A Dynamic Replication Mechanism for Data Grid Federation Systems, Based on Replica Placement Cost

Musa Sule Argungu, Suki Arif, Mohd Hasbullah Omar

Abstract: ***Data replication plays a major role in making data more available in the distributed computing systems, such as data grid federation systems. Also, it is typical of a data replication mechanism to place file replicas in storage locations that offer lower transfer time for data items, without regards to the distance between host and the requesting client. Although file transfer time is a measure of file size and bandwidth usage, which drastically affect storage usage as well as jobs completion time, it requires additional parameters in order to make replica placement more cost effective. In this write-up, a replica placement mechanism is proposed that computes optimum replica placement cost from file value, transfer time and site distance for efficient data replication in a federated data grid environment. The proposed mechanism was evaluated in OptorSim simulator and has effectively improves jobs execution time, bandwidth usage and storage usage within a federated data grid environment, compared to the existing mechanisms.***

Keywords: Data Grid, Data Grid Federation, Data Replication Mechanism, Replica Placement Cost

I. INTRODUCTION

The core goal of this write-up is to develop a dynamic replication mechanism for improving the performance of data grid federation systems, based on an optimum replica placement cost (ORPC). The proposed ORPC aims to minimise job completion time, storage usage and bandwidth consumption within a federated data grid environment. The mechanism determines an optimum replica placement cost for each region of the federation using file value, file transfer time and site distance as design metrics. DGF belongs to grid computing paradigm, and it is formed by joining more than one data grid system or computing clusters together according to research by Jagatheesan& Moore, (2004).Thus, DGF provides a means for managing different types of data grid systems connected from different locations and computing background. According to Jagatheesan & Moore, (2004), a point of difference between DG and DGF is that data grid provides the facility for naming, organizing, as well as management of data on diverse distributed computing resources, while the DGF provides a way for naming, organizing, and managing data on multiple data grids,which is achieved using a federating mechanism or software. Figure 1 is a layered architecture of grid systems showing the relationship between data grid and data grid federation.

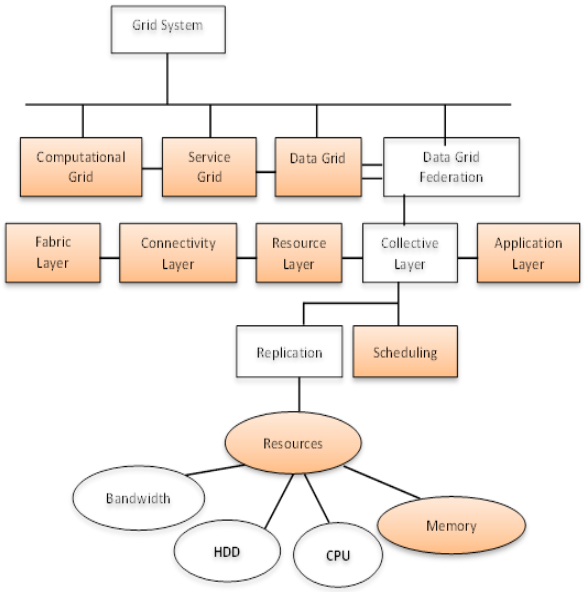

Fig. 1 The relationship between data grid and data grid federation

Figure 2 shows an abstract view of DG systems connected via a federation middleware reported by Wang, Chen, Hsu, & Huang, (2011) to form a grid federation over Peer-to-Peer wide area network (WAN) connections.

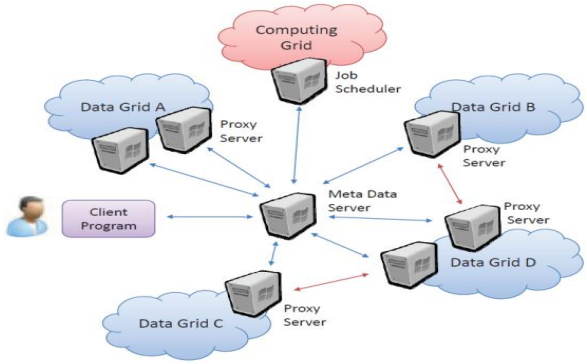

Fig. 2 A middle were architecture for federating heterogeneous data grid systems

Revised Manuscript Received on May 22, 2019.

Musa Sule Argungu, Department of Computer Science, University of Science and Technology, Aliero, Kebbi State Nigeria, InterNetWorks Research Laboratory, School of Computing, Universiti Utara Malaysia.06010Sintok, Kedah, Malaysia

Suki Arif, InterNetWorks Research Laboratory, School of Computing, Universiti Utara Malaysia.06010Sintok, Kedah, Malaysia

Mohd Hasbullah Omar, InterNetWorks Research Laboratory, School of Computing, Universiti Utara Malaysia.06010Sintok, Kedah, Malaysia

Data replication aims to improve data locality, by duplicating more copies of the most popular files and placing these replicas in suitable locations that provide minimum job completion time to the users of the grid system according to Mansouri&Asadi, (2014). Previous researchers attempted to address the issue of replica placement for improving the performance of grid systems. However, most of the researchers focused on data replication within the conventional data grid systems as observed by Abawajy&Deris, (2014), while others targeted DGF systems according to Mohamad, Ahmad, Rose, Mohamad, &Deris, (2016) and Ciubăncan&Dulea, (2017).

As mentioned earlier, this study seeks to develop a replication mechanism by computing the optimum replica placement cost for DGF systems. The proposed ORPC addresses the issues regarding file value, file transfer time and the distance between replica sites (file transfer cost),before placing new file replicas. The issue of site distance inreplica placement has been considered by Mansouri and Asadi(2014),in the conventional data grid system. Furthermore, Madi, M. K., (2012) has considered the issue of file transfer time.

However, the efforts of previous researchers failed short of addressing the overall replica placement cost, by ignoring to consider site distance and file transfer time together as vital element for replica placement cost according to Almomani&Madi, (2014). Thus, to find ideal location for placing new replica files, there is need to determine file value, file transfer time and site distance, from which the replica placement cost for each site is computed. Thereafter, determines the optimum replica placement cost from the total RPC for all the sites in the federation. The site that provides less RPC compared to the optimum RPC is selected for replica placement. Section 3 discusses how to compute RPC as well as optimum RPC for efficient replica placement decision in DGF environment. To the coverage of this research, no other mechanism has incorporated all these factors in a single mechanism for replica placement in the research domain. Before describing the proposed mechanism, the next section discusses the related literature reviewed by this research work.

II. RELATED LITERATURE

The previous section introduced the concept of data grid federation, which also distinguished data grids from DG federation based on their composition. Regarding data replication, the fundamental difference between DG and DGF systems is the way the replication mechanisms handle replica placement decision (Chang, Chang, & Wang 2008). Some mechanisms find suitable sites for replica placement on the individual regions of the DGF systems, as reported by Park, Kim, Ko, & Yoon, (2003), while others consider the entire federation system according to Ranjan, Buyya, & Harwood, (2005). Thus, ORPC considers the entire federation system for replica placement by taking account of the individual DGF regions in a stepwise progression. This section explores some of the popular research works on data replication, based on which the proposed ORPC mechanism was developed.

Research by (Madi, 2012) proposed a replication mechanism called dynamic replica creation mechanism (DRCM), for data grid systems. The mechanism aimed to minimize jobs completion times, network bandwidth consumption and storage element usage, by considering file transfer time and file sizes. However, the algorithm has not addressed the issue of site distance, which is a requirement for RPC computations.

Also, researchby (Meroufel&Belalem, 2013), proposed a replication placement scheme aimed at ensuring desired availability of data with minimum replication, without degrading system performances regarding jobs completion times, by considering sites workloads (load balancing) and response time. However, the scheme fails to consider site distance, which is a vital factor for replica placement cost.

A dynamic replica placement strategy was proposed by Mansouri & Asadi, (2014)called enhanced latest access largest weight (ELALW). The researchers maintained that replication should be used wisely due to limited storage capacity of the grid sites. Thus, it is vital to design an effective approach for replica replacement. ELALW replaces existing file replicas based on the number of future requests; the replica size; and the number of copies of the file replica. However, the existing ELALW mechanism failed to address the issue of replica placement cost by not considering site distance. Section 3 explains the solution to these shortcomings proposed by this write-up.

III. THE PROPOSED ORPC MECHANISM

The proposed ORPC contains a formula for computing replica placement cost, which is determined from file value, file transfer time and distance between replica sites. These factors are explained shortly.

Computing replica placement cost (RPC)

Replica placement cost (RPC) ensures that newly created replica files are put on site locations that provide minimum replication cost. The RPC is designed based on the work of Almomani & Madi, (2014). It encapsulates distance between sites, file transfer time and file value. Distance between sites is determined using shortest path algorithm. File transfer time determines site with shortest possible transfer time from source to destination. The PRC is computed from the following the following Equation 1.

$$RPC_{si} = \frac{\Sigma_1^n FV_{si} * FTT * D\left(Source, destination\right) file_i}{m} \quad (1)$$ Where,

$RPC_{si,}$ = Replica Placement Cost

n = total number of sites within the grid,

m = number of sites that request the replica from the given location,

FV = file value for $site_{i,}$ FTT =file transfer time and

D = the distance between source to destination site.

For replica placement, the mechanism finds the site that provides optimum replica placement cost then replicates new files on such sites.

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication

Optimum replication cost is determined from Equation 3 using Equation 2, as follows:

$$RPC_{Average} = \frac{TotalRPC_{\forall allsites}}{NumberofSites} \quad (2)$$

Where

ToTal RPC is the total cost of replica placement for all sites; NumberOfSites represents the total number of sites in the DGF system

Thus, the optimum RPC is determined from $RPC_{average}$ average as follows:

$$RPC_{Optimum} <= 50\%(RPC_{Average}) \quad (3)$$

Where

$RPC_{Optimum}$ is the replica placement cost that provides the ideal cost of replication, which is determined from 50% of the $RPC_{Average}$.

The file transfer time (FTT) and file value (FV) and are explained further in subsection 3.1.2and 3.1.3, respectively. The next subsection 3.1.1 explains how distance between replica sites is determined.

Computing site distance

The distance between replica site narrows search time, download and transfer time for data files if selected carefully, which improves on job completion time as well as minimises bandwidth consumption. From Equation 1, the RPC computes the distance between replica site and the site requesting for the file. The D(source, destination) is programmed using Java High-Level Programming Language. The source and the destination are presented to the compiler in the form of array elements.

In this write-up, site distance is determined using Dijkstra's shortest distance algorithm reported by Katre & Thakare, (2017). The Dijkstra's algorithm was modified by this write up tosuit the stated goal of finding the shortest distance between the source and destination sites, within a federated grid environment. The modification involved computation of total logical connections (TLC) within the DGF system.

The modified Dijkstra's algorithm is shown in Figure 3. The TLC is a measure if direct logical connections (DLD) and the indirect logical connections (ILD), as indicated on line (3) of the algorithm. After that, line (8) groups the sites (set of vertices) according to their respective TLC values, and assigns the site with TLC value greater than or equal to TLC_Averageas the source site.

From the pseudo-code, Graph is the set of vertices of the input graph and source is the starting site or vertex. If only the shortest path between the source site and target site needed to be found, the algorithm can be terminated to stop the search after line (17) if u = target and the rest of the algorithm is ignored.

```
1.  functionDijkstra (Graph, source);  //the function takes inputs
    graph and source site, then compute shortest distances
    between the sites;
2.  use site connectivity sample workload data file;
3.  compute TLC = DLD + ILD; //compute total logical
    connections;
4.  compute TLC_Average; for all sites
5.  TLC_Average = Σ_i^n TLC ÷ Σ_i^n Sites ;
6.  set dist[source] ← 0;  // Initialization
7.  create vertex set Q;
8.  group vertices based on TLC value;
9.   for each group
10.  set source←ILD (site_i) =>TLC_Average;
11.    for each vertex v in Graph:
12.    ifv ≠ source
13.     set dist[v] ←INFINITY //Unknown distance from source
        to v
14.   set prev[v] ← UNDEFINED   // Predecessor of v
15.  doQ.add_with_priority(v, dist[v])
16.   WhileQ is not empty  // Main loop
17.     u ← Q.extract_min() // Removes and returns best vertex
18.    for each neighbor v of u// Only v that is still in Q
19.    alt ← dist[u] + length(u, v)
20. ifalt<dist[v]
21. dist[v] ← alt
22. prev[v] ← u
23. do Q.decrease_priority(v, alt)
24.        End if
25.      End for
26.    End while
27.   End if
28.  End for
29.  End for
30.  Returndist[ ], prev[ ]
```

Fig. 3 Dijkstra's algorithm Pseudo-code for finding distances between replica sites

What follows is an illustration of how the algorithm computes the shortest distance between sites, using graph abstraction (Boneva, Kreiker, Kurb{\'a}n, Rensink, & Zambon, 2012).

Firstly, initialise the shortest distances array D, the priority queue Q, the distance from the source site to itself (D[A] = 0) is zero, and the rest of the array is ∞(infinity). Thus;

D(A) = 0;

D(B) = D(C) = D(D) = D(E) = D(F) = ∞; //Distances not yet determined

D = {0, ∞, ∞, ∞, ∞, ∞};

Q = {A(0), B(∞), C(∞), D(∞), E(∞), F(∞)};

Observe that the priority elements in the queue Qare given in parenthesis. The set of vertices vis inserted into the priority queue Q, with a priority D[v]. Figure 4Shows the graphical illustration for computing distance between sites for six sites A...F.The final phase is shown in Figure 5.

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication

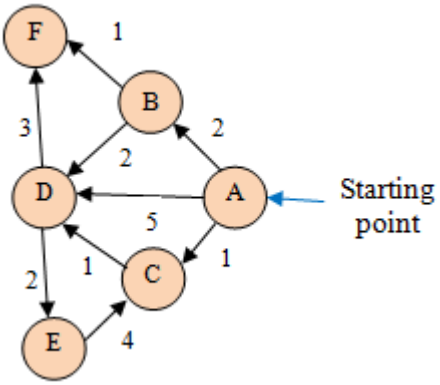

Fig. 4 Graphical illustration for computing distance between sites

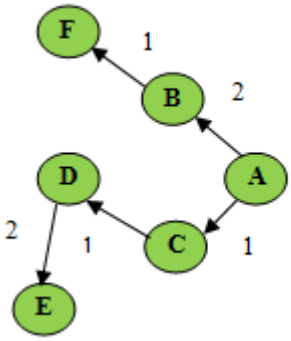

Fig. 5 The resulting shortest paths distance tree for six sites

The algorithm execution commences, by extracting A from the queue as the minimum element from the priority queue. Then A is marked as visited. In the same vein, the rest of the elements B, C...Fare processed and re-arranged into shortest path graph, after series of iterations.

Although the algorithm needs to perform series of iterations to find the shortest paths, however, its favoured by the Dijkstra's lower time complexity of O(|E|+|V|Log(|V|)), which also performs "decrease-key operation" in just O(1)amortized time. The amortized time "is the average time taken per operation," if many operations are performed at an instance according to Palúch&Majer, (2017).

Computing the transfer time. File transfer time(FTT) is determined based on the work of (Mansouri & Asadi, 2014), computed from the following formula:

$$FTT = \frac{FileSize}{Bandwidth} \qquad (4)$$

File transfer time (FTT) is proportional to FileSize variable. Thus, the lower the File Size, the lower the FTT value, which has a direct impact on jobs completion time. In other words, the higher the bandwidth, the higher the FTT value, which also influences significantly on jobs completion time.

Computing the file value. File value (FV) signifies the relevance of a data file to the DGF users and other files within the system. FV_{si}refers to file value with respect to a particular site, which is determined according to the following formula outlined by Almomani & Madi, (2014).

$$FV_{si} = \frac{FileValue}{NoR_{ni}} \qquad (5)$$

Where,

FileValue is the file value with respect to the DGF system as a whole, which is determined from Equation6 below:

$$FV\ (t, f) \leftarrow FLT\ (t, f) + FW\ (t, f) \qquad (6)$$

NoR_{si} = number of requests for the popular file from a specific site, s

File value computed from popular file evaluation step in replica creation stage (not included in this write up). So also, file life time (FLT) and file weight (FW)are part of popular file evaluation for a dynamic replica creation and eviction mechanism proposed by the authors of this write-up in another version.

IV. RESULTS AND DISCUSSION

As mentioned earlier, OptorSim is used in this research work for performance evaluation. The performance of the proposed ORPC mechanism was evaluated against the existing ELALW and DRCM mechanisms. ORPC outperforms ELALW by 30.47% and DRCM by 26.05% in Jobs Completion Time metric, regarding efficiency. In addition, the efficiency of ORPC over ELALW and DRCM regarding SE usage is 42.10% and 40.01%, respectively.

Regarding the effective network usage (ENU), the performances of both mechanisms under study are at close range. However, due to the less number of replications by ORPC, its performance over ELALW shows the efficiency of 4.55% and 2.28% over DRCM. Figures 6-8 show the simulation results for both mechanisms under review. The advantage of ORPC over the existing mechanisms is that, ORPC performs limited number of replications in comparison, which saves bandwidth usage as well as storage usage.

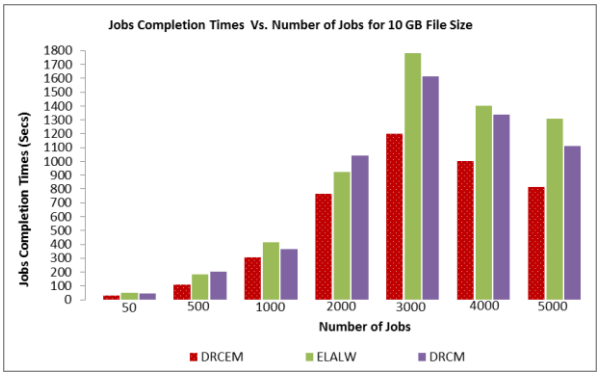

Fig. 6 Jobs times for different number of submitted jobs of 10 GB files size

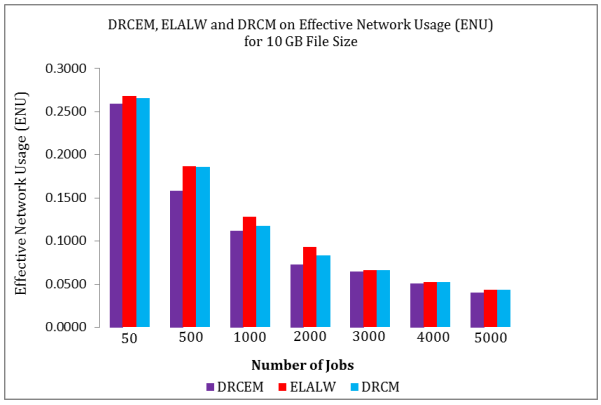

Fig. 7 ENU for different number of submitted jobs of 10 GB files size

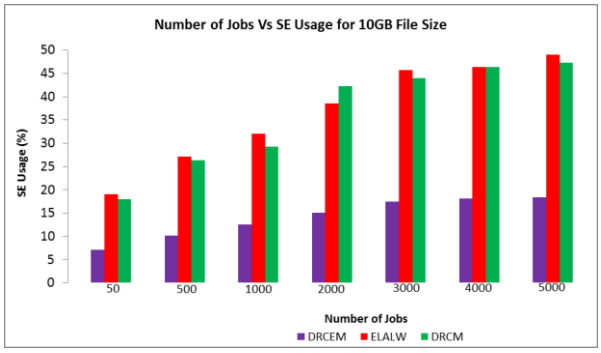

Fig. 8 SE Usage of DRPM and existing Mechanisms for 10GB files size

V. CONCLUSION AND FUTURE WORK

From the ongoing discussion, it suffices to say that ORPC is a better alternative mechanism for replica placement in DGF systems. The ORPC replication is guided by the optimum replication cost, which encapsulates file value, file transfer time and site distance. In this regard, performing fewer replications has two advantages, thus; saves storage by not performing unnecessary replications and boast jobs time; whereas in the other mechanisms, valuable time is used to perform unnecessary replications. Although replication is done to improving data availability, this needs to be with caution in order: not to constraints the users by slowing down their jobs completion times due to distance free replica placement decision, and not constraints the grid resources by creating bottlenecks as a result of high FTT. In the near future, this mechanism will be extended to include resources failures in replica placement decision. Also, it will be implemented in a federated cloud data centers.

REFERENCES

1. Abawajy, J. H., &Deris, M. M. (2014).Data replication approach with consistency guarantee for data grid. IEEE Transactions on Computers, 63(12), 2975-2987.
2. Almomani, O., &Madi, M. (2014).A GA-Based Replica Placement Mechanism for Data Grid. International Journal of Advanced Computer Science and Applications (IJACSA), 5(10).
3. Boneva, I., Rensink, A., Kurban, M. E., & Bauer, J. (2007). Graph abstraction and abstract graph transformation. CTIT Technical Report Series, (LNCS4549/TR-CTIT-07-50).
4. Chang, R. S., Chang, H. P., & Wang, Y. T. (2008, March). A dynamic weighted data replication strategy in data grids. In Computer Systems and Applications, 2008.AICCSA 2008. IEEE/ACS International Conference on (pp. 414-421). IEEE.
5. Charrada, F. B., Ounelli, H., &Chettaoui, H. (2010, November).An efficient replication strategy for dynamic data grids. In P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC), 2010 International Conference on (pp. 50-54). IEEE.
6. Ciubăncan, M., &Dulea, M. (2017, September).Implementing advanced data flow and storage management solutions within a multi-VO grid site.In Networking in Education and Research (RoEduNet), 2017 16th RoEduNet Conference (pp. 1-4).IEEE.
7. Gueye, M., Sarr, I., &Ndiaye, S. (2009, March). Database replication in large-scale systems: optimizing the number of replicas.InProceedings of the 2009 EDBT/ICDT Workshops (pp. 3-9).ACM.
8. Katre, P. R., &Thakare, A. (2017, April).A survey on shortest path algorithm for road network in emergency services. In Convergence in Technology (I2CT), 2017 2nd International Conference for (pp. 393-396). IEEE.
9. Madi, M. K. (2012). Replica Creation Algorithm for Data Grids (Doctoral dissertation, Universiti Utara Malaysia).
10. Mansouri, N., &Asadi, A. (2014).Weighted data replication strategy for data grid considering the economic approach. Int. J. Comput. Elect. Auto. Control Inf. Eng, 8, 1336-1345.
11. Meroufel, B., &Belalem, G. (2013).Managing data replication and placement based on availability. AASRI Procedia, 5, 147-155. http://dx.doi.org/ 10.1016/j.aasri.2013.10.071.
12. Mohamad, Z., Ahmad, F., Rose, A. N. M., Mohamad, F. S., &Deris, M. M. (2016).Implementation of Sub-Grid-Federation Model for Performance Improvement in Federated Data Grid. Malaysian Journal of Applied Sciences, 1(1), 55-67.
13. Moore, R. W., Jagatheesan, A., Rajasekar, A., Wan, M., & Schroeder, W. (2004, April).DATA GRID MANAGEMENT SYSTEMS.In NASA/IEEE MSST 2004 Twelfth NASA Goddard Conference on Mass Storage Systems and Technologies (p. 1).
14. Palúch, S., &Majer, T. (2017, May).Effective and fast implementation of k-shortest paths algorithms.In Carpathian Control Conference (ICCC), 2017 18th International (pp. 284-289).IEEE.
15. Park, S. M., Kim, J. H., Ko, Y. B., & Yoon, W. S. (2003, December). Dynamic data grid replication strategy based on Internet hierarchy. In International Conference on Grid and Cooperative Computing (pp. 838-846).Springer, Berlin, Heidelberg.
16. Ranjan, R., Buyya, R., & Harwood, A. (2005, July).A model for a cooperative federation of distributed clusters. In High-Performance Distributed Computing, 2005. HPDC-14.Proceedings.14th IEEE International Symposium on (pp. 295-296). IEEE.
17. Wang, C. M., Chen, H. M., Hsu, C. C., & Huang, C. C. (2011, May).Fedmi: A federation middleware for integrating heterogeneous data grids.In Parallel and Distributed Processing with Applications (ISPA), 2011 IEEE 9th International Symposium on (pp. 127-134).IEEE.