A Dynamic Replication Strategy based on Exponential Growth/Decay Rate

Mohammed Madi, Suhaidi Hassan and Yuhanis Yusof

College of Arts and Sciences
Universiti Utara Malaysia, 06010 UUM Sintok
M A L A Y S I A
E-mail : s91499@studmail.uum.edu.my, suhaidi@uum.edu.my, yuhanis@uum.edu.my

ABSTRACT

Data Grid is an infrastructure that manages huge amount of data files, and provides intensive computational resources across geographically distributed collaboration. To increase resource availability and to ease resource sharing in such environment, there is a need for replication services. Data replication is one of the methods used to improve the performance of data access in distributed systems. In this paper, we include issues arising in data replication domain and also we propose a dynamic replication strategy that is based on exponential growth or decay rate. The purpose of the proposed strategy is to identify which files to be replicated. This is achieved by estimating number of accessed of a file in the upcoming time interval. The greater the value, the more popular the file is and therefore will be selected to be replicate.

Keywords
dynamic replication, data grid, exponential growth and decay

1.0 INTRODUCTION

The Internet is a networking infrastructure, which connects together millions of computers worldwide. In order to share information among these millions of computers; there is a need of Web services. Such a service is offered in Grid computing.
Grid is a service-built on top of the Internet for sharing computing power and data storage capacity. Scientists built the
Grid based on the electrical power grid (Foster, 1999), where a lot of instruments are connected to the Internet and share their resources via large-scale computer networks. Grid technology can be classified as Computational Grid and Data Grid. Computational Grid focuses on supplying computing power while Data Grid deals with data, such as sharing and management of large amount of data.

A Data Grid (Venugopal, 2006) is a geographically-distributed collaboration in which all members require access to the datasets produced within the collaboration. In Data Grids (Foster, 2001)(Foster, 2002), distributed scientific and engineering applications often require access to a large amount of data or they continuously generate several terabytes, even petabytes, of raw data in data grid. Therefore one of the tasks in Data Grid is to manage the huge amount of data and facilitate data and resource sharing. In order to achieve this task, data must be copied and stored in several physical locations to vouch the efficient access, without a large consumption of the bandwidth and access latency. In other words, such a system requires replica management services that create and manage multiple copies of files. Creating replicas can reroute the client requests to certain replica sites and offer a higher access speed than a single server (Tang, 2005). Moreover, replication services lead to the following two main benefits (Goell, 2007):

i. **Increased availability:** at fail occurrence in any site, a system can resort to replicated data that is stored at other sites. Hence, increases the data availability.
ii. **Increased performance:** in as much as the data is stored at multiple sites, a client can get the required data from the nearest site, thus increases the performance of the system.

The main aim of data replication is to increase data access performance from the perspective of clients (Tang, 2005), by replicating one or more files to other sites. In this paper, we include a dynamic replication model that uses exponential growth/decay rate in determining files that are to be replicated.
We acknowledged the fact that files that are requested in present, are apt to be requested in the near future. Therefore, popularity of a file depends on the number of access made to the file by the users. With this, popular data files can be identified by analyzing the access histories. Our proposed method is designed by tracking user's behavior of requesting a file, and notes the change to this request, whether is a growth or decay change. The rest of this paper is structured as follows. Section 2 discusses the concept of dynamic replication strategy and reviews on some of replication scenarios and related issues.
Section 3 provides a brief description on existing work in dynamic replication strategies. We include the details of our proposed replication strategy in Section 4 and make a conclusion in Section 5.

2.0 DYNAMIC REPLICATION STRATEGY

To achieve the desired end of the replication process, a replication must be carried out in accordance with certain strategies that organize and manage data. These strategies are done based on several factors such as demand for data, network conditions and cost of transfer (Venugopal, 2006). The replication strategies can be categorized based on whether the strategy is static or dynamic. If the resource conditions are fairly stable in data grid over a long time, then static strategies are applied for replication, since it create and manage the replicas manually (Ranganathan, 2001). A replica can be persisted until it is deleted by users or its duration is expired (Tang, 2006). This means that static replication will not adapt to the network or access pattern changes, which considered as an inherent characteristic of data grid architecture, when client access pattern changes, instead of benefiting from the replication strategy, it would be useless and will bring a burden on the system. On the contrary, dynamic replication changes the location of replicas and creates new replicas to suit the situation automatically. Meaning, dynamic replication takes into consideration the changes of the grid environment (Tang , 2006). In most cases, dynamic replication is triggered when the popularity of a file passes a threshold (Bell, 2002). Therefore we can say that dynamic replication algorithms aim at finding out the potential popular files.

2.1 Replication Scenario And Issues Related To Dynamic Replication

There are three types of replication scenarios and they are briefly presented as follows:

i. Read-only replication:
Data is generated at some site and can be read by other sites, which are offering a read-only replica services, such as read, search and compare requests. The data can be stored at different replicated servers. The scenario that we discuss in this paper is the read-only one.

ii. Read-write replication (update transactions):
It contains data that can be updated by other sites. Managing of updatable replicas would be somewhat simple if only a single site is authorized to update the data. But, as the data can be modified by users from multiple sites, the consistency of the data may be compromised. To maintain the consistency of data, the order in which the transactions are executed must be maintained.

iii. Exchanging data with mobile users:
Providing data and collecting data from mobile users is a key part of many applications. Many applications require data to be available to remote users, including sales people, delivery drivers and employees working away from the office that need to interact with clients and collect data.

There are three unavoidable issues that have to be addressed by any dynamic replication strategy, in order to achieve optimal replication.

i. Replica management
Replica management is the process of creating or deleting replicas at storage site (Stockinger, 2002), this definition leads to a very important question that is which file should be replicated. It would be unwise to copy all data files available at data resources; this will affect the storage capacity. But determining of the file should be based on the objective of the replication strategy. Possible objectives of a replication strategy are to exploit popularity by replicating the most requested datasets (Chang, 2006), to minimize the update cost (Lamehamedi, 2002) or to maximize economic objectives (Bell, 2003).

ii. Determination of the number of replicas
Knowing the number of replicas would be easy if the owner of popular file is the only responsible for creating the replica of this file. But it becomes extremely challenging in Grid systems to determine the number of replicas, especially with the huge and rapid increase of nodes and resources. As a result, not only the owner is responsible for replication, but there are other nodes do so. To address this issue, the role of the algorithm in its ability to answer this question: how many copies need to be created.

iii. Determination of the replica location
Replica location is a problem of finding the physical locations to store replicas of the identified data in a large-scale wide area Data Grid system (Chang, 2007). These three decisions or determination issues have been tackled by many researchers and in many ways (Chang, 2006)(Foster, 2001) (Ranganathan ,2001)(Chang, 2007)(Ranganathan, 2001). Nevertheless, this paper only includes existing work on identifying data or files to be replicated and this is presented in the following section.

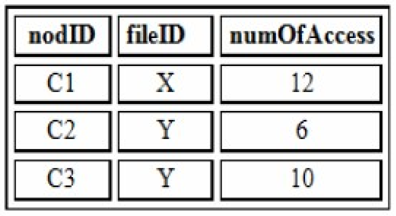
3.0 RELATED WORK

In this section, we introduce some of the studies undertaken involving dynamic replication strategies. In (Ranganathan, 2001), the performance of five distinct strategies has been evaluated using simulation framework; Best Client, Cascading,
Caching+ Cascading, and Fast Spread. The evaluation has been done using three different kinds of access patterns. The research does not include consistency issues and the data used in the work was read-only data. The three different access patterns are:

- Random access pattern, which has no locality in patterns.
- Data contain a small amount of temporal locality; some accessed files are likely to be accessed again. Temporal Locality means that the potential access to the popular file in the past is more than others.

- Data contain small amount of geographical and temporal locality, the files recently accessed by client are likely to be accessed by nearby clients.

Two dynamic replication mechanisms [multi-tier] are proposed in the multi-tier architecture for Data Grids, including Simple Bottom-Up (SBU) and Aggregate Bottom-Up (ABU). The SBU algorithm replicates any data file that exceeds a pre-defined threshold. The main shortcoming of SBU is the lack of consideration to the relationship with historical access records.
For the sake of addressing the problem, ABU is designed to aggregate the historical records to the upper tier until it reaches the root.
Let us consider the data shown in Figure 1. It is an example of access history for two files, X and Y. In addition, the predefined threshold is 10. According to SBU algorithm, if the parent P1 has enough space, file X will be replicated, since the value of its *numOfAccess* is greater than threshold. In return, file Y will be overlooked, although from the viewpoint of the overall system (looking the system as a whole) it was accessed for 16 times (6 + 10). This means that file Y is more popular compared to file X and therefore should be replicated instead of file X. But SBU algorithm processes the access history individually, and does not consider the relation among the accessed files. In the contrary, Aggregate Bottom Up (ABU) takes into consideration the relation among the files, since it aggregates the files included under the same node, and the file with the highest rate will be replicated. Revert to last example and apply ABU, the records after aggregation are < P1 , X , 12> and < P1 , Y , 16 >.



nodID	fileID	numOfAccess
C1	X	12
C2	Y	6
C3	Y	10

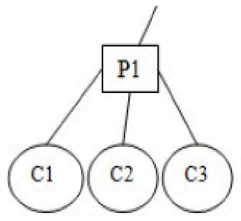


Figure 1: An example of the history and the node relation

The dynamic replication algorithm proposed in (Tang, 2006) determines popularity of a file by analyzing data access history.
The researcher believes that the popular data in the past will remain popular in the near future. The history table is in the format of < *FID* , *NOA* >, which indicates that the file *FID* (file ID) has been accessed *6OA* (number of access) times. Having analyzed data access history, the average number of access, NOA, is computed. Files with NOA's value that is greater than the computer average NOA will be replicated. Hence, the order of which files to be replicated depends on the NOA. The larger the NOA, the more popular the file is and will be given a higher priority during the replication process. .

Nevertheless, these replication strategies did not consider the time period of when the files were accessed. If a file was accessed for a number of times in the past, while none was made recently, the file would still be considered popular and hence will be replicated. The algorithm proposed in (Chang, 2006) called Last Access Largest Weight (LALW) tried to solve this problem. The key point of this algorithm is to give different weights to files having different ages. A hierarchical architecture used in supporting LALW mechanism is shown in Figure 2. It consists of several clusters; each cluster includes a number of sites and one header. The header in each cluster manages sites information about file accesses with other headers. The format of the access rate of a file stored in a header is < FileID , clusterID , Number >, which indicates that the file with FileID has been accessed Number of times by the cluster clusterID. The number of file accessed should be aggregated for the same FileID. Having gathered the access information for the files from all headers, LALW algorithm is then invoked to carry out the replication stages.

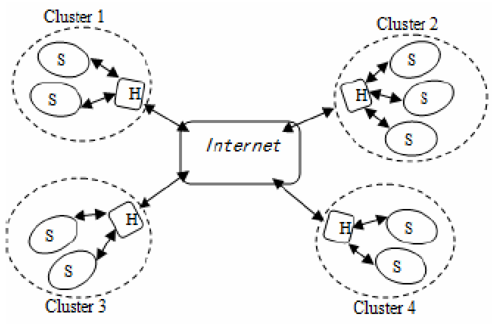


Figure 2: A hierarchical architecture for Grid Cluster

The LALW algorithm is similar to other algorithms by means of using information on access history to determine popularity of a file. But the innovation is included by adding a tag to each access history record of a file. The weight of the record decays to half of its previous weight after a constant time interval.
Older access history records have smaller weights; it means that a more recent historical record is more important. An *Access*
Frequency is calculated to represent the importance of access histories in different time intervals and this is achieved using the formula stated as below. Furthermore, an example of calculating the access frequency for five files is provided in Figure 3.

$$Af(f) = a_0 + \sum_{t=1}^{N_T} \left(a_f^t \times 2^{-(N_T - t)} \right), \forall f \in F$$

Where: N_T is the number of time intervals, F is the set of files that have been requested, and a_f^t indicates the number of accesses for file f at time interval t.

(a) The first time interval

$t_1\,(2^0)$	
FileID	Access Times
A	20
B	17
C	15
D	10

Phase 1:
AF(A) = $20*2^0$ = 20
AF(B) = $17*2^0$ = 17
AF(D) = $15*2^0$ = 15
AF(E) = $10*2^0$ = 10
Phase 2: (p is A)
AFavg(p) = 20/1 = 20
AFavg(all) = (20+17+15+10)/(4*1) = 15.5
R_number(p) = floor(20/15.5) = 1

(b) Sample of access data history during four intervals

$t_1\,(2^{-3})$	
FileID	Access Times
A	20
B	17
C	15
D	10

$t_2\,(2^{-2})$	
FileID	Access Times
A	15
B	20
C	13
D	5
E	25

$t_3\,(2^{-1})$	
FileID	Access Times
A	12
B	24
C	20
E	20

$t_4\,(2^0)$	
FileID	Access Times
A	10
B	15
C	30
E	18

Phase 1:
AF(A) = $20*2^{-3} + 15*2^{-2} + 12*2^{-1} + 10*2^0$ = 22.5
AF(B) = $17*2^{-3} + 20*2^{-2} + 24*2^{-1} + 15*2^0$ = 34.1
AF(C) = $15*2^{-3} + 13*2^{-2} + 20*2^{-1} + 30*2^0$ = 45.1
AF(D) = $10*2^{-3} + 5*2^{-2}$ = 3.7
AF(E) = $25*2^{-2} + 20*2^{-1} + 18*2^0$ = 34.2

According to the result above and based on LALW: the most popular file in the fourth time interval is file C

Figure 3: An example of finding the popular file using LALW algorithm

4.0 EXPONENTIAL GROWTH/DECAY RATE

Many real world phenomena can be modelled by functions that describe how things grow or decay as time passes. Examples of such phenomena include the studies of populations (Kapitza, 2003)(Kremer, 1993) and bacteria(Kreft, 1998).

Exponential growth/decay is a positive or negative growth in which the rate of growth is proportional to the current size (Richards, 1959)(Bartlett, 1996). This paper proposes to apply the exponential growth/decay rate in determining popularity of a file. This is due to the fact that each file has its own number of accessed and this value increases by the increase of access rate and vice versa. If the access rate increases, so does the growth/decay rate. Having calculated these different rates, then the average is calculated and according to this average the popular file is detected.

We describe an exponential growth/decay model for an access number of files in access history. The process of accessing files in data grid environment follows an exponential model. If we use a_0 to represent the number of access for file f at time t, and $a(t)$ to represent the number of access at time $t+1$, our exponential decay/growth model would be given by:

$$N^{t+1} = N_f^t \times (1+r)$$

Assume T is the number of intervals passed, F is the set of files that have been requested, and N_f^t indicates the number of access for the file f at time interval t, then we get the sequence of access numbers:

$$N_f^0\ N_f^1\ N_f^2\ N_f^3 \ldots.\ N_f^{t-1}\ N_f^t$$

So, according to the exponential decay/growth model we can write:

$$N_f^1 - N_f^0 \times (1+r_0),\ N_f^2 - N_f^1 \times (1+r_1)$$
$$N_f^3 - N_f^2 \times (1+r_2)$$

$N_f^t - N_f^{t-1} \times (1+r_{t-1})$, this implies that

$r_{t+1} = \dfrac{N_f^t}{N_f^{t+1}} - 1$. Therefore the average rate for all intervals is

$$r = \frac{\sum_{i=0}^{t-1} r_i}{t-1}$$

Having known the average accessed rate (growth or decay) for a file during the past intervals, we can estimate the number of access for upcoming time interval:

$$N^{t+1} = N_f^t \times (1+r)$$

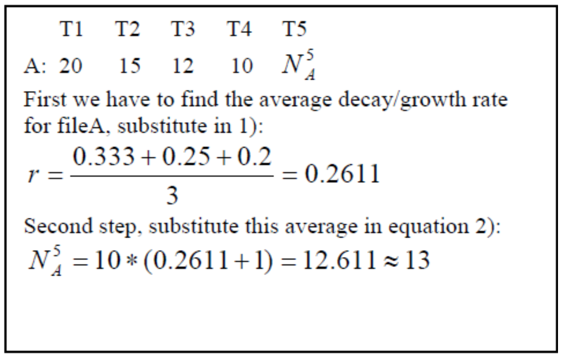
In order to avoid extreme cases where the growth or decay rate is equal to infinity, we are assuming that all files have been accessed for at least once. Using the same data depicted in Figure 3(sample of access history during four time intervals), we calculate the estimation for number of access of a file for the upcoming time interval, using the proposed exponential growth/decay rate model. Based on this estimation, we can then identify the most popular file. Figure 4 shows the process of estimating the upcoming number of access for file A, B and C. In addition, we also obtained the value of 1.76 as the number of access for file D and 13.1 for file E. Based on these estimations values, the order of the most popular file is given as C, B, E, A and D. Such a result is similar to the one obtained using the LALW model.

It is worth mentioning that the true advantage of our approach is that base of exponential decay varies based on the access rate of the file. That means the decay/growth rate of each file is not the same. Contrary to LALW approach which assume that the base of exponential decay is constant and equals ½ , that means all of files decay in the same rate regardless the access rate of each one. As a result, the declension rate of weight will be slower.

4.0 CONCLUSION

Exponential growth and decay are mathematical changes. The rate of the change continues to either increase or decrease as time passes. In exponential growth, the rate of change increases over time - the rate of the growth becomes faster as time passes. In exponential decay, the rate of change decreases over time - the rate of the decay becomes slower as time passes.

We adopted the exponential growth and decay in determining popularity of a file since it is noted that monetary (growth or decay) of number of access is more significant than the access frequency. Even though the proposed model generated a similar sequence of files as the LALW model, it has yet to be tested in a grid environment. It is our intention to implement the presented replication strategy in a grid environment, for example by using *OptorSim*, a grid simulator. Furthermore, the strategy can be tested on a larger of number of files and of different types.



	T1	T2	T3	T4	T5
A:	20	15	12	10	N_A^5

First we have to find the average decay/growth rate for fileA, substitute in 1):

$$r = \frac{0.333 + 0.25 + 0.2}{3} = 0.2611$$

Second step, substitute this average in equation 2):

$$N_A^5 = 10 * (0.2611 + 1) = 12.611 \approx 13$$

Estimation of next number of access for file A

	T1	T2	T3	T4	T5
B:	17	20	24	15	N_B^5

First we have to find the average decay/growth rate for fileA, substitute in 1):

$$\alpha = \frac{(-.15)+(-0.167)+(0.6)}{3} = 0.0943$$

Second step, substitute this average in equation 2):

$$a_B^5 = 15*(0.0943+1) = 16.45 \approx 16$$

Estimation of next number of access for file B

	T1	T2	T3	T4	T5
C:	15	13	20	30	N_C^5

First we have to find the average decay/growth rate for fileA, substitute in 1):

$$r = \frac{0.154+(-0.35)+(-0.333)}{3} = 0.176$$

Second step, substitute this average in equation 2):

$$N_C^5 = 30*(-0.176+1) = 24.7 \approx 25$$

Estimation of next number of access for file C

Figure 4: An example of finding the popular file using Exponential model

REFERENCES

Bartlett, A. A. (1996). The Exponential Function. XI: The New Flat Earth Society The Physics Teacher, 34, 342-343.

Bell, W. H., Cameron, D. G., Capozza, L., Millar, A. P., Stockinger, K., & Zini, F. (2002). Simulation of Dynamic Grid Replication Strategies in OptorSim. Lecture notes in computer science, 46-57.

Bell, W. H., Cameron, D. G., Carvajal-Schiaffino, R., Millar, A. P., Stockinger, K., & Zini, F. (2003). Evaluation of an economy-based file replication strategy for a data grid.

Bonhoeffer, S., May, R. M., Shaw, G. M., & Nowak, M. A. (1997). Virus dynamics and drug therapy (Vol. 94, pp. 6971-6976): National Acad Sciences.

Chang, H. P. (2006). A Dynamic Data Replication Strategy Using Access-Weights in Data Grids.

Chang, R. S., & Chen, P. H. (2007). Complete and fragmented replica selection and retrieval in Data Grids. Future Generation Computer Systems, 23(4), 536-546.

Foster, A. C. I., Salisbury, C. K. C., & Tuecke, S. (2001). The Data Grid: Towards an Architecture for the Distributed Management and Analysis of Large Scientific Datasets. Journal of Network and Computer Applications, 23.

Foster, I. (2002). The Grid: A New Infrastructure for 21st Century Science. Physics today, 55(2), 42-47.

Foster, I., & Kesselman, C. The Grid: Blueprint for a New Computing Infrastructure. 1999. San Francisco: Morgan Kaufmann Publishers, 24(677), 8.

Goel, S., & Buyya, R. (2007). Data replication strategies in wide area distributed systems. Enterprise Service Computing: From Concept to Deployment.

Hunt, R., & Institute of Terrestrial, E. (1978). Plant Growth Analysis.

Kapitza, S. P. (2003). The statistical theory of global population growth. Formal Descriptions of Developing Systems.

Kreft, J. U., Booth, G., & Wimpenny, J. W. (1998). BacSim, a simulator for individual-based modelling of bacterial colony growth. Microbiology, 144(12), 3275-3287.

Kremer, M. (1993). Population Growth and Technological Change: One Million BC to 1990. Quarterly journal of economics-cambridge massachusetts-, 108, 681-681.

Lamehamedi, H., Szymanski, B., Shentu, Z., & Deelman, E. (2002). Data Replication Strategies in Grid Environments.

Newling, B. E. (1974). Urban Growth and Spatial Structure: Mathematical Models and Empirical Evidence. Comparative Urban Structure.

Ranganathan, K., & Foster, I. (2001). Design and Evaluation of Dynamic Replication Strategies for a High Performance Data Grid.

Ranganathan, K., & Foster, I. (2001). Identifying Dynamic Replication Strategies for a High-Performance Data Grid. Lecture notes in computer science, 75-86.

Richards, F. J. (1959). A Flexible Growth Function for Empirical Use. Journal of Experimental Botany, 10(2), 290-301.

Stockinger, H., Samar, A., Holtman, K., Allcock, B., Foster, I., & Tierney, B. (2002). File and Object Replication in Data Grids. Cluster Computing, 5(3), 305-314.

Tang, M., Lee, B. S., Tang, X., & Yeo, C. K. (2006). The impact of data replication on job scheduling performance in the Data Grid. Future Generation Computer Systems, 22(3), 254-268.

Tang, M., Lee, B. S., Yeo, C. K., & Tang, X. (2005). Dynamic replication algorithms for the multi-tier Data Grid. Future Generation Computer Systems, 21(5), 775-790.

Venugopal, S. (2006). Scheduling Distributed Data-intensive Applications on Global Grids: University of Melbourne, Dept. of Computer Science and Software Engineering.

Venugopal, S., Buyya, R., & Ramamohanarao, K. (2006). A taxonomy of Data Grids for distributed data sharing, management, and processing. ACM Computing Surveys