REPLICATION STRATEGY FOR CLOUD COMPUTING

Phyu Phyu[1], Thin Thin Swe[2], Sandar Pa Pa Thein[3]
[1] *Lecturer, Faculty of Information Science, Pathein, Myanmar*
[2] *Lecturer, Faculty of Information Science, Pathein, Myanmar*
[3] *Lecturer, Faculty of Computing, Pathein, Myanmar*

ABSTRACT

Replication is one of the important roles in cloud computing to improve data availability, fault tolerance and throughput for user and control storage cost. As data access pattern changes every time, the nature of popular files is unpredictable and unstable. The data popularity is taken into account as an important factor in replication. Data popularity in replication impacts an efficient storage because it is able to reduce waste storage for unpopular files. The current HDFS replica placement policy does not consider bandwidth and DataNodes' storage utilization. Bandwidth and storage utilization are considered in the proposed data placement algorithm in order to achieve faster file access rate and load balancing. This paper introduces a replication strategy for cloud computing. The proposed strategy contains two potions; replica allocation and replica placement. This proposed strategy will be efficient for cloud computing.

Keyword : - *Replication, utilization, replica allocation, replica placement….*

1. Introduction

Cloud storage is emerging as a powerful paradigm for sharing information across the Internet, which satisfies people's mobile data demand anywhere and anytime. Rather than relying on a few central large storage arrays, such a cloud storage system consolidates large numbers of geographically distributed computers into a single storage pool and provides large capacity, high performance storage service at low costs in unreliable and dynamic network environment [5].

To manage replication in cloud environment, there are two main problems to impact system performance. They are:

(i) Replica Allocation Problem: The replication degree of files should be able to adapt the changing pattern of data access.

(ii) Replica Placement Problem: There is still a problem which algorithm should be optimal to get the best place to store the replica efficiently.The number of HDFS replication is fixed which approved as three [1]. But the file's access frequencies for all files is not the same, for example, the number of access time for some popular hot files is great while others is less relatively. When the number of hot files accessed will surge, the system will appear hot issues. However, HDFS adopts the fixed Replication mechanism; it can't solve the hot issue well. If the hot issue is solved by increasing the number of replicas for all files, then the number of all file' copies in the whole system will increase, it' not necessary for the general non-hot file. It greatly increases the DataNode' storage expenditure and causes the huge waste for the increasingly tense storage space. The concept of popularity of files is introduced to replication strategies for selecting a popular file in reality. File popularity represents whether a file has been hot in recent time intervals, which is computed by file access rate. In order to handle the fluctuation of file access rate, access histories of files are used to calculate the popularity in recent time period.

In this paper, therefore, data popularity based replication method is proposed to overcome the problems of static replication in HDFS and to support better efficiency in cloud storage. Firstly, the rate of change of popularity growth rate is calculated by analyzing the access histories with first order differential equation. Secondly, the replication degree for each file is calculated according to population growth rate. Finally, the replicas will be placed based on proposed data placement algorithm. The rest of this paper is organized as follows. Section 2 describes related works and background theory is presented in section 3 Section 4 presents proposed system architecture.

Evaluation results of replica degree based on popularity growth rate is presented in section 5 and finally, section 6 describes the conclusion and future work.

2. Related Works

Today most popular storage system for cloud computing such as Google File System (GFS) [5] and Hadoop distributed file system (HDFS) [6] are widely used and well known. Replication management has been active research issue in cloud and grid computing. A cost effective replication management scheme for cloud storage cluster is proposed by Qingsong Wei [2]. In that paper, blocking probability is used as a criterion to place replicas among data nodes to reduce access skew, this paper can improve load balance and parallelism. However, this method isn't good for very large file that is file size is Terabyte and also reduces the performance of the system.

One approach Latest Access Largest Weight (LALW) algorithm [7] that is proposed by R.S. Chang and H.P.Chang for data grids. LALW finds out the most popular file in the end of each time interval and calculates a suitable number of copies for that popular file and decides which grid sites are suitable to locate the replicas.

A. Hunger and J. Myint compare two data popularity-based replication algorithms: PopStore and Latest Access Largest Weight (LALW) [1]. In that paper, both algorithms find more popular files according to the time intervals through the concept of Half-life. However, this paper did not consider for load balance in replica placement.

A model for a dynamic data replication strategy is proposed by Da-Wei Sun and et.al., [4]. In that paper, a mathematical model is formulated to describe the relationship between system availability and the number of replicas. This paper considered only the most popular file at each interval and did not consider load balancing.

In [8], a solution for bandwidth-aware data placement in Hadoop is proposed by periodically measuring the bandwidth between clients and DataNodes and placing the data blocks in DataNodes that have maximum end to end bandwidth.

In [3], X.L. Ye, M.X. Huang, D.H. Zhu and P. Xu proposes a static block placement strategy which focuses on load balancing. The optimal nodes are chosen by the novel strategy based on the remaining utilization of nodes' space, and the load balancing is achieved. However, this paper did not consider dynamic behavior and network bandwidth between nodes.

Therefore, in the proposed system, both bandwidth and storage utilization are considered in replica placement. This system can not only improve storage utilization but also achieve speedy file access rate.

3. Proposed System Architecture

The proposed system flow diagram is shown in figure 1.

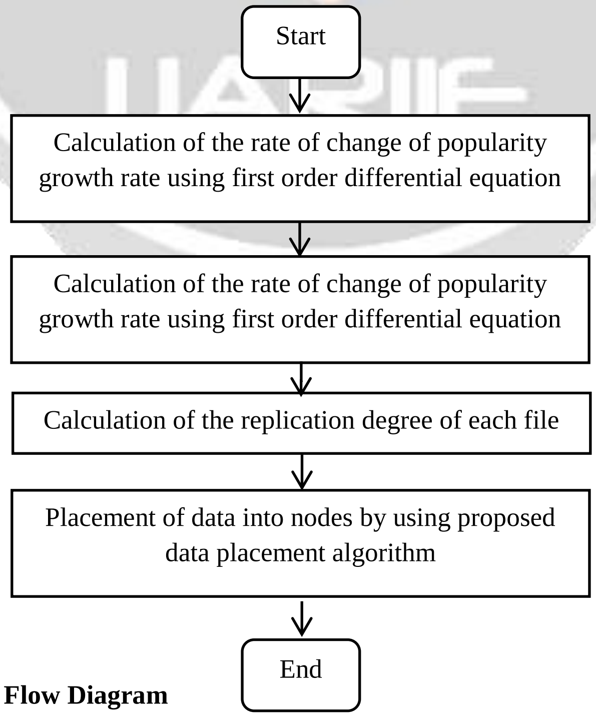
Start
Calculation of the rate of change of popularity growth rate using first order differential equation
Calculation of the rate of change of popularity growth rate using first order differential equation
Calculation of the replication degree of each file
Placement of data into nodes by using proposed data placement algorithm
End

Figure 1. Proposed System Flow Diagram

3.1. Proposed Popularity Growth Rate Algorithm

At first step, we calculate changes of file popularity with first order differential equation. The assumption of popularity is that the popularity of an item grows at a definite time is relative to the total popularity of the item at that time. From the first order differential equation, we compute the growth or decay constant, k.

$$k = \frac{\ln(\frac{P(t)}{P_0})}{t} \quad (1)$$

where P(t) denotes popularity at time t, P0 is starting popularity and k is growth or decay constant. From Yahoo Hadoop audit log file data source [10], we compute changes of file popularity. The audit log is split into smaller files according to timeslot duration. After that, extraction of fields such as Date, Time and src is performed. Then, from the src link, access frequency is counted in each timeslot. Then, the rate of change of file popularity, k, is computed with above mentioned equation 1.

According to popularity growth rate, replica degree for each file is assumed into five groups shown in table 1.

Table 1. Assumption for Replica Degree based on Popularity Growth Rate.

Popularity Growth Rate (k)	Replica Degree
0.01 to 0.19	1
0.20 to 0.39	2
0.40 to 0.59	3
0.60 to 0.79	4
0.80 to 0.99	5

If the accessed file is new and there is no access record history, the replica degree for this file will be assigned 3 as like HDFS default replica number.

3.2. Proposed Data Placement Algorithm

After determining the number of replicas, we will consider how to place these replicas efficiently in order to maximize system performance and load balancing. The existing Hadoop block placement strategy does not take into account DataNodes' utilization, which leads to in an imbalanced load. In proposed algorithm, the replicas will be placed based on DataNodes' utilization and network bandwidth. Therefore, Iperf is used to measure bandwidth in our proposed algorithm. Iperf is an open source package for measuring bandwidth HDFS client and different DataNodes [9]. These bandwidths are compared and DataNodes with highest bandwidth are chosen. If there are more number of DataNodes with highest bandwidth, then among them the DataNodes are selected according to DataNodes'utilization. Our proposed algorithm will overcome the issues of concerned with bandwidth and storage utilization by selecting DataNodes according to bandwidth and storage utilization. We can carry out the disk utilization rate model as

$$U(D_i) = \frac{D_i(use)}{D_i(total)} \quad (2)$$

Where, $U(D_i)$ is the disk utilization rate of the ith DataNode. $D_i(use)$ is the used disk capacity of the ith DataNode, and its unit is GB. $D_i(total)$ is the total disk capacity of the ith DataNode, it is a fixed value of each DataNode, and its unit is GB. The bandwidth and DataNode utilization are used in proposed data placement algorithm as shown in table 2 and algorithm 1.

Table 2. Notations Used in Data Placement Algorithm

Notation	Description
DN	DataNodes list
BW	Bandwidth

U	Storage utilization
RP	Replica List
TU	Total Disk Capacity

Algorithm 1: Data Placement Algorithm

Input: DataNodes List DN= {DN_1, DN_2,.., DN_n }, Replica List RP ={ RP_1, RP_2, RP_3,…., RP_n }, SELECTED [ ] = false

Output: DataNodes List DN

```
1. for each DataNode DN do
2.   Calculate bandwidth BW using Iperf package
3.   Calculate storage utilization of each   DataNode DN by using (2)
4. end for
5. Sort DataNode List DN in descending order of bandwidth
6. for each replica RP do
7.   for each DataNode DN do
8.     if $U(DN_i) + U(RP_i) > TU(DN_i)$ and    !(SELECTED[i] ) then
9.       Place replica $RP_i$ in that Datanode $DN_i$
10.      $U(DN_i)$ += $U(RP_i)$
11.      SELECTED[$DN_i$] = true
12.      break
13.    end if
14. end for
10. end for
11. return DataNode list DN
```

Figure 2. Data Placement Algorithm

In this algorithm, we assume the same rack in cluster of cloud storage.

4. Evaluation Results

In this section, the probabilistic model is applied to evaluate the analysis of availability. In this model, data file F is stripped into n blocks denoted as B= {b1,b2,b3,…,bn} and stored into different DataNodes. There are rn replicas of each block in data file F, and all blocks at the same site will have the same available probability as all blocks are stored in DataNodes with the same configuration. The block availability of data block Bn is denoted as BA_n. $P(BA_n)$ is the probability of data block Bn in an available state. $P(\overline{BA_n})$ is the probability of data block Bn in an available state. Let the number of replicas of data block bn be r_n and the available and unavailable probability of each replica of data block bn are $P(b_n)$ and $P(\overline{b_n}) = 1-P(b_n)$. So, the availability and unavailability of data block Bn are calculated as

$$P(BA_n) = 1 - (1 - p(b_n))^{r_n} \quad (3)$$

$$P(\overline{BA_n}) = (1 - p(b_n))^{r_n} \quad (4)$$

Then, the availability and unavailability of file f_n are calculated as

$$P(FA_n) = (1 - (1 - p(b_n)^{r_n})^{b_n} \quad (10)$$

$$P(\overline{FA_n}) = 1 - (1 - (1 - p(b_n)^{r_n})^{b_n} \quad (11)$$

To evaluate the availability of data block, the 1GB of data is stored in Cluster based storage system. The average failure rate of each DataNode in the system is 0.01, 0.02, 0.03, 0.04 and 0.05.

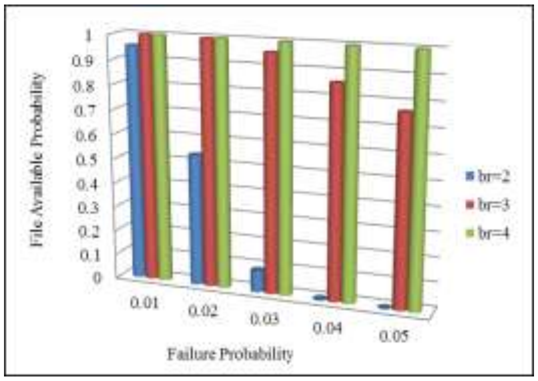

Figure. 3. File Availability with Replication Factor (r) 2, 3 and 4 in the eighty DataNodes

It can be observed from the evaluation results, and file availability is a rapidly fall on replication factor 2 in the eighty number of data nodes in the system. According to the evaluation results, the file availability depends on the replication factors and failure probability. From the experiment, it is obvious that the expected availability values can be satisfied by increasing replication degree in the existence of failure on cluster.

5. Conclusion

In cloud storage environment, data can be stored with some geographical or logical distance and this data is accessible for cloud based applications. In this paper, a dynamic replication management scheme is proposed for HDFS. At each time intervals, the proposed system collects the data access history in cloud storage. According to access frequencies for all files that have been requested, a popular file can be calculated and replicated them to suitable ataNodes in order to achieve load balance of system. As a future work, many experiments have to be done in order to get the efficiency of proposed data placement algorithm. Moreover, additional replica placement policies such as rack-awareness will be considered for overall system improvement.

REFERENCES

[1] A. Hunger and J. Myint, "Comparative Analysis of Adaptive File Replication Algorithms for Cloud Data Storage", *2014 International Conference on Future Internet of Things and Cloud*, 2014.
[2] B. Gong, B. Veeravalli, D. Feng L. Zeng, and Q. Wei, "CDRM: A Cost-Effective Dynamic Replication Management Scheme for Cloud Storage Cluster", *2010 IEEE International Conference on Cluster Computing*, Sep. 2010, pp. 188–196.
[3] D.H. Zhu and P. Xu, M.X. Huang, and X.L. Ye, "A Novel Blocks Placement Strategy for Hadoop", *2012 IEEE/ACIS 11th International Conference on Computer and Information Science*, pp. 3-7, 2012.
[4] D.-W. Sun, G.-R. Chang, S. Gao, L.-Z. Jin, and X.-W. Wang, "Modeling a Dynamic Data Replication Strategy to Increase System Availability in Cloud Computing Environments", *Journal of Computer Science and Technology*, vol. 27, no. 2, Mar. 2012, pp. 256–272.
[5] H. Gobioff, S. Ghemawat, and S.-T. Leung, "The Google File System", *Proceedings of 19th ACM Symposium on Operating Systems Principles* (SOSP 2003), New York, USA, October, 2003.
[6] H. Hardware, and P. Across, "The Hadoop Distributed File System: Architecture and Design", 2007, pp. 1–14.
[7] H.-P. Chang, R.-S. Chang, and Y.-T. Wang, "A dynamic weighted data replication strategy in data grids", *2008 IEEE/ACS International Conference on Computer Systems and Applications*, Mar. 2008, pp. 414–421.
[8] Madhu Kumar S D, and Shabeera T P, "Bandwidth-Aware Data Placement Scheme for Hadoop", *2013 IEEE Recent Advances in Intelligent Computational Systems (RAICS*), 2013.
[9] https://iperf.fr/iperf-download.php.
[10] https://webscope.sandbox.yahoo.com.