Distributed Positioning of Replica using Grass Growing Structure

Worawit Fankam-ai and Peraphon Sophatsathit

Advanced Virtual and Intelligent Computing (AVIC) Center, Department of Mathematics and Computer Science, Faculty of Science, Chulalongkorn University, Bangkok, Thailand

fworawit@gmail.com, peraphon.s@chula.ac.th

Abstract - This paper proposes a distributed data replication scheme based on Grass Growing Structure to reduce bandwidth consumption and access latency in distributed systems. The replication will be multicast to the nearest nodes using Depth Limit Search algorithm within a predefined limiting distance. Performance is measured by means of effective network usage and mean job time. The EU Data Grid Testbed is employed as the benchmarking assessment to compare the proposed approach with conventional Centralized and Flooding algorithms. The results yield less access latency, good scalability, and reliability than those comparable approaches.

Index Terms - Distributed Positioning, Replica location, Grass Growing Structure.

1. Introduction

Replication is a common method used to improve the performance of data access in distributed systems. It improves not only data access efficiency, but also data availability and fault tolerance. In order to achieve higher replication performance, there must be an efficient replica scheme to manage the replication process. Replica scheme mainly includes replication strategy and replica selection strategy to find the best-fit replica, replication consistency, and replica positioning mechanisms. Replication strategy determines when and where to create a replica, taking into account of the factors such as number of data requests, network condition, and storage availability of each replica site.

In this paper, we propose a replication positioning algorithm based on Grass Growing Structure. The focus on replica positioning mechanisms is to determine where the new replica should be located so that multicast traffic will be minimal. The algorithm is inspired by the growing of grass with adequate irrigation which will flourish more than those depleting of water. Grass areas that receive water represent distribution of data replicas, while grass trunk represents network topology. Thus, data replicas spread around starting from the initial source node. Data are replicated only via the designated routes. No superfluous distribution that occupies limited bandwidth to be wasted. Thus, performance of this algorithm will be compared with Flooding algorithm [1][5][11] and Centralized Location algorithm [1][5][11] measured by Mean Job Time and Effective Network Usage [2][3][4][5].

2. Related Work

Some principal definitions of replication location model and two related replica location algorithms [1][5][11] are described as a basis for the development of the proposed algorithm.

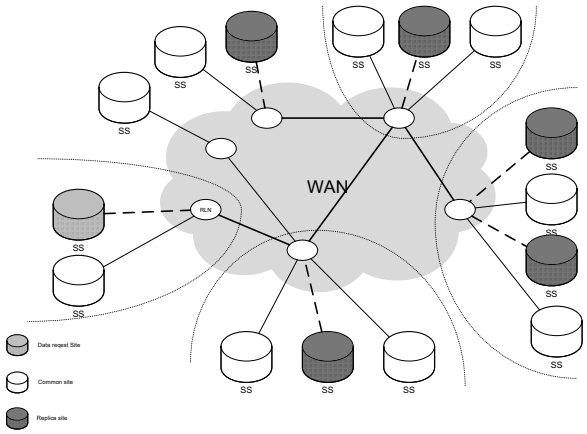

Fig. 1 Replica Location Server Model

A. Terminology

A logical file name (LFN) is a unique logical identifier for desired data content. The replica location service must identify one or more physical copies (replicas) of the logical file. Each physical copy is identified by a physical file name (PFN), which specifies its location on a storage site.

A number of storage sites (SS) collaborate to share their storage capabilities to all users. A replica location node (RLN) aggregates LFN to PFN mappings from one or more SSs and collaborates with other RLNs to build a distributed catalog of LFN mappings.

RLNs offer both a query interface to clients and a registration interface that SSs can enlist PFN to LFN mapping for files stored locally. RLNs also organize into a search network to allow remote searches. Nodes in this network distribute compressed information on the set of LFN mappings stored locally in the form of node digests.

Depth Limit Search (DLS) [8], like the normal depth-first search, is an uninformed search. It works exactly like depth-first search, but avoids the completeness drawbacks by imposing a maximum limit on the depth of the search. Even if

the search could still expand a vertex beyond that depth, DLS will not do so and thereby will not follow infinitely deep paths or get stuck in cycles. Therefore depth limited search will find a solution if it is within the depth limit, which guarantees at least completeness on all graphs.

B. Definitions

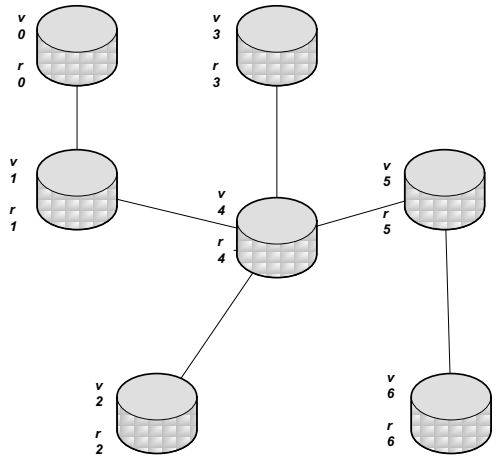

Fig 2. Centralized Location

- V is the aggregation of sites in the data grid system
- V_{SS} is the set of nodes that storage desired data(files) and their copies
- l_{vi} is the location of the node v_i
- V_{rln} is the set of sites that aggregate information about LFN to PFN mapping or some information about nodes
- Flooding algorithm. Each progress of distribution location starts from the node $v_0 \in V_{SS}$ that stores the information about the location of V_{SS}, but does not include the corresponding relationship between files and V_{SS}. That is to say, the algorithm does not know the location of V_{SS} before locating one replica r. In all the information for any node $v_i \in V_{SS}$ stored in the node v_0, the corresponding location l_{vi} is unknown. Thus, this algorithm can be costly in terms of wasted bandwidth while a message may only have one destination to be sent. Moreover, messages can duplicate in the network which increase the load on the network bandwidth. Worse yet, duplicate packets may circulate forever unless certain precautions are taken [9][10].
- Centralized Location Algorithm [1]. In Fig. 2, V_4 is only one V_{rln} which contains all the information about location of V_{SS} and LFN to PFN mappings.

3. Distributed Positioning of Replica using Grass Growing Structure

The proposed approach to position the replica will follow grass growing pattern, aka Grass Growing Structure. The area of growing grass with adequate irrigation will flourish better than the one depleting of water. Grass areas that receive water represent the position of data replicas, having grass trunks as the replica links that form the distribution topology. Fig. 3 shows an example of grass growing structure. The left figure represents grass trunk and right one is link distribution topology. If grass receives water at node 4, then node 6,7,8 will flourish better than other nodes. So the nearest nodes to the initial source node 4 are 2,6,7,8

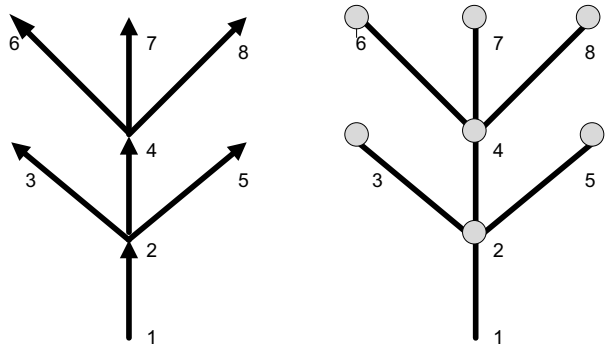

Fig. 3 Grass Growing Structure

The first stage positioning of replicas is to set up a random initial source node. Selection of neighboring nodes uses DLS algorithm [8] to find the path between source node and neighboring nodes. The procedure starts stepping each node with a vector to all directly attached nodes and advertising its current vector to all neighboring nodes. In the process, it finds the shortest distance and updates the distance cost. The nearest neighboring nodes just selected will be used as the second stage positioning of replica source nodes. This process repeats until one of the nearest neighbour nodes is the destination node. At which point, data replication commences. Thus, considerable network traffic is reduced compared with conventional flooding algorithm. However, we imposed a stopping criterion on the Grass Growing Structure algorithm by limiting the DLS depth to 2 to prevent indefinite depth search. The *GrassGrowing* algorithm is shown below.

Function *GrassGrowing*
Begin
  Location replica *r* starts from initial node v_0;
  // Select a node v_i using DLS algorithm
  If DLS limit != 2
    Replicate data on node v_i
  Else repeat *GrassGrowing*
End;

4. Simulation and evaluation

To measure the performance of the Grass Growing Structure algorithm, we employed a simulation using OptorSim [2][3][4] with network topology from EU Data Grid Testbed [6] as shown in Fig. 4. The results were compared with Flooding and Centralized Location algorithms which performed on the same testbed.

Performance measurement is carried out by means of mean job execution time [2][3][4] and effective network usage (ENU) [2][3][4]. Details are described below.

A. Grid Configuration

This research used OptorSim [2][3][4] as the data grid simulator to simulate real data grid environment. This simulator is developed in Java under the funding of the European Data Grid project (EU Data Grid).

The OptorSim simulation ran on Intel Xeon 2.1GHz. There were three input configuration files. They are:

1. Network topology file that described the links between different sites, the available network bandwidth, and size of disk storage on each site.
2. Data file that contained the number of replicas and information on how they distribute.
3. Optorsim configuration file is the set number of tests and data request randomization procedure.

B. Network Topology Testbed

Fig. 4 shows the EU Data Grid Testbed [6] as the network simulation topology. Site S0 is the CERN (European Organization for Nuclear Research) location. The star denotes a router and the circle denotes a site. Each link shows the available bandwidth between two connecting sites. In this experiment, each Testbed site, excluding CERN, was assigned a computing and storage element. The CERN was allocated a Storage Element to hold all the master files but was not assigned any Computing Elements (CEs). A CE ran jobs that used data files stored on Storage Elements (SEs). Nodes without Computing or Storage Elements acted as network nodes or routers.

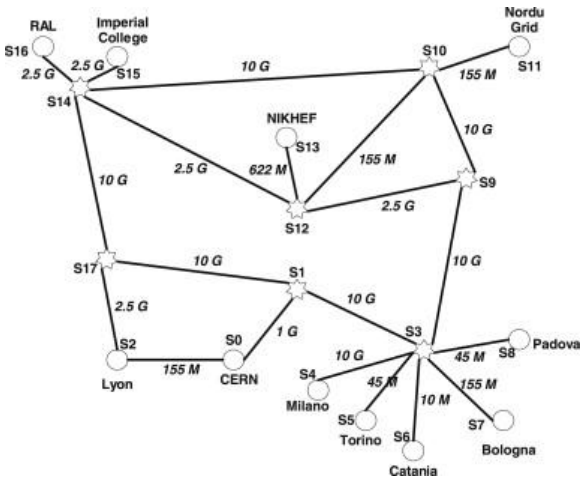

Fig. 4 EU Data Grid Testbed sites and their associated network topology. The numbers indicate bandwidth between the two ending sites in Mbit/s(M) or Gbit/s(G). Stars denote routers and circular nodes denote replica sites.

The mean job execution time is defined as the total time to execute all the grid jobs divided by the number of jobs. ENU (r_{ENU}) is defined as network usage after executing all the grid jobs as follows:

$$r_{ENU} = \frac{N_{remote\_file\_accesses} + N_{file\_replications}}{N_{local\_file\_accesses}}$$

where $N_{remote\_file\_accesses}$ is the number of times the CE reads a file from different SE sites. $N_{local\_file\_accesses}$ is the number of times a CE reads a file from an SE on the same site. For a given network topology, a low value of r_{ENU} indicates that replication is a better optimization strategy than locating another site.

Assuming S0 is the starting point, the Centralized Algorithm places all the data in S0. All sites must retrieve the desired data from S0. Flooding Algorithm starts distributing data through routers and individual site (enclosed by parenthesis) in the following order: (S0), (S2), S1, S17, S3, S14, (S4), (S5), (S6), (S7), (S8), S9, (S16), (S15), S10, S12, (S13), and (S11). The proposed Grass Growing Structure Algorithm replicates the data in the following order: (S0), (S2), S17, S14, S1, S3, (S4), (S5), (S6), (S7), (S8), and (S9).

C. Simulation Results and Discussion

Based on randomization procedures, data requests were issued on EU data grid testbed to measure the performance of the proposed replication method. Figure 5 shows ENU comparison of Grass Growing Structure Algorithm, Flooding, and Centralized Location algorithms. From the outset, the Centralized Location algorithm performs the best, having lowest r_{ENU} while the simulation is still in transient state. As more runs elapse, the graph steadily increases and levels at 0.8 after 500 runs. In the meantime, both Grass Growing Structure Algorithm and Flooding algorithms start with high r_{ENU} but gradually drop to 0.2 after 500 runs.

Fig. 6 shows the mean job time of the three algorithms. The Centralized Location algorithm expends the highest ratio among all algorithms. However, after 100 runs, the ratio begins to level off and reaches a steady state at 500 runs.

Table 1 illustrates comparative Percentage of Storage Filled/Available [11] that is calculated from SE usage multiplied by available SE storage. Notice that the Centralized Algorithm has the highest percentage because data are stored (filled) at only one (S0) available location.

One noteworthy benefit precipitates from this work is reliability and robustness of the underlying network. The proposed Grass Growing Structure Algorithm positions the replicas at appropriate sites not only to reduce the traffic in comparison with the other two algorithms, but also increase reliability and robustness to the system. The distribution can thus be more widely dispersed and reachable by all clients in the network.

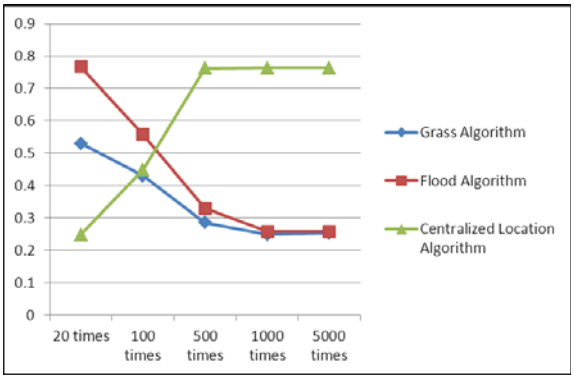

Fig. 5 Effective Network Usage ($^{r}_{ENU}$)

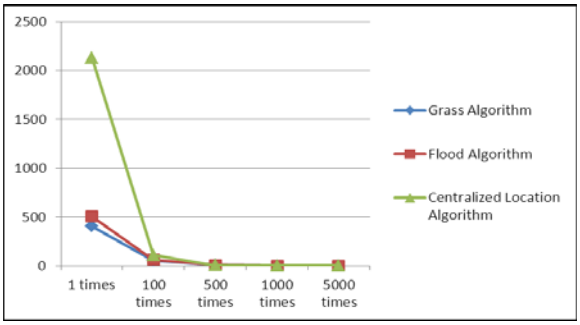

Fig. 6 Mean Job Time

Table 1 Summary of Storage Filled/Available percentage of all replicas by each algorithm

Algorithm	percentage
Grass Growing Structure Algorithm	0.378084
Flood Algorithm	0.480568
Centralized Location Algorithm	0.823892

5. Conclusion and Future Work

In this paper, we propose a replica positioning algorithm called Grass Growing Structure as a means for data replication in a distributed environment. The proposed approach is based on natural grass growing process depending on the amount of water irrigation, whereby data are replicated at the designated location accordingly. This introduces a simple yet effective selection and replication of data over the network. Performance of the proposed algorithm is assessed in comparison with conventional Centralized Location and Flooding algorithms. The results were proved to be satisfactory in terms of ENU and mean job time.

In our future work, we will extend our simulation to incorporate wider network topology testbeds to assess the performance of the Grass Growing Structure.

References

[1] Runqun Xiong, Junzhou Luo, and Aibo Song, "An Effective Replica Location Algorithm Based on Routing-Forward in Data Grid", in *Proceedings of The fifth Annual China Grid Conference*, pp. 31-36, July 16-18, 2010.

[2] David G. Cameron, Ruben Carvajal-Schiaffino, A. Paul Millar, Caitriana Nicholson, Kurt Stockinger, and Floriano Zini, "Evaluating Scheduling and Replica Optimisation Strategies in OptorSim", in *the 4th International Workshop on Grid Computing (Grid2003)*, IEEE Computer Society Press, pp. 52-59, November 17, 2003.

[3] William H. Bell , David G. Cameron , Luigi Capozza, A. Paul Millar , Kurt Stockinger, Floriano Zini, "OptorSim - A Grid Simulator for studying dynamic data replication strategies", *International Journal of High Performance Computing Application*, vol. 17, no. 4, pp. 403-416, February, 2003.

[4] David G. Cameron, Ruben Carvajal-Schiaffino, A. Paul Millar, Caitriana Nicholson, Kurt Stockinger, and Floriano Zini, "UK Grid Simulation with OptorSim", in *e-Science All-Hands Meeting*, Nottingham, UK, September 2003.

[5] Coles, J., "The evolving grid deployment and operations model within EGEE, LCG and GridPP", in *Proceedings of the First International e-Science and Grid Computing, 2005*, pp. 97-115, July 11, 2005.

[6] Ian Foster, C. Kesselman, "Globus : A meta-computing infrastructure toolkit", *International Journal of Supercomputer Application*, pp. 115-128, 1997.

[7] A. Chervenak, I. Foster, A. Iamnitchi, C. Kesselman, W. Hoschek, P. Kunszt, M. Ripeanu, H. Stockinger, K. Stockinger, and B. Tierney, "Giggle: A Framework for construction Scalable Replica Location Services", *Global Grid Forum*, pp. 1-17, 2001.

[8] Russel, Stuart J., Norvig, Peter (2003), "Artificial Intelligence: A Modern Approach (2nd ed.)", Prentice-Hall, pp. 88.

[9] Wikipedia contributors. (2013, March 20). Flooding Computer Network.Available: http://en.wikipedia.org/wiki/Flooding_computer_networking.

[10] A. Tanenbaum, D. Wetherall, Prentice Hall, "Computer Networks, 5th Edition", pp. 368-370.

[11] David G. Cameron, A. Paul Millar, Caitriana Nicholson, Rubén Carvajal-Schiaffino, Kurt Stockinger, Floriano Zini, "Analysis of Scheduling and Replica Optimisation Strategies for Data Grids Using OptorSim", *Journal of Grid Computing*, vol. 2, pp. 57-69, 2004.