OXFORD UNIVERSITY PRESS The Computer Journal

A Dynamic Data Replication Strategy Based on Categorization

Journal:	*The Computer Journal*
Manuscript ID:	COMPJ-2013-09-0582
Manuscript Type:	Original Article
Date Submitted by the Author:	14-Sep-2013
Complete List of Authors:	Bsoul, Mohammad; The Hashemite University, Alsarhan, Ayoub; The Hashemite University, Hammad, Maen; The Hashemite University, Otoom, Ahmad; The Hashemite University,
Key Words:	Data Grid, Replication, Categories, Plain Caching, Fast Spread, Simulation

A Dynamic Data Replication Strategy Based on Categorization

Mohammad Bsoul Ayoub Alsarhan Maen Hammad
and Ahmad Otoom

September 14, 2013

Abstract

Data Replication is copying the data from a certain location to another location. Replication is used in Data Grid to have two or more copies of the same data at different locations. In this paper, a Category-based dynamic replication strategy (CDRS) is proposed. The strategy takes into account that the replicas exist on a node belong to different categories. Each of these categories is given a value that determines its importance for the node. When the node's storage is full, the node starts to store only the replicas that belong to the category with the highest value. The results of the simulations show that the new proposed strategy achieved better performance than Plain Caching and Fast Spread strategies in terms of total transit time and total bandwidth consumption.

Data Grid; Replication; Categories; Plain Caching; Fast Spread; Simulation.

1 Introduction

A Data Grid is a system that comprises a group of geographically distributed computer and storage resources located at different locations, and allows users to share data and other resources [4,9,13].

In the Data Grid, when a file is requested, large amount of time and bandwidth could be consumed to send the file from the server to the client. Therefore, it could be appropriate to make a number of replicas of the same file at different places. There are two types of replication: Static replication [7] and dynamic replication [2,3,5,8,10,14–16,19–27]. Static replication specifies the location of replicas before the run time (design phase), while dynamic replication specifies their location during the run time. Dynamic replication is considered better than static replication because it can adapt to changes in user behavior. However, All the existing dynamic replication strategies are replica-based and not category-based and which means that the decision to copy the replica is taken based on the importance of the replica alone and not the importance of the category the replica belongs to. Some of the well-known existing dynamic replication strategies are Plain Caching, and Fast Spread [6,11,17]. Plain Caching stores the requested replica on the storage of the requesting node. In this strategy, the requester searches its storage for the requested replica. If it is there, it uses it. If not, a request is sent from the requester to all nodes on a predetermined path. The first node on the predetermined path that has the requested replica on its storage will send the replica to the requester. If the requester's storage is full, a group of replicas (that contains one or more replicas) needs to be deleted in order to store the new replica. This strategy shows the case of caching all the requested replicas. On the other hand, Fast Spread stores a replica of the requested file at each node along its path to the requester. If the storage of one of these nodes is full, a group of replicas needs to be deleted in order to store the new replica. Fast Spread is considered one of the best replication strategies especially for random request patterns [18]. However, the above two strategies do not consider the importance of existing replicas to determine if they need to be replaced. Thus, if the node's storage is full, a group of replicas needs to be deleted in order to store the new replica even if that group is more important than the new replica.

The aim of this paper is proposing a new strategy that takes into consideration the number of requests (NOR) of the category to which the requested replica belongs when determining if the requested replica must be copied in the case the node's storage is full. The NOR of a category is increased by one each time one of replicas that belongs to it is requested by its node.

The rest of this paper is organized as follows: Section 2 gives a description of the employed network structure. Section 3 describes the categorization process. Section 4 presents the new proposed strategy. Section 5 describes the metrics for measuring the strategies' performance. Section 6 explains how the simulation is configured. Section 7 discusses the simulation results. Section 8 concludes the paper and describes future work to improve our work.

2 Network structure

We employed the network structure in [3]. The employed network structure consists of a Server node and a number of Client nodes that issue requests. The Server node has the main storage and it contains all the data in the Data Grid. On the other hand, each Client node has a small storage if compared with that of a Server node and therefore cannot accommodate all the requested replicas. Therefore, some of the replica requests are served non-locally.

In this structure, there is a predetermined path (e.g. shortest path) from each Client node to the Server node. When a Client node requests a replica, it first searches its own storage. If it is stored there, it just uses it. If it is not stored in its storage, the client traverses the predetermined path until it finds a a copy of the replica. Then, this copy is sent to that client. Fig. 1 shows the tree of the used network topology. The used network topology is a complete graph.

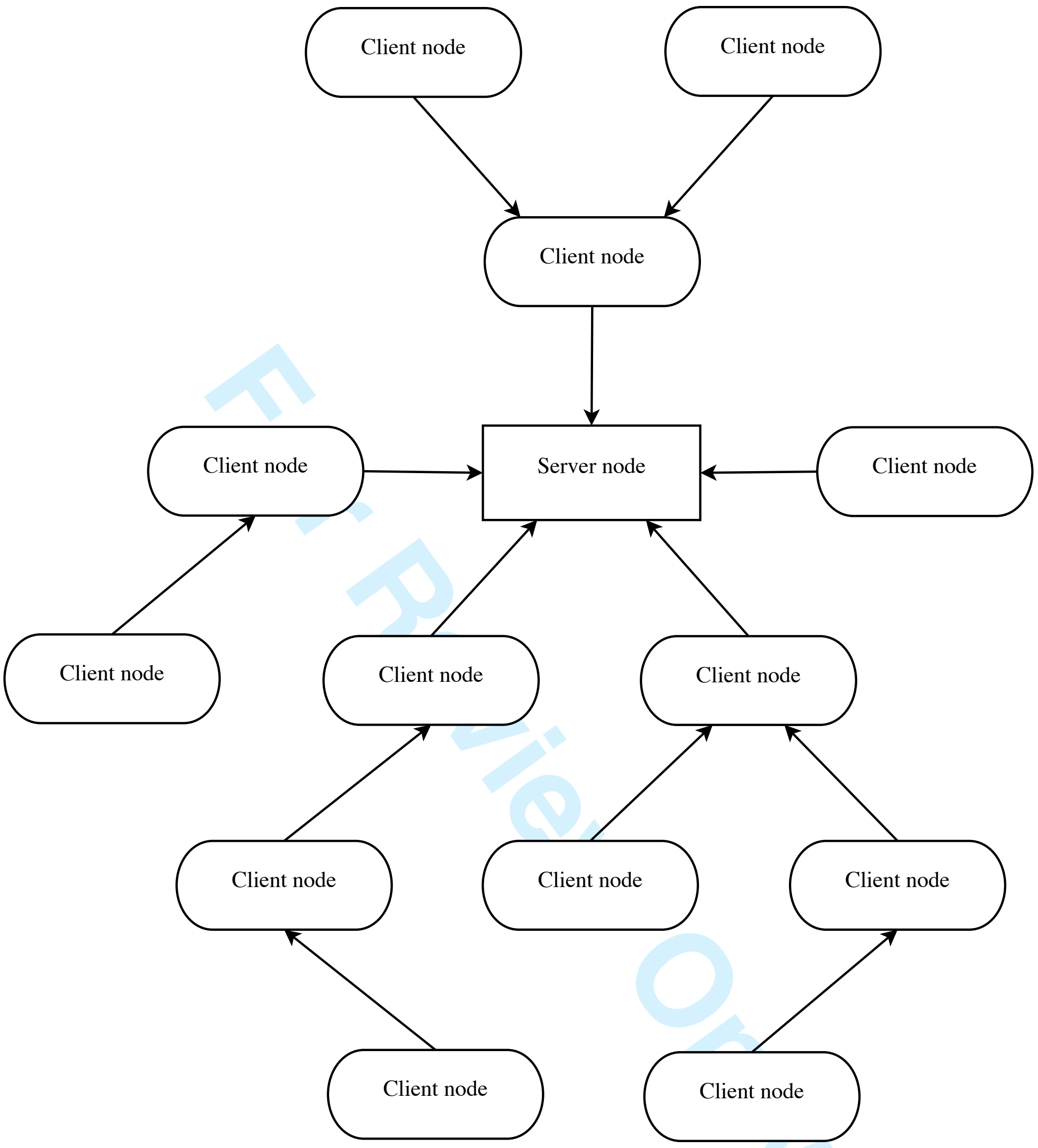

Figure 1: Network structure.

3 Categorization process

Categorization is defined as "*the act of distributing things into classes or categories of the same type*" [1]. In this paper, we divide the replicas into categories and each category contains replicas of the same type. For example, the medical replicas can be put in one category. The Grid user is usually interested in replicas of specific category and therefore storing these replicas in its storage will allow it to use them locally without the need to consume time and bandwidth because of bringing them from other nodes. Each replica must contain information about its category. So when the user receives the replica, it can use this information to add the replica to the category it belongs to.

4 CDRS strategy

As mentioned in the introduction section, Plain Caching stores a replica of the file on the storage of the requesting node, while Fast Spread strategy stores a replica of the file at each node along its path to the requesting node. However, if the nodes' storage is full and there is no space for storing the replica, a group of replicas needs to be removed from the storage in order to store the new replica. The question arises whether this group of replicas that needs to be removed is more important than the new replica. Plain Caching and Fast Spread strategy do not take this into account and replaces that group with the new replica even if it is more important than the new replica.

On the other hand, the CDRS takes this into consideration and replaces that group only if the new replica belongs to the category with the highest number of requests (CWHNOR). For a given node, the category with the smallest NOR is the least important category, while the category with the largest NOR is the most important one and called CWHNOR.

In this strategy, each node stores the NOR of the different categories. The node's NOR for a given category is increased by one each time one of the category's replicas is requested by the node.

Pseudocode 1 shows the pseudocode of this strategy. For definitions of variables used in the pseudocode, refer to Table 1.

When a node requests an existing replica, it just uses it. However, if the replica does not exist on that node, it starts searching for it on every node on the predetermined path starting from the node that comes directly after the requesting node on the predetermined path (RN + 1) and ending by the main server.

When the requested replica is found on one of the nodes on the predetermined path, it is brought backward to the requesting node. CDRS does not necessarily copy that replica to every node it visits when it is brought backward. It is copied to the visited node in two cases. The first case is if the visited node has enough free storage space (FSS) to store the requested replica. The second case is if the node's FSS is less than the size of the requested replica, and this replica belongs to the node's CWHNOR. In this case, the strategy starts deleting the replicas that belong to the category with smallest NOR from the largest replica to the smallest replica. The largest replicas are deleted first because they occupy more storage space. If the category gets empty (all of its replicas were deleted) and FSS is still less than the size of the requested replica, the strategy starts deleting the replicas that belong to the category with the second smallest NOR and so on. The deletion process continues until FSS is greater than or equal to the size of the requested replica. If the requested replica does not belong to node's CWHNOR and the node's FSS is less than the size of the requested replica, it will not be stored on the node's storage.

By storing only the NOR of the replica categories instead of the NOR of the replicas reside on the node, the consumed space to store the NOR will be less and the search time will be reduced significantly. For example, suppose there are 10 categories and 1000 replicas belong to each category. In this case, the strategy stores the NOR for 10 categories only. Additionally, if it needs to find the category with the smallest NOR in order to start deleting the replicas that belong to it, it has only to search 10 values (NOR) only. However, if this strategy is replica-based then it needs to store and search 10000 (10×1000) values (NOR of all replicas).

Table 1: Definitions of pseudocode' variables of CDRS.

Variable	Definition
RR	Requested replica.
RN	Requesting node.
CN	Checked node.
FSS	Free storage space.
NOR	Number of requests.
$NSPList$	The list that contains the nodes on the predetermined path (e.g. shortest path) from RN to the main server.
$CategoryList$	The list that contains the node's categories sorted in inreasing order based on their NOR.
$CWHNOR$	The category with highest number of requests.
$ReplicaList$	The list that contains the replicas that belong to a category sorted in decreasing order based on their sizes.

Pseudocode 1: CDRS' pseudocode

```
1  if RR exists on RN then
2      Use $RR$;
3  else
4      for $i = 2$ to $NSPList.size$ do
5          if RR exists on NSPList(i) then
6              for $j = NSPList(i-1)$ to 1 do
7                  if $CN.FSS \geq RR.Size$ then
8                      Copy $RR$;
9                      $CN.FSS = CN.FSS - RR.Size$;
10                 else
11                     if $RR.Category = CN.CWHNOR$ then
12                         for $x = 1$ to $CN.CategoryList.size$ do
13                             for $y = 1$ to $CN.CategoryList(x).ReplicaList.size$ do
14                                 Delete $CN.CategoryList(x).ReplicaList(y)$;
15                                 if $CN.FSS \geq RR.Size$ then
16                                     Copy $RR$;
17                                     $CN.FSS = CN.FSS - RR.Size$;
18                                     Goto 6;
19 $RN.Categorylist(RR.Category).NOR = RN.Categorylist(RR.Category).NOR + 1$;
20 if $RR.Category \neq RN.CWHNOR$ then
21     if $RN.Categorylist(RR.Category).NOR >$
       $RN.CategoryList(RN.CWHNOR).NOR$ then
22         $RN.CWHNOR = RR.Category$;
```

5 Comparison metrics

In the current work, there are n nodes $N_1, N_2, \cdots, N_n$, m categories $C_1, C_2, \cdots, C_m$ and w replicas $R_1, R_2, \cdots, R_w$. Each category contains a set of replicas.

In order to determine the best performing strategy, a suitable set of metrics need to be defined. Two metrics are used to measure the performance of different strategies: total transit time and total bandwidth consumption. Both metrics need to be minimized. Transit time is the elapsed time between sending the requested replica and receiving it. The sum of all transit times for the duration of the simulation is calculated. Bandwidth consumption is the bandwidth consumed for replica transfers occur when a node requests a replica that does not exist on it. The sum of all bandwidth consumptions for the duration of the simulation is calculated. If the requested replica exists on the requesting node, both transit time and bandwidth consumption are considered zero.

Table 2 shows the metrics used to measure the performance of strategies. Since the calculated values of total transit times and total bandwidth consumptions through the duration of the simulation are expected to be so large, the constant C is used to minimize their values. In the simulations, the value of C is set to 0.001.

Table 2: Metrics of strategies.

$M_1 = TTT \times C$	TTT = Total transit time, and C is a constant.
$M_2 = TBC \times C$	TBC = Total bandwidth consumption, and C is a constant.

6 Simulation setup

In this paper, an event-driven simulator written in Java is used for evaluating three different replication strategies which are Plain Caching, Fast Spread, and our new proposed strategy named CDRS. Both Plain Caching and Fast Spread use least recently used (LRU) algorithm to select the replicas that need to be replaced in case the node's free storage is not enough to hold the new replica. LRU [12] discards the least recently used replicas first. The performance of these strategies is measured under three different scenarios. In all scenarios, there is a category named the most wanted category (MWC) that contains 10% of the total number of replicas. Each node has its own MWC. The probability of making a request for replicas in MWC is higher than the probabiltiy of making a request for the rest of replicas. The probability of requesting a replica in MWC is 30% in the first scenario, 50% in the second scenario, and 70% in the third scenario. The selected scenarios have better representation for the real situation where some information (replicas) are requested more than the others. For instance, the probability of requesting an account information for a customer who lives in the same area of a bank branch is higher than the probability of requesting an account information for a customer who lives in a different area. For this bank branch, the account information for same area customers can be considered the MWC.

In each simulation, all the nodes are set to employ one of the strategies under a specific scenario. For each strategy, the simulation is run three times; one for each of the scenarios. Since three strategies are evaluated under three scenarios; the total number of simulation runs is nine. Under each scenario, the performance of CDRS strategy is compared with the performance of the other two strategies. In all simulations, the used network topology is a complete graph. The predetermined path between a Client node and a Server node is set to be the shortest path between them.

The inter-arrival times of nodes' requests and replicas' sizes follow uniform distribution. The communication latency (CL) between two directly connected nodes is calculated as follows:

$$CL = \frac{FS}{NB} + \frac{D}{PS}$$

Where FS is the frame size, NB is the network bandwidth, D is the distance between the two directly connected nodes, and PS is the propagation speed.

Since total transit time and total bandwidth consumption are the only metrics measured in the simulations, it is assumed that the time required for replacing a group of replicas and storing the new replica is negligible (equal to zero).

Table 3 shows the simulation parameters and their values.

7 Simulation results and discussion

In this section, the performance of various strategies is measured under the three different scenarios mentioned in the previous section. Figs. 2 and 3 show the total transit time and total bandwidth consumption, respectively, achieved by various replication strategies under the three different scenarios.

In Scenario one, the performance of three replication strategies is evaluated under the assumption that the probability of making a request for replicas in MWC is 30%, while the probabiltiy of making a request for the rest of replicas is 70%.

Table 3: Simulation parameters.

Parameter	Value
Number of nodes	100
Number of replicas	1000
Number of categories	10
Number of replicas that belong to each category	100
Length of each replica	Between 100 and 1000 Megabit
Number of generated requests	1000000
The inter-arrival times of nodes requests	Between 0 and 99
Storage space for every client node	50000 Megabit
Storage space for server node	So large so it can accommodate all the replicas on the Data Grid.
Network bandwidth	10 Mbps
Distance between every two directly connected nodes	Between 1 and 1000 Kilometer
Propagation speed	2×10^5 Kmps
C	0.001

From scenario one of Figs. 2 and 3, it can be seen that CDRS achieved a better performance than the performance achieved by Plain Caching and Fast Spread. This is because, when the node's storage is full, the CDRS strategy replaces a group of replica with the requested replica only if the requested replica belongs to the CWHNOR which will have the highest number of requests. As a result, this replica is expected to be requested more than the replaced group of replicas.

In scenario two, it is assumed that the probability of making a request for replicas in MWC is 50%, while the probabiltiy of making a request for the rest of replicas is 50%. From scenario two of Figs. 2 and 3, it appears that the CDRS achieved the best performance. Additionally, it can be seen that all the strategies achieved a better performance in this scenario. This is because of increasing the probability of requesting a replica in MWC from 30% to 50%. Hence, the possibility of serving the requests locally is higher in this scenario.

In scenario three, the probability of making a request for replicas in MWC is 70%, while the probabiltiy of making a request for the rest of replicas is 30%. Scenario three of Figs. 2 and 3 shows that CDRS achieved the best performance in this scenario too. This scenario shows that there is a proportional relation between the probability of MWC and the performance of the three strategies.

In all scenarios, it can be seen that Plain Caching strategy has the worst performance because it only stores a replica of the file on the storage of the requesting node. However, Fast Spread strategy achieved a better performance than Plain Caching strategy because it stores a replica of the file at each node along its path to the requesting node and not just at the requesting node as in Plain Caching strategy.

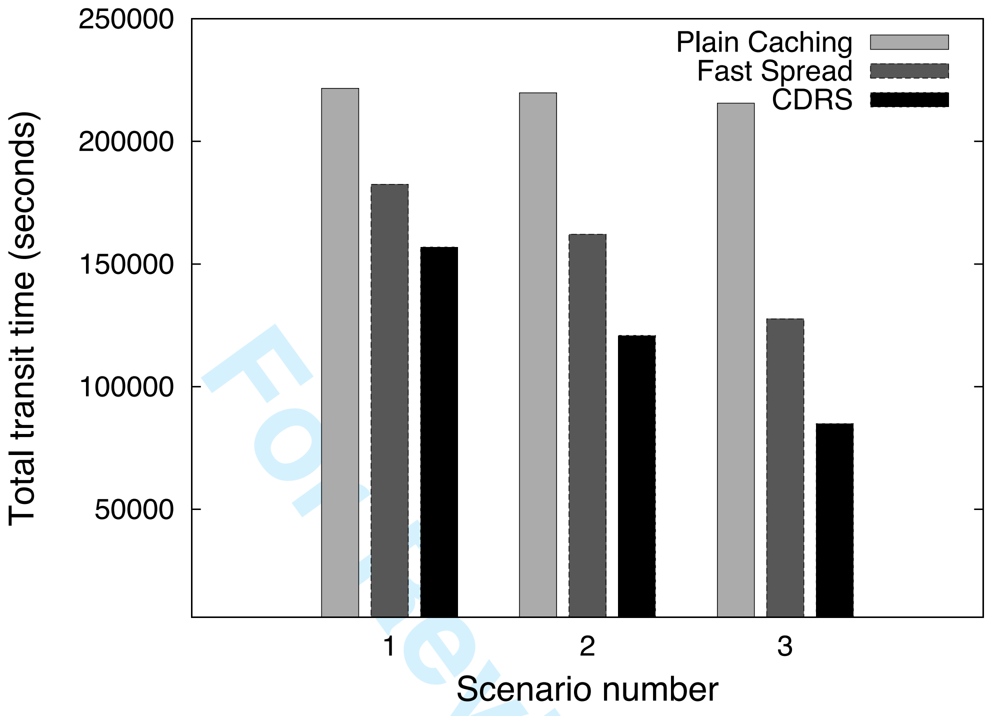

Figure 2: Total transit time achieved by three replication strategies under three different scenarios.

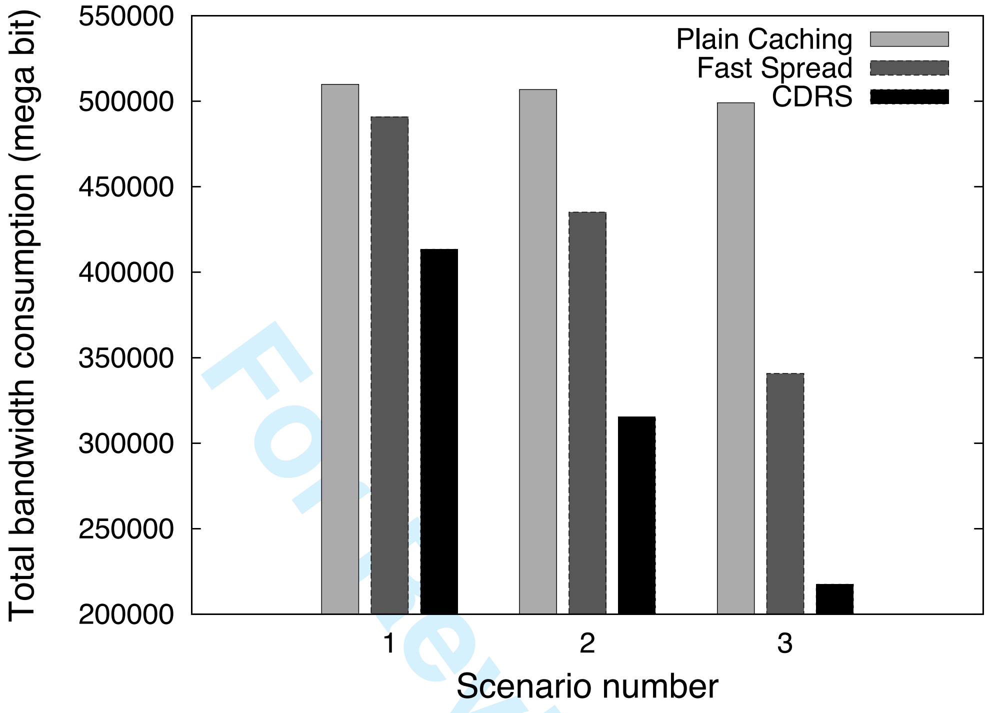

Figure 3: Total bandwidth consumption achieved by three replication strategies under three different scenarios.

Table 4 shows the CDRS' percentage decrease in total transit time, while Table 5 shows the CDRS' percentage decrease in total bandwidth consumption.

8 Conclusion

In this paper, a new replication strategy named CDRS has been presented. The performance of this strategy has been compared with Plain Caching and Fast Spread by event-driven simulations under different scenarios. The evaluation shows that CDRS achieved superiority over Plain Caching and Fast Spread in terms of total transit time and total bandwidth consumption.

Table 4: CDRS' percentage decrease in total transit time.

Compared with Plain Caching	
Scenario1	29.211%
Scenario2	45.006%
Scenario3	60.623%
Compared with Fast Spread	
Scenario1	14.014%
Scenario2	25.431%
Scenario3	33.470%

Table 5: CDRS' percentage decrease in total bandwidth consumption.

Compared with Plain Caching	
Scenario1	18.904%
Scenario2	37.777%
Scenario3	56.416%
Compared with Fast Spread	
Scenario1	15.771%
Scenario2	27.516%
Scenario3	36.168%

In future work, there are two main areas of consideration: considering more factors to determine the importance of different categories, and undertaking further experimental investigations.

References

[1] Categorization. `http://www.thefreedictionary.com/categorization`. Accessed: 2013-07-02

[2] Bsoul, M., Al-Khasawneh, A., Abdallah, E.E., Kilani, Y.: Enhanced fast spread replication strategy for data grid. Journal of Network and Computer Applications **34**(2), 575–580 (2011)

[3] Bsoul, M., Al-Khasawneh, A., Kilani, Y., Obeidat, I.: A threshold-based dynamic data replication strategy. J. Supercomput. **60**(3), 301–310 (2012)

[4] Cameron, D.G., Millar, A.P., Nicholson, C., Carvajal-Schiaffino, R., Stockinger, K., Zini, F.: Analysis of scheduling and replica optimisation strategies for data grids using optorsim. Journal of Grid Computing **2**(1), 57–69 (2004)

[5] Chang, R.S., Chang, H.P.: A dynamic data replication strategy using access-weights in data grids. J. Supercomput. **45**(3), 277–295 (2008)

[6] Chang, R.S., Chang, H.P., Wang, Y.T.: A dynamic weighted data replication strategy in data grids. In: AICCSA '08: Proceedings of the 2008 IEEE/ACS International Conference on Computer Systems and Applications, pp. 414–421. IEEE Computer Society, Washington, DC, USA (2008). DOI http://dx.doi.org/10.1109/AICCSA.2008.4493567

[7] Cibej, U., Slivnik, B., Robic, B.: The complexity of static data replication in data grids. Parallel Computing **31**(8), 900–912 (2005)

[8] Dong, X., Li, J., Wu, Z., Zhang, D., Xu, J.: On dynamic replication strategies in data service grids. In: ISORC '08: Proceedings of the 2008 11th IEEE Symposium on Object Oriented Real-Time Distributed Computing, pp. 155–161 (2008)

[9] Figueira, S., Trieu, T.: Data Replication and the Storage Capacity of Data Grids, pp. 567–575 (2008)

[10] Hong, L., Xue-dong, Q., Xia, L., Zhen, L., Wen-xing, W.: Fast cascading replication strategy for data grid. In: CSSE '08: Proceedings of the 2008 International Conference on Computer Science and Software Engineering, pp. 186–189. IEEE Computer Society, Washington, DC, USA (2008). DOI http://dx.doi.org/10.1109/CSSE.2008.624

[11] Horri, A., Sepahvand, R., Dastghaibyfard, G.: A hierarchical scheduling and replication strategy. IJCSNS International Journal of Computer Science and Network Security **8**(8) (2008)

[12] J.O'Neil, E., E.O'Neil, P., Weikum, G.: The LRU-K page replacement algorithm for database disk buffering. In: Proceedings of the 1993 ACM SIGMOD international conference on Management of data, pp. 297–306. ACM (1993)

[13] Lamehamedi, H., Szymanski, B., Shentu, Z., Deelman, E.: Data replication strategies in grid environments. In: Proceedings of the Fifth International Conference on Algorithms and Architectures for Parallel Processing (ICA3PP02, pp. 378–383. Press (2002)

[14] Mansouri, N., Dastghaibyfard, G.: A dynamic replica management strategy in data grid. Journal of Network and Computer Applications **35**(4), 1297–1303 (2012)

[15] Mavromoustakis, C.X., Karatza, H.D.: Performance evaluation of opportunistic resource-sharing scheme using socially oriented outsourcing in wireless devices. Comput. J. **56**(2), 184–197 (2013)

[16] Park, S., Kim, J., Ko, Y., Yoon, W.: Dynamic data grid replication strategy based on internet hierarchy. In: In Second International Workshop on Grid and Cooperative Computing (GCC2003, pp. 838–846 (2003)

[17] Ranganathan, K., Foster, I.: Design and evaluation of dynamic replication strategies for a high-performance data grid. In: International Conference on Computing in High Energy and Nuclear Physics. Beijing, China (2001)

[18] Ranganathan, K., Foster, I.T.: Identifying dynamic replication strategies for a high-performance data grid. In: GRID '01: Proceedings of the Second International Workshop on Grid Computing, pp. 75–86. Springer-Verlag, London, UK (2001)

[19] Rasool, Q., Li, J., Oreku, G.S., Munir, E.U.: Fair-share replication in data grid. Information Technology Journal **7**(5), 776–782 (2008)

[20] Sashi, K., Thanamani, A.: Dynamic replication in a data grid using a modified BHR region based algorithm. Future Gener. Comput. Syst. **27**(2), 202–210 (2011)

[21] Shorfuzzaman, M., Graham, P., Eskicioglu, R.: Adaptive popularity-driven replica placement in hierarchical data grids. J. Supercomput. **51**(3), 374–392 (2010)

[22] Tang, M., Lee, B.S., Yeo, C.K., Tang, X.: Dynamic replication algorithms for the multi-tier data grid. Future Generation Computer Systems **21**(5), 775–790 (2005). DOI DOI:10.1016/j.future.2004.08.001

[23] Wang, Z., Li, T., Xiong, N., Pan, Y.: A novel dynamic network data replication scheme based on historical access record and proactive deletion. J. Supercomput. **62**(1), 227–250 (2012)

[24] Wu, J.J., Lin, Y.F., Liu, P.: Optimal replica placement in hierarchical data grids with locality assurance. J. Parallel Distrib. Comput. **68**(12), 1517–1538 (2008)

[25] Zhang, J., Lee, B.S., Tang, X., Yeo, C.K.: Improving job scheduling performance with parallel access to replicas in data grid environment. J. Supercomput. **56**(3), 245–269 (2011)

[26] Zhao, W., Melliar-Smith, P.M., Moser, L.E.: Low latency fault tolerance system. Comput. J. **56**(6), 716–740 (2013)

[27] Zhao, W., Xu, X., Xiong, N., Wang, Z.: A weight-based dynamic replica replacement strategy in data grids. In: APSCC '08: Proceedings of the 2008 IEEE Asia-Pacific Services Computing Conference, pp. 1544–1549. IEEE Computer Society, Washington, DC, USA (2008). DOI http://dx.doi.org/10.1109/APSCC.2008.41