Dynamic Replication Management Scheme for Hadoop Distributed File System

May Phyo Thu, Khine Moe Nwe , Kyar Nyo Aye
University of Computer Studies,Yangon
mayphyothu@ucsy.edu.mm, khinemoenwe@ucsy.edu.mm, kyarnyoaye@gmail.com

Abstract

Replication is an essential corner stone for data storage not only for traditional storage systems but also for cloud environment. Data popularity is a key factor in data replication as popular files are accessed most frequently and then they become unstable and unpredictable. Moreover, replicas placement is one of key issues that affect the performance of the system such as load balancing, file access rate etc. Therefore, we focus these factors in this paper. Although the current Hadoop Distributed File System (HDFS) replica placement policy can achieve both fault tolerance and read/write efficiency, but this cannot achieve load distributions. Moreover, the current HDFS replica placement policy does not consider bandwidth and DataNodes' storage utilization. To address these challenges, this paper proposes a dynamic replication management scheme; it includes replica allocation and replica placement algorithms. Bandwidth and storage utilization are considered in the proposed data placement algorithm in order to achieve faster file access rate and load balancing. We calculated the popularity growth rate and replica degree based on popularity growth rate as verification. Our proposed scheme will be effective for large-scale cloud storage.

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].

Depending on the size of cloud enterprises, cloud storage systems may consist of a cluster of storage nodes or even geographically distributed data centers. To build such a cloud storage system, an increasing number of companies and academic institutions have started to rely on the HDFS. HDFS provides reliable storage and high throughput access to application data. It is suitable for applications that have large data sets, typically the Map/Reduce programming framework for data-intensive computing.

In large-scale distributed system, replication is a general technology that can improve the efficiency of data access and the fault-tolerance performance. Data replication can reduce the waiting time of users when they access files, improve the fault tolerant and load balancing.

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. 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

In cloud storage environment, data can be stored with some geographical or logical distance and this data is accessible to cloud based applications. Data is replicated and stored in multiple data nodes to provide high availability and load balancing. Data replication can increase fault tolerance and data availability but one of the challenges in data replication to select suitable data nodes to place replicas.

Replication management has been active research issue in cloud and grid computing. Many researches tried out replication strategies to replicate data efficiently. 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.

Since the DataNode selection for the block placement has been random, the network bandwidth of the allotted DataNode may be less than the maximum available bandwidth. Sometimes, the block may be placed in a DataNode with minimum bandwidth and it is accessed more frequently. At the same time, there may be DataNodes with greater bandwidth and having less load. 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.

According to literature reviews, some paper considers bandwidth-aware replica placement strategy. Others consider storage utilization-aware replica placement strategy. However, not only bandwidth but also DataNodes' utilization is essential for replica placement. Bandwidth improves read access performance and storage utilization achieves load balancing. 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. Background Theory

Today most popular storage system for cloud computing such as Google File System (GFS) [5] and Hadoop distributed file system (HDFS) [1], [6] are widely used and well known. In GFS, three components are client, master and chunk server, while in HDFS these three components are client, NameNode and DataNode. HDFS is developed by Yahoo, which is similar with the Google file system (GFS) developed by Google. However, HDFS is more light-weighted and open-source platform. HDFS is designed to store very large data sets reliably, and to stream those data sets at high bandwidth to user applications. It stores each file as a sequence of blocks; all blocks in a file are of the same size, except the last one. The blocks of a file are replicated for reading performance and fault tolerance.

The default replica placement policy of HDFS is to put one replica on one node in the local rack; another on a node in a remote rack; and the last on a different node in the same remote rack. This replica placement policy cuts the inter-rack write traffic, which generally improves writing performance. With this policy, the replicas of a file do not evenly distribute across the racks. It does not take into account DataNodes' utilization which leads to an imbalanced state of system.

Moreover, the default HDFS block placement policy tries to place blocks randomly, which may easily result in the load imbalance occurrence as well as poor parallelism and low performance of the system. In HDFS, some nodes are too busy and other nodes are idle so much. This condition may accelerate load imbalance and traffic jams, even lead some nodes crash [6].

In HDFS clusters, end-users can specify the default replication factor (number of replicas) for all data. Replication factor and replica placement are key issues of replication management. Although static replication management strategy for HDFS is useful in many ways and easy to manage, it is rarely able to adapt the dynamic circumstances such as highly accesses to some popular files or rarely used files. Therefore, the issue of dynamic replication management strategy for HDFS has drawn considerable attention.

4. Proposed System Architecture

The proposed system flow diagram is shown in figure 1.

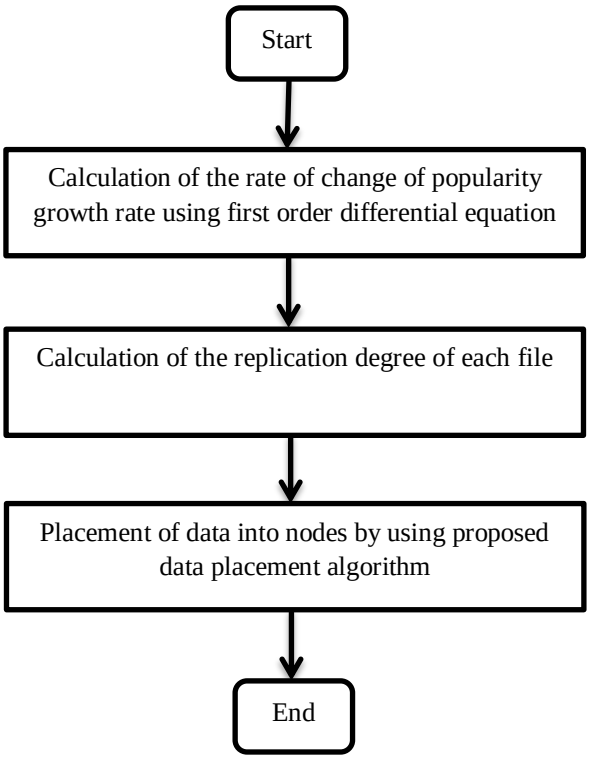

Figure 1. Proposed System Flow Diagram

The proposed scheme includes three-step processes: the rate of change of popularity growth rate will be calculated using first order differential equation in the first step and the number of replicas of each file will be calculated in the second step and then the replicas will be placed into nodes based on proposed data placement algorithm in the third step.

4.1. Proposed Popularity Growth Rate Algorithm

In this step, the rate of change of popularity growth rate will be calculated using first order differential equation. LALW and Pop-Store algorithms applied half-life strategy which means the weight of the records in an interval decays to half of its previous weight.

The idea of popularity is the assumption that the rate at which a popularity of an item grows at a certain time is proportional to the total popularity of the item at that time. In mathematical terms, if *P(t)* denotes the total population at time t, then this assumption can be expressed as

$$\frac{dP}{dt} = k\,P(t) \qquad (1)$$

where *P(t)* denotes population at time t and k is the growth constant or the decay constant, as appropriate. If k > 0, we have growth, and if k <0, we have decay.

It is a linear differential equation which solve into

$$P(t) = P_0\, e^{kt} \qquad (2)$$

Then,

$$k = \frac{\ln(\frac{P(t)}{P_0})}{t} \qquad (3)$$

Where P_0 is the initial population, i.e. p (0) = P_0, and k is called the growth or the decay constant.

In this step, popularity growth rate will be calculated by using Yahoo Hadoop audit log file [10] as data source. The Yahoo HDFS User Audit log format is shown in figure 2.

2016-12-10 11:11:59,693 INFO org.apache.hadoop.hdfs.server.namenode.FSNamesystem.audit: **ugi**=hduser **ip**=/134.91.100.59 **cmd**=delete **src**=/app/hadoop/tmp/test.txt **dst**=null **perm**=null

Figure 2. HDFS User Audit Log Format

To get the frequency count of each file, user audit log is split into small files based on timeslot duration and number of records. Then the required fields are extracted such as Date, Time, IP and src. After that, the user access frequency is counted from src source link from figure 2. In each time slot, the access frequency is counted and stored for individual files. Then popularity growth rate of each file is calculated on each time slot according to table 1 and algorithm 1.

Table 1. Notations Used in Popularity Growth Rate Algorithm

Notation	Description
$P(t_f)$	Popularity values of file f
$AF(t_f)$	Total access frequency of file f at each time slot
inLog	The input log file
k	Popularity growth rate

Algorithm 1: Popularity Growth Rate Algorithm

Input: inLog

Output: *k*

1. Read inLog
2. Calculate access frequency of each file by using

$$P(t_f) = AF(t_f) , \forall f \in F$$

3. Calculate popularity growth rate k of each file by substituting $P(t) = P(t_f)$ in (3)
4. return k.

Figure 3. Popularity Growth Rate Algorithm

In order to verify our proposed popularity growth rate algorithm, we suppose three files (x_1, x_2 and x_3) in three time slots. Each time slot duration is set as 10 seconds, therefore, (t1= t2 = t3= 10 seconds). Let $P_0 = 1$, $P(t) = AF(t_f)$ and calculate k by using equation (3).Suppose access frequencies of file x_1,x_2 and x_3 in time slot 1 are 40, 1100 and 200. In time slot 1, the growth rate k in file x_1, x_2 and x_3 is 0.3688, 0.7003 and 0.5298. Also, in time slot 2, access frequencies of file x_1, x_2 and x_3 are 400, 100 and 900. Therefore, the growth rate k in file x_1, x_2 and x_3 for time slot 2 is 0.5991, 0.4605 and 0.6802. Also, in time slot 3, access frequencies of file x_1, x_2 and x_3 are 2200, 1200 and 20. Therefore, the growth rate k in file x_1, x_2 and x_3 for time slot 3 is 0.7696, 0.7090 and 0.2996.

According to popularity growth rate, replica degree for each file is assumed into five groups shown in table 2.

Table 2. 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.

4.2. 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. Since the

DataNode selection for the block placement is random, the network bandwidth of the allotted DataNode may be less than or greater than the available bandwidth.

In a cluster, different DataNodes may have different 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.

During the process of placement, the disk utilization rate of DataNode is one of the most important factors to affect the load balancing in HDFS. Because HDFS is established on the cheap hardware facilities, different DataNodes may have different disk capacity. Therefore, the capacity of DataNode stored should be proportional to its total disk capacity, in the condition of effective load balancing. We can carry out the disk utilization rate model as

$$U(D_i) = \frac{D_i(use)}{D_i(total)} \quad (4)$$

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 3 and algorithm 2.

Table 3. Notations Used in Data Placement Algorithm

Notation	Description
DN	DataNodes list
BW	Bandwidth
U	Storage utilization
RP	Replica List
TU	Total Disk Capacity

Algorithm 2: 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 (4)
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 4. Data Placement Algorithm

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

5. Evaluation Results

The proposed algorithms will be implemented and tested in simulated cloud environment. To evaluate the proposed popularity growth rate algorithm, Yahoo Web Scope User Audit log dataset is used. However, in this evaluation, the first 100 files for only 5

time slots are used as case study. Each time slot duration is set as 10 seconds. Figure 5 shows the file access frequency table for 15 files in 5 time slots. Figure 6 shows the graph for the file access pattern of 100 files in 5 time slots. According to figure 6, file access pattern can change during time slots. Figure 7 shows the popularity growth rate table for 15 files in 5 time slots. According to growth rate, replica degree is calculated as shown in figure 8.

According to the number of replicas in figure 8, figure 9 shows the graph for total number of created replica of 100 accessed files for 5 time slots. Then the total number of created replicas will be placed based on proposed data placement algorithm. The efficiency of data placement algorithm will be improved by eliminating unnecessary replicas in existing system. Moreover, proposed data placement algorithm will be verified as a future work.

	Access Frequency				
	Timeslot 0	Timeslot 1	Timeslot 2	Timeslot 3	Timeslot 4
1	100	200	300	51	200
2	150	100	50	100	300
3	200	400	400	800	100
4	100	28	100	31	50
5	700	180	500	600	200
6	400	54	300	200	24
7	50	800	100	300	430
8	300	100	23	18	200
9	1000	100	900	110	800
10	8	400	7	800	9
11	88	200	110	700	469
12	23	40	87	100	243
13	500	390	22	23	46
14	200	30	47	2	432
15	800	11	46	22	300

Figure 5. File Access Frequency

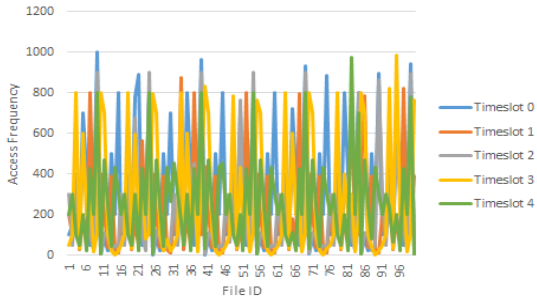

Figure 6. File Access Pattern

	Popularity Growth Rate				
	Timeslot 0	Timeslot 1	Timeslot 2	Timeslot 3	Timeslot 4
1	0.46	0.53	0.57	0.38	0.53
2	0.5	0.46	0.39	0.46	0.57
3	0.54	0.6	0.6	0.67	0.46
4	0.46	0.38	0.45	0.33	0.39
5	0.65	0.52	0.62	0.64	0.52
6	0.59	0.39	0.57	0.52	0.32
7	0.39	0.67	0.45	0.57	0.61
8	0.57	0.46	0.31	0.29	0.53
9	0.69	0.46	0.68	0.47	0.67
10	0.21	0.6	0.19	0.67	0.22
11	0.45	0.53	0.47	0.65	0.62
12	0.31	0.38	0.44	0.46	0.55
13	0.62	0.6	0.31	0.31	0.38
14	0.53	0.34	0.38	0.07	0.61
15	0.67	0.24	0.38	0.31	0.57

Figure 7. Popularity Growth Rate

	Number of Replicas				
	Timeslot 0	Timeslot 1	Timeslot 2	Timeslot 3	Timeslot 4
1	3	3	3	2	3
2	3	3	2	3	3
3	3	4	4	4	3
4	3	2	3	2	2
5	4	3	4	4	3
6	3	2	3	3	2
7	2	4	3	3	4
8	3	3	2	2	3
9	4	3	4	3	4
10	2	4	1	4	2
11	3	3	3	4	4
12	2	2	3	3	3
13	4	4	2	2	2
14	3	2	2	1	4
15	4	2	2	2	3

Figure 8. Number of Replicas

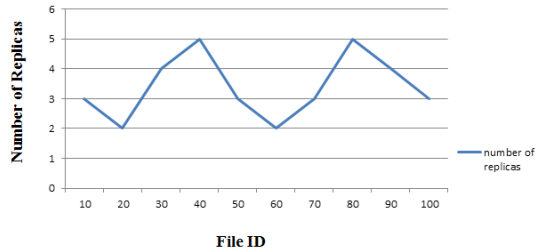

Figure 9.Total Number of Created Replica

6. Conclusion

In cloud storage environment, data can be stored with some geographical or logical distance and this data is accessible for cloud based applications. Data is replicated and stored in multiple data nodes to provide for data availability. 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 DataNodes 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. Proposed data placement algorithm has to be verified as future work. 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.