ISBN 978-981-14-1684-2
Proceedings of 2019 the 9th International Workshop on Computer Science and Engineering (WCSE 2019)
Hong Kong, 15-17 June, 2019, pp. 663-667

Replication Based on Data Locality for Hadoop Distributed File System

May Phyo Thu [1+], Khine Moe Nwe [2] and Kyar Nyo Aye [3]

[1,2,3] University of Computer Studies, Yangon, Myanmar

Abstract. Replication plays an important role for storage system to improve data availability, throughput and response time for user and control storage cost. Due to different nature of data access pattern, data popularity is important in replication because of the unstable and unpredictable nature of popular files. Also, replicas placement is important in consideration of system's performance. In data-parallel applications, data locality is a key issue and this consequence of this issue occurs the decrement of system' performance. Therefore, this paper proposes a data locality-based replication for Hadoop Distributed File System (HDFS). In replica allocation, data popularity is considered for maintaining less replicas for unpopular data and also, disk bandwidth, CPU utilization and disk utilization are considered in the proposed replica placement algorithm in order to get better data locality and more effective storage utilization. Our proposed scheme will be effective for HDFS.

Keywords: Replication, Data Locality, Data Popularity.

1. Introduction

Cloud storage is a technology that provides users' demand data every time and everywhere by sharing information across the Internet. It provides virtual storage to users over the network. There are already known cloud storage products such as Google File System (GFS) [1], Hadoop Distributed File System (HDFS) and Simple Storage Service (S3) etc. Among them, HDFS is open-sourced and platform-independent [2]. Moreover, it is reliable and support faster access of data stored in them. HDFS split incoming data into predefined-sized data blocks and distribute them to nodes with static replication.

Data locality is one of the key issues in considering the performance of Hadoop. In order to provide data locality, Hadoop performs collocation of data with assigned nodes. However, as Hadoop randomly places data to nodes, there is a condition that is when assigned node load data blocks from different node that stores that same data block. Therefore, data locality problem has occurred. Data locality is interval between data block and the assigned node. This minimizes network congestion and increases the overall throughput of the system. The types of data locality are node locality, rack locality and rack-off locality. The greater node locality, the more throughput of the system.

Static replication is used in implementation of Hadoop. Static replication is not good because data access pattern always changes every time. Therefore, file popularity factor is need to be considered in replication. File popularity can be estimated from data access pattern. The consideration of file popularity in replication results in efficient storage because it avoids replicating unnecessary replicas.

In this paper, therefore, dynamic replication method is designed to support better locality in HDFS. Firstly, changes of file popularity are computed by analysing data access pattern with first order differential equation. After that, the calculation of replicas is performed. Finally, the replicas are placed to nodes in order to improve data locality.

[+] Corresponding author. Tel.: + 95-9451236768; fax: +95-01 610633.
E-mail address: mayphyothu@ucsy.edu.mm.

The contributions of this system are as follows:

(1) Changes of file popularity in timeslots is analysed using first order differential equation.

(2) Increment and decrement of the number of replicas for each file is computed.

(3) While the replicas are placed into nodes, the load of nodes such as disk utilization, CPU utilization and bandwidth utilization are considered.

(4) The predefined threshold is used to compute the overloaded condition of cluster.

(5) If the overloaded condition of that assigned nodes occurs, proposed replica replacement algorithm is used.

(6) This proposed replacement algorithm considers not only outgoing blocks but also the access frequencies for blocks.

The rest of this paper is as follows. Section 2 describes related works and the proposed system architecture is presented in section 3 and finally, section 4 concludes the conclusion and future work.

2. Related Works

Cloud storage provides a storage services that is hosted remotely on servers and users can access this through Internet. Data is replicated and maintained in several nodes to provide high availability and load balancing. There were several researches on data replication in Hadoop. R.S. Chang and H.P. Chang proposed an algorithm for data grids, Latest Access Largest Weight (LALW) [3]. It detected only the most popular file in each timeslot and computed the number of replicas for that popular file and determined which nodes were suitable to place these replicas. This did not consider unpopular files and so did not eliminate unnecessary replicas.

A. Hunger and J. Myint introduced a replication algorithm that is based on file popularity: PopStore [4]. In that paper, they detected more popular files at timeslots using Half-life approach. However, file popularity does not have always decay over the time because the characteristics of data access is dynamic. This paper did not consider this condition and data locality in replica placement.

There have been many researches concerning with improving data locality on data replication in Hadoop. Scarlett [5] presented a replication method that replicated proactively files according to prediction of data popularity. It targeted to data received at least three accesses concurrently. However, they did not think about node popularity occurred by relative popular data arrangement.

For achieving improved data locality, the authors proposed a distributed adaptive data replication algorithm (DARE) based on the access frequencies of data blocks [6]. It was a reactive approach and it retained remote data retrieval and evicted aged replicas. It automatically increased the number of replicas when data was replicated to the fetched node. However, it had the limitation of the optimized number of replicas.

Also, access count prediction-based data replication scheme for Hadoop was proposed in [7]. This scheme determined whether generation of a new replica or use of data as cache selectively using the predicted data access count. It placed replicas into nodes using the structure of circular linked list without consideration of utilization factor of this nodes.

3. Proposed System Architecture

The basic idea of replication is based on the different replication degree per data file. Keeping the fixed number of replicas causes wasteful storage for unpopular data and inefficiency for popular data. Also, maintaining too much replicas than current access count for a file does not always guarantee better locality for all blocks. The proposed system flow diagram is presented in figure 1. The aim of system is to design a replication strategy that attempts to improve data locality by increasing the number of replicas for popular data while maintaining the less replicas for unpopular data.

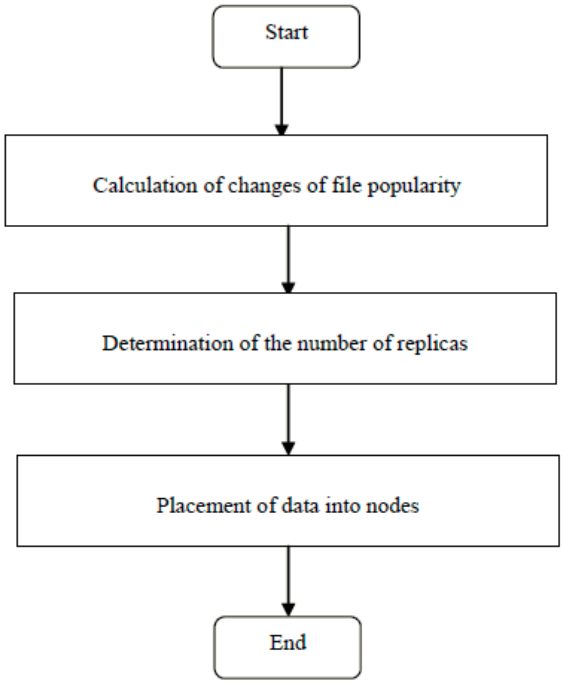

Fig 1: Proposed System Flow Diagram

3.1. Replica Allocation

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, P_0 is starting popularity and k is growth or decay constant. From Yahoo Hadoop audit log file data source [8], 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.

At second stage, the number of replicas for each file is defined using changes of file popularity that is the outcome of the first stage. Initially, existing replicas will be assumed as 3 as like the default replica of HDFS. If k is less than 0.0, then existing replicas is decreased by 1. If k is greater than 0.0, then existing replicas is increased by 1. If k is equal to 0.0, then existing replicas is unvaried. Otherwise, if it is new file, then existing replicas is determined 3 as like the default replica of HDFS.

3.2. Replica Placement

At the third step, replicas are placed into assigned nodes to achieve greater data locality. We will make the assumption that the incoming jobs must have to access these replicas at next timeslot. The entering job is split into tasks and assignment of task with nodes in the cluster is performed. Each input block has one map task. It is assumed that one data block represents one data file. We will let that maximum replicas are total nodes in the cluster and minimum replicas is 1. Node locality of task is checked and if there has node locality, then placement of task at that assigned node is performed. If the condition, that is lack of replica at computing node for map task will occur, prefetching needed replica block into this node. In this system, the load of assigned node is considered to avoid overloaded condition while loading into assigned nodes. That replica is loaded if the load of assigned node is less than predefined threshold. Otherwise, replacement of needed replica block with existing block at assigned node is performed.

The default placement policy of Hadoop is randomness and it assumes that all nodes within cluster have equality condition. Moreover, it does not consider utilization of nodes in placement. This condition results in imbalance load to Hadoop. The proposed system considers inequality condition of nodes within the cluster. In this system, we consider disk utilization, disk bandwidth and CPU utilization as the load of nodes. We can carry out the disk utilization as

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

Where, $U(D_i)$ is the disk utilization of the i[th] DataNode, $D_i\ (use)$ is the used disk capacity of the i[th] DataNode and $D_i\ (total)$ is the total disk capacity of the i[th] DataNode. Then, we can carry out the disk bandwidth as

$$BW(D_i) = \frac{T_b}{T_s} \tag{3}$$

Where, $BW(D_i)$ is the disk bandwidth of the i[th] DataNode, T_b is the total number of bytes transferred and T_s is the total time between the first request for service and the completion of the last transfer. We can carry out the CPU utilization as

$$CU(D_i) = 100\% - (\%\ of\ time\ that\ is\ spent\ in\ idle\ task) \tag{4}$$

Where, $CU(D_i)$ is the CPU utilization of the i[th] DataNode. To compute the load of assigned node, the coefficients of storage utilization, disk bandwidth and CPU utilization are set as $\propto$, β and γ. Then, we can carry out the load of node as

$$Load(D_i) = \propto U(D_i) + \beta\, BW(D_i) + \gamma\, CU(D_i) \tag{5}$$

To compute the overloaded condition of cluster, the predefined threshold O_i of the i[th] cluster is let as the sum of maximum disk utilization, maximum disk bandwidth and maximum CPU utilization in cluster is divided by the number of nodes within cluster. Therefore, we can carry out the predefined threshold O_i of cluster C_i as

$$O_i = \frac{Max_i(U) + Max_i(BW) + Max_i(CU)}{N} \tag{6}$$

That replica is placed at that node if the load of assigned node is less than predefined threshold. Otherwise, replacement of needed replica block with existing block at assigned node is performed. The proposed data replacement algorithm is based on Least Recently Used (LRU) [9]. It is more reliable and efficient than LRU because it takes into account not only outgoing blocks but also access frequencies for blocks in replacement. The proposed data replacement algorithm and data placement algorithm are as follows:

Table 1: Data Replacement Algorithm

Algorithm 1: Data Replacement Algorithm
Step 1: It compute total access frequencies of all blocks at that assigned node as the replica is loaded into the assigned node.
Step 2: That replica is selected to evict from the node if only one block that has minimum access frequencies is found.
Step 3: If there have more than one block that have minimum access frequencies are found, outgoing block is chosen to remove from that assigned node as LRU.

Table 2: Data Placement Algorithm

Algorithm 2: Data Placement Algorithm
Step 1: The entering job is split into tasks and assignment of task with nodes in the cluster is performed. **Step 2:** Node locality of task is checked and if there has node locality, placement of task at that assigned node is performed. **Step 3:** If there has no node locality, remote data retrieval is performed from step 4 to step 7. **Step 4:** Disk utilization, Disk bandwidth, CPU utilization and load of assigned node is calculated. **Step 5:** The predefined threshold of cluster is calculated. **Step 6:** If the load of assigned node is greater than predefined threshold, replacement of needed replica block with existing block at assigned node is performed using algorithm 1. **Step 7:** If the load of assigned node is less than predefined threshold, that replica is placed at that node.

4. Conclusion

Cloud storage provides a storage services that is hosted remotely on servers and users can access this through Internet. Data is replicated and stored in multiple data nodes to provide for data availability. This paper proposes a data locality-based replication for HDFS. In this paper, unpopular replica can be maintained due to the calculation of popularity of certain time. This factor would be effectiveness on utilization of disk bandwidth, CPU and disk utilization of a node while replicating the data through proposed replica placement algorithm. By the way, the proposed replica algorithm will be performed better data locality and effective storage utilization. The data access frequencies obtained from Yahoo audit log file data source. As future work, in order to be a proposed efficient data placement algorithm, many experimental evaluations need to be performed to proof its efficiency. And as well, this data replication scheme will be implemented in various distributed file systems as an ongoing research.

5. References

[1] 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.

[2] H. Hardware, and P. Across, "The Hadoop Distributed File System: Architecture and Design", 2007, pp. 1–14.

[3] 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.

[4] 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.

[5] G. Ananthanarayanan et al., "Scarlett: Coping with skewed content popularity in mapreduce clusters," in *Proc. Conf. Comput. Syst. (EuroSys)*, 2011, pp. 287–300.

[6] C.L. Abad, Yi Lu, R.H. Campbell, "DARE: Adaptive Data Replication for Efficient Cluster Scheduling", *IEEE International Conference on Cluster Computing (CLUSTER 2011)*, pp.159-168, 2011.

[7] D. Lee, J. Lee, and J. Chung, "Efficient Data Replication Scheme based on Hadoop Distributed File System", *International Journal of Software Engineering and Its Applications* Vol. 9, No. 12 (2015), pp. 177-186,2015.

[8] https://webscope.sandbox.yahoo.com.

[9] Andrew S. Tanenbaum. Modern Operating Systems. Prentice-Hall, 1992.