J Supercomput (2012) 60:301–310
DOI 10.1007/s11227-010-0466-3

A threshold-based dynamic data replication strategy

Mohammad Bsoul · Ahmad Al-Khasawneh · Yousef Kilani · Ibrahim Obeidat

Published online: 13 August 2010

Abstract Data replication is the creation and maintenance of multiple copies of the same data. Replication is used in Data Grid to enhance data availability and fault tolerance. One of the main objectives of replication strategies is reducing response time and bandwidth consumption. In this paper, a dynamic replication strategy that is based on Fast Spread but superior to it in terms of total response time and total bandwidth consumption is proposed. This is achieved by storing only the important replicas on the storage of the node. The main idea of this strategy is using a threshold to determine if the requested replica needs to be copied to the node. The simulation results show that the proposed strategy achieved better performance compared with Fast Spread with Least Recently Used (LRU), and Fast Spread with Least Frequently Used (LFU).

Keywords Data grid · Replication strategy · Fast spread · Least recently used · Least frequently used · Simulation

1 Introduction

A Data Grid consists of a collection of geographically distributed computer and storage resources located in different places, and enables users to share data and other resources [1, 6, 11].

M. Bsoul (✉)
Department of Computer Science and Applications, The Hashemite University, P.O. Box 150459, Zarqa 13115, Jordan
e-mail: mbsoul@hu.edu.jo

A. Al-Khasawneh · Y. Kilani · I. Obeidat
Department of Computer Information System, The Hashemite University, P.O. Box 150459, Zarqa 13115, Jordan

In the Data Grid, a user located somewhere may need to run an intensive job on a large data set. This user may choose to get the data from where it exists to the local computing resource and run the job there. On the other hand, it may be better to transfer the job to where the data exists, or both the job specification and the data may be sent to a third location that will perform the computation and return the results to the user.

When a user requests a file, large amount of bandwidth could be spent to send the file from the server to the client. Moreover, the delay involved could be high. Thus, it could be beneficial to create replicas of the same file at different locations. The main goals of using replication are to reduce access delay and bandwidth consumption [13]. There are two kinds of replication: static and dynamic replication. Dynamic replication [2, 5, 7, 12, 14, 15, 17, 18] has an advantage over static replication [4] because it can adapt to changes in user behavior. Some of the well-known dynamic replication strategies that have already been implemented are No Replication or Caching, Best Client, Cascading Replication, Plain Caching, Caching plus Cascading Replication, and Fast Spread [3, 8, 10].

Fast Spread is one of the best replication strategies especially for random request patterns [13]. In this strategy, which is our main concern in this paper, a replica of the file is stored at each node along its path to the user. If the storage of one of these nodes is full, a group of replicas (that contains one or more replicas) needs to be deleted in order to store the new replica. The problem is if this group of existing replicas is more important than the new replica. Even in this case, this group must be deleted. The aim of this paper is to propose a new strategy that is based on Fast Spread but better than it. This is achieved by using a dynamic threshold that determines if the replica should be stored at each node along its path to the user. If the decision taken based on the threshold value is storing the new replica, then the new replica is considered more important (achieved higher number of requests by its node) than the group of replaced replicas. This new strategy is named Modified Fast Spread (MFS) strategy.

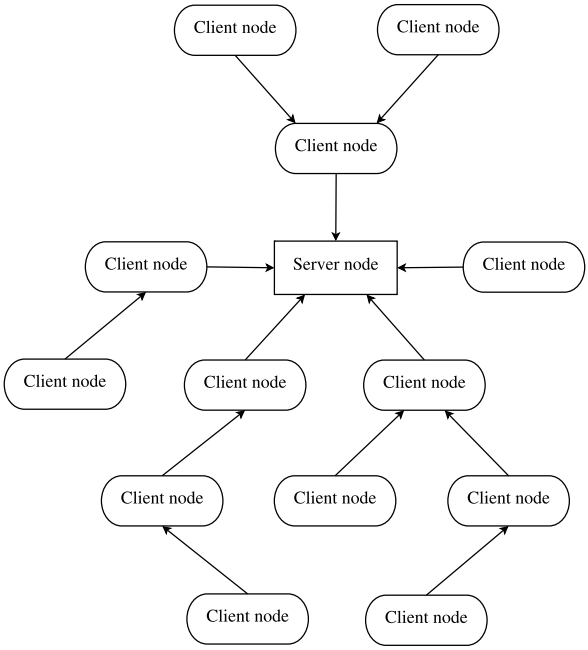
The rest of this paper is organized as follows. Section 2 gives a description of the employed network structure. Section 3 presents the new proposed strategy. Section 4 describes the metrics for measuring the strategies' performance. Section 5 explains how the simulation is configured. Section 6 discusses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

2 Network structure

The employed network structure consists of a Server node and a number of Client nodes that interact with each other as appears in Fig. 1. The Server node is the node with the main storage which contains all the data that belongs to the Data Grid. On the other hand, Client nodes are the nodes that create the requests. Each of these nodes includes a storage that is relatively small if compared with the Server node's storage and cannot hold all the requested replicas. Thus, some of the requested replicas are brought from the other nodes.

In this structure, there is a 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

Fig. 1 Network structure



there, it just uses it. Otherwise, it keeps looking for it on every node on the shortest path. When it finds it, it brings it backward to itself so it can use it.

3 MFS strategy

As mentioned in Sect. 1, Fast Spread strategy stores a replica of the file at each node along its path to the user. The problem is if the nodes' storage is full and there is no space for storing the replica. In this case, a group of replicas needs to be removed from the storage in order to store the new replica. But, what if that group of replicas that needs to be removed is more important than the new replica? Fast Spread strategy does 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 MFS takes this into consideration and replaces that group only if it is less important than the new replica. For a given node, the replica with the smallest number of requests (NOR) is the least important replica, while the replica with the largest NOR is the most important one. If two or more replicas have the same NOR, the replica with the smallest size is considered the most important one. This is because it occupies less space.

In this strategy, each node stores the NOR of each replica that resides on it. The node's NOR for a given replica is increased by one each time it is requested by it.

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

```
1  Initialise SOS to 0;
2  if RR exists on RN then
3  |  Use RR;
4  else
5  |  for i = 2 to NSPList.size do
6  |  |  if RR exists on NSPList(i) then
7  |  |  |  for j = NSPList(i − 1) to 1 do
8  |  |  |  |  if CNFSS ≥ RR.Size then
9  |  |  |  |  |  Copy RR;
10 |  |  |  |  else
11 |  |  |  |  |  PNOR = NOR × (RR.Size − CNFSS) / RR.Size;
12 |  |  |  |  |  for x = 1 to ReplicaList.size do
13 |  |  |  |  |  |  if SOS < RR.Size − CNFSS then
14 |  |  |  |  |  |  |  SOS = SOS + SizeList(x);
15 |  |  |  |  |  |  else
16 |  |  |  |  |  |  |  Break;
17 |  |  |  |  |  |  end
18 |  |  |  |  |  end
19 |  |  |  |  |  if Σ_{y=1}^{x−1} NORList(y) < PNOR then
20 |  |  |  |  |  |  for y = 1 to x − 1 do
21 |  |  |  |  |  |  |  Delete ReplicaList(y), SizeList(y), NORList(y);
22 |  |  |  |  |  |  end
23 |  |  |  |  |  |  Copy RR;
24 |  |  |  |  |  end
25 |  |  |  |  end
26 |  |  |  end
27 |  |  end
28 |  end
29 end
```

Pseudocode 1 MFS strategy

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 shortest path from RN + 1 to the main server, where RN + 1 is the requesting node's successive node on the shortest path.

When the requested replica is found on one of the nodes on the shortest path, it is brought backward to the requesting node. In the original Fast Spread replication strategy, that replica is copied to every node it visits when it is brought backward to the requesting node. In contrast to Fast Spread, MFS 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 sufficient free storage space to store the requested replica. The second case is if the node's free storage space is less than the size of the requested replica, and this replica was found more important than a group of existing replicas that their sizes are greater than or equal to the size still needed to make the node's storage able to store it. In this case, that group of replicas is replaced with this replica. The requested replica is considered more important than that group of replicas if its partial number of requests (PNOR) is greater than the NOR of that group. PNOR equals NOR in case the checked node's free storage space (CNFSS) is equal to zero. PNOR calculation is shown in Pseudocode 1 line 11.

Table 1 Definitions of pseudocode variables of MFS strategy

Variable	Definition
RR	Requested replica.
RN	Requesting node.
CNFSS	Checked node's free storage space.
NOR	Number of requests of RR.
PNOR	Partial number of requests of RR.
SOS	The variable that contains the sum of sizes of a group of replicas on the checked node.
NSPList	The list that contains the nodes on the shortest path from *RN* to the main server.
ReplicaList	The list that contains the existing replicas on the checked node sorted in increasing order based on their NOR. If two or more replicas have the same NOR, these replicas are sorted in decreasing order based on their sizes. If they also have the same size, they are sorted randomly.
SizeList	The list that contains the sizes of the corresponding replicas in *ReplicaList*.
NORList	The list that contains how many times each replica in *ReplicaList* has been requested by the checked node.

Table 2 Metrics of strategies

$M_1 = TRT \times C1$	$TRT =$ Total response time, and $C1$ is a constant.
$M_2 = TBC \times C2$	$TBC =$ Total bandwidth consumption, and $C2$ is a constant.

4 Comparison metrics

In the current work, there are n nodes $N_1, N_2, \ldots, N_n$, m groups $G_1, G_2, \ldots, G_m$ and w replicas $R_1, R_2, \ldots, R_w$. Each group 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 response time and total bandwidth consumption. Both metrics need to be minimized. Response time is the elapsed time between sending a request for a replica and receiving the requested replica. If the requested replica exists on the requesting node, the response time is considered zero. The sum of all response times for the duration of the simulation is calculated. Bandwidth consumption is the bandwidth consumed for data transfers that happen 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.

Table 2 shows the metrics used to measure the performance of strategies.

5 Simulation setup

In this paper, an event-driven simulator written in Java is used for evaluating three different replication strategies which are Fast Spread with LRU [9], Fast Spread with LFU [16], and our proposed strategy named MFS. Fast Spread with LRU discards the least recently used replicas first, while Fast Spread with LFU discards the replicas

Table 3 Simulation parameters

Parameter	Value
Number of nodes	20
Number of replicas	1000
Length of each replica	Between 100 and 1000 Megabit
Number of generated requests	100,000
The inter-arrival times of nodes' requests	Between 0 and 99
Number of groups	10
Storage space for every client node	50,000 Megabit
Storage space for server node	So large so it can hold all the replicas on the Data Grid
Number of replicas within each group	100

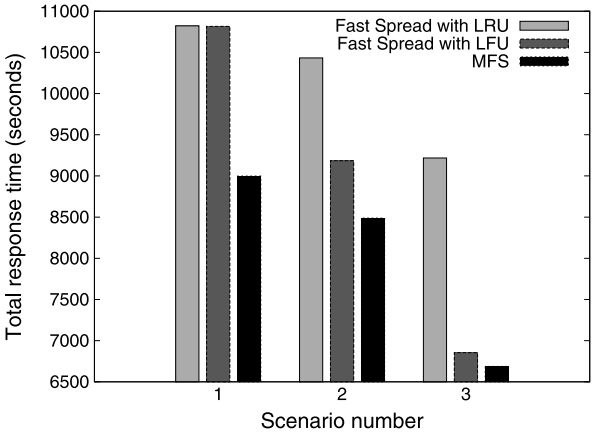


Fig. 2 Total response time achieved by three replication strategies under three different scenarios

that are used the least first. The performance of these strategies is measured under three different scenarios. In the first scenario, the probability of requesting any of the replicas is the same. In the second and third scenarios, there is a group named the most wanted group (MWG) that contains 10% of the total number of replicas. Each node has its own MWG. The probability of making a request for replicas in MWG is higher than the probability of making a request for the rest of replicas. The probability of requesting a replica in MWG is 30% in the second scenario, while it is 50% in the third scenario. The inter-arrival times of nodes' requests and replicas' lengths follow uniform distribution. In each simulation, all the nodes are set to employ one of the strategies under a specific scenario.

Table 3 shows the simulation parameters and their values.

6 Simulation results and discussion

In this section, the performance of various strategies is measured under the three different scenarios mentioned in the previous section. Figures 2 and 3 show the total

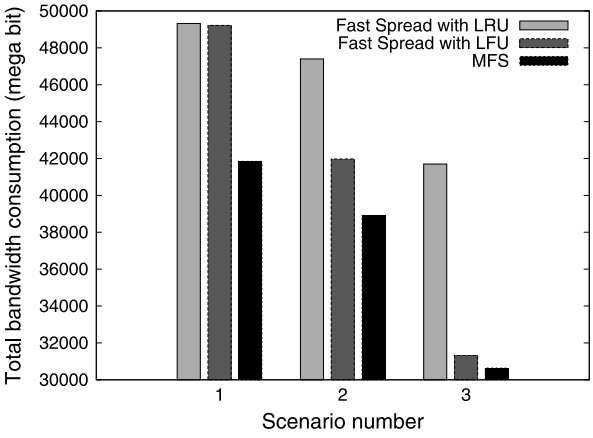


Fig. 3 Total bandwidth consumption achieved by three replication strategies under three different scenarios.

response time and total bandwidth consumption, respectively, achieved by various replication strategies under different scenarios.

6.1 Scenario one

In this experiment, the performance of three replication strategies is evaluated under the assumption that the probability of requesting any of the replicas is the same. From scenario one of Figs. 2 and 3 it can be seen that the MFS strategy is the superior one. This is because, when the node's storage is full, the MFS strategy only replaces the old group of replicas with the new one if it is found that it is more important than that group in terms of NOR. In the other two strategies, when the storage is full, a group of existing replicas must be replaced with the new replica even if it is less important than the replaced group of replicas in terms of NOR. Thus, any replica in the replaced group might be requested in the future and in this case the node must bring it from another node which takes time and consumes the valuable bandwidth.

6.2 Scenario two

It is assumed in this experiment that the probability of making a request for replicas in MWG is 30%, while the probability of making a request for the rest of replicas is 70%. From scenario two of Figs. 2 and 3 it appears that the performance of all strategies is better than their performance in the previous scenario. The reason is that most of the replicas kept in each node's storage belong to MWG, and these replicas are requested more frequently than the other replicas in other groups. As a result, the probability of finding the requested replicas locally is relatively high if compared with the previous scenario. It also appears that the improvement in the performance of Fast Spread with LFU and MFS is larger than the improvement in the performance of Fast Spread with LRU. This is because of using NOR in Fast Spread with LFU and MFS to determine the replicas that will be deleted first. Therefore, the nodes keep most of the replicas in those MWG that usually have the largest NOR. MFS also achieved the best performance in this scenario.

Table 4 MFS percentage improvement in total response time

Compared with Fast Spread with LRU
Scenario 1 16.86%
Scenario 2 18.64%
Scenario 3 27.46%
Compared with Fast Spread with LFU
Scenario 1 16.81%
Scenario 2 07.58%
Scenario 3 02.44%

Table 5 MFS percentage improvement in total bandwidth consumption

Compared with Fast Spread with LRU
Scenario 1 15.16%
Scenario 2 17.90%
Scenario 3 26.54%
Compared with Fast Spread with LFU
Scenario 1 14.98%
Scenario 2 07.30%
Scenario 3 02.19%

6.3 Scenario three

This experiment differs from the previous experiment in the value of probability. In this experiment, the probability of making a request for replicas in MWG is 50%, while the probability of making a request for the rest of replicas is 50%. Scenario three of Figs. 2 and 3 shows that all the strategies achieved the best performance in this scenario. This is because of increasing the probability of requesting a replica in MWG from 30 to 50%. Hence, the possibility of serving the requests locally is higher in this scenario. MFS keeps it superiority in this scenario, too.

It is clearly seen in the above figures that MFS is the best strategy under all scenarios in terms of total response time and total bandwidth consumption. The reason is that when the node's storage is full and a request to non-existing replica is made, the MFS strategy only replaces a group of existing replicas with this replica if it is more important (has larger NOR) than that group. In case of replacement, the possibility of requesting this replica is higher than the possibility of requesting the replicas in the replaced group, thus reducing the total response time and total bandwidth consumption. When the requested replica is found on the node's storage, the response time and bandwidth consumption for this request are both zero.

Table 4 shows the MFS percentage improvement in total response time, while Table 5 shows the MFS percentage improvement in total bandwidth consumption.

7 Conclusion

In this paper, a new replication strategy named MFS has been presented. The performance of this strategy has been compared with Fast Spread with LRU and Fast Spread with LFU by event-driven simulations with different scenarios. The evaluation shows that the MFS strategy is the best strategy in all scenarios in terms of total response time and total bandwidth consumption.

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

References

1. Cameron DG, Millar AP, Nicholson C, Carvajal-Schiaffino R, Stockinger K, Zini F (2004) Analysis of scheduling and replica optimisation strategies for data grids using OptorSim. J Grid Comput 2(1):57–69
2. Chang RS, Chang HP (2008) A dynamic data replication strategy using access-weights in data grids. J Supercomput 45(3):277–295. http://dx.doi.org/10.1007/s11227-008-0172-6
3. Chang RS, Chang HP, Wang YT (2008) 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. IEEE Comput Soc. Washington, pp 414–421. http://dx.doi.org/10.1109/AICCSA.2008.4493567
4. Cibej U, Slivnik B, Robic B (2005) The complexity of static data replication in data grids. Parallel Comput 31(8):900–912. http://dx.doi.org/10.1016/j.parco.2005.04.010
5. Dong X, Li J, Wu Z, Zhang D, Xu J (2008) 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. IEEE Comp Soc. Washington, pp 155–161. http://dx.doi.org/10.1109/ISORC.2008.66
6. Figueira S, Trieu T (2008) Data replication and the storage capacity of data grids, Springer. Berlin, Heidelberg, pp 567–575. http://dx.doi.org/10.1007/978-3-540-92859-1_50
7. Hong L, Xue-dong Q, Xia L, Zhen L, Wen-xing W (2008) Fast cascading replication strategy for data grid. In: CSSE '08: proceedings of the 2008 international conference on computer science and software engineering. IEEE Comp Soc. Washington, pp 186–189. http://dx.doi.org/10.1109/CSSE.2008.624
8. Horri A, Sepahvand R, Dastghaibyfard G (2008) A hierarchical scheduling and replication strategy. Int J Comput Sci Netw Secur 8(8) 30–35
9. O'Neil J, O'Neil P, Weikum G (1993) The LRU-K page replacement algorithm for database disk buffering. In: Proceedings of the 1993 ACM SIGMOD international conference on management of data. ACM, New York, pp 297–306
10. Ranganathan K, Foster I (2001) 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
11. Lamehamedi H, Szymanski B, Shentu Z, Deelman E (2002) Data replication strategies in grid environments. In: Proceedings of the fifth international conference on algorithms and architectures for parallel processing, pp 378–383
12. Park S, Kim J, Ko Y, Yoon W (2003) Dynamic data grid replication strategy based on Internet hierarchy. In: Second international workshop on grid and cooperative computing, pp 838–846
13. Ranganathan K, Foster I (2001) Identifying dynamic replication strategies for a high-performance data grid. In: GRID '01: proceedings of the second international workshop on grid computing. Springer, London, pp 75–86
14. Rasool Q, Li J, Oreku GS, Munir EU (2008) Fair-share replication in data grid. Inf Technol J 7(5):776–782
15. Tang M, Lee BS, Yeo CK, Tang X (2005) Dynamic replication algorithms for the multi-tier data grid. Future Gener Comput Syst 21(5):775–790. doi: 10.1016/j.future.2004.08.001

16. Prischepa V (2004) An efficient web caching algorithm based on LFU-K replacement policy. In: Proceedings of the spring young researcher's colloquium on database and information systems. IEEE, New York, pp 23–26
17. Wu JJ, Lin YF, Liu P (2008) Optimal replica placement in hierarchical data grids with locality assurance. J Parallel Distrib Comput 68(12):1517–1538. http://dx.doi.org/10.1016/j.jpdc.2008.08.002
18. Zhao W, Xu X, Xiong N, Wang Z (2008) A weight-based dynamic replica replacement strategy in data grids. In: APSCC '08: proceedings of the 2008 ieee asia-pacific services computing conference. IEEE Comput Soc. Washington, pp 1544–1549. http://dx.doi.org/10.1109/APSCC.2008.41