ISSN 2278-3091
Volume 9, No.1.1, 2020
International Journal of Advanced Trends in Computer Science and Engineering
Available Online at http://www.warse.org/IJATCSE/static/pdf/file/ijatcse6791.12020.pdf
https://doi.org/10.30534/ijatcse/2020/6791.12020

Binary Voting Fragmented Database Replication Model

Noraziah Ahmad[1,2*], Ainul Auni Che Fauzi[1], Syifak Izhar Hisham[1], Zarina Mohd[3], Mohd Helmy Abd Wahab[4]

[1]Faculty of Computer Systems & Software Eng., Universiti Malaysia Pahang, 26300 Kuantan, Pahang, Malaysia
[2]IBM Centre of Excellence, Universiti Malaysia Pahang,26300 Kuantan, Pahang, Malaysia
[3]Faculty of Informatic and Computing, Universiti Sultan Zainal Abidin, 22200 Besut, Terengganu, Malaysia.
[4]Department of Computer Engineering, Faculty of Electrical and Electronic Eng.,
Universiti Tun Hussein Onn Malaysia, 86400 Batu Pahat, Johor, Malaysia.
*Corresponding author E-mail: noraziah@ump.edu.my

ABSTRACT

Replication is a useful technique for distributed database systems. Existing techniques such as Read One-write-All (ROWA) and Branch Replication Scheme (BRS) are the popular techniques being used. However, these techniques have their weaknesses in terms of replication time taken and communication costs. Consequently, ROWA and BRS take long executing time for a transaction since they have to replicate its data to all servers. In this research, the some-data-to-some-sites technique Binary Voting Fragmented Database Replication (BVFDR) model is proposed. This technique only considers the adjacent servers binary vote assignment to its logical grid structure on fragmented data copies in order to manage transactions in the systems. Thus, it minimizes the storage capacity needed since we store database that has been fragmented. An experiment has been carried out in three replicated servers. The results have been compared with existing techniques such ROWA and BRS. The result shows that BVFDR is able to preserve data consistency and outperformed in terms of time taken for a complete transaction compare to existing techniques. Overall, BVFDR able to manage fragmented data replication and transaction management in distributed database environment by preserving data consistency through the synchronization approach.

Key words: Data Grid; Distributed Database; Fragmentation; Replication; Computational Intelligence.

1. INTRODUCTION

Nowadays, huge numbers of data are generated around the world distributed across data grid. Data grid is emerging as the main part of the infrastructure for large-scale data intensive applications. One of the biggest problems that data grids users have to overcome today is to improve the management of data. Providing reliable services along with high data availability and the performance are the important requirements that need to be essentially met.

The concept of replication is used to ensure these requirements. The process of data replication is keeping several copies of a data file at multiple servers in order to achieve better performance, availability and reliability. Replication strategy is one of effective solutions to meet the requirement of service response time by preparing data in advance to avoid the delay of reading data from servers [1]. Replication is commonly used in data intensive applications where data is shared and need to be accessed from different servers, situated at different geographical locations.

As everything has pros and cons, replication too has some drawbacks associated with it. As the number of replicas of data files increases, the cost of creating and maintaining these replicas also increases. One way to provide reliable data with low cost and minimum response time, a database fragmentation can be combined with data replication. In order to fragment a file, it will split data into fragments, which should be allocated to sites over the network in the allocation stage [2]. Every part that produced after fragmentation is called as a database fragment. Fragmentation in distributed database is very beneficial in terms of practice, dependability and effectiveness of a system. Fragmentation phase is the process of distributing a database table into a set of smaller tables. The process of distributing the generated fragments over the database system servers is called allocation.

Each part of the fragmented distributed database may be copied. When a data is updated at one site, the changes are noted and kept locally. Then, they are submitted and applied at all of the distant servers. In order to ensure the consistencies for all the replicated data, synchronous replication can be practiced. It can be divided into several methods, i.e., copy all data to all servers, copy all data to some servers and copy some data to all servers. A proper synchronization method is required to preserve the integrity and reliability of the data in distributed environments.

In this paper, we manage fragmented database replication and management of data transaction for online website using a new proposed technique called Binary Voting Fragmented Database Replication model. This technique is the combination of replication and fragmentation. Combining these two techniques will increase data availability and data reliability as one server goes down, users can still query or update data by accessing the replica servers.

The paper is organized as follows. In Section 2, we reviewed previous techniques on replication in distributed database. In Section 3, we explained the proposed technique. Next, we presented the results and discussions, including the real time result in Section 4. Finally, we concluded our work in Section 5.

2. RELATED WORKS

In this section, it reviews about replication, database fragmentation and the existing techniques that used the same technique with BVFDR which is replication in distributed database system.

2.1. Replication

The general idea of replication is to store several copies of the same data in different sites across the grid. This clearly scales up the performance by reducing remote access delay and mitigating single point of failure. In addition, data replication helps overcoming long wide-area data transfer latencies by keeping data close to locations where queries are originated. Indeed, through replication, data grid can achieve high data availability, improved bandwidth consumption, and better fault tolerance [3]. Replication is commonly used in data intensive applications where data are shared and need to be accessed from different sites, situated at different geographical locations.

2.2. Database Fragmentation

Fragmentation in a single database needs to be divided into two or more pieces such that the combination of the pieces yields the original database without any loss of information. Each piece that produced after fragmentation is known as a database fragment [4]. Fragmentation in distributed database is very useful in terms of usage, reliability and efficiency [5]. Fragmentation phase is the process of clustering the information accessed simultaneously by applications in fragments, while the process of distributing the generated fragments over the database system sites is called allocation phase [6]. In order to fragment a database, it is possible to use two basic methods which are vertical fragmentation and horizontal fragmentation. Other than these two methods, it is also possible to execute mixed or hybrid fragmentation on a class by combining both techniques [6]. In the object model, vertical fragmentation breaks the class logical structure (its attributes and methods) and distributes them across the fragments, which will logically contain the same objects, but with different structures. On the other hand, horizontal fragmentation distributes class instances across the fragments, which will have exactly the same structure but different contents. Thus, a horizontal fragment of a class contains a subset of the whole class extension [6]. Each partition/fragment of a distributed database may be replicated [7]. Changes applied at one site are captured and stored locally before being forwarded and applied at each of the remote locations.

2.3. Existing Techniques

2.3.1. Read-One-Write-All Monitoring Synchronization Transaction System (ROWA-MSTS)

Read-One-Write-All Monitoring Synchronization Transactions System (ROWA-MSTS) has been developed based on ROWA technique. The ROWA-MSTS technique handles each site either it is operational or down. The researcher used VSFTPD (GPL licensed FTP server for UNIX systems) as an agent communication between replicated servers [8]. In ROWA-MSTS techniques, replicas consistencies are guaranteed by the consistency of execution on one replica, but the client replicas are only updated and cannot provide accurate responses to queries. Synchronous replication methods guarantee that all replicas are maintained consistently at all times by executing each transaction locally only after all replicas have agreed on the execution order. Hence, a very strict level of consistency is maintained. Figure 1 shows the framework of ROWA-MSTS in distributed environment.

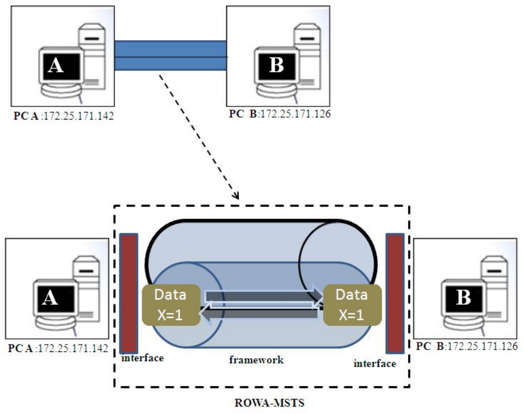

Figure 1: The framework of ROWA-MSTS
(Source: Noraziah et al., 2010)

However, this strategy practices all-data-to-all-sites replication protocol. That means, all servers will have the same data. There will be a lot of data redundancy and waste of space. In addition, the execution time for a transaction will be high since the primary server has to wait for all other neighbour servers to proceed with the transaction.

2.3.2. Branch Replication Scheme (BRS) Protocol

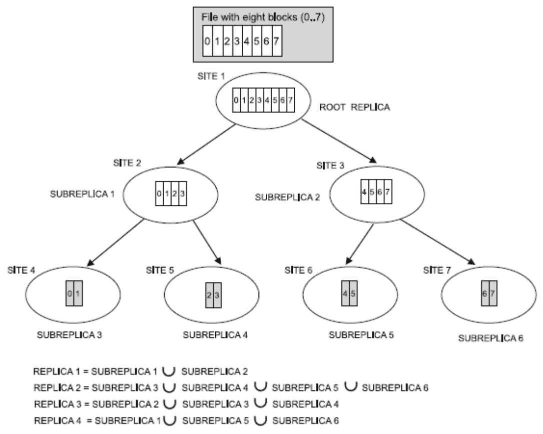
File with eight blocks (0..7)
0 1 2 3 4 5 6 7
SITE 1
0 1 2 3 4 5 6 7
ROOT REPLICA
SITE 2
SUBREPLICA 1
0 1 2 3
SITE 3
SUBREPLICA 2
4 5 6 7
SITE 4
0 1
SUBREPLICA 3
SITE 5
2 3
SUBREPLICA 4
SITE 6
4 5
SUBREPLICA 5
SITE 7
6 7
SUBREPLICA 6
REPLICA 1 = SUBREPLICA 1 ∪ SUBREPLICA 2
REPLICA 2 = SUBREPLICA 3 ∪ SUBREPLICA 4 ∪ SUBREPLICA 5 ∪ SUBREPLICA 6
REPLICA 3 = SUBREPLICA 2 ∪ SUBREPLICA 3 ∪ SUBREPLICA 4
REPLICA 4 = SUBREPLICA 1 ∪ SUBREPLICA 5 ∪ SUBREPLICA 6

Figure 2: Data replication in BRS
(Source: Pérez et al., 2010)

The goals of Branch Replication Scheme (BRS) are to increase the performance, fault tolerance and scalability. In BRS, by using hierarchical topology each replica is composed of a different set of organized subreplicas using a hierarchical topology. In order to increase the scalability and performance of the system for read and write operations, BRS uses parallel I/O. The main features of BRS are root replica, parallel replication; fine grain replication, partial replication of popular file fragments. Parallel data access better resource usage. This technique needs low space per storage to support replica. Figure 2 shows data replication in BRS.

In BRS, replicas are created as close as possible to the clients that request the data files. The root replica grows towards the clients in a branching way, where the replicas will be split into several subreplicas. By using this approach, the growing of replica tree is driven by client needs. This means a replica is expanded or attracted towards the clients [9].

In addition, replication does not have to be for the entire replica. Subreplicas can also be replicated based on the previous conditions. Assume that accesses to a file are not uniformly distributed and that, as a result, the subreplica *Ri* storage node is overloaded. BRS can replicate only this subreplica to discharge this node. Consequently, the growth of the replication tree might not be symmetric and different branches could have different depths [9].

In order to maintain consistency among the updates by clients, a mechanism is proposed. Clients only can modify the data located in the terminal replica, or referred as the leaf nodes of the replication tree. Thus, the location of the replica is reduced to the location of the deepest subreplicas that support the range of data requested by the application. Data update is performed bottom_up, from the children replicas to the parent until the root replica is reached. Only updated blocks are propagated. Assume, for the example in Figure 2, that block 3 of replica 2 (located in SITE 5) is written. The consistency algorithm sends block 3 to the replica's parent (SITE 2), that again sends block 3 to its parent (SITE 1). As the replica in SITE 1 is the root, the algorithm stops. Thus replica updating can be executed minimizing the number of steps (3) and the amount of information sent (only 1 block in this example). The amount of data transferred would be a minimum of 8 blocks.

A problem may occur when a client tries to write in a subreplica which is not terminal, because that subreplica has been split into other subreplica. In this case, the error “write not allowed” is sent to the client. This may only happen because the client opens the file in the read-only mode. Thus, the client has to open the file for writing or updating and look for the replica that contains the data range needed by the client. Besides, the drawback for BRS is it is costly because it requires many servers.

2.3.3. Popular Group of File Replication (PGFR) Algorithm

Popular Group of File Replication (PGFR) considers dependency between files (data) for data replication. It also replicates a group of dependent files to the requested grid sites and reduces mean job execution time, bandwidth consumption as well as avoiding unnecessary replication. The proposed algorithm is based on three assumptions: jobs in a grid site have similar interests in files, jobs have the temporal locality of file accesses, and all files are read-only [10]. Based on this assumption, and file access history, PGFR builds a connectivity graph to recognize a group of dependent files in each grid site and replicates the most Popular Groups of Files to each grid site, thus increasing the local availability. This paper used OptorSim simulator to evaluate the efficiency of PGFR algorithm [10]. The simulation results show that PGFR achieves better performance compared to the existing algorithm; PGFR minimized the mean job execution time, bandwidth consumption, and avoiding unnecessary replication. PGFR discovers the most Popular Group Files for a grid site according to their file access history, and then replicates the most dependent files to each grid site. Therefore, later when a user of that grid site requests some files, they will be available locally. Thus, PGFR decreases access latency and bandwidth consumption. To evaluate the efficiency of the algorithm, data grid simulator has been used, OptorSim. PGFR then is compared to five existing algorithms which are No-Replication, Least Recently Used (LRU), Least Frequently Used (LFU), EcoModel Zipf-like distribution, and PDDRA [11]. The first four techniques exist in the OptorSim. In the simulation performance evaluation metrics of Mean Job Execution Time (MJET), Total number of replications, and Effective Network Usage (ENU) were used. The simulation has been running for different file access patterns. The simulation results showed that PGFR reduces MJET and ENU.

3. BVFDR MODEL

In this section, we proposed Binary Vote Fragmented Database Replication (BVFDR) model by considering the distributed database fragmentation. The following notations are defined:

a) S is a transaction.
b) R is a relation in database.
c) T is a tuple in fragmented
d) x is an data in T which will be modified by element of S.
e) y is an data in T which will not be modified by element of S.
f) R_1 is a vertical fragmented relation with data x derived from R.
g) R_2 is a vertical fragmented relation without data x derived from R_1.
h) P_k is a primary key.
i) $P_{k,x}$ is a primary key and data x.
j) $P_{k,y}$ is a primary with data y, where $y \neq x$
k) $R_{1_{(Pk,x)}}$ and $R_{1_{(Pk,y)}}$ are a horizontal fragmentation relation derived from R_1.

l) ε and ρ are groups for the transaction *S*.
m) λ = ε or ρ where it represents different group for the transaction *S* (before and until get quorum).
n) S_ε is a transaction that comes before S_ρ, while S_ρ is a transaction that comes after S_ε.
o) *U* is a union of all data objects managed by all transactions *S* of BVFDR.
p) $\left\lceil\frac{n}{2}\right\rceil$ is the greatest integer function (i.e., *n*=9, $\left\lceil\frac{9}{2}\right\rceil = 5$).

3.1. Implementation of BVFDR

To demonstrate BVFDR transaction, nine servers that logically organized in 3 × 3 are considered based on BVFDR two-dimensional logical design. The number of replicated data, d, can be 3, 4 or 5. The 3 replication servers are deployed as in Figure 3. Each server or node is connected to one another through a local network. An experiment has been done where two transactions request to update same instant at two different servers.

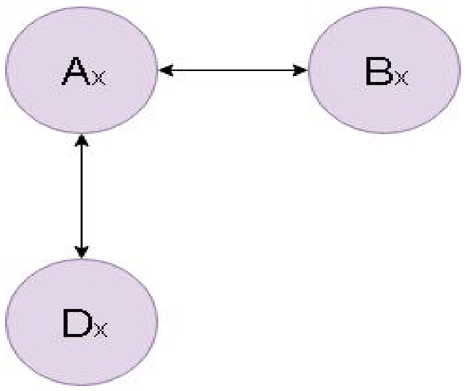

Figure 3: Three replication servers connect each other.

and the primary replication server should be connected each other logically. Each server has been assigned with vote 0 or 1. Vote 0 means the server is free locked and able to proceed with a new transaction. In contrast, vote 1 means the server is busy which means it is already locked. Hence, new transaction cannot be initiated on that server.

Using BVFDR model, each primary replica copies other database to its neighbour replicas. Client can access other database at any server that has its replica. We assume that data *x* located in primary Server A while Server B and Server D are the neighbour replicas. If two transactions, $\boldsymbol{S}_\varepsilon$ and $\boldsymbol{S}_\rho$ request to update instant *x* at two different servers, A and D at the same time, the result as shown in Table 1.

Table 1: Result for two transactions initiate at two sites

Replica / Time	A	B	D
t1	unlock(*x*)	unlock(*x*)	unlock(*x*)
t2	begin_transaction	begin_transaction	begin_transaction
t3	$S\varepsilon_x$ write lock(*x*), counter_w(*x*)=1		$S\rho_x$ write lock(*x*), counter_w(*x*)=1
t4	$S\varepsilon_x$ propagate lock:B		S_{ρ_x} propagate lock:A
t5	$S\rho_x$ fail to get lock:A, counter_w(*x*)=1	S_{ε_x} lock(*x*) from A	
t6	$S\varepsilon_x$ get lock:B, counter_w(*x*)=2		
t7	obtain quorum, release lock: D		
t8			S_{ρ_x} release lock
t9			S_{ε_x} lock(*x*) from A
t10	$S\varepsilon_x$ get lock:D and H, counter_w(*x*)=3		
t11	update x		
t12	S is fragmented into S1 and S2		
t13	S_1 is fragmented into $S_{1_{(Pk,x)}}$ and $S_{1_{(Pk,y)}}$		
t14	commit $\hat{S}_{\varepsilon_x} \in S_\varepsilon$	commit $\hat{S}_{\varepsilon_x} \in S_\varepsilon$	commit $\hat{S}_{\varepsilon_x} \in S_\varepsilon$
t15	unlock(x)	unlock(x)	unlock(x)

From Table 1, at time equals to 1 (*t1*), instant *x* at all servers are unlocked. At *t2*, the transaction begins. At *t3*, there are two transactions, $\boldsymbol{S}_{\varepsilon_x}$ and $\boldsymbol{S}_{\rho_x}$ request to update instant *x* at the same time. Both of transactions initiate lock. Based on Table 1, *t2* and *t3* is the *Initiate Lock.* At this time, the target set for each server has changed to 1 which means the server is busy. Hence, write counter for both server *A* and *D* now is equal to 1. Next, propagate lock at server *B* at *t4.* At *t5*, $\boldsymbol{S}_{\varepsilon_x}$ lock(*x*) from *A* and write counter for $\boldsymbol{S}_{\varepsilon_x}$ now equal to 2 at *t6*. At the same time, $\boldsymbol{S}_{\rho_x}$ propagate lock at server *A*. Since $\boldsymbol{S}_{\varepsilon_x}$ already lock instant *x* at server *A*, the target set for the server has now been equal to 1. Hence, $\boldsymbol{S}_{\rho_x}$ will not success to get lock from it. Then, at *t7*, $\boldsymbol{S}_{\varepsilon_x}$ obtain majority quorum and release lock $\boldsymbol{S}_{\rho_x}$ at server *D*. Based on Figure Table 1, at *t8* $\boldsymbol{S}_{\rho_x}$ release lock At *t9*, $\boldsymbol{S}_{\varepsilon_x}$ lock(*x*) from *A* at server *D* write counter for $\boldsymbol{S}_{\varepsilon_x}$ now equal to 3 at *t10.* Therefore, instant *x* is updated at *A* at *t11,* the relation *S* is fragmented into S_1 and S_2 using vertical fragmentation. The relation S_1 fragmented again at *t13* using horizontal fragmentation into $\boldsymbol{S}_{\boldsymbol{1}_{(Pk,x)}}$ and $\boldsymbol{S}_{\boldsymbol{1}_{(Pk,y)}}$. Finally, at *t17*, $\boldsymbol{S}_{\varepsilon_x} \in \boldsymbol{S}_\varepsilon$ is commit to all sites and at *t14*, instant *x* at all replica servers will unlock and ready for the next transaction to take place. Based on Table 1, *t15, t16* and *t17* is the *Database Fragmentation and Commit.*

4. RESULTS AND DISCUSSION

In this section, the times taken to update data and complete a transaction are compared between two existing techniques, namely ROWA and BRS, and the technique proposed in this research which is BVFDR.
In ROWA, the data is replicated to all sites, means the data is available at all sites. Hence, ROWA has high data availability. In BRS, a data is chunk into several subreplica and allocated in a server. This server is organized in the form of tree structure. When a data is updated, it will be replicated using bottom up scheme. A problem may occur when a client tries to write in a subreplica which is not terminal, because that subreplica has been split into other subreplicas. In this case, the error handling control sent the message to prevent the write authorization to the client. In BVFDR, the server is organized in a form of two-dimensional grid structure. Each site has a premier data. When data is updated, it will replicate to its neighbours. When data is not available at a particular site, user still can access the data through other replica site. Therefore, the data availability is high in BVFDR.

4.1. Time Taken Comparison

In this section, the time captured is compared to BVFDR in order to prove BVFDR technique requires the lowest time to update a data.
Table 2 shows the executing time comparison between ROWA, BRS with BVFDR. From Table 2, it is proved that BVFDR requires the lowest time to complete a transaction. Total time taken for BVFDR is 29 ms. The highest time taken is BRS with 623 ms. This is because BRS required 8 replica copies. Moreover, the replicas can be more than 8 due to client request but cannot be lowered than 8 replicas. BVFDR has 95.35% improvement from BRS technique in terms of time taken. Hence, time taken for the transaction also can be increased when the techniques does not the database fragmentation which causes of large amount of bandwidth every time data been updated. Besides that, ROWA took 72 ms for a transaction due to its maximum numbers of replica copies. BVFDR has 59.72% improvement from ROWA technique. For ROWA, it would be a greater different result when the server involved is 9 or more.

Table 2: Time Taken Comparison

Replication Technique	Initiate Lock (ms)	Propagate Lock (ms)	Obtain Majority Quorum (ms)	Datanase Fragmented & Commit (ms)	Total Time Taken (ms)	BVFDR Improvement:
ROWA	16	23	2	31	72	59.72 %
BRS	50	398	8	162	623	95.35 %
BVFDR	8	11	0	10	29	-

5. CONCLUSION

In order to preserve data availability and data consistency of the website, managing transactions is very important. With the aim of managing fragmented database replication and transaction management, we design a new model called Binary Voting Fragmented Database Replication (BVFDR). From the experiment result, we can say that managing replication and transaction through BVFDR able to preserve data consistency in a shorter time so it is very useful for critical data update such us bank data, e-commerce and etc.

ACKNOWLEDGEMENT

Appreciation conveyed to University Malaysia Pahang Internal Grant under RDU170398; Research Management, Innovation and Commercialisation Centre (RMIC), Universiti Sultan Zainal Abidin; and Universiti Tun Hussein Onn Malaysia for financial support of this project.

REFERENCES

[1] Shaoming Pan, Lian Xiong, Zhengquan Xu, Yanwey Chong, Qingxiang Meng (2018), A dynamic replication management strategy in distributed GIS, *Computer and GeoSciences*, Vol112, 1-8. https://doi.org/10.1016/j.cageo.2017.11.017

[2] Hossein Rahimi, Fereshteh-Azadi Parand, Davoud Riahi (2018), Hierarchical simultaneous vertical fragmentation and allocation using modified Bond Energy Algorithm in distributed databases, *Applied Computing and Informatics*, Vol14, 127-133.

[3] Hamrouni T, Slimani S, Charrada F B (2015), A Survey of Dynamic Replication and Replica Selection Strategies Based on Data Mining Techniques in Data Grids, *Engineering Applications of Artificial Intelligence*, Vol48, 140–158.

[4] Apers PMG (1988), Data Allocation in Distributed Database Systems, *ACM Transaction on Database System*, Vol13, 263–304.

[5] Ainul Azila, Noraziah A, Fauzi AAC, Deris MM, Saman MYM, Zain NM, Khan N (2011), Lowest Data Replication Storage of Binary Vote Assignment Data Grid, *Procedia-Social and Behavioral Sciences*, Vol28, 127-132.

[6] Baiao F, Mattoso M, Zaverucha G (2000), Horizontal Fragmentation in Object DBMS: New issues and Performance Evaluation, *Proceeding of IEEE International Conference on Performance, Computing, and Communications*, 108-114.

[7] Deris MM, Abawajy JH, Taniar D, Mamat A (2009), Managing Data using Neighbour Replication on a Triangular-Grid Structure, *International Journal of High Performance Computing and Networking*, Vol6, No1, 56-65. https://doi.org/10.1504/IJHPCN.2009.026292

[8] Noraziah A, Abdalla AN and Roslina MS (2010), Data Replication Using Read-One-Write-All Monitoring Synchronization Transaction Systems in Dis-

tributed Environment, *Journal of Computer Science*, Vol6, No10, 1033-1036.

[9] Pérez, JM, Carballeira FG, Carretero J, Calderón A, and Fernández J (2010), Branch Replication Scheme: A New Model for Data Replication in Large Scale Data Grids, *Future Generation Computer Systems*, Vol26 12-20.

[10] Leila Azari, Amir Masoud Rahmani, Helder A. Daniel, Nooruldeen Nasih Qader (2018), A data replication algorithm for groups of files in data grids, *Journal of Parallel and Distributed Computing,* Vol113, 115-126.

[11] NazaninSaadat, Amir MasoudRahmani (2012), PDDRA: A new pre-fetching based dynamic data replication algorithm in data grids, *Future Generation Computer Systems* Vol28, No4, 666-681. https://doi.org/10.1016/j.future.2011.10.011