International Journal of Security and Its Applications
Vol.9, No.8 (2015), pp.439-448
http://dx.doi.org/10.14257/ijsia.2015.9.8.39

The Dynamic Replication Mechanism of HDFS Hot File based on Cloud Storage

Mingyong Li*, Yan Ma and Meilian Chen

College of Computer and Information Science, Chongqing Normal University, Chongqing, China
cqnu_lmy@163.com,cqnu_mayan@163.com,47053746@qq.com

Abstract

As an open source cloud storage scheme, HDFS is used by more and more large enterprises and researchers, and is actually applied to many cloud computing systems to deal with huge amounts of data. HDFS has many advantages, but there are some problems such as NameNode single point of failure, small file problem, hot issues, etc. For HDFS hot issues, this paper proposes a dynamic Replication mechanism of HDFS hot file based on cloud storage（HDFS-DRM）. The mechanism includes a Replication of the dynamic adjustment mechanism and adding, deleting duplicate node selection mechanism in two parts, by increasing the NameNode, BlockMap parameters, it records the number of reading requests of each file in a certain period of time to decide whether to increase or decrease the number of copies. The mechanism presents a replica placement method based on stage historical information and node load and selects the appropriate node to add or delete copies of documents to improve the utilization efficiency of the data node storage space effectively. Experimental results show that, HDFS - DRM in hot files case, compared to native HDFS file system access latency is significantly reduced, HDFS-DRM can solve the hot issues successfully.

Keywords: *Cloud storage, HDFS, hot files, dynamic Replication*

1. Introduction

Because of HDFS (Hadoop Distributed File System) [1-2] has good stability and efficiency, it used by many big enterprises to handle mass data. But it also has many drawbacks, such as the single pint of failure of the NameNode, the problem of small file, hot issues and so on. These problems are restricting further development of HDFS .This thesis mainly focus on hot issues of the HDFS.

In large-scale distributed system, Replication is a general technology that can improve the efficiency of data access and the fault-tolerance performance. Replication indicates that the data Replication saved in different location, which can help users to recover data in simple, when the original data is lost. Data Replication can reduce the waiting time of users when they access files and improve the fault tolerant and load balancing [3]. HDFS adopts the Replication mechanism [4]. Each file block has one Replication or several copies (HDFS). The HDFS Replication 'placement mechanism is that the two copies are stored in different data nodes on the same rack ,while the other Replication is stored in data nodes of another rack ,because the possibility is lower, of which the data node on different racks out of work simultaneously . The number of HDFS Replication is fixed which tacitly approved as three. But the file's access frequencies is not the same, for example, the number of time of some popular hot files be accessed is great while others is less relatively. When the number of hot files be accessed surge, the system will appear hot issues [5]. If large numbers of users access data block of the same file which on the same data node at the same time. It may generate the performance reduction which is caused by hot files, and the problem such as the increasing latency of user's access. Hot files change

ISSN: 1738-9976 IJSIA

dynamically, natural event (earthquake, typhoon, tsunami) and social emergency will make the related files become hot files in short time. As time goes on and the emergence of new hot events, the new hot file will appear and the old hot file' heat will disappear.

HDFS adopts the fixed Replication mechanism, it can't solve the hot issue well. If we solve the hot issue by increasing the number of Replication, 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. So this thesis adopts HDFS hot file' dynamic Replication mechanism which is based on the cloud-storage to solve this problem.

HDFS（Hadoop Distributed File System）is a distributed file system of Hadoop system; it is open-sourcing accomplishment of the Google File System (GFS) [6]. Hadoop is a distributed system infrastructure which is supported by the Apache foundation. HDFS has feature with high fault tolerance and is designed to deploy on the low-cost hardware, it is also used as the distributed storage and management for the mass data. HDFS adopts the master /slave framework which is demonstrated as Figure 1. A HDFS cluster is consisted of a NameNode and several DataNode, the framework usually has a Secondary NameNode as NameNode' backups. When this Name Node appears single point of failure, the reserved node will take the responsibility of NameNode immediately .The DataNode is deployed on several racks. The NameNode is in charge of the file system' NameNode and the file which is accessed by client .DataNode is in charge of data, which is on the node. DataNode stores and manages data in itself. HDFS exposures the file system's name node, users can store data in the form of the file on it. Each file is divided into one or several Block, these block data can be stored in a series of DataNode. Block size is fixed and the default size is 64 MB. With the increasing amount of data, it also can be set as 128MB. NameNode executes the operation of the file system's name node, such as opening, closing and renaming files or catalogues, it is also responsible for determining the mapping from data block to specific DataNode. DataNode is responsible for handling the client's file system read and write requests. Creating, deleting and Replicationing the block data under the NameNode's unified scheduling. The single NameNode structure in the clusters greatly simplifies the framework of system.

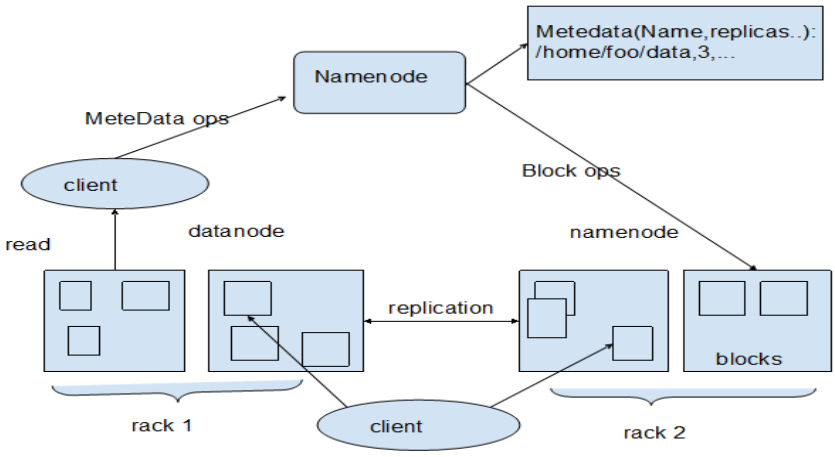


Figure 1. The System Structure of HDFS

2. Framework of Dynamic Replication mechanism system based on HDFS

Dynamic replication strategy refers to the replication's number and layout will dynamically change with the current status of the system, which is widely applied to network technology and distributed file management system. Although the

implementation of dynamic replication strategy is complicated, but it is more suitable for diverse environments. Based on issues such as file access frequency, history request mode and node diversities ,many scholars propose the regulatory mechanisms [7], such as Bartal, have implemented a dynamic replication strategy based on counter . The advantage of this method is easy to implement, the disadvantage is that the statistics of counter only consider the total number of the files being accessed, whereas ignore the distinction between situation of history access and recent access to the file. If a high number of visits to the file in the past, but the most recent visit times is 0, then if you still maintain a high degree of redundancy, it not only fails to improve the performance of the system, but also a waste of system resources. The improved ideas proposed by this text is to use a file's heat properties [9-10] to measure the popularity of the file, and then dynamically adjust the number of file replication. For the high heat file, the system automatically increases the number of copies, and selects the high-performance nodes to store replication when the replication is diffusing, to reduce the latency of data access, balance load, and improve the processing capability of the system. As shown in figure 2:

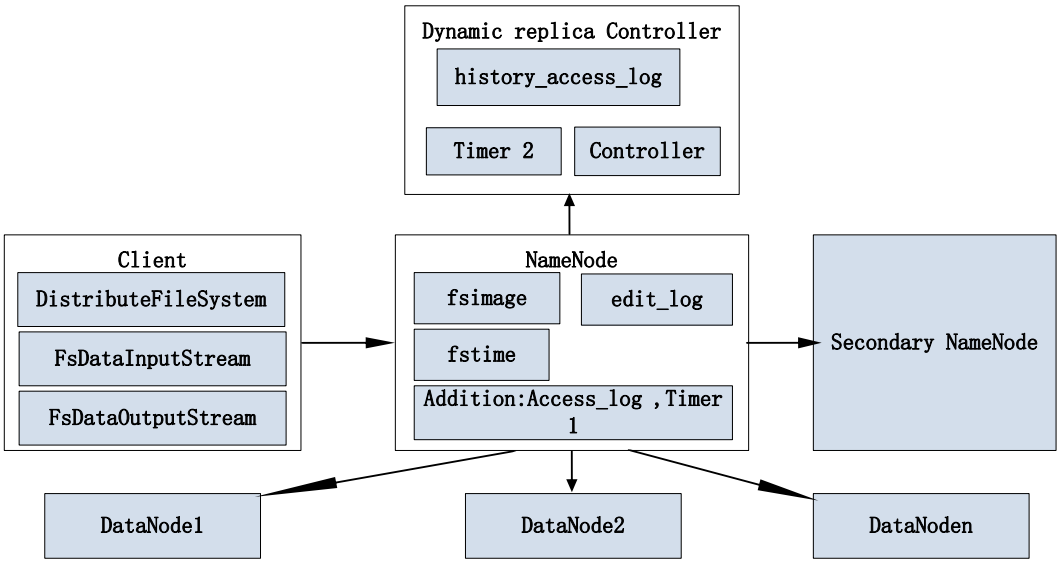


Figure 2. Framework of HDFS-based Dynamic Replication Mechanism (HDFS- DRM)

The main improvement of Dynamic Replication Mechanism based on HDFS:

2.1. Increasing access_log and Timer1 in NameNode

(1) access_log: When Client sends reading files request to NameNode, the request will be saved in the access_log. Every time t, updated information stored in access_log will be transferred to history _access_log in the Dynamic Replica Controller.
(2) Timer1: this mechanism has set a timer such as Timer1 which is in NameNode. When Timer1 completes each round, namely after Timer1 = a, connectCounter will be checked, if the parameter is greater than P d which will be reduced by 1.

2.2. Adding a Separate Dynamic Replica Controller, its Main Functions

(1) history_access_log: History_access_log is located in the Dynamic Replica Controller, it receives updated information of access_log in NameNode every time t.
(2) Timer2: In order to avoid increasing system's burden, which caused by the adding and subtracting of the number of copies within a short time because of reading request of a certain file surging in a short time and then falling sharply, the paper

introduces another timer Timer2, its period is b. The timer is stored in Dynamic Replica Controller and controlled by it.

(3) Determining the position of the Replication node which need to add or delete: Dynamic Replica Controller mainly through the historical information to determine the appropriate location of the node.

(4) Adjusting the number of copies: Modifying the NameNode BlockMap in order to adjust the number of file copies.

3. The Improvement of the BlockMap

In order to achieve the dynamic adjustment mechanism of transcript, it needs to increase the variable to distinguish the standard transcripts from the present transcripts in NameNode, it also needs to record the situation of the Block to be read at the same time so as to judge whether it is a hot file. Hence, the two new variables which are added to this mechanism in BlockMap are named as NumReplica and ConnectCounter. As shown in Figure 3, the two variables situate after the BlockID that in the BlockMap, and the triad including DN、prev and next can be expanded.

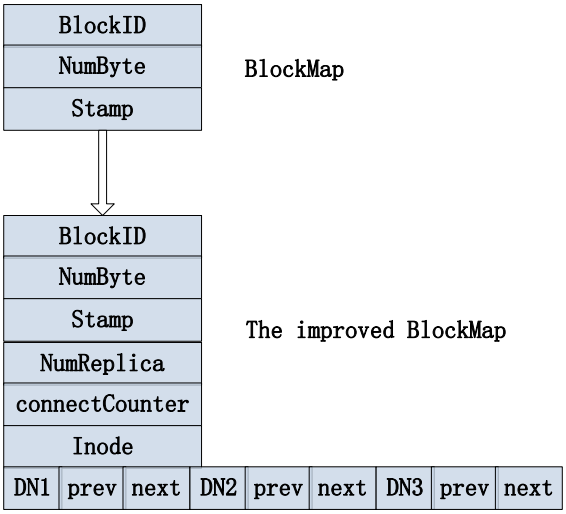


Figure 3. The Improved BlockMap

NumReplica: it is an important parameter of the transcript of dynamic adjustment and it indicates dynamic transcripts. When in the initial system, the value of the parameter NumReplica is equal to the minimal copies which are set in the fsimage. With the dynamic adjustment of the copies, the NumReplica will change dynamically, but the system can dynamically adjust copies to keep consistent with it with the change of NumReplica.

ConnectCounter: it represents the number of users to simultaneously read in the file over a period of time and the ConnectCounter is the sum of multiple Replication reads in the file, the number of copies is determined by the NumReplica. when a new user send read request to the file ,the ConnectCounter in the BlockMap that belongs to the file will be added 1. In order to avoid the problem that ConnetCounter only rise, the mechanism sets a time counter Timer1 where situates in the NameNode. When the time counter Timer1 have completed each cycle, namely have the ConnectCounter being checked after Timer1 equals a. If this parameter is greater than the Numst, which is the number of standard copies being set in the fsimage, the ConnectCounter will minus 1. This operation represents a read request has been completed. Specific process is shown in Figure 4:

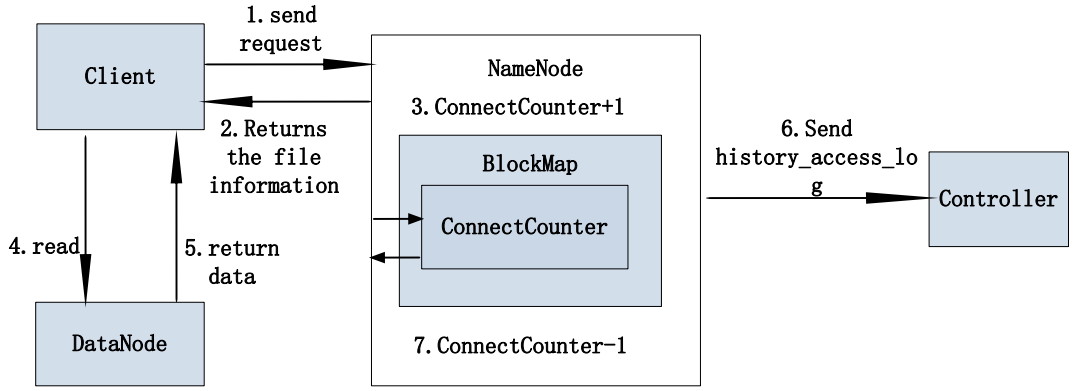


Figure 4. The Process of Dynamical Adjustment of ConnectCounter

Introducing another timer Timer2 whose cycle value is b so as to avoid a surge and then a sharp decline in the file read requests in a short time, and also avoid a system burden be caused by the increase or decrease of the system copies when the parameter ConnectCounter exceeds the threshold value, and then decline sharply in a short time. When ConnectCounter exceeds the threshold value Lth that is set in advance by system, the name node will send a message of addInfo to the system controller to make the value of NumReplica add one. When the system controller receives the message will activate the timer Timer2, and send an ACK message to the name node, and give a new Lth value at the same time. When the value of the parameter ConnectCounter is less than Numst in a period , the system controller will compare the NumReplica which in the name node with the parameters of standard Replication which is set in the fsimage . If NumReplica equals to the number of standard copies, the number will not be modified and exit the operation; If NumReplica is greater than the number of standard copies, then the name of the node will minus the NumReplica parameter with one.

$$Lth = 0.7 * Lmax * NumReplica \quad (1)$$

It's the maximum number of links that each Replication could carry in this fomula. The dynamic adjustment process of the hot file NumReplica is shown as Figure 5:

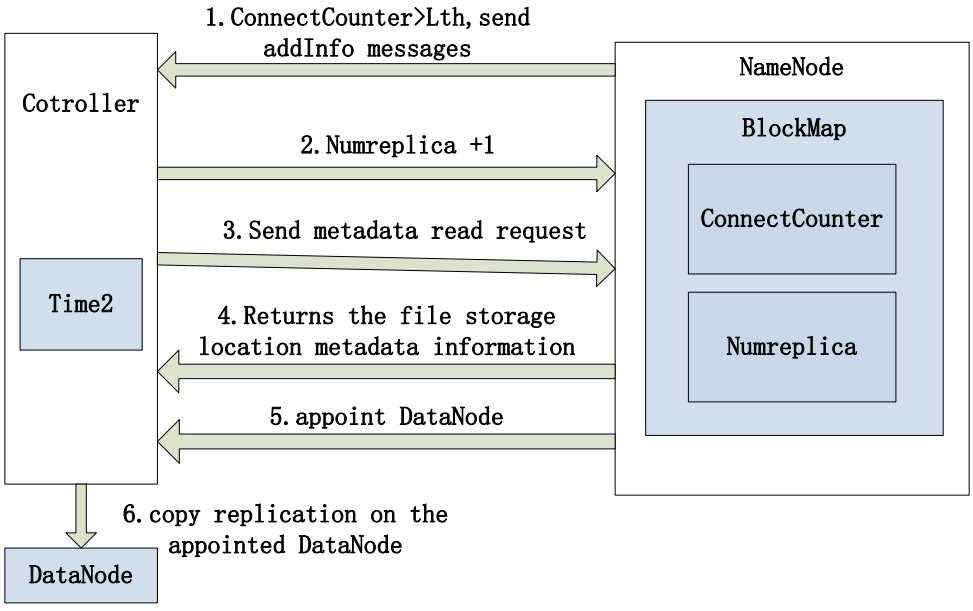


Figure 5. The Dynamic Adjustment Process of the Hot File NumReplica

4. The Node's Selection of Adding and Deleting Replication

The node's selection of adding and deleting Replication mainly includes two aspects: Selecting the appropriate node which can increase or delete the Replication of the document. In order to achieve the greatest degree to improve the performance of the system, it also needs to study the location of Replication [11], so we propose the method which base on the historical information in a period and the node load for Replication' location .

Using an undirected graph G (V, E) to represent the topological structure of HDFS, where V is the set of nodes that represent the data nodes in different domains. Now all the data is divided into N node domain, respectively V1, V2 ... Vn, each domain contains a number of nodes. E is the collection between two nodes' link in network. The link e that between any two nodes ni and nj is given a weight d(i,j),which represents the delay between them. In this mechanism, a value is assigned to each node X, Xi represents whether the Replication of X file has been saved on the node ni. If Xi = 1, it indicates that there is a Replication of the file on the node ni, otherwise there is not. Vh ' represents all Xi=1 in the domain Vh or indicates the collection of nodes that own the file Replication. Selection process of the addition to the Replication' node. The process of increasing the selection of Replication' node is shown as Figure 6.

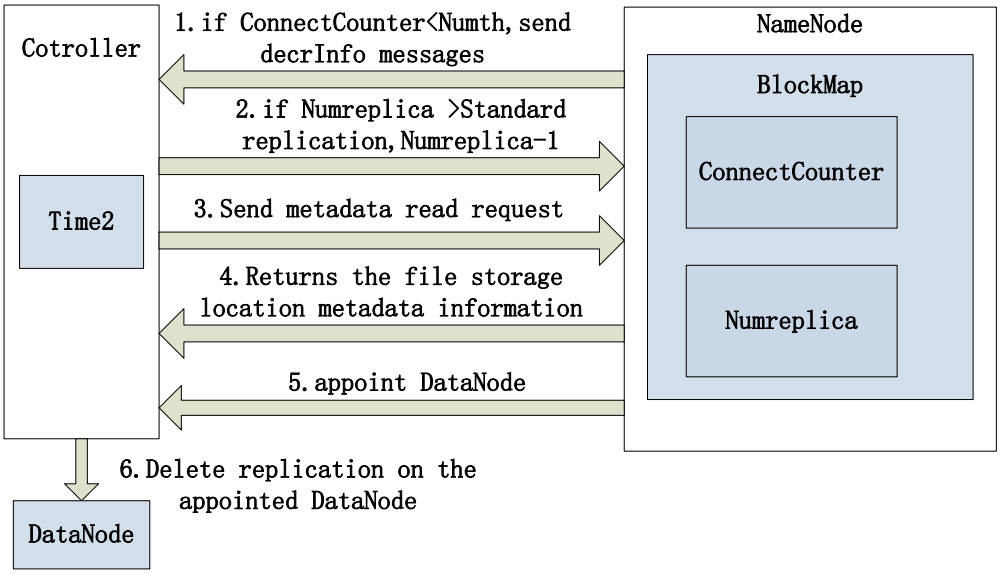


Figure 6. The Process of Increasing the Selection of Replication' Node

4.1. The Selection of Node that Increase the Replication

To determine which node to increase the Replication of the document, it should firstly judge to add the file' Replication to which domain. According to the historical information recorded by the controller, we can identify the quantity of client request in each domain.

$$R = r1 \cup r2 \cup \cdots \cup ri \cup \cdots \cup rn \quad (2)$$

R represents the collection of the quantity of requests sent by the client, which ri represents the quantity of request sent by the client in the domain Vi .

When choose to increase the Replication' node, not only to consider the data nodes' number of client requests data , but also consider the data nodes' hardware load. Do not select the high load data nodes. We use the CPU usage rate (Ucpu), disk storage occupancy rate (Ustor), network bandwidth usage (Ubd) and disk I / O throughput (Ui / o) to represent these four parameters load data nodes, the DataNode periodically send status information to the NameNode. Ldn represents load data nodes.

$$Ldn = k1 * Ucpu + k2 * Ustor + k3 * Ubd + K4 * Ui/o \quad (3)$$

k1, k2, k3, k4 is the weight coefficient, k1 +k2+ k3 +k4=1, According to the actual usage, when the four parameter values are 0.8 ,it is the high-load node, so we set the Ldn' threshold value as 0.8. All Ldn> 0.8 nodes called high load node, all the high load nodes constitute the domain Vg .

After exclude the high-load nodes to constitute domain Vg, we select domain with the minimum number of requests to increase Replication and also need to increase the node of Replication which in domain V+:

$$V+ = \min\ ((\frac{r1}{v1}, \frac{r2}{v2} \cdots \frac{rn}{vn}) - Vg) \quad (4)$$

After determining which domain to increase replica nodes, the system controller also needs to determine the replica nodes' location in this domain. Because of the historical information of the system controller, we can determine the location where all the users who send request in the domain V. Recording the domain which have user's request in domain V as Vr.

$$Vr = Vr1UVr2U \cdots UVriU \cdots UVrn \quad (5)$$

Vri is a collection of nodes that all requests sent in Vi. Gri (Vri, E) is a diagram that is constituted by Vri and the edge e which link the nodes. By reading the name of the node in the BlockMap can learn all copies distributed in the system, then you can get VR, VR = VR1UVR2U ... UVRiU ... UVRn, the VRi is the collection of all the nodes that have this file Replication in Vi. GRi (VRi, E) is covered by the scope of VRi. In summary, we can get:

$$G' = Gri - (Gri \cap GRi) \quad (6)$$

4.2. The Selection of Node that Delete Replication

This system adopts the simple strategy that deletes Replication. When the additional copies are created in datanode, BlockMap will record the time of its creation. When you need to delete the Replication of the file, checking the creation time of all copies of the file, then deleting copies that created with the longest time. In this system, we set the time with 3 days. A feature of the hot file is its locality, that's these hot files' heat may greatly decrease over time. These hot files are generally holding heat time about 2-3 days, after which the document is no longer hot file, so we set the time that delete Replication as 3 days in this system. We can modify and set the parameters through the configuration files of system.

5. Experiment and Analysis

Experimental scheme is as follow: Comparing the system adopted HDFS-DRM mechanism and the HDFS primary system non-adopted dynamic replication mechanism. Experimental environment is as follow: HDFS configuration is 1 NameNode and 20 DataNode. Node configuration is as follow: Intel's third-generation Core i5-3470@3.20GHz Quad-core, the internal storage with 4GB DDR3 1666MHZ, Gigabit Ethernet, 1TB hard disk. Nodes are connected by 1000Mb switch. The node operating system is Ubuntul13.10, Hadoop hadoop-1.0.3 version, java version 1.6.0. The Block size is set as 128M. Setting the case1 as no hot files and access number as 10 users; case2 as normal hot files and access number as 50 users; case3 as high hot files and access number as 100 users.

Through the experiment, the read time of system adopted HDFS-DRM mechanism and the primary system non-adopted the dynamic replication mechanism under 3 kinds of case files is shown in Figure 7.

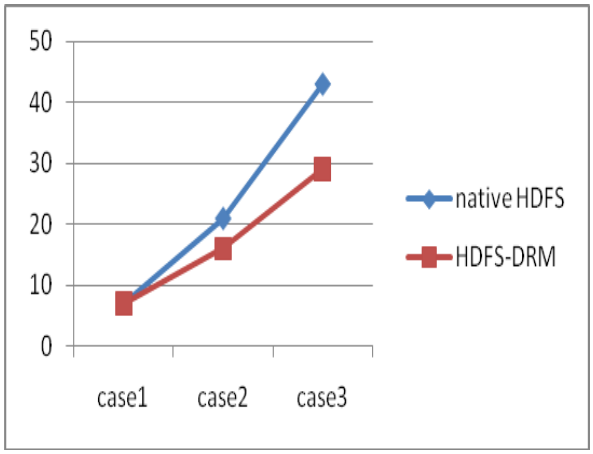


Figure 7. The Read Time of HDFS-DRM Mechanism and Native HDFS

Be knowable by the above picture, the performance of HDFS-DRM system compared to native HDFS has increased significantly, the time cost that HDFS-DRM read the file is obvious lower than that of native HDFS, after appearance of the hot files, and with the increasing popularity of HDFS-DRM files, the time cost of HDFS-DRM mechanism increase gently, while that of the native HDFS increases rapidly. The mechanism of HDFS-DRM has superiority in high heat file case.

6. Conclusion

At the era of big data, cloud computing technology is widely used. Because of the cloud shortage distributed file system is received more and more attention, the HDFS is adopted by more and more enterprises. Hot issues seriously affect the further development and application of HDFS. It proposes dynamic Replication mechanism (HDFS-DRM) of the hot file HDFS which based on cloud storage for the hot issue of HDFS. The mechanism includes a Replication of the dynamic adjustment mechanism and the node's selection of adding and deleting Replication mechanism in two parts. Recording each file' quantity of read quest in a certain period to determine whether increase or decrease the number of copies by increasing the parameter in NameNode and BlockMap .It proposes the method of replica placement, which depends on the historical information at some stage and the node load. Selecting the appropriate node to add or delete file Replication for improving the utilization efficiency of the data node' storage space. The experimental result indicates that (HDFS-DRM files in hot situations, compared to the native HDFS file system significantly reduces access latency). Hot issues can be satisfactorily resolved by HDFS-DRM. I hope to further improve the mechanism of replica's placement and removement in the future to get the minimum access latency of the topology.

Acknowledgement

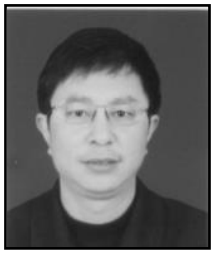
This work is sponsored by Chongqing Education Committee Science and technology project (KJ1500320), Chongqing Education Committee humanities and social sciences project (15SKG040),Chongqing Education Science "The 12th Five Year " planning 2013 year Education Technology special Project (2013-JS-003), Chongqing Higher Education Institute of Higher Education Scientific Research Project(CQGJ13C423), Chongqing graduate education reform project(yjg143091) and Chongqing Normal University fund projects(13XLQ02) .

References

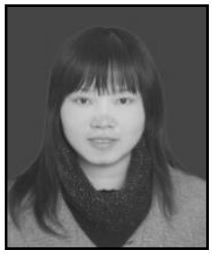
[1] D. Borthakur, "HDFS architecture［EB/OL］［**2012**-07-25］.http://hadoop. Apache.org /common/docs/r0.20.0/hdfsdesign.html.
[2] K. Shvachko, K. Hairong and S. Radia, "The hadoop distributed file system［C］/ / Proc of the 26th IEEE Symposium on Mass Storage Systems and Technologies", (**2010),** pp. 1-10.
[3] K. Ranganathan, A. Lamnitchi, and I. Foster, "Improving data availability through dynamic model-driven replication in large peer-to-peer communities", In: the 2th IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGrid, 02), (**2002),** pp. 376-381.
[4] J. Shafer, S. Rixner and A L. Cox, "The Hadoop distributed file system: Balancing portability and performance [C]", In: (**2010),** pp. 122-133.
[5] M, Rabinovich, I. Rabinovich and R. Rajaraman, "A dynamic object replication and migration protocol for an Internet hosting service [C]", Proc. of the 19th IEEE International Conference on Distributed Computing Systems, (**1999)**, pp.101-113.
[6] S. Ghemawat, "The Google File System. In proceedings of the 19th Symposium on Operating Systems Principles", SOSP'03, October19-22, **2003**, Bolton Landing, New York, USA, pp. 29-43.
[7] J. Wang, A. Chen and J. Peng, "Research on the resource deployment policy of cloud storage based on community theory［C］‖2011 International Conference on Internet Technology and Applications (iTAP2011), Wuhan: IEEE Press, (**2011),** pp. 1-4.
[8] Y. Bartal, A. Fiat and Y. Rabani, "Competitive algorithms for distributed data management [C]", Proceedings of 24th ACM symposium on Theory of Computing, (**1992**), pp. 39-50.
[9] X. Liang, "Study of Data Replication Technology in Cloud Storage Environment [D]", Nanjing University of Posts and Telecommunications, (**2013),** p. 3.
[10] J.-g. Dong, W.-w. Chen, H.-j. Wu and L.-j. Tian, "Load balancing study in cloud storage based on dynamic replica technology", Application Research of Computers, (**2012)** September.
[11] D. S. Li, X. Nong and X. Lu, "Dynamic Self-Adaptive Replica Location Method in Data Grids", In the Proceedings of IEEE International Conference on Cluster Computing, December 1-4, (**2003),** pp. 442-445.

Authors

Mingyong Li, he was born on October 27, 1979 in Hubei Province, China. He received the Bachelor's degree in educational technology in 2003 at Central China Normal University, and he received the Master of Educational Technology in 2008 at Chongqing Normal University. His main research interests include cloud computing, cloud storage, hadoop HDFS.



Yan Ma, he was born on October 13, 1960 in Yunnan Province, China. He received the Master of Educational Technology in 1993 at Central China Normal University, and he received the doctor's degree of Education in 2008 at Southwestern University. His main research interests include artificial intelligence, semantic meshes, education



Meilian Chen was born on September 9, 1981 in Hubei Province, China. Study in Chongqing normal University. Major in Computer application. And research interests on cloud computing and mobile learning.