum amount is got for the numbers of copies of every file from the following relation.

$$iNOR(f) = \frac{\bar{\varphi}}{\varphi_f} \tag{13}$$

If the number of copies present in the system for each file is lower than optimal number of copies, this method begins to produce file copies in order that their numbers are equal to optimal number of copies. But if number of present file copies is higher than optimal number of copies, they will be deleted in order that their numbers are equal to optimal numbers.

In [33] total cost of access to all file copies is obtained by using parameters of file reading cost, storage cost of file in the site and transmission cost between two sites. Now, number of file copies in the system is reached in a way that the cost becomes the least possible amount.

In [24] to clime file availability depends on the network error rate and accuracy of replications positioning service.

In this study, in order to find number of replications required for each file in the system, it is assumed that M^f is number of replications required for each file f. q and p are probability of a node and a link availability and A^f is availability required for file f. The probability that none of M^f of file replications f is available is equal to $(1-pq)^{M^f}$. Therefore, in order to insure availability required for file f, $1-(1-pq)^{M^f} \geq A^f$ should be satisfied. Thus, M^f can be obtained.
In [43], following value is calculated in order to determine number of popular file replications (P):

$$AF_{avg}(p) = \frac{AF(p)}{N_T} \tag{14}$$

Where its parameters have been defined in (8) then following amount is obtained:

$$AF_{avg}(f) = \frac{[AF(f)]_{sum}}{N_F * N_T} \tag{15}$$

for all $f \in F$, where $[AF(f)]_{sum}$ shows sum of all AFs for requested files. Number of replica required for popular file is obtained from following relation:

$$Num_{system}(p) = \frac{AF_{avg}(p)}{AF_{avg}(f)} \tag{16}$$

2.9. **Methods of prefetching files.** Another question outlined here is whether or not files required for a task execution is predicted before they are needed and stored them close to place of task execution to

if necessary fetch them from closer places and reduce time of task implementation.

Some researchers have studied these methods as follows:

In [48], when a file is requested, a sequence of files related to it is fetched in requesting site. In order to find files related to a file, concept of space locality has been used meaning files that are semantically close to that file. In this method, it has been assumed that site requires sequentially files that are semantically similar. In addition, the authors used concept of virtual organization. Virtual organization is not the number of sites that are geographically close to each other rather it means sites that have similar requests even if they are not geographically close to each other. Therefore, despite local differences in access pattern to files in different sites, sites member in virtual organization access continuously to files that are conceptually similar to each other. There is a data base in each site where a sequence of files requested continuously by that site is placed. It should be noted that files placed in a sequence, are called adjacent files. Corresponding to this sequence, there is a sequence of times in which each member of sequence shows request time for corresponding file.

Process of function of this algorithm is such that when a site requires a file for task execution and there is no local access to that file, among remote sites containing file, one is chosen and the request for file transfer is sent to that remote site. That remote site sends file to requesting site and at the same time it tries to reach files adjacent to requested file from its database and sends a message to local site including list of files adjacent to requested file. At the end, local site chooses some of adjacent files and replicates them in order to reduce response time in the system.

In [26], a method has been presented that is called predictive hierarchical fast spread (PHFS). In order to find files related to requested file, space locality has been emphasized. Tree topology has been used in architecture of this paper. This method is a modification of fast spread. In fast spread, as requested file moves toward requesting node, it is replicated in all nodes of this path. In this method, other files related to requested file, are replicated in all nodes present in the path in addition to requested file. Replication is hierarchical meaning many replications are provided in higher levels and numbers of replications are reduced by moving to downward levels.

In order to find files related to requested file, users' files access information is gathered from the system and stored in users' access log

files. Then, PHFS exerts data mining methods such as clustering and association rules on users' access log files in order to extract proper information from log files. By exerting data mining methods on users' access log files, a cluster of files is obtained showing files that are semantically close to each other. The basis of this inference is space locality in a way that files related to requested file are the most probable next requests.

In this method, a predicted working set is formed for each requested file where files related semantically to requested file are placed. In order to identify files related to requested file, semantic relation between two files is shown by a number ranged from 0 to 1. Number 1 shows complete semantic relation and 0 shows lack of relation between two files. Having these quantitative values for semantic relation between two files, it is not difficult to find files that are semantically relevant to a specified file more than other files.

For this purpose, consider $(0 < \alpha < 1)$ constant as a criterion. Files that are relevant to a specified file for example A, more than αwill be files related to A and files that are less relevant than α will be as irrelevant files. α value is determined in design stage of system based on different factors such as memory available in the system and mean relation among files. It will be constant during system activity.

Despite files present in PWS that belong to A have been made based on space locality, but it is rational that these files contain limits of geographical and time localities. On the other words, it is highly possible that files present in PWS are accessed from a geographical region and a specific time interval.

In [41], a method has been presented to prefetch files required for the site. In this method, tier structure has been used to store file access pattern. This structure is similar to a tree in which the root is the highest level. Sites are placed in the second level and files accessed by sites are placed in the third level. In this method, a tier is used for each virtual organization.

When a file is requested by a site, at first the request should be added to tree structure. In order to add a file, time interval between request of new file and the last request is calculated. If it exceeds threshold and there is a node in tier for that file, only one unit will be added to the amount of file access but if there is no node, a node will be produced in level 3 for that file and access number 1 is put for it. If time interval is less than threshold, requested file will be added as a child node for the last node of previous request. This node is

introduced as the last request. Thereby, after a file is requested by the site, tier related to virtual organization will be up to dated.

When a file is requested by a site and there is no local access to that file, a request will be sent to other sites to find the file. When a remote site finds requested file in its SE, a sequence of files related to requested file is obtained using tier related to its virtual organization. Then among files in which number of their editions in a certain time period is more than threshold are deleted (these files are changed frequently during time so they are not good candidates for fetching). Then, other files present in the sequence are sent from sites that have the least relational cost to the requesting site with requested file.

2.10. **Particular methods.** Some researchers have attempted to deal with other aspects of data replication. For example:

In [9], files fragmentation and fragmented replications are outlined. In this method, it is assumed that file replications stored in different places are divided into blocks. When a site requires a file for a task execution, authors suggest that instead of fetching file from a site, different parts of file are fetched from different places in parallel form so that file access time is reduced. On the other hand, different parts of file can be replicated separately meaning if a site requires one or more parts of a file more than other parts, replication of those parts in the site is sufficient. It causes storage space to be saved.

In this method, it is assumed that blocks are interlocked in a fragmented file replication. If this assumption is deleted, problem solving will be difficult. An example has been taken in the paper showing that this method does not represent an optimal solution.

In [16], a study has been represented on efficiency of algorithms of dynamic data replication for real time file accesses. Network structure used in this study is tree topology containing three layers. All produced data have been stored in layer zero. Sites of layer 1 only have storage capacity and sites of layer 2 have both storage capacity and processing power. Tasks of all components in this study have been defined in a way that requested file will be accessed by requesting site in a certain time. In this study, 8 algorithms of dynamic data replication have been presented on this architecture:

- *Catching algorithm*: file requested by sites of layer 2 are cached directly from layer 0 and are replicated only in requesting site.
- *Cascading algorithm*: requested file is replicated in target site and if number of file requests in site of layer 0 is more than

threshold, the file will be replicated in site of layer 1 in path of target site and then it will be replicated in target site.

- *Fast spread algorithm*: required file is replicated in path of reaching the target in layer 1 in addition to target site.
- *Catching-pp algorithm*: requesting site searches for the file in other sites of layer 2. If it is found, the file will be fetched otherwise, it will be requested from site of layer 0, and then it will be replicated.
- *Cascading-pp algorithm*: if requested file is found in site of layer 1, it will be fetched otherwise all sites in layer 2 will be searched. If the file is found, it will be fetched otherwise, layer 0 will be searched for it. The rest of this method is as same as cascading.
- *Fast spread-pp algorithm*: the file is searched in site of layer 1. If it is not found, it will be searched in other sites of layer 2 and again if it is not found, layer 0 will be searched for. The rest of this method is as same as fast spread.
- *Cascading-Enh algorithm*: at first, requested file is searched in other sites of layer 2. If it is not found, it will be searched in site of layer 1 and again if it is not found, site of layer 0 will be searched for. The rest of this method is as same as cascading.
- *Fast spread-Enh*: at first, requested file is searched in sites of layer 2. If it is not found, it will be searched in site of layer 1 and again if it is not found, it will be searched in site of layer 0. The rest of this method is as same as fast spread.

Results of simulation showed that efficiency of algorithms is in a descending order as follows:

Fast spread Enh, Fast spread-pp, cascading Enh, fast spread, cascading-pp, caching-pp, cascading.

In [10], a data replication algorithm has been presented for dynamic period. In this research, it has been assumed that a dynamic period is based on number of tasks accepted in grid system.

If number of replications produced in period T_n is shown by $ReplicaT_n$, $n+1_{th}$ period i.e T_{n+1} is calculated as follows:

$$T_{n+1} = \frac{T_n}{\frac{ReplicaT_n}{T_n} + 1} \tag{17}$$

According to this formula, when number of replications made in period T_n is increased, duration of T_{n+1} will be reduced and vice versa. Thereby, automatically the period duration is up to dated. In order

to evaluate this method, best client method with static and dynamic periods has been evaluated in terms of two parameters of effective network usage (ENU) and total response time. Results of simulation showed that both parameters of ENU and total response time in dynamic period have had considerable improvement compared to those in static period. A question outlined here is that how much is the primary length of period? Experimental results showed that primary length of period influences two parameters of ENU and total response time. The less the length of this period, the better values is obtained for these parameters.

In [5], a framework has been defined for data replication in grid environment. Topology used in this method is complete graph topology. There is a server node and several work stations all of which are interrelated. Server node contains high storage capacity so that all data of grid environment are stored there. Nodes of work station request files but they have limited storage capacity so that they can store limited number of files. In this framework, a path is introduced from each work station to server node. This path is the shortest one between each work station and server node.

In this method, when a node requires a file and cannot access it locally, it sends its request to the closest node on the path. After receiving request, each node searches its SE for the file. If it is not found, the request will be sent to the next node in the shortest path otherwise requested file replication is sent from this node to requesting node on the same shortest path.

It can be said that the advantage of this method is graph topology but this topology has been changed into a tree topology by using server node and user nodes and importantly using the shortest path.

The author has compared methods of "*no replication*", "*plain caching*" and "*fast spread*" based on parameter of "*locally accessed file numbers*" under this framework and results of simulation showed that method of no replication has the worst efficiency for above mentioned parameter and the other methods have approximately the same efficiency but the method of plain caching have a little better efficiency than fast spread. These results suggest that above mentioned parameter is a necessary but not sufficient parameter. If parameters such as "*consumed bandwidth*", "*time of task implementation*", "*network delay*" are considered, certainly efficiency of fast spread will be better than plain caching.

It is noteworthy that there are some reviews on different methods of data replication like [4,15].

3. Main results

Before discussing about simulated results of the methods mentioned above, 6 basic strategies are stated and many researchers have compared the efficiency of their methods with above mentioned ones: these methods have been defined on multi-tier architecture.

- *No replication method*: data are only stored where it has been produced and there is no extra replication. This strategy is used as a basis for comparison with other strategies and it is not practical.
- *Best client method*: accesses of different users to present data in each node will be stored by that node. Accesses will be checked at certain intervals. If access numbers to a certain file exceed threshold, replication mechanism replicates it in a node that has the highest number of requests for that file.
- *Plain caching method*: file requested by the user is stored in user node.
- *Cascading method*: as the best client method, number of requests for files present on each node is kept by that node. When number of file accesses exceeds threshold in one node, a replication from that file is produced in lower level of the tree on the path that reaches the best client node (the node that has the highest file request). Process of moving to a lower level in the tree continues until the specific file reaches the best client node provided that best client node requests file for a long time otherwise, it is possible that file moves down in the tree and then it stops.
- *Cascading and caching method*: this strategy is combination of caching and cascading methods. Requested file is cached in requesting node and it move downward in hierarchy of tree levels.
- *Fast spread method*: requested file is replicated in all nodes on the path while going to the requesting node.

Before continuing, we should mention that the threshold is mentioned in previous methods is static, but in [27], the fuzzy cascading method is using "*file size*" and " *nods' different levels*" to calculate a dynamic threshold, this method decrease the response time and use the sources efficiently.

FL method proposed in [17] has less consumed bandwidth and time delay compared to 6 methods expressed in [38] in a way that methods present in [38] has consumed bandwidth of 1.7 times and time delay of 1.73 times more than FL method.

Comparing results of three methods proposed in [44] to three methods of No replication, Best client and Fast spread showed that three proposed method has less access cost than these three methods.

Simulated results of method expressed in [49] with Economy method present in simulator optorism have shown that method proposed in [49] has lower task implementation time, lower number of replications and generally higher efficiency.

Considering parameter of time of task implementation, method presented in [14] has less mean time of task implementation compared to two methods of LRU and LFU

BHR method presented in [37] has lower time of task implementation compared to two methods of LRU and Oldest (in this method, at first older files are deleted).

HRS method presented in [8] has lower time of task implementation and mean inter-cluster number of communications compared to three methods of LRU, LFU and BHR

Mean time of task implemented in [23] is lower than that of BHR and LRU

Method presented in [42] that is a modification of BHR method has lower time of task implementation and lower mean occupied space compared to methods of LRU, LFU and BHR

Considering parameter of mean response time, Simulated results of method presented in [40] perform better than no replication method but it spends more time compared to fast spread because in fast spread method, requested file is replicated in all nodes on the path and thus response time is improved but load resulted from it is high on the nodes. Considering parameter of percentage of storage usage, presented method occupies less space than fast spread method

Comparing results of PRA method presented in [48] to two methods of no replication and the best client shows that PRA method has less mean response time and lower consumed bandwidth

Comparing method presented in [22] to three methods of fast spread, LRU, Economy zipf for parameter of mean response time,

results showed that this method has been improved by 40 compared to two methods of LRU and Economy zipf and by 9% compared to fast spread method.

Comparing results of method presented in [50] with 4 methods of no replication, LRU, LFU and LALW showed that this algorithm has performed better than 4 methods considering parameter of mean time of total task implementations. It has been improved by 30% for parameter of effective network usage compared to these four methods. It should be noted that definition of effective network usage is as follows:

$$ENU = \frac{N_{remotefileaccess} + N_{filereplication}}{N_{remotefileaccess} + N_{localfileaccess}} \tag{18}$$

Each algorithm that reduces this parameter has better efficiency.

FIRE method presented in [3] has been compared to two methods of LRU and LFU under two different access patterns. Considering parameter of total time of task implementation under serial access pattern, this method has been improved by 25% compared to LRU method and by 14% compared to LFU method. This method has had worse efficiency compared to methods of LRU and LFU under random access pattern.

Comparing results of method proposed in [11] to methods of no replication and best client shows that when number of tasks in grid environment is increased, total time of task implementations has lower value in proposed method compared to other methods. In this algorithm, parameter of effective network usage has been 28-53% optimal compared to those two algorithms.

Method presented in [43] has been compared to three methods of no replication, LRU and LFU. Results showed that proposed method has lower mean time of total task execution and lower number of replications than three methods. But this method has had worse result for parameter of storage usage compared to three other methods because no replication method that does not replicate any file has the best performance considering this parameter. In two methods of LRU and LFU, when a file is needed, it will be replicated but in this method, files are replicated automatically so storage usage is more than those three methods. Proposed method performs better than other methods considering effective network usage.

Method presented in [51] has had total time of task implementation approximately 10 percent lower than methods of LRU and LFU. It also had better performance for parameter of effective network usage compared to those two methods.

In [6], fast spread method has been simulated with three sustitution methods of LRU, LFU and the method presented in this research that

is called EFS. Efficiency of these three methods has been evaluated with three scenarios. In the first scenario, probability of requesting all files is equal. In the second and third scenarios, there is a group of files that are requested more than others files. This group is shown with MWG that constitutes 10% of total file copies. In the second scenario, probability of requesting each files of group MGW is about 30%. In the third scenario, this probability is 50%. Simulated results have shown that EFS has lower response time and lower bandwidth under three scenarios compared to methods of LRU and LFU.

In tables 1 and 2, several characteristics of methods reviewed in this paper have been shown briefly

Ref	Replication Decision	Architecture	Reduce Response Time	Bandwidth consumption consideration	Load Balancing consideration	Fault tolerance consideration	Storage Assumption
[5,10,19]	Centralized	Hierarchical	Yes	No	No	No	Limited
[6]	Centralized	Tree	No	No	No	No	Limited
[8]	De-Centralized	Hierarchical	Yes	No	No	No	Limited
[11]	Decentralized	Graph	Yes	No	No	No	Unlimited
[12]	Decentralized	Graph	No	No	No	No	Unlimited
[14]	Decentralized	Graph	Yes	No	No	No	Limited
[16]	Centralized	Graph	Yes	No	No	No	Unlimited
[17]	Centralized	Tree	No	No	No	No	Unlimited
[20]	Decentralized	Siblingtree	Yes	No	No	No	Limited
[21]	Decentralized	Graph	Yes	Yes	No	No	Unlimited
[22]	Decentralized	Graph	Yes	Yes	No	No	Limited
[23]	Decentralized	Tree	Yes	Yes	No	No	Limited
[24]	Decentralized	Tree	No	No	No	No	Unlimited
[29]	Decentralized	Graph	Yes	No	No	No	Limited
[35]	Decentralized	Graph	Yes	No	Yes	No	Limited
[36]	Centralized	Graph	No	No	No	No	Limited
[37]	Decentralized	Graph	Yes	Yes	No	No	Limited
[38]	Decentralized	Graph	Yes	Yes	No	No	Limited
[39]	Decentralized	Tree	Yes	Yes	No	No	Limited
[40]	Centralized	Graph	Yes	Yes	No	No	Unlimited
[41]	Centralized	Graph	Yes	Yes	No	No	Limited
[45]	Decentralized	complex	No	No	No	No	Limited
[47]	Decentralized	Graph	Yes	Yes	No	No	Limited
[48]	Centralized	Tree	Yes	No	No	No	Limited
[50]	Centralized	Star	Yes	Yes	No	No	Limited
[52]	Centralized	Hierarchical	Yes	Yes	No	No	Limited
[53]	Centralized	Graph	Yes	No	No	No	Limited
[54]	Decentralized	Tree	Yes	Yes	No	No	Limited
[55]	Centralized	Graph	No	No	No	No	Limited
[56]	Centralized	Hierarchical	Yes	No	No	No	Limited

TABLE 1. Features of replication techniques studied in survey.

Ref	Storage Utilization	Reduce Access cost	Simulation	Complexity	Threshold Based	Optimal number of Replica	Replication cost consideration
[5,10,19]	No	No	Yes	-	No	No	No
[6]	No	Yes	No	Np-complemet	No	No	No
[8]	No	No	Yes	-	Yes	No	No
[11]	No	No	Yes	-	No	No	No
[12]	No	Yes	Yes	-	No	No	No
[14]	No	No	Yes	-	No	Yes	No
[16]	No	No	Yes	-	Yes	Yes	No
[17]	No	Yes	No	o(nhk)	No	No	Yes
[20]	Yes	No	Yes	-	Yes	No	No
[21]	No	Yes	Yes	-	No	No	No
[22]	No	No	Yes	-	No	No	No
[23]	No	No	Yes	-	Yes	No	No
[24]	No	Yes	No	o(nh)	No	Yes	No
[29]	No	No	Yes	-	No	No	No
[35]	No	No	Yes	-	No	No	No
[36]	No	Yes	No	-	No	Yes	Yes
[37]	No	No	Yes	-	Yes	Yes	No
[38]	No	No	Yes	-	No	No	Yes
[39]	No	No	Yes	-	Yes	No	No
[40]	No	No	Yes	-	Yes	No	No
[41]	No	No	Yes	-	No	Yes	No
[45]	Yes	No	Yes	-	No	No	No
[47]	No	No	Yes	-	Yes	No	No
[48]	No	No	Yes	-	Yes	No	No
[50]	No	No	Yes	-	Yes	No	No
[52]	Yes	No	Yes	-	No	No	No
[53]	No	No	Yes	-	Yes	No	No
[54]	No	No	Yes	-	Yes	No	No
[55]	No	Yes	Yes	-	No	No	No
[56]	No	No	Yes	-	No	No	No

TABLE 2. Features of replication techniques studied in survey.

REFERENCES

[1] S. Abdi and S. Mohamadi, **The impact of data replication on job scheduling performance in hierarchical data grid**, J. Applications Of Graph Theory in Wireless Ad Hoc Networks and Sensor Networks. 2, 3(2010),15-25.

[2] S. Abdi and S. Mohamadi, **Two level job scheduling and data replication in data grid**, J. Grid Computing and Applications.1, 1(2010), 23-37.

[3] A.R. Abdurrab and T. Xie, **FIRE: A file reunion based data replication strategy for data grids**, Proceeding of the 10th IEEE/ ACM International Conference on Cluster, Cloud and Grid Computing. (2010), 215-223.

[4] T. Amjad, M. Sher and A. Daud, **A survey of dynamic replication strategies for improving data availability in data grids**, J. Future Generation Computer Systems, 28, 3(2012), 337-349.

[5] S.M. Bsoul, **A framework for replication in data grid**, Proceeding of the International Conference on Networking, Sensing and Control Delft,the Netherlands. (2011), 233-235.
[6] M. Bsoul, A. Al-Khasawneh and E. Eddien, **Enhanced fast spread replication strategy for data grid**, J. Network and Computer Applications. 34, 2(2011), 575-580.
[7] R.S. Chang and H.P. Chang, **A dynamic data replication strategy using access-weights in data grids**, J. Supercomputing. 3, 45(2008), 277-295.
[8] R.S. Chang, J.S. Chang and S.Y. Lin, **Job scheduling and data replication on data grids**, J. Future Generation Computer Systems. 23, 7(2007), 846-860.
[9] R.S. Chang and P.H. Chen, **Complete and fragmented replica selection and retrieval in data grids**, J. Future Generation Computer Systems. 23, 4(2007), 536-546.
[10] F.B. Charrada, H. Ounelli and H. Chettaoui, **Dynamic period vs static period in data grid replication**, Proceeding of the International Conference on P2P, Parallel, Grid, Cloud and Internet Computing. (2010), 565-568.
[11] F.B. Charrada, H. Ounelli and H. Chettaoui, **An efficient replication strategy for dynamic data grids**, Proceeding of the International Conference on P2P, Parallel, Grid, Cloud and Internet Computing. (2010), 50-54.
[12] U. Cibej, B. Slivnik and B. Robic, **The complexity of static data replication in data grids**, J. Parallel Computing. 31, 8-9(2005), 900-912.
[13] C. Dan.wei, Z. Shu.tao, R. Xun.yi and K. Qiang, **Method for replica creation in data grids based on complex networks**, The Journal of China Universities of Posts and Telecommunications. 17, 4(2010), 110-115.
[14] N.N. Dang, S. Hwang and S.B. Lim, **Improvement of data grids performance by combining job scheduling with dynamic replication strategy**, Proceeding of the Sixth International Conference on Grid and Cooperative Computing.(2007).
[15] S. Dayyan and M.R. Khayyambashi, **IA comparative study of replication techniques in grid computing systems**, J. Science and Information security, 11, 9(2013).
[16] A. Dogan, **A study on performance of dynamic file replication algorithms for real-time file access in data grids**, J. Future Generation Computer Systems. 25, 8(2009), 829-839.
[17] X. Dong, J. Li, Z. Wu, D. Zhang and J. Xu, **On dynamic replication strategies in data service grids**, Proceeding of the 11th IEEE Symposium on Object Oriented Real-Time Distributed Computing. (2008), 155-161.
[18] I. Foster and C. Kesselman, **The grid: blueprint for a future computing infrastructure**, Morgan Kaufmann Publishers.(1999).
[19] I. Foster, C. Kesselman and S. Tuecke, **The anatomy of the grid**(2001),1-25.
[20] M. Garmehi and Y. Mansouri, **Optimal placement replication on data grid environments**, IEEE.(2007),190-195.
[21] Q. Gu, B. Chen and Y. Zhang, **Dynamic replica placement and location strategies for data grid**, Proceeding of the International Conference on Computer Science and Software Engineering. (2008), 35-40.

[22] L. Hong, Q.X. Dong, L. Xia, L. Zhen, W. Xing, **Fast cascading replication strategy for data grid**, Proceeding of the International Conference on Computer Science and Software Engineering. (2008), 186-189.

[23] A. Horri, R. Sepahvand and Gh. Dastghaibyfard, **A hierarchical scheduling and replication strategy**, J. Computer Science and Network Security. 8, 8(2008), 30-35.

[24] S. Jeon, C.H. Youn and H.J. Kim, **Optimal file replication scheme (C0-RLS) for data grids** , Advanced communication Technology, IEEE. (2004), 1055-1059.

[25] P.G. JeyaSheeli and M. Archanaa, **Efficient centralized data replication algorithm for data grids**, Proceeding of the International Conference on Computing, Electronics and Electrical Technologies. (2012), 900-904.

[26] L.M. Khanli, A. Isazadeh and T.N. Shishavan, **PHFS: A dynamic replication method, to decrease access latency in the multi-tier data grid**, J. Future Generation Computer Systems. 27, 3(2010), 233-244.

[27] L.M. Khanli, M. Khojand and M. Fatan, **Predictive fuzzy cascading method in data grid**, Proceeding of the First International Conference on Internet. (2009).

[28] M.Ch. Lee, F.Y. Leu and Y.P. Chen, **PFRF: An adaptive data replication algorithm based on star-topology data grids**, J. Future Generation Computer Systems. 28, 7(2011), 1045-1057.

[29] M.K. Madi and S. Hassan, **Dynamic replication algorithm in data grid: survey**, Proceeding of the International Conference on Network Applications, Protocols and Services. (2008).

[30] N. Mansouri and Gh.Dastghaibyfard, **A new dynamic replication algorithm for hierarchy networks in data grid**, Proceeding of the International Conference on P2P, Parallel, Grid, Cloud and Internet Computing. (2011), 187-192.

[31] N. Mansouri and Gh.H. Dastghaibyfard, **A dynamic replica management strategy in data grid**, J. Network and Computer Applications. 35, 4(2012), 1297-1303.

[32] V. Mansouri, M. Garmehi, M. Sargolzaei and M. Shadi, **Optimal number of replicas in data grid environment**, IEEE. (2008), 173-186.

34. D.S. Na, C.H. Youn, S. Jeong, E.B. Shim, E.Y. Lee and E.K. Park, **An efficient replication scheme for data grids**, IEEE. (2004), 392-396.

[33] S. Naseera and K.V.M. Murthy, **Agent based replica placement in a data grid environment**, Proceeding of the First International Conference on Computational Intelligence, Communication Systems and Networks. (2009), 426-430.

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

[35] S.M. Park, J.H. Kim, Y.B. Ko and W.S. Yoon, **Dynamic data grid replication strategy based on internet hierarchy**, J. Grid and Cooperative Computing. (2004), 838-846.

[36] R.M. Rahman, K. Barker and R .Alhajj, **Replica placement in data grid: considering utility and risk**, Proceedings of the International Conference on Information Technology: Coding and Computing. 1, (2005), 354-359.

[37] K. Ranganathan, A. Lamnitchi and I. Foster, **Improving data availability through dynamic model-driven replication in large peer-to-peer communities**, Proceeding of the Global and Peer-to-Peer Computing on Large Scale Distributed Systems Workshop. (2002).

[38] Q. Rasool, J. Li, G.S. Oreku, S.Zhang and D. Yang, **A load balancing replica placement strategy in data grid**, IEEE. (2008), 751-756.

[39] N. Saadat and A.M. Rahmani, **PDDRA: A new pre-fetching based dynamic data replication algorithm in data grids**, J. Future Generation Computer Systems. 28, 4(2011), 666-681.

[40] K. Sashi and A.S. Thanamani, **Dynamic replication in a data grid using a modified BHR region based algorithm**, J. Future Generation Computer Systems. 27, 2(2011), 202-210.

[41] K. Sashi and A.S. Thanamani, **A new replica creation and placement algorithm for data grid environment**, Proceeding of the International Conference on Data Storage and Data Engineering. (2010), 265-269.

[42] S.S. Sathya, S. Kuppuswami and R. Ragupathi, **Replication strategies for data grids**, IEEE. (2006), 123-128.

[43] M. Shorfuzzaman, P. Graham and R. Eskicioglu, **Distributed popularity based replica placement in data grid environments**, Proceeding of the 11th International Conference on Parallel and Distributed Computing, Applications and Technologies. (2010), 66-77.

[44] P.K. Suri and M. Singh, **An effective two-level job scheduling algorithm and two-phase dynamic replication strategy for data grid**, Proceeding of the International Conference on Advances in Computing, Control, and Telecommunication Technologies. (2009), 232-236.

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

[46] T. Tian, J. Luo, Z. Wu and A. Song, **A prefetching-based replication algorithm in data grid**, IEEE. (2008), 526-531.

[47] X. You, G. Chang, X. Chen, C. Tian and C. Zhu, **Utility-based replication strategies in data grids**, Proceedings of the Fifth International Conference on Grid and Cooperative Computing. (2006).

[48] W. Zhao, X. Xu, Z. Wang, Y. Zhang and S. He, **Improve the performance of data grids by value-based replication strategy**, Proceeding of the Sixth International Conference on Semantics, Knowledge and Grids.(2010),313-316.

[49] W. Zhao, X. Xu, Z. Wang, Y. Zhang and S. He, **A dynamic optimal replication strategy in data grid environment**, IEEE. (2010).

[50] H. Zhong, Z. Zhang and X. Zhang, **A dynamic replica management strategy based on data grid**, Proceeding of the Ninth International Conference on Grid and Cloud Computing. (2010), 18-23.

[51] The MONARC project. http://monarc.web.cern.ch/MONARC/.

[52] LHC Accelerator Project. http://lhc.web.cern.ch/lhc/.

[53] http://public.web.cern.ch/Public/Content/Chapters/AboutCERN/AboutCERN-en.html.
[54] The European Data Grid project. http://eu-datagrid.web.cern.ch/eu-datagrid/.

Zahra Miroei
Department of Computer, Science and Research Branch, Islamic Azad University, Kerman, Iran. e-mail: miroeizahra@yahoo.com

Marjan Kuchaki Rafsanjani
Department of Computer Science, Faculty of Mathematics and Computer, Shahid Bahonar University of Kerman, Kerman, Iran. e-mail: kuchaki@uk.ac.ir