*J. Basic. Appl. Sci. Res.*, 3(7)546-553, 2013

ISSN 2090-4304
**Journal of Basic and Applied Scientific Research**
*www.textroad.com*

# New Replication Method in Data Grids Based on Weighted Files

**M. Reza Salehnamadi, B. Ghaheri**

Islamic Azad University- South Tehran Branch, Tehran, IRAN

## ABSTRACT

A new replication mechanism is presented in this paper. The aim is to decrease the job execution time and increase the data availability in the system. This mechanism used weighted criteria to highlight a file as a more popular file. The data files are arranged on the basis of the weight criterion and if the profit of replicating a file is more than the loss of deleting the candidate files, the replication work will be done. The method of presented algorithm is the greedy strategy to do the replication work.
Two new functions, newbio and newzipf, were introduced as a prediction function, and are used to predict the future access to files. Also it is proved that operation of new models with different prediction functions in term of different schedulers and various data access patterns, is better than other models. The results of these comparisons show that the presented models, have better operation rather in comparison with the older and more popular models.
**KEYWORDS:** Data Grids, file Replication, weighted file, scheduling.

## I. INTRODUCTION

Grid computing is the new generation of distributed networks and like internet allows its users to share the files. Moreover, provides common resources of data for users. There are some problems of data delivery protocols and data possession in cloud computing [10-12]. In all grid applications, excessive movement, transposition and transition of volume and data integrating are unavoidable cases .Some techniques are designed for optimal use of grid resources. One of these techniques is replicating that leads to improve the data application in grid. The main idea of replication is to make copies from data and to distribute them in various locations and different sites in grids as data can be retrieved easily and while a copy in one location is lost that data don't become inaccessible Data replication is considered as an important technique for decreasing the data access latency (for reading the data) and reducing the consumption of bandwidth and consequently decreasing the job execution time. The mechanism of production and managing the copies of one file is called Replica Management(RM). The RM services include: making new copies, recording the location of the copies in copy's catalogue and searching the catalogue to find the requested copy. Different mechanisms are presented for data replication but all of them should answer to three questions. The first one is that which data must be copied? Second is that, when must the new copy be produced? The last one is that, where must this new copy be located finally? Usually when the requests to one file exceed from a threshold value, replicating will be performed and the location of that copy will select accidentally or among the selected sites.

There are two different methods for replicating. One method is static which generating and managing copies will do manually and offline in it. The second type is dynamic methods that can change the locations of copies and start to make new copies automatically for adapting to these changes. As sources and files of data in grid are constantly changing, it is clear that dynamic methods are better. Ranganathan and Foster in [2], simulated six different replication strategies for three different user access patterns. These six strategies were composed of: No Replication or Caching, Best Client, Cascading Replication, Caching Plus Cascading Replication, and Fast Spread. These strategies have been assessed by three access patterns to different data that are stated below.

1) Random access pattern: that there is no locality in this pattern.
2) Data includes small temporal locality. In such a manner that some of that accessed files be under request again.
3) The data include small geographical and temporal locality. Geographical locality means the files that have been under the access of customers recently, and also its neighbors have access to it.

The results of simulation showed that the adoption of replication strategy with suitable access pattern will economize the use of band width and reduce data access latency. Also if access pattern includes small geographical locality, we will have noticeable bandwidth use and latency reduction. In [3], considering the limited storage space of system, two new criterions have been presented for evaluating the system reliability. These two new criteria are system file missing rate (SFMR) and system byte missing rate (SBMR). It is obvious that when the sizes of all files are equal, these two criteria will be the same. Also an online optimizing algorithm that minimizes data missing rate is presented. They simulated this algorithm with the aim of increasing data availability in limited saving space using four different access prediction functions. In [4] a dynamic replication mechanism named LALW, has been presented. LALW selects a popular file for replication with giving weight to the files. Then it calculates

**Corresponding author:** M. Reza Salehnamadi, Islamic Azad University- South Tehran Branch , Tehran, IRAN,
e-mail: m_saleh@azad.ac.ir.

the number of required replications and determines the sites that copies must be placed in them. The importance of records will be different with attributing different weights to historical data access records. In this manner, that newer data access record, has more weight. It means that it has more importance and is more suitable for determining recent conditions of data access. The simulation results show that LALW increases the effective use of net successfully. In [5] a replication strategy has been presented according to economic model (Eco). Its authors have used an auction protocol for decision making about data replication to optimize, replicating operation in long term. They pointed that their offered strategy works better than other replication strategies with sequential file access pattern.

In this paper a new replication mechanism is presented. The aim is to decrease the job execution time and increase the data availability in the system. This mechanism used weighted criteria to highlight a file as a more popular fileThe method of presented algorithm is the greedy strategy to do the replication work.

Two new functions, newbio and newzipf, were introduced as a prediction function, and are used to predict the future access to files. In contrast to other techniques, this technique uses different prediction functions in term of different schedulers and various data access patterns, so is better than other models.

The reminder of the paper is organized as follow: in section II, proposed dynamic replication strategy will be described, in section III, simulation detailed will be listed, finally in section IV, performance will be evaluated and compared with other methods.

## II. DYNAMIC REPLICATION STRATEGY

Data grids are generated for managing and sharing of excessive volumes of distributed data. Data replication is an accepted technique in data grid that is applied in decreasing the access latency and reduction of network bandwidth. Also replication result in increasing the data availability and consequently increase in reliability and fault tolerance of system. In this paper, a dynamic data replication mechanism is presented to maximize the data availability assuming limited replication storage.

### *A. System model*

Users submit their jobs to grid for execution. Each job needs to access several files to be performed. If one job requests a file and that file doesn't exist in local site, may be that job become idle or its execution be delayed. Some of data files, are more demandable that those are called, popular files, whereas some other files will be rarely used. If the popular files can be find and replicated into the required sites, then a great step are taken to decrease the data access and therefore decrease the job execution time[4].

The mentioned method idea, is that the file that was used more in the past time, will use more than others in the future. This case is named Temporal Locality[4]. In this manner, the popular file will be recognized with analysis of performed access by the users. In presented model, a giving weight criterion is defined to these files, that in its bases the file that has more weight is counted more popular file.

Consider a set of j = ($j_1$,$j_2$, $j_3$,… ,$j_n$) jobs. The jobs usually need to more than one file to be done. Consider the collection of files as *f*=($f_1$,$f_2$,$f_3$,… , $f_n$) and the size of each $f_i$ file is equal to $S_i$. The storage element is the only place that a file can be placed in it. To increase the system reliability and performance, each file can has some copy in grids that in this case each one of those copies must be saved in different storage elements. Because saving some copies of one file in one storage element, not only don't help to increase file's availability but also will damaged it by vein use of storage space[3].

Each storage element has the data availability that is indicator of possibility of existing one file in it. Also it is supposed that all the saved files in storage element have the same availability. The file availability in the storage element *j* is shown by $p_{SEj}$. Since it is possible that there were more than one copy of file, so the availability of each $f_i$ file that is shown by $P_i$ will be calculated in this manner:

$$P_i = 1 - \prod_{j=1}^{k} (1 - p_{SEj})$$

K, shows the number of $f_i$ files copies.

It is obvious that for each operation of accessing to a file, the possibility of unavailability is gained form (1- $P_i$) junction, of course with this supposition that the access operation of files will done separate from each other[3].

### *B. Predict functions for dynamic replication*

Practically there aren't anything about the next requests of users but it can be partly predict it. For each access to $f_i$ file in T time, a value called $V_i$ is regarded that is indicator of the times that this file will be accessed in the future (>=T). it can be predicted by the following two predict functions.

- Bio prediction: By the Binomial distribution according to access record of file in the past time, the $V_i$ value that is related to the file will be obtained.
- Zipf prediction: By the zipf distribution according to the access record of file in the past time, the $V_i$ value that is related to the file will be calculated.

The mentioned functions, demonstrate the domain of possible values for $V_i$ in the future.

*C. Definition of replication strategy*

The method of presented algorithm is the greedy strategy to do the replication work. *Greedy* algorithms build up a solution piece by piece, always choosing the next piece that offers the most obvious and immediate benefit. Although such an approach can be disastrous for some computational tasks, there are many for which it is optimal [13].

Our Greedy Algorithm is demonstrated in fig. 1 This algorithm will perform automatically by the replication manager of applicant site whenever the request of one file is occurred [1].

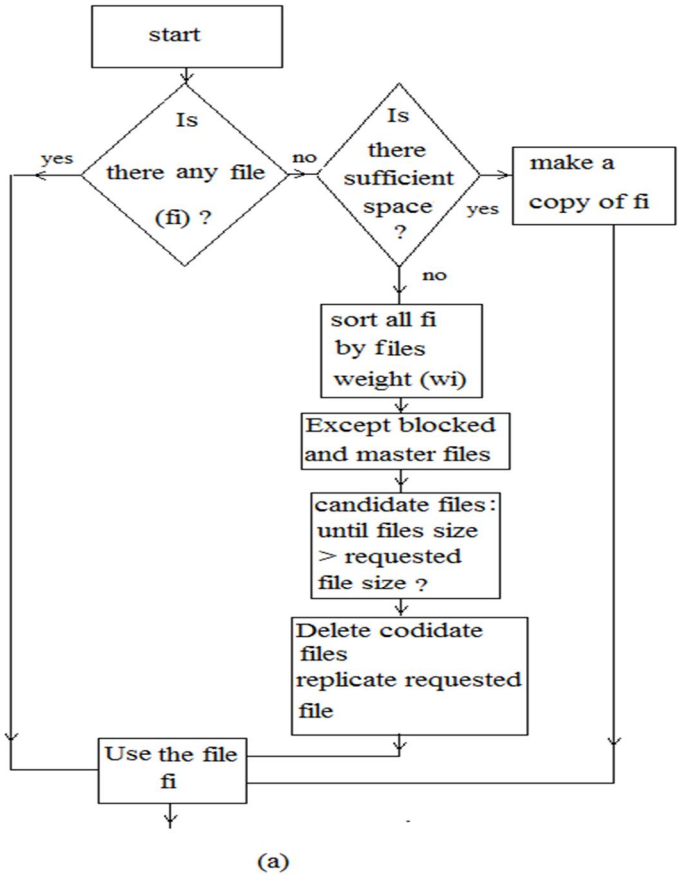
Optimization algorithm goes through four steps to decide about the file replication.

In the first step, if the requested file exists in the storage element, there isn't any need to replicate and copy it. Since as mentioned before, numerous replications of the file in storage element don't improve the availability. In contrast it can cause to waste the storage space.

In the second step, if there is enough free space to keep the requested file, constantly the file replication will be done. In the third step, if the enough free space doesn't exist, from the existed files some candidates will be selected to delete and replace that this selection is done on the base of the files weight. This matter is valuable in the long time because the less valuable file will be replaced by the more valuable file. In selecting the candidate files to replace, it must be regarded that the original copies can not be deleted in the system. Also the files that are accessible and used by the job can't be deleted.

The fourth and the most complicated step, the replacement will be done just in situations that the profit of saving new copy be more than the expense of deleting the existed files.

The profit of replicating $f_i$ file is obtained by $\Delta P_i = (P_i^{'} - P_i) \times V_i$ . That $P_i$ is the present availability of $f_i$ file and $P_i^{'}$ is the availability of $f_i$ file after replication.




Replication strategy():
1- if requested file (fi) exists in the local site, do noting.
2- if requested file (fi) does not exists in this site,and the site ha s sufficient free space, store a copy of file (fi) in this site.
3- if requested file (fi) does not exists in this site and the site dosen't have enough space:
- Sort all files in this SE by increasing order of file weight ($W_i$). calculate the file weight $W_i = P_i \times V_i + AF_{ave}(fi)$.
- Fetch the files from sorted list in order r & add it into cadidates list and continue until the sum of candidates file size become greater or equal than the requested file size, The file must not be locked or master file.

4- Replicate the requeste d file e if only the value gained by replicating the file fi is greater than the accumulative value loss by deleting the candidates file (fi) from the SE. where

$\Delta P_i \times V_i > \sum_{j \in \text{candidates}} \Delta P_i \times V_i$

(b)

Fig. 1 Replication strategy

Also the expense of deleting the candidate files will be obtained by the relation of $\sum_{j \in Candidates}(P_j - P_j^{'}) \times V_j$ that $P_j$ is the present availability of candidate file and $P_j^{'}$ is the availability after deleting the candidate file.

*D. Description of weight criterion in the demonstrated algorithm*

To give value into the files a synthetic relation is used as the weight of file that is the following.

$$W_i = P_i \times V_i + AF_{avg}(f_i)$$

The first part of this weight criterion, actually is prediction of the future access to the file. $P_i$ is the availability of file $f_i$. $V_i$ is the predicted value for the next requests of file $f_i$ according to prediction functions.

The second part of the weight relation uses the meaning of half-life [4]. It means that the value of access record of each file, after passing a constant time interval, will decrease half of the previous amount. Thus the newer records have more value to calculate the weight. If $N_t$ be the number of time interval passed, $F$ is the collection of files that are requested and $a_f^t$ is the number of performed access to file $f$ in the time interval $t$. AF for file $f$ is calculated as follows[4]:

$$\text{AF}(f) = \sum_{t=1}^{N_t} (a_f^t \times 2^{-(N_t - t)}) \quad ,\forall f \in F$$

The average of access in each time interval for each file will be calculated as follows:

$$AF_{avg}(f) = \frac{AF(f)}{N_t}$$

## III. *SIMULATION'S* DETAILS

OptorSim v2.1 simulator is used to evaluate the algorithm efficiency. The simulator is implemented as the part of EU DataGrid project to test the algorithms of dynamic replication. It is constructed in this manner that its components are so near the actual data grid [7].

The Optorsim architecture has some sites that execute the jobs. Each site consists of computing elements (CE), storage elements (SE), and replica manager (RM) that replica manager, has the replica optimizer (RO) in itself [8]. The computing elements are responsible to execute the jobs. The data files are placed on the storage element. The users submit the jobs to grid to be done. Then a resource broker will schedule the jobs to be done. The pattern of delivering the jobs to the grid is selectable. When executing a job is started in one CE, its data files will be processed according to the special access pattern. If the file of requested data doesn't exist in the site, it must be read from the remote site, or one copy of it must be replicated in the site. The duty of managing the files of each site is related to replica manager. The replica manager consists of one part named replica optimizer, that replication algorithm placed in it and this part performs the work of replication or deletion of the files automatically [9].

The simulator's topology that is shown in fig. 2 is used[6]. In this topology there are both CE and SE in all the sites except CERN. The bandwidth between two sites are shown in the fig. 2 It is supposed that the original copy of all the files at the beginning of the simulation placed in CERN site that has the large storage space but doesn't have the computing element.

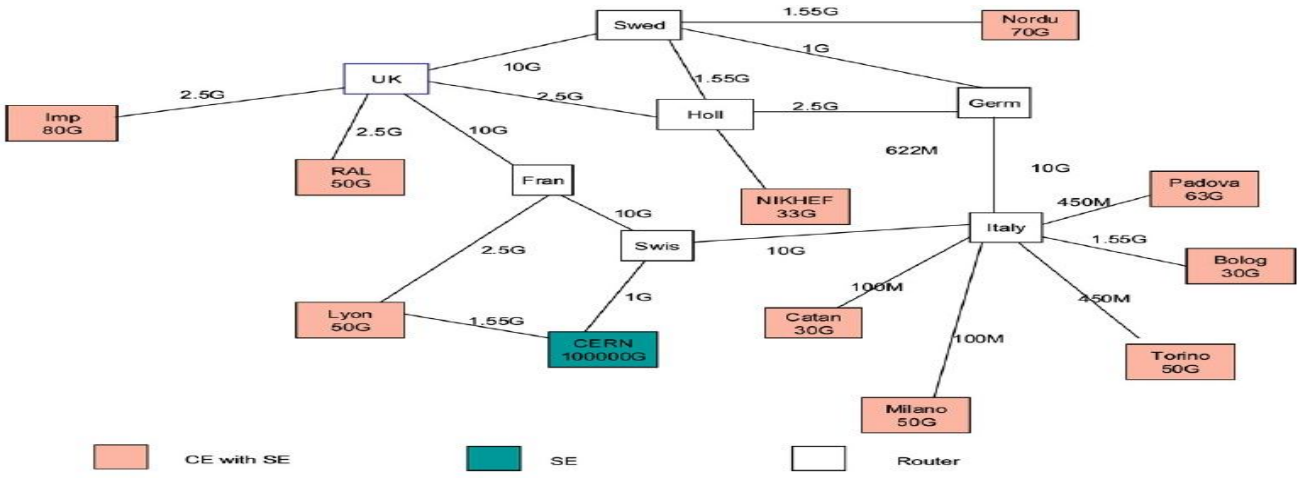



Fig. 2 Grid Topology in the simulation

The amounts of system parameters are shown in table 1. The default amounts of parameters are used. Suppose that there are 200 data files and 10,000 jobs enter into the grid. The size of all the files are 1G and each job needs 3 to 20 files. As it is presented in the table, each storage media has 30 to 100000 GB space. It is supposed that the data availability in each site is 99%.

Table 1. simulation parameters

| parameter | value |
|---|---|
| **Number of files** | 200 |
| **Number of jobs** | 10,000 |
| **Storage available in each SE** | 30G-100,000G |
| **File size** | 1G |
| **Number of files requested by a job** | 3-20 |

## IV. PERFORMED EVALUATIONS

The algorithm is performed with two different predict functions to calculate the file weight and named them NewBio and NewZipf associated with the Binomial and Zipf prediction function respectively. Proposed model is compared with the traditional LFU (least frequently used), LRU (least recently used) models and the economical EcoBio, EcoZipf models that all of them implemented in the simulator [7].

LRU optimizer: this model always replicates the file and to create the free space deletes the oldest file in the storage element.

LFU optimizer: this model always replicates the file, create the free space and selects the files that had got the minor access number in the past time, to be replaced.

EcoBin optimizer: this model decides about replication of the file on the bases of economical model of Binomial. Prediction of files value is done on the bases of Binomial distribution.

EcoZipf optimizer: this model decides about replication of the file on the bases of economical model of Zipf. Prediction of files value is done on the bases of Zipf distribution.

The above economical models, use greedy strategy such as our algorithm also use the amounts of profit and loss to determine the replacing files.

### *A. Evaluation parameters*

The operation of above mentioned models are evaluated from "the total job time" point of view. Also other parameters such as "system file missing rate" and "system byte missing rate" are used. The definition of these parameters according to data availability is as follows[3]:

System File Missing Rate:

$$SFMR = \frac{\sum_{j=1}^{n}\sum_{i=1}^{m_j}(1-P_i)}{\sum_{j=1}^{n} m_j}$$

Where n indicates the total number of jobs. $m_j$ indicates the number of file access operation of each job. $P_i$ shows the probability of availability of file $f_i$ as defined in equation (1).

In the same way, the system byte missing rate can be defined as follows:

System Byte Missing Rate:

$$SBMR = \frac{\sum_{j=1}^{n}\sum_{i=1}^{m_j}(1-P_i) * S_i}{\sum_{j=1}^{n}\sum_{i=1}^{m_j} S_i}$$

Where $n$, $m_j$ and $P_i$ have the same meaning as before mentioned. $S_i$ indicates the size of file $f_i$. It is clear that the values for SFMR and SBMR become similar if the sizes of all files are the same. It also substantiated that smaller value for SFMR and SBMR show better system data availability [3].

### *B. Evaluation of the Total Job Time*

The total job time are compared for all six replication schemes. Fig. 3 present the results of this comparison. The NewBio, NewZipf schemes are the replicating schemes that were presented.

The job times were measured on the basis of seconds and by supposing the sequential access pattern and random scheduler. The total job time consists of the time of data transferring and job execution. It is obvious that the recently presented strategies have the minor total job time. NewBio has the least total job time in contrast with other strategies. EcoBio has the most total job time and after that EcoZipf, LFU and LRU are in the next ranks.


Fig. 3 total job execution time for sequential access pattern and random scheduling

*C. System data missing rate for different access patterns*

The order of files request is determined by the access pattern. All of the four access patterns existed in simulator that named Sequential access pattern are used by model, Random access pattern, Random walk Gaussian, Random walk Zipf to evaluate the models.

Fig. 4 shows the amount of SFMR for each six increasing schemes and four access patterns. In this evaluation the random scheduler used to schedule the jobs.

It is obvious that both of our suggested strategies for all of the access patterns perform better than the economical EcoBio, EcoZipf models. The just exception case is the NewZipf strategy for sequential access pattern. The economical EcoBio model has the highest rate of SFMR for all the access patterns. The other economical EcoZipf model has the second rank for all the access patterns. The NewBio has the least amount for all the access patterns and it has insignificant amount. The LFU, LRU strategies has the better operation rather than economical models. LFU do a little better than LRU. But both of these strategies have more SFMR amount rather than the new models. The diagram shows that the new presented models have better operation rather than the economical and traditional LFU, LRU models.

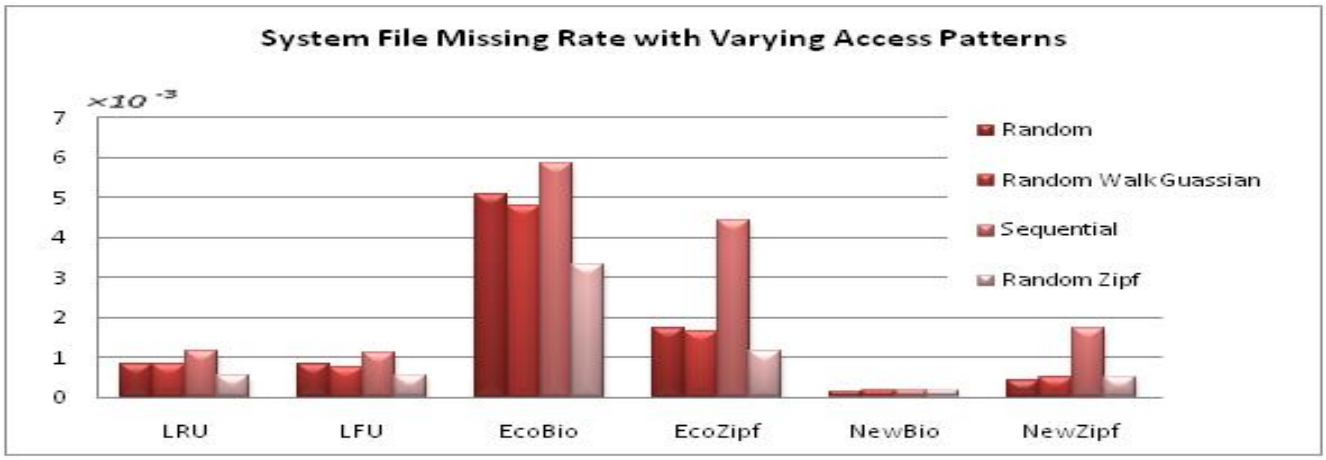



Fig. 4 SFMR with varying access patterns

But it is clear that NewZipf do worse than LFU, LRU for sequential pattern. It is may be for this reason that our algorithm just do the replication work when the profit amount of replicating file, be more than its replacement expenses. But EcoZipf prediction function is not suitable for access pattern, because it caused incompatibility in calculating the file weight. In other word it causes mistakes in distinction of popular and valuable files for replication. In this reason the system file missing rate has more amounts for all the replication schemes with sequential access amount.

*D. System data missing rate for different schedulers*

In the next step, the effect of scheduler type on the system file missing rate is considered. The recourse broker in simulator can schedule the jobs for sites on the basis of different algorithms. In Optor Sim, four different schedulers were implemented that consist:

- Random scheduler: schedule the job for the random site.
- The shortest queue scheduler: the job is accepted for the candidate site that has the shortest size of queue.
- Access cost scheduler: the job execution consigned to the Computing element that has the least expense of access to the file.
- Queue Access Cost Scheduler: the job is scheduled for the site that has the least access cost for current job and all of the waiting jobs in the queue.

Fig. 5 shows the effect of scheduler type on the system file missing rate. In this comparison the sequential access pattern is considered.

For all the models, the shortest queue and access cost schedules has the nearly same SFMR amounts. The operation of two new models for different schedulers has insignificant differences. The just excepted case is the NewZipf model to random schedulers that has high SFMR amount. It is for this reason that Zipf predict function to estimate the file value in the future used a pattern that is not coordinate with random scheduler.

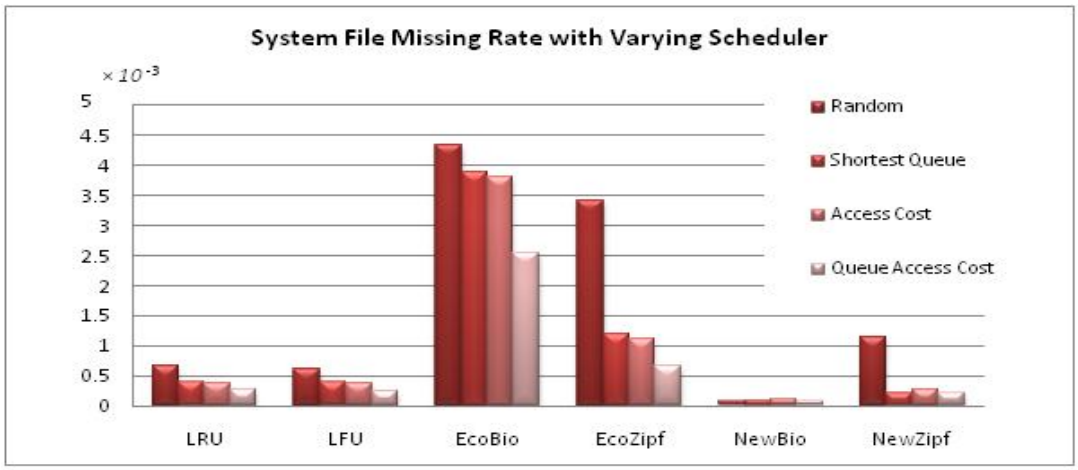



Fig. 5 SFMR with varying scheduler

Overall comparison of proposed algorithm with LALW and MinDMR algorithms adding up in table 2.

Table 2- comparison of proposed algorithm with two others

| parameter | LALW | MinDMR | Proposed Algorithm |
|---|---|---|---|
| **Algorithm start time** | Automatically after every 40 job in grid | When requested file is not in site | When requested file is not in site |
| **Prediction of popular files** | Yes | no | no |
| **Prediction of request no. of files in future** | no | Yes, with binomial function and Zipf and current queue | Yes, with binomial function and Zipf |
| **Weight formula of popular file determination** | $AF(f) = \sum_{t=1}^{N_t} (a_f^t \times 2^{-(N_t - t)})$ | $W_i = P_i \times Vi / (Si * Ci)$ | $W_i = P_i \times V_i + AF_{avg}(f_i)$ |
| **Prediction of reject-able files** | no | no | no |
| **File rejection decision** | On requested time and conditional | Reject before the request | Rejection in some requests |
| **Weight of reject-able File determination** | LRU | Files with less weight | Files with less weight |
| **Replica condition** | Replica with no condition | When the replica benefit is more than it's cost | When the replica benefit is more than it's cost |
| **Decision on replication number** | Popular file's weight divided by requested files weights average | Only one copy | Only one copy |
| **Place of replication** | In the best client | in requesting client | In requesting client |

## CONCLUSION

A new replica mechanism is presented in this paper. This mechanism uses a weighted criteria to highlight a file as a more popular file. Availability of files in distributed system obtain with a limited copy space condition. It minimize the data miss rate and maximize the availability of files, meanwhile with limited storage space and low "time to data access". In fact, this algorithm is a greedy algorithm. Performance of system level availability is more than file level availability. So the main idea is: some files that were accessed more frequently in past, will be accessed in future more than others. The mechanism assigns an availability factor for each storage element in grid. This factor is symptom of probability of file existence. It is assumed that all files in a storage element have the same availability and in addition, each file has only one copy in storage element.

Based on two prediction function, binomial and zipf, two new functions, newbio and newzipf are used to predict the future access to files. Optorsim was used to simulation the performance of proposed algorithms compared with LRU, LFU, Ecozipf and Ecobio models. Simulation shows that for all access pattern, proposed algorithms have better performance than others.

To compare the effect of scheduling on miss rate, four scheduling models were used with sequential access pattern. Simulation shows that two new algorithms have same effect, only exception is newzipf on random scheduling that has high miss rate. This is because the newzipf prediction is not matches with random scheduler.

Overall results is that scheduling type and file access pattern have not effectively effect on system data miss rate, and effective parameter is the replication mechanism.

## REFERENCES


[1] M. Reza Salehnamadi, B. Ghaheri, "A dynamic Data Replication Strategy in Data Grids", Proceeding of International Conference on Signal Processing, Communication and Networking, Paris, pp, 109- 114, 2010

[2] K. Ranganathan, I. Foster, "Identifying dynamic replication strategies for a high-performance data grids", Proceeding of 3rd IEEE/ACM international workshop on grid computing, USA, Lecture notes on computer science, vol 2242, Springer, pp 75-86, 2002

[3] M. Lei, SV. Vrbsky, X. Hong, " An on-line replication strategy to increase availability in data grids ", Future Generation Computer Systems, 28, 85-98, 2008

[4] Ruay-Shiung Chang, Hui-Ping Chang, " A dynamic data replication strategy using access-weights in data grids", Journal Supercomputer, 45, 277-295, 2008

[5] W. Bell, D. Cameron, R. C. Schiaffino, A. Millar, K. Stockinger, "Evaluation of an economy-based file replication strategy for a Data Grid", International Workshop on Agent Based Cluster and Grid Computing at CCGrid, Tokyo, Japan, IEEE Computer Society Press, 2003

[6] D.G . Cameron, R.C. Schiaffino, J. Ferguson, P. Millar, C .Nicholson, K. Stockinger, F. Zini, "OptorSim v2.0 installation and user guide", http://edg-wp2.web.cern.ch/edg-wp2/optimization/optorsim.html, 2004

[7] OptorSim- A replica optimizer simulation. http://edg-wp2.web.cern .ch/ edgwp2/optimization/optorsim.html.

[8] D.G. Cameron, A.P. Millar, C. Nicholson, R. C. Schiaffino, F. Zini, K. Stockinger, "Analysis of scheduling and replica optimisation strategies for data grids using OptorSim", Journal of Grid Computing, (2) 57-69, 2004

[9] w. Bell, D.G. Cameron, L. Capozza, A.P. Millar , K. Stockinger, F. Zini, "Optorsim: A grid simulator tool for studying dynamic data replication strategies", International Journal of High Performance Computing Applications 17 (2003) 4.

[10] M. Pajouhesha, Amir M. Bidgoli b, M. H. Yektaie, " A New Reliable Data Delivery Protocol in Wireless Sensor Network" *J. Basic. Appl. Sci. Res.*, 2(1), TextRoad Publication, 2012, pp. 132-137.

[11] M. Javeria Anwar, M. Younus Javed, S. Rehman, N. Asad, "Applying Provable Data Possession with Elgamal in Cloud Computing "*J. Basic. Appl. Sci. Res.*, 2(7), TextRoad Publication, 2012, pp. 7091-7094

[12] F. Mohammad Hosseinzadeh1, S. Javanmard, "A Numerical Analysis of Homogeneous Cloud Seeding Agent Based on Sensitivity Tests in Different Conditions", *J. Basic. Appl. Sci. Res.*, 2(7), TextRoad Publication, 2012, pp. 7328-7342

[13] www.cs.berkeley.edu/~vazirani/algorithms/chap5.pdf