J. Basic. Appl. Sci. Res., 3(4)928-939, 2013
© 2013, TextRoad Publication

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

A Prediction-Based Replica Replacement Strategy in Data Grid

Maryam Mehraban, Ahmad Khademzadeh, M.Reza Salehnamadi

Department of Computer Engineering, South Tehran Branch, Islamic AZAD University, Tehran, Iran

ABSTRACT

Data Grid provides services for sharing and managing huge amount of data files around the world. For the high latencies of the Internet, it becomes a big challenge to access such large and widely distributed data fast and efficiently on data grids. Data replication is one of the methods. However; it is bounded by two factors: size and number of available storage and bandwidth of sites within the Data Grid. Therefore, Replica replacement is the crucial step to replication strategy. This paper proposed a replica replacement strategy called Prediction Replica replacement (PRA) was applied into replica replacement during data replication. The PRA framework can automatically decide on which replica file to be deleted whenever the storage element of the grid site is full. The performance evaluation of PRA and other replication algorithms are carried out by optorSim simulation. The result shows that Prediction Replica replacement Strategy performs better than other replication strategies.
KEYWORDS: Data Grid, Replica, Replacement, Replication, OptorSim.

1. INTRODUCTION

In recent years, applications such as bioinformatics, climate transition, and high energy physics produce large datasets from simulations or experiments. Managing this huge amount of data in a centralized way is ineffective due to extensive access latency and load on the central server. In order to solve these kinds of problems, Grid technologies have been proposed. Data Grids aggregate a collection of distributed resources placed in different parts of the world to enable users to share data and resources [1, 2, 3]. Data replication is an important technique to manage large data in a distributed manner. The general idea of replication is to place replicas of data at various locations. Data replication has been used in database systems [4], parallel and distributed systems [5, 6, 7], mobile systems [8], and Data Grid systems [9, 10, 11].

Data replication has two direct improvements on the performance of Data Grid. One is to speed up the data access, which leads to a shorter execution time of the grid jobs; and the other one is to save the bandwidth between nodes, which can avoid the network congestion while the sudden frequently requirement of some data. But Replication is also bounded by two factors: the number and size of storage available at different sites within the Data Grid and the bandwidth between these sites [12]. Sites have limited storage space and cannot accommodate replicas of every data file on the grid, while network have limited capacity for transferring them. A grid must therefore have a replica management system that manages the data files in grid environment with the aim to optimize the performance of the grid. One of the most important strategies is replica replacement strategy, which is the main focus of this paper. The main rule of replica replacement is to make a room for the newly created replica by finding the victim replica to be replaced by the newly created replica. Replica replacement strategy plays a vital rule in enhancing the performance of grid.

Due to the capacity of the storage device of each site is limited, replica replacement, deleting replicas to make room for new replicas, is indispensable whenever the capacity of the SE (Storage Element) is insufficient during replication. This caused another problem: which replicas will be replaced from the SE. Common disadvantage of all replacement replica strategies is disability in General prediction to access information in the near future A replacement strategies wisely is the method that chosen to eliminate replica with the least likely to be available in the future and the hot replica with cold replica to prevent Therefore, how to select replicas that will be less likely accessed in the future, is one of the key problems in the process of replication, and also is the core problem to be solved in this paper. Therefore, if there is not enough storage during replication, a well-designed replication replacement algorithm will be needed. Our proposed Replica Replacement technique, PRA finds the best candidate replicas for replacement. At present, most replica replacement strategies have been adapted from page replacement algorithms in Operating

System (OS), such as LRU (Least Recently Used), LFU (Least Frequently Used). LRU and LFU could cause replicas that are beneficial for future jobs to be wastefully removed. Both of this method, basically lists all files according to their access frequency from the time the replica was created and also keep tracks on how often these replicas are being accessed throughout their existence. These strategies do not provide enough weight age to judge a replica's future access. Therefore, deleting replica according to the LRU and LFU method could cause the valuable replica in the future to be removed as well. In addition, when the removed replica is requested in the near future, it will then have to replicate again into the storage element. This might take longer job execution time since

***Corresponding Author**: Maryam Mehraban, Department of Computer Engineering**,** South Tehran Branch, Islamic AZAD University, Tehran, Iran. E-mail address: Mar.mehraban@gmail.com, maryam.mehraban65@yahoo.com

the replication process of transferring the replica back into the storage element will have to take place again. Using an efficient algorithm has also played an important role in reduce Mean Job Time, Total Number of Replications, Effective Network Usage and Percentage of Storage Filled/Available. This paper proposed a replica replacement strategy called Prediction Replica replacement (PRA) was applied into replica replacement during data replication. The PRA framework can automatically decide on which replica file to be deleted whenever the storage element of the grid site is full based on information such as data-access frequency, Priority replica, age, free space on storage elements to which data will be replicated or deleted and Future transfer cost. The difference between PRA and the old methods, PRA allocated greater weight to the newly created replicas, Prevent removal of replicas that have been stored in the storage element recently. Also, reduces the computational overhead. The reason of that is because the PRA invoke the deletion function with minimum number if there is a need to perform the replacement process. In other words, the process of replacement in PRA occurs with minimum number as it deletes minimum number of files to make a free space for the newly created replica. Such an approach considers both the users satisfaction by deleting the less valuable file and resource satisfaction by deleting only one file. Also, Deleting files According to file size and Storage space required is done. Since most of the datasets in the scientific data grid scenario are read-only, the overhead of updates will not be considered in the replication strategies.

The main contribution of this paper is providing efficient mechanism for Replica replacement in Grid data. More specifically, the following contributions are achieved in this paper:

1) Present a new replica replacement strategy, which is Prediction Replica Replacement Strategy, abbreviated to PRA. The PRA improves the temporal locality property and apply of decisions based on information such as data-access frequency, Priority replica, age, free space on storage elements to determine the victim file.

2) The PRA strategy is evaluated and our simulation results show that the jobs with PRA strategy took the least relative delay-time to complete, also shows better performance and efficiency of the data access on Data Grids; compared with other two LRU and LFU strategies.

The rest of this paper is organized as follows: Section 1.1 provides a brief description on existing work in dynamic replication strategies and how they determine the victim file. We include the details of our proposed strategy in Section 2. We show the simulation setup, parameters configurations and performance metrics used to compare with related replication strategies; and the performance results are then presented in section 3. Finally, we summarize some conclusions in Section 4.

1.1 Related Works

In [13, 14, 15], they proposed a prediction-based replica replacement algorithm using a two-stage process to evaluate the popularity of a replica. They considered some bandwidth with replica size. It is combined prediction and cost function together to predict. The simulation results demonstrated that their algorithm contributed to better grid performance. The work in [16] suggested a replica replacement algorithm based on economic model and opportunity cost, the files have been evaluated using zipf-like distribution prediction model and then weighted using the file transfer cost model. If the needed replica has a higher weight than the replica with the lowest weight in the local storage, that file will be deleted and the new replica will be transferred into the local site.

L.H. Ai and S.W. Luo [17] present a job-attention replica replacement strategy, abbreviated to JARRS. JARRS makes its replacement strategy based on grid locality analysis. The locality analysis is based on the traditional replica replacement strategies.

Jianhua.J, Huifang Ji2, Gaochao Xu[18] present an Associated Replica Replacement Algorithm Based on Apriori Approach for Data Intensive Jobs in Data Grid, abbreviated to ARRA. ARRA is introduced in two parts. In the first part, access behaviors of data intensive jobs are analyzed based on Apriori algorithm, which is a famous data mining algorithm. In the second part, replica replacement rules are generated and applied.

Madi.M, Yusof Y and Hassan S [18] present an exponential based replica replacement strategy, abbreviated to ERRS. File Evaluation stage - in this stage assigned a prediction value based on exponential growth/decay model to each file according to historical information; File Elimination - in this stage the victim file is chosen Based required storage capacity and replica value.

2. The proposed Model

The proposed replica replacement strategy termed as PRA is applied when the selected site for placing the newly created replica has insufficient storage capacity to store the underlying replica. Prior to that, it is assumed that information about access history on which file to be replicated and where the replica is to be stored is available [19]. PRA is a strategy that selects a victim file from the files that are stored in target storage in order to make sufficient storage space for the underlying replica.

The PRA performs replica replacement through two main stages:

✓ File Evaluation stage - in this stage we assign a prediction value to each file according to historical information and transfer cost. also, This stage is composed of three parts:

- Evaluation of replica popularity -.we apply Half-life model to determine the replica popularity.

- Evaluation of future transfer cost - The cost factor often affected by some factors such as replica size and bandwidth.
- Prioritization of replica - we determine replica value base on replica popularity and future transfer cost.

✓ File Elimination Stage - in this stage we eliminate the files from being a victim Based on the size of the newly created replica and needed storage capacity.

Details of the stages are included in the following subsections.

2.1 File Evaluation Stage

In this stage, firstly Evaluates replica popularity and future transfer cost locally and then determines replica value for selection the victim file.

2.1.1 Evaluation of replica popularity

We believe that popular replicas are more likely to be available in the future. However, recent access frequency is more important, but older frequency access records are also important. Each site maintains the history of files accessed in that site. At a regular time interval each site calculates replica popularity Based on the age and data-access frequency of previous periods. We apply concept of **Half-life** to determine the replica popularity and Integration the age and data-access frequency. **Half-life** is mentioned in many domains, such as physics, chemistry, and medicine. Half-life indicates the time required for the quantity to decay to half of the initial value, where the quantity may be radioactive element or chemical element.

Information gathered at different time intervals has different weights in order to distinguish the importance between history records. The rule of setting weight uses the concept of **Half-life** .In our algorithm, the weight represents the quantity, and a time interval represents the time for half-life. That is, the weight of the records in an interval decays to half of its previous weight. Setting different weight is used to evaluate the importance for history records. Older history records have smaller weights. It means that the recent history access are worthier for referencing than previous [20]. The selection of the popular file is based on the weights given to the file. Figure 1 indicates the concept. At the first time interval, there is only an access history in the first time interval, which is T1. The weight of records in this time interval is 2^0. At the second time interval, there are two access history in two time interval T1 and T2. Because T1 has existed for a time interval, the weight of T1 becomes 2^{-1} and the weight of T2 is 2^0. We can derive the weight of records in different table at the *n*th time interval from the above rule [20].

The weight of T1, T2, …, and T*n* is $2^{-(n-1)}$, $2^{-(n-2)}$, …$2^{-(n-n)}$. This weight is used to find a more popular file, as explained in the following [20].

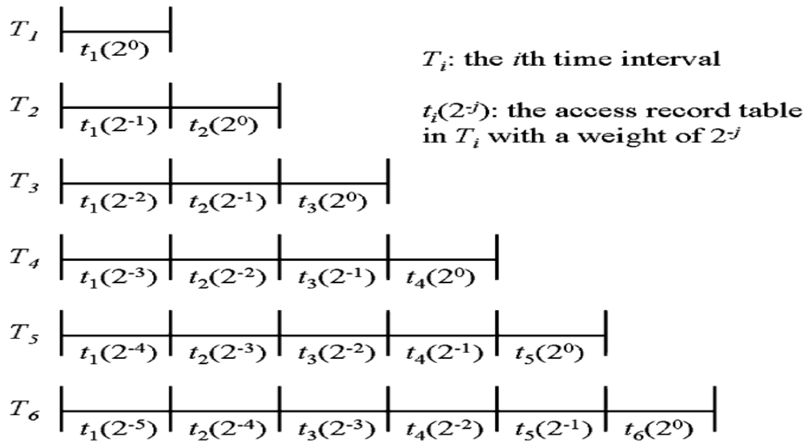

T_1
$t_1(2^0)$
T_2
$t_1(2^{-1})$ $t_2(2^0)$
T_3
$t_1(2^{-2})$ $t_2(2^{-1})$ $t_3(2^0)$
T_4
$t_1(2^{-3})$ $t_2(2^{-2})$ $t_3(2^{-1})$ $t_4(2^0)$
T_5
$t_1(2^{-4})$ $t_2(2^{-3})$ $t_3(2^{-2})$ $t_4(2^{-1})$ $t_5(2^0)$
T_6
$t_1(2^{-5})$ $t_2(2^{-4})$ $t_3(2^{-3})$ $t_4(2^{-2})$ $t_5(2^{-1})$ $t_6(2^0)$
T_i: the ith time interval
$t_i(2^{-j})$: the access record table in T_i with a weight of 2^{-j}

Fig 1: The diagram of half-life for the weight in different time interval [20]

Then we define an *Access Frequency* (*AF*) [20] to exhibit the importance for access history in different time intervals. Assume N_T is the number of time intervals passed, *F* is the set of files that there are in a target storage element. a_{ij} indicates the number of accesses for the file i at time interval j. The Access Frequency for file F is represented as**:**

$$AF(f) = \sum_{t=1}^{N_T} \left(a_f^t \times 2^{-(N_T - t)} \right), \forall f \in F \qquad (1)$$

For instance, a file *X* has been accessed 4 times and 5 Time and 10 Time in the first time interval, the second time Interval and the Third time interval respectively. Then *AF(X)* is $(4\times2^{0}) + (5\times2^{-1}) + (10\times2^{-2})$. *AF* puts different weights for access records of different time intervals. According to Equation (1) we calculate the *AF*s of all files that have been requested. A file with the largest *AF* is chosen as the popular file.

One of the important points in this method is choosing appropriate the length of Time intervals; because that if the length of time interval is too short, the information about data access history is not enough. On the contrary, the information could be overdue and useless if the length is too long. Using the concept **Half-life** causes more weight to be given recent access history. Prevent removal of replicas that have been stored in the storage element recently. The reason this work is that the newly created replicas likely can be accessed many times in the future. While because that newly created replicas and low access frequency may be removed. It will also prevent the creation of multiple replicas.

2.1.2 Evaluation of future transfer cost

The cost factor is an important factor during the whole replica replacement process, which is often affected by some factors such as replica size and bandwidth. Too much bandwidth consumption may block the network and improve the possibility of fault appearance during the transfer process. Consequently, the lower the total cost is, the better the performance of replacement algorithm is. In [21], they chose the best one-the one with the largest bandwidth to transfer the related replica in the beginning of the transfer process. However, some problems happened in the real applications. Because that the replacement of the storage element, the needed file in the best site will be deleted, which will cause the whole replacement to be delayed greatly. In order to address these problems, some changes will be made in our algorithm [16]. We define the bandwidth *B*(*f*) as the mean of the bandwidth of all the replicas in the data grid and N_R is the number of replica of file f and B_i is the bandwidth of the *i*-th replica, which is:

$$B(f) = \frac{\sum_{i=1}^{N_R} B_i}{N_R} \qquad (2)$$

So the cost of the replica, C(f), is defined as:

$$C(f) = \frac{S(f)}{B(f)} \qquad (3)$$

S(f) : the size of the file

2.1.3 Prioritization of replica

Prioritization of replicas determined based on replica value in each storage element. We define the replica value V(F) That it consists of future transfer cost of replica and popular replica.

$$V(f) = \alpha\, AF(f) + \beta\, C(f) \qquad (4)$$

The factor weightings α and β defined in V(F) must be chosen such that

$$\alpha + \beta = 1 \qquad (5)$$

α and β are coefficients of the formulas; and Their value Determined based on The importance of access history or popular replica. A resource with highest rank ensures better aggregate response time compared to resource with lower rank. In this experiment, considered α=0.7 and β=0.3. Using of these coefficients is because that improved results and Prioritization of replicas. So that in the list of Candidate replicas, the first priority is the replicas that had the lowest access with the lowest cost replacement. Then, replicas will be deleted that had the lowest access with the highest cost replacement. In this case, Due to the low access number, used less likely in the future. The third priority is the replicas that have the highest popularity and lowest cost replacement. However; Total Access is high but because that Low cost replacement, they are appropriate candidates for removal. Even in their case it is possible that due to the low cost, they can be accessed remotely. In the end, it still was not enough space available, copies of which have the greatest value and replacement cost are selected.

2.2 File Elimination Stage

This stage uses the results of file evaluation stage in order to decide which file to be the victim and which one to be eliminated from deletion. Firstly, all replicas that have not been achieved during its presence in the storage space are the victims. If it did not create enough space for the newly created replica; our algorithm beginning to replace files that accessed to them.

One approach is to select the less valuable file to be the victim for deletion function. However, this approach has a drawback which is the increasing of the number of victim files until there is enough space for the underlying replica, as it depends on the file value [19]. For example, assume that we have 9 files stored in one storage element with free storage space 300 MB as shown in Table 1. Also assume that we have a file with size 900 MB need to be placed.

In the above example shown in Table 1, the victim file is File2, as it is the less valuable file, but we still need to delete one more file, so the next victim file is File7. So we need to delete two files in order to room one file, in some cases the number of victim files may reach to five as they have small size. That means the system may lose 5 files that are considered stable file to room one file [19].

Table 1: example of data files stored in on storage element with their corresponding file value and file size

File name	File Value	File Size
File 1	45	400
File 2	32	500
File 3	40	700
File 4	55	1200
File 5	50	1100
File 6	60	1300
File 7	35	900
File 8	45	800
File 9	65	1500

Another approach is to delete the file, which has a larger size compared to other files. However, this approach is infeasible as in some cases the file that got large size may have the highest value among other files and the system still needs it. Therefore**,** our approach considered three criteria including file size and predicted transfer cost and file popularity to reduce the number of files replicated and Bandwidth consumption.

The steps of the elimination stage are as follows:

1- Calculate file value based on popularity and future transfer cost.

2-Calculate how much of storage capacity we need in order to room the underlying replica, by applying the following equation:

$$S = F\ Size - free\ space \qquad (6)$$

$$RS = S / 2^K \qquad K >= 0 \qquad (7)$$

Where RS is the required space to host the underlying replica and F Size is size of Input file

3-Eliminate the files those sizes greater than or equal to RS

4-If there is no file that sizes greater than or equal to RS, PRA executes the file replacement algorithm considering the N value, at first K=0. Then we increment K by one and this process is continued until a set of files that meet the criterion is located.

5- Sort the files in ascending order based on File Value. Then identify the victim file. File that has the lowest File Value. The above algorithm is shown by flowchart.

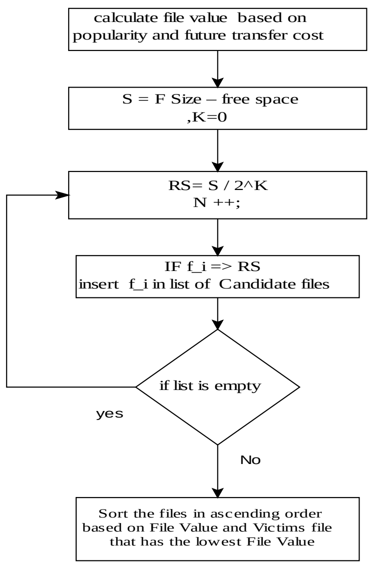

calculate file value based on popularity and future transfer cost
S = F Size – free space ,K=0
RS= S / 2^K N ++;
IF f_i => RS insert f_i in list of Candidate files
if list is empty
yes
No
Sort the files in ascending order based on File Value and Victims file that has the lowest File Value

Fig 2: The steps of the elimination stage

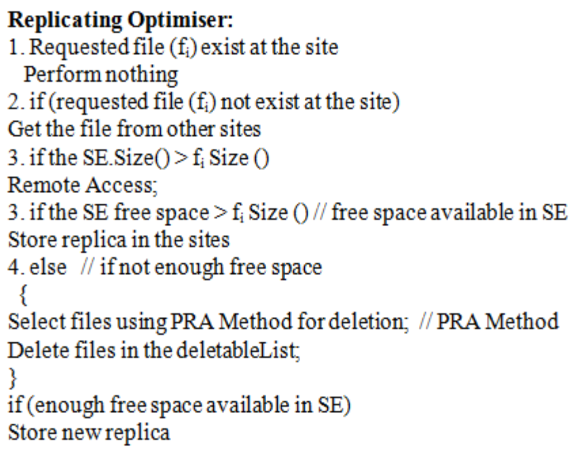
We refer to the example shown in Table 1. We target files that sizes greater than or equal to RS. Therefore, in the above example: RS = 900 – 200 = 700. Completing the elimination stage, that is removing files with sizes less than or equal to 700, we obtain files in Table 2. The file with the least value is File7; hence it will be identified as the victim file.

Table 2: the targeted files for deletion function

File name	File Value	File Size
File 7	35	900
File 3	40	700
File 8	45	800
File 5	50	1100
File 4	55	1200
File 6	60	1300
File 9	65	1500

Perhaps the ineffectiveness of this method is that when we're forced to delete files with size up and the maximum value. But Remove several small files to obtain enough space, In addition to being the victim of more files, may be Total values of several small files more be than a big file. Such an approach considers two important factors. These factors are deleting the less valuable file and resource satisfaction by deleting the minimum number of files.

The pseudo code of replica replacement algorithm is depicted as bellow.

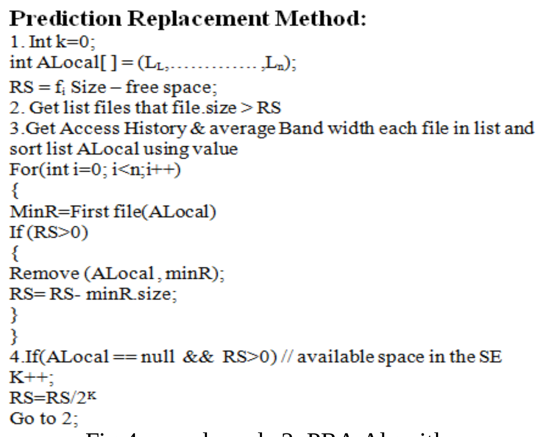


```
Replicating Optimiser:
1. Requested file ($f_i$) exist at the site
   Perform nothing
2. if (requested file ($f_i$) not exist at the site)
Get the file from other sites
3. if the SE.Size() > $f_i$ Size ()
Remote Access;
3. if the SE free space > $f_i$ Size () // free space available in SE
Store replica in the sites
4. else   // if not enough free space
  {
Select files using PRA Method for deletion;  // PRA Method
Delete files in the deletableList;
}
if (enough free space available in SE)
Store new replica
```

Fig 3: pseudo code 1.Modified Replication Optimiser

PRA Optimizer will direct the Storage Element to store replicas that are created by the optimizer as well as to remove files according to the criteria set by the PRA algorithm. The Storage Element will then execute the command and thus stores or removes the particular file. The algorithm as shown in the pseudo code 1, described the overall operation of the implemented replication optimiser. In the pseudo code 1, if Requested file exist at the site do nothing and if requested file there is not at sit and the file size is larger than the storage space can be accessed remotely. But if there are not enough storage space and The file size is smaller than the storage space, used PRA Method for deletion.

In the pseudo code 2, generated a list of file that are larger than required space and placed in the array Alocal. Then we will remove the lowest File Value. If the list was empty, by using the formula $RS=RS/2^K$, The list is filled with smaller replica.



```
Prediction Replacement Method:
1. Int k=0;
int ALocal[ ] = ($L_L$,............ ,$L_n$);
RS = $f_i$ Size – free space;
2. Get list files that file.size > RS
3.Get Access History & average Band width each file in list and
sort list ALocal using value
For(int i=0; i<n;i++)
{
MinR=First file(ALocal)
If (RS>0)
{
Remove (ALocal , minR);
RS= RS- minR.size;
}
}
4.If(ALocal == null  &&  RS>0) // available space in the SE
K++;
RS=RS/$2^K$
Go to 2;
```

Fig 4: pseudo code 2. PRA Algorithm

3. Simulation Setup and Metrics

OptorSim is used to evaluate the performance of PRA algorithm. OptorSim was developed by European Data Grid projects and is written in Java. It provides a framework to simulate the real grid environment. It is developed to test the Dynamic Replication strategies. Using the modules, we can easily compare the effectiveness of different replica optimization algorithms within this environment. OptorSim contains a number of elements including Computing Elements (CEs), Storage Elements (SEs), Resource Broker (RB), Replica Manager (RM) and Replica Optimiser (RO) [22].

Grid topology is that of the CMS testbed which consists of 8 routers and 20 grid nodes. In the CMS configuration, each grid site is allocated a CE and initially empty storage capacity of 50GB, except for the CERN and FNAL sites [17]. For the CMS testbed, CERN and FNAL were given SEs of 100 GB capacity and no CEs. All master files were stored at one of these sites. Every other site was given 50 GB of storage and a CE with one worker node[23].Jobs stored in Storage Element and Processed in the Computing Elements. Routers send requests to other sites. Data Replication strategies assumes that the data are read-only. Grid topology shown in Figure 5.

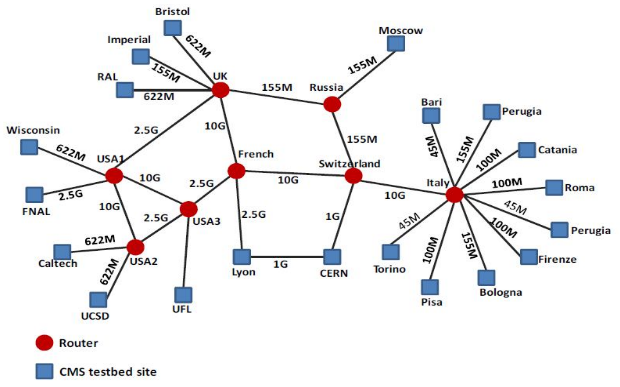


Fig 5: Grid topology for CMS [23]

We ran the simulation process with 100,200,300,400,500 jobs including 6 job types. Jobs are submitted with a fixed probability such that some jobs are more popular than others. Each job is submitted at 25 millisecond intervals. Each job type requires specific files for execution. The order of the files accessed in a job is sequential and has been set in the job configuration file. The number of files in our simulation process is 97. The size of the files is randomly generated from 100MB to 2000MB.

4.1 Simulation Setup

The performance metrics we chose to evaluate the proposed system are: Mean JobExecution Time (MJET), Efficient Network Usage (ENU), Average Storage Usage (ASU) and Total Number of Replications (TNR). These metrics are described below.

Mean Job Execution Time - The mean job execution time is defined as the total time to execute all the jobs divided by the number of jobs completed. The total time includes the time that elapses from when a job enters the queue in a site to await execution until the time when the job finishes its processing and leaves the site. An ordinary grid user would require the fastest possible turnaround time for the job, and so this metric is considered the most important of the evaluation metrics. It is defined by [19, 24] and calculated by the following equation:

$$MJET = \sum_{i=1}^{n} \frac{Job\ Arrival\ Time - Job\ Departure\ Time}{No.of\ Jobs\ Completed} \tag{8}$$

where i is the number of jobs processed through the system.

Efficient Network Usage - File replication is essential to a distributed Grid system but it takes time and uses network bandwidth. Thus, a good balance must be found where any replication is in the interest of reducing future network traffic. We define effective network usage as a measure of how well the optimization strategy uses the network resources. It is defined by [19, 24] and calculated by the following equation:

$$ENU = \frac{N_{remote\ file\ accesses} + N_{file\ replication}}{N_{remote\ file\ accesses} + N_{local\ file\ accesses}} \tag{9}$$

Where $N_{remote\ file\ accesses}$ is the number of accesses that Computing Element reads a file from a remote site, $N_{file\ replication}$ is the total number of file replication occurs, and $N_{local\ file\ accesses}$is the number of times that Computing Element reads a file locally and $N_{local\ file\ accesses}$ is the number of times that Computing Element reads a file from a remote site that is The ratio files transferred to files Requested. A lower value indicates that the utilization of network bandwidth is more efficient and the optimization strategy used is better at placing files in the right places.

Average Storage Usage - Storage usage can be calculated for each site as a percentage of capacity reserved by files according to the total capacity for the underlying storage. The average of the all storage elements in the grid can reflect the total system storage cost. It is defined by [24] and calculated by the following equation:

$$ASU = \frac{\sum_{i=1}^{n} \frac{U}{S} (site\ i)}{n} * 100\% \qquad (10)$$

Where,
U: is the storage usage for each site in MB
n: is the number of sites in the grid
c: is the total capacity of the storage medium.

Total Number of Replications - Represents the number of replication has been done. so a low value indicates that the Victim Files is selected correctly. This metric represents the total created replicas for files requested by the client in a simulation session. Increase the replica number indicates that number of replicaton has increased. Replication causes not only increased bandwidth consumption, But also I/O disk and CPU is consumed. Therefore, must be controlled the frequency of replication to avoid heavy server and load network. The PRA is compared with LFU and LRU strategies.

3.2 SIMULATION RESULTS

The order in which a job requests files is determined by the job's access pattern. There are various access pattern generators used in optorSim. In this work Sequential Access Pattern Generator is used. When compared to LRU and LFU, PRA Algorithm performs better. An ordinary grid user would want the fastest possible turnaround time for their jobs and thus consider Mean Job Execution Time the most important evaluation metric [15].

This algorithm minimizes the Mean Job Execution time and thus the data access time can be improved and even with increasing the numbers of job, it shows better performance. Fig 6 demonstrates the Mean Job Execution Time. Choosing the correct victim replica reduces the number of replacements. Since PRA strategy is implemented based on the past access frequency And future transfer cost and file (replica) size it is able to show better accuracy as the time increases and consequently a better prediction of replica value would happen[24]. This makes PRA to reduce the changes of replicas to be deleted or replaced unless it is worthwhile. As a result, PRA is able to process jobs faster than LFU and LRU algorithm.

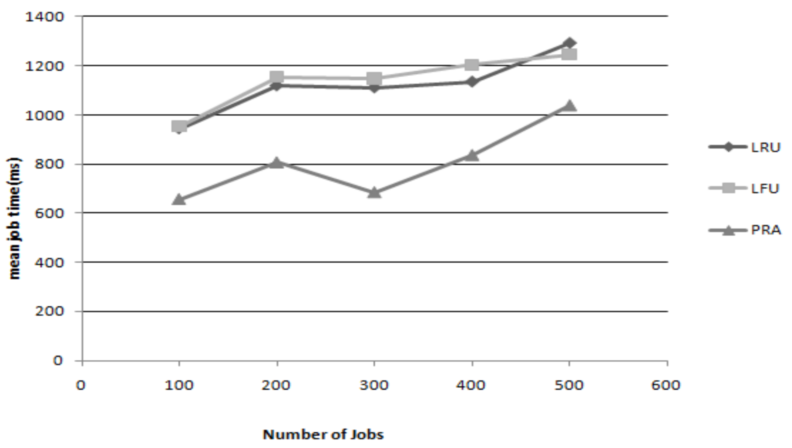


Fig 6: Mean Job Execution Time

Fig 7 demonstrates the Number of Replications. It is important to reduce the number of replication; and it means that selected the correct victim replica.

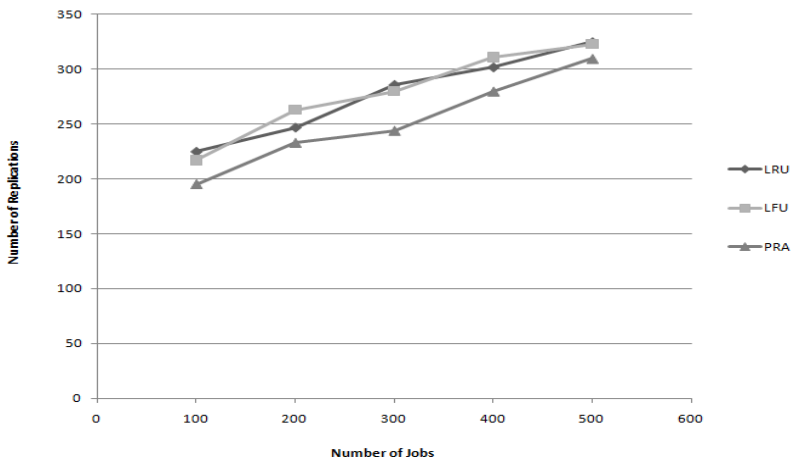


Fig 7: Number of Replications

Figure 8 demonstrates Average Storage Used**.** The difference between PRA and other algorithms is the replica replacement techniques used in deciding the replicas for deletion. As regards that The proposed algorithm is considered several factors simultaneously for select the victim replica And Prevents deletion of important replica and select the suitable victim replica, it could cause Consumption storage capacity is reduced. Thus, PRA has advantages to dynamically choose the replicas for replacement while satisfying storage capacity constraints. The ASU of our algorithm is lower compared to the LFU or LRU algorithm. The reason is that LFU and LRU always replicate, so the large value of file replication will increase the ASU value.

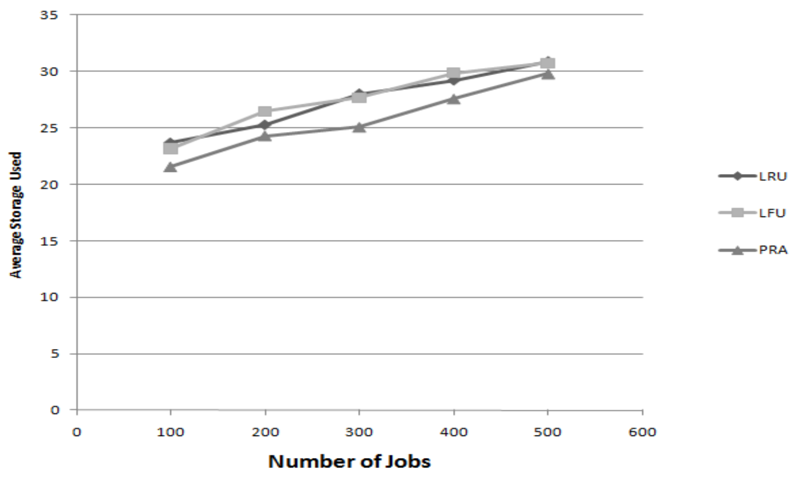


Fig 8: Average Storage Used

Fig 9 demonstrates Effective Network Usage. Effective Network Usage used to quantify use of network bandwidth. The ENU values are ranged between 0 and 1.The less ENU the better performance is, the data locality increases, and reduced bandwidth consumption. Thus the LFU and LRU strategy shows a poor performance in utilizing the bandwidth usage available in the network. This is because in LRU and LFU number of deleted files resulting from performing the replacement process is large. Consequently performing the replication will be increased as well. As a result the ENU will be large. Better performance of the proposed strategy is because victim replica less valuable in the future. It indicates that with PRA algorithm we can expect less replicates which leads to its better performance than LRU and LFU.

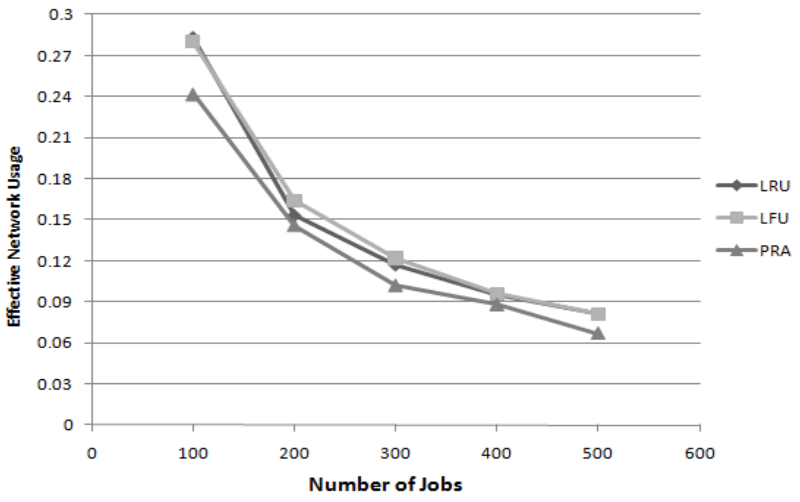


Fig **9:** Effective Network Usage

The simulation also provides wider evaluation range for the algorithms as the job loads increases. While the performances of LRU and LFU were almost similar in most of the cases, the PRA showed improvement in the mean job time taken to execute the assigned jobs. It also has the lowest number of replicas created which are optimally placed in the network. Therefore, this indicates that PRA is the most suitable replica replacement strategy that can be used in the data grid environment [24].

4. Conclusion

In this research, a Prediction Replica replacement algorithm is proposed. The algorithm consists of two stages. First, concept of half-life was introduced and using the half-life to determined the replica popularity and Integration the age and data-access frequency. Then future transfer cost is calculated using two factors: replica size and bandwidth. To calculate the bandwidth, all replica of each file is considered. Finally, Replica value is calculated according to replica popularity and future transfer cost of replica in the first stage. Of course, Different weights to each factor is given. Giving more weight to the popularity has Leads to Prioritization of replicas. In the second stage, PRA (The proposed model) within the list of replicas that Provider space required, deletes or replaces the replicas that are least worth. The proposed model simulated with optorsim and Compared with LRU and LFU.

The simulation results show that PRA successfully increases data grid performance than other replication strategies such as LRU and LFU**.** By using PRA, the mean job execution time and Number of Replications can be minimized, the network is used more effectively and storage space is saved. In other words, the mean job execution time using PRA is about 30% faster than LFU, and faster about 28% than LRU. The reason of that is because the PRA invoke the deletion function with minimum number if there is a need to perform the replacement process. In other words, the process of replacement in PRA occurs with minimum number as it deletes minimum number of files to make a free space for the newly created replica. However, in LRU and LFU the deletion function is invoked many times in one replacement process and need to check every deletion process the storage space of the underlying Storage Element. As a result, LRU and LFU will take longer time to perform the replacement process. The less Effective Network Usage the better performance is. Thus the LFU and LRU strategy shows a poor performance in utilizing the bandwidth usage available in the network. Effective Network Usage using PRA is about 14% faster than LFU, and faster about 11% than LRU. This is because in LRU and LFU number of deleted files resulting from performing the replacement process is large. Therefore, the probability of reading the files remotely will be increased; consequently performing the replication will be increased as well. As a result the ENU will be large. The large number of deleted files by LRU and LFU affects the Number of Replications metric as the number of Replications will be decreased. PRA outperforms the LFU by 10% and LRU by 9% in the Number of Replications metric. Average Storage Used using PRA is about 7% faster than LFU, and faster about 6.5% than LRU. This is because in proposed model considering various factors, Prevent the removal of important replica and several replication. These factors are: locality, size, transfer cost and popularity of replicas considerations effectively to achieve the best performance possible.

Also, this strategy by predicting the transfer cost that will be incurred for replicas at grid sites, replacement of replicas with high cost can be avoided.

As this approach deals only with data replacement strategies, further works can be combined with scheduling algorithm and replication algorithm to improve the overall system performance. We have not tested our strategies in the real grid systems. It will be an important part of our future work. We will also make some extensions to our current approach to further improve its performance. Another further works can be added parallel transfer for improve access time and Job Execution Time.

REFERENCES

[1] Chervenak A, Foster I, Kesselman C, Salisbury C, Tuecke S. The data grid: towards an architecture for the distributed management and analysis of large scientific datasets. Journal of Network and Compute Applications 2000;23(3):187–200.

[2] Foster I. The grid: a new infrastructure for 21st century science. Physics Today 2002;55(2):42–7.

[3] Allcock B, Bester J, Bresnahan J., Chervenak AL, Kesselman C, Meder S, Nefedova V, Quesnel D, Tuecke S, Foster I, Secure. efficient data transport and replica management for high-performance data-intensive computing, in: Proceedings of the First Eighteenth IEEE Symposium on Mass Storage Systems and Technologies, 2001.

[4] Wolfson O, Milo A. The multicast policy and its relationship to replicated data placement. ACM Transactions on Database System 1991;16(1):181–205.

[5] Bae MM, Bose B. Resource placement in torus based networks. IEEE Transactions on Computers 1997; 46(10):1083–92.

[6] Loukopoulos T, Lampsas P, Ahmad I. Continuous replica placement schemes in distributed systems, in: Proceedings of the 19th International Conference on Supercomputing, 2005, pp. 284–292.

[7] Rehn-Sonigo, V, Optimal replica placement in tree networks with QOS and bandwidth constraints and the closest allocation policy, Technical Report, No. 6233, INRIA, 2007.

[8] Tu M, Li P, Xiao L, Yen I-L, Bastani FB. Replica placement algorithms for mobile transaction systems. IEEE Transactions on Knowledge and Data Engineering 2006;18(7):954–70.

[9] Rahman RM, Barker K, Alhaij R. Replica placement strategies in data grid. Journal of Grid Computing 2008;6(1):103–23.

[10] Ranganathana K, Foster I. Identifying dynamic replication strategies for a high performance data grid, in: Proceedings of the International Grid Computing Workshop, 2001, pp. 75–86.

[11] Abawajy, JH, Placement of file replicas in data grid environments, in: ICCS 2004, in: Lecture Notes in Computer Science, 2004, pp. 66–73.

[12] Srikumar, V., Rajkumar, B., Kotagiri, R.: A Taxonomy of Data Grids for Distributed Data Sharing, Management, and Processing. ACM Computing Surveys 38, 2006.

[13] Teng, M., Junzhou, L.: A prediction-based and cost-based replica replacement algorithm research and simulation. In: 19th International Conference on Advanced Information Networking and Applications (AINA 2005), 2005, pp. 935–940.

[14] T. Tian and J. Luo, "A Prediction-based Two-Stage Replica Replacement Algorithm," in 11th International Conference on Computer Supported Cooperative Work in Design, (CSCWD 2007), 2007, pp. 594-598.

[15] Tian, T., Luo, J.: A VO-Based Two-Stage Replica Replacement Algorithm. In: Network and Parallel Computing, 2010, pp. 41–50.

[16] Zhao, W., Xu, X., Xiong, N., Wang, Z.: A Weight-Based Dynamic Replica Replacement Strategy in Data Grids. In: Asia-Pacific Services Computing Conference, 2009, pp. 1544–1549.

[17] L.H. Ai and S.W. Luo, Job-attention Replica Replacement Strategy, in: Proceedings of 8th ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing, 2007, pp. 837-840.

[18] Jianhua.J, Huifang Ji2, Gaochao Xu . ARRA: an Associated Replica Replacement Algorithm Based on Apriori Approach for Data Intensive Jobs in Data Grid, Key Engineering Materials Vols. 439-440, 2010, pp 1409-1414.

[19] Madi.M, Yusof Y, Hassan S, A Novel Replica Replacement Strategy for Data Grid Environment. ICSECS 2011, Part III, CCIS 181, , 2011 pp. 717–727.

[20] Ruay-Shiung Chang, Hui-Ping Chang, Yun-Ting Wang A dynamic data replication strategy using access-weights in data grids. Springer Science+Business Media, LLC 2008, DOI 10.1007/s11227-008-0172-6, 2008,pp. 277–295.

[21] Ma Teng, and Luo Junzhou, "A Prediction-based and Cost-based Replica Replacement Algorithm Research and Simulation", Proceedings of the 19th International Conference on Advanced Information Networking and Applications, IEEE Computer Society, 2005, pp. 935-940.

[22] David G.Cameron, Ruben Carvajal-Schiaffion, Jamie Ferguson,A.paul Miller, Caitriana Nicholson, Kurt Stockinger, Floriano Zini, " OptorSim v2.1 Installation and User Guide", October 4, 2006.

[23] D. G. Cameron, A. P. Millar, C. Nicholson, University of Glasgow, Glasgow G12 8QQ, OPTORSIM: A SIMULATION TOOL FOR SCHEDULING AND REPLICA OPTIMISATION IN DATA GRIDS, , .

[24] Soosai Al, Sulaiman Na, Dynamic Replica Replacement Strategy in Data Grid, In: Proceeding :Computing Technology and Information Management (ICCM), 2012 8th.