International Journal of Database Theory and Application
Vol.9, No.2 (2016), pp. 1-8
http://dx.doi.org/10.14257/ijdta.2016.9.2.01

Research on Metadata Management in Cloud Computing

Dongyu Li

Batotou Vocational & Technical College Department of Computer and Information Engineering, China

Abstract

Existing metadata management methods bring about lower access efficiency in solving the problem of renaming directory. This paper proposes a metadata management method based on directory path redirection (named as DPRD) which includes the data distribution method based on directory path and the directory renaming method based on directory path redirection. Experiments show that DPRD effectively solves the lower access efficiency caused by the renaming directory.

Keywords*:Cloud Computing; Directory Path; Redirection; Metadata*

1. Introduction

With the prevalence of Internet application and data-intensive computing, there are many new application systems in cloud computing environment. These systems are mainly characterized by [1-3]: (1) the enormous files stored in the system, some even reach trillions level, and it still increase rapidly; (2) The user number and daily access are quire enormous, reaching billions level. For example, the number of pictures currently stored in Facebook [4] is more than 140 billion, and both the number of users and daily access are over one billion. Such massive files and accesses may result in that the expandability and high performance will be the bottleneck of the efficient application of cloud computing, and meanwhile, the efficient management of metadata is a key technology that can break through this bottleneck.

At present, the metadata management methods mainly include the look-up table [5], sub-tree partition, Hash and directory path fixed number. The typical application of look-up table should be the single metadata server method adopted by Google, which mainly stores the metadata table in a server, and it is applicable for the storage system with few files. However, for the distributed storage system with numerous files, it will be a bottleneck of the system performance. The sub-tree partition method [6-7] separates the only name space of file system into independent sub-tree according to the directory levels, and each metadata server in the metadata server cluster will be responsible for one or several sub-trees. It has a good expandability, but the directory is distributed in several metadata servers, and it requires directory traversal among metadata servers to locate the metadata, which may result in the low efficiency access. Hash method [8-9] will locate the storage position of this file according to the file identifier (such as the full path name of the file). It breaks through the limitations of single metadata server, but it requires migrating related metadata when renaming a directory. The directory path fixed numbering (marked as DPFN) [10-11] endows the globally unique ID (DPID) for each directory path, and DPID remains unchanged in the life cycle of the directory path, and the metadata of all files (or sub-directories) in the directory path will be placed and achieved according to its hash value of DPID. It can solve the metadata migration issue caused by directory renaming, but it uses a directory path index server to manage the mapping relationship between directory path and DPID, which brings a low efficiency access.

Aiming at the low efficiency access for solving the directory renaming problem in current metadata management methods, the metadata management method based on the redirection of directory path (marked as DPRD) is proposed in this paper. Firstly, it uses the method of hash directory path for distributing and achieving the metadata, breaks through the bottleneck of the directory path index server, and realizes the concurrent access of multiple users. Secondly, it only records the redirected directory path space, and the space is distributed into all metadata servers, which reduces the searching space of directory path. In the end, adopting the directory path redirection method, it realizes the correct access to metadata without migrating metadata during renaming directories. It can effectively solve the low efficiency access caused by the directory renaming.

2. Analysis of the Directory Path Fixed Number

The directory path fixed number(DPFN) supports not to migrate metadata when renaming a directory, which basic idea is: each directory path is given a globally unique fixed ID, which remains unchanged in the life cycle of its directory path; a directory path index server (marked as DPIS) is used to maintain the mapping between directory path and its DPID; placing and achieving the metadata of an object has to get its DPID from the directory path index sever, and then remaining operations are completed in the MDS corresponding to DPID hash value. The metadata distribution and access pattern are shown in Figure 1.

Based on the above mentioned idea, the process of DPFN achieves the file f is:

(1) User send a request to DPIS for achieving the DPID of the file f;

(2) search the DPID of the file f in the entire directory path space;

(3) Return the DPID of the file f to user;

(4) An user send access request to the MDS hash(DPID) according to the hash value of the DPID;

(5) search the metadata of the file f in the MDS hash(DPID);

(6) return the metadata of two user;

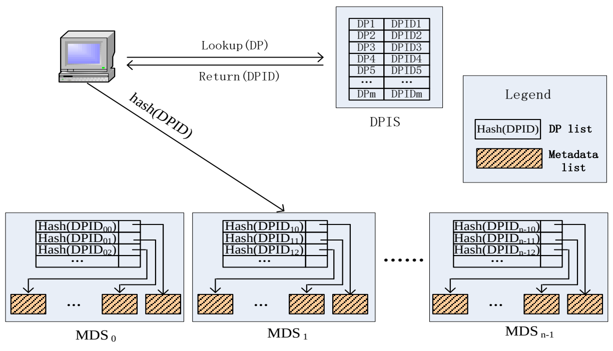


Figure 1. Metadata Distribution and Access in DPFN

Although DPFN solves the issue of metadata migration caused by renaming directory, but it brings a low efficiency access. It is mainly because: (1) all users must firstly access DPIS for the DPID, resulting in multi-user accesses are serial; (2) the DPIS records the mapping between directory path and its DPID, resulting in the search space is large.

3. Metadata Distribution Method Based on Directory Path

The directory path of a file refers to the path from the root directory to its parent directory. The directory path of file f is marked as DPf. As shown in Figure 2, the directory path of file 6 DPfile6=\D2\D3.

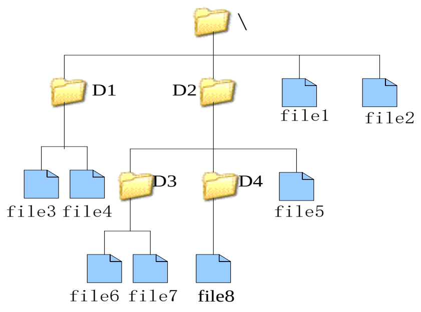
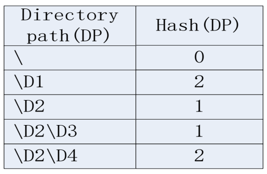


Figure 2. A File System

The method for DPRD to distribute metadata is that: it hashes the directory path of a file, and then it will distribute the metadata of the file to the metadata server corresponding to the hash value, namely the MDS storing the metadata of file f is: Hf=hash (DPf).

For instance, for the file system showed in Figure 2, if the Hash value of each directory path is shown in Figure 3, then the directory and file metadata distribution is shown in Figure 3.



Directory path(DP)	Hash(DP)
\	0
\D1	2
\D2	1
\D2\D3	1
\D2\D4	2

Figure 3. DP and Hash(DP)

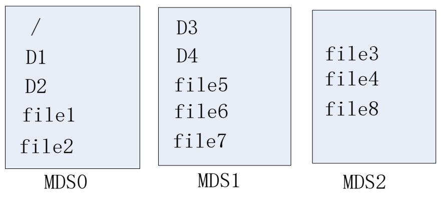


Figure 4. Metadata Distribution

4. Efficient Metadata Organization Method

As for the file under the same directory, the Hash value of the directory path name is the same. Therefore, the metadata of all files in the same directory is distributed to the same metadata server. In order to be convenient for searching and directory operation, DPRD employs three-layer structure to organize the metadata, as shown in Figure 5. The bottom layer is physical layer, which stores the metadata in the form of heap file, and it can realize the efficient utilization of storage space. In the physic layer, all metadata are distributed randomly, while most of the directory operation requires operating all files in a directory or the sub-tree which the directory is its root. In order to improve the directory operation efficiency, DPRD establishes a logic layer over the physical layer, which can virtualizes the randomly scattered files into a logic whole through indexed mode. The top layer is mapping layer, which is a list formed by the directory path in the same MDS (marked as DT). Each item of DT is a pair <hash (DP), pointer>, in which hash (DP) is the hash value of a directory path, and the pointer points to the logic block of DP.

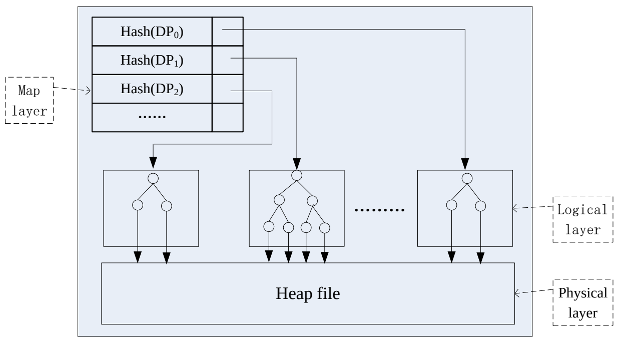


Figure 5. Metadata Organization

5. Renaming Directory Based on Redirection of Directory Path

When a directory dir is renamed as dir′, those directory path DP_{dir} containing dir will turn to be DP_{dir}', and the metadata server storing these file under DP_{dir} will change from Hash(DP_{dir}) to Hash(DP_{dir}′). In order to maintain correct location data in the following operations, it should migrate the metadata under DP_{dir} from Hash(DP_{dir}) to Hash(DP_{dir}′) synchronously, which greatly influence the system performance. At first, DPRD employs the directory path redirection method to avoid metadata migration when renaming, and then it will take advantage of the periodical fluctuation of the load, which will migrate the metadata whose directory paths are redirected to correct position when the system load is relatively light. Thus it improves the efficiency of renaming operation.

The redirection method of directory path is that: each MDS is placed a redirection directory table (marked as RDT), which records directory paths whose metadata are not local. Each item of RDT is a pair <hash (DP), pointer>, in which hash (DP) is the hash value of a directory path, and the pointer points to the logic block of DP. The pointer is composed of A and B of two parts, in which A is the MDS number which hosts the metadata of a redirected directory path, and B is its logical block first address. With the redirection of directory path, the metadata distribution and organizational structure is shown in Figure 6.

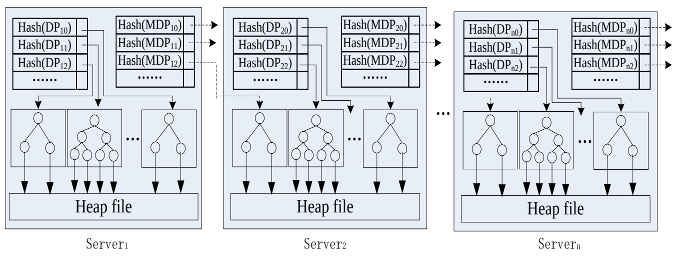


Figure 6. Metadata Organization with Directory Path Redirection

When comparing DPFN and DPRD, it can be seen that: (1) multiuser accesses are carried out serially in DPFN and concurrently in DPRD. (2) DPFN records entire directory path space, while DPRD only records the redirected directory path space, and the directory space is distributed into N metadata server by the hash method of directory path, so each metadata server stores nearly 1/N of the redirected directory space, which greatly shortens the searching space of each access. (3) When the directory path remains unchanged, DPFN still accesses the directory path indexed server to gain DPID, while DPRD will find metadata in the corresponding metadata server of directory path directly. In summary, DPRD is superior to DPFN in access efficiency.

6. Renaming Directory Based on Redirection of Directory Path

A. Experimental environment

Our testing infrastructure had 126machines on 4 racks connected by Gigabit Ethernet switches. Intra-rack bisection bandwidth was ≈14Gbps, while inter-rack bisection bandwidth was ≈6.5 Gbps. Each machine had two 2.4GHz Intel Xeon CPUs, 4GB of main memory, and two 7200RPM SCSI disks with 200GB each. Machines ran Red Hat Enterprise Linux AS 4 with kernel version 2.6.9.

In the following experiments, the distribution of nested directories in each sub-tree by depth and that of the sub-directories count in per directory are based on the generative, probabilistic model[12,13] which closely accords the situation of the observed directory depths with metadata snapshots from more than ten thousand file systems from 2001 to 2004. Each client machine had 4 parallel threads, each with one outstanding request issued as soon as the previous completed. MDS and client respectively run on a separate computer.

B. Experiment result and analysis

At first, under different directory redirection rate (represented by r), metadata operation performance of DPRD was tested. The directory redirection rate is the proportion of redirected directory paths and total directory paths. In this test, a great number of directories should be made first, and then files would be created under each existing directory. Finally, the metadata of each file would be read. The test result is shown in Figure 7, from which it can be seen that, when r=0, the system performance is the best, but with the increase of r, the system performance declines gradually, for when r=0, a network access can achieve metadata, but when r >0, these redirected directory may get their metadata needing two network access. When r is fixed, with the increase of MDS, the growth rate of throughout will decline gradually. It may result from the fact that when the increase of MDS brings the growth of throughout, the probability of hitting the redirected directory increases.

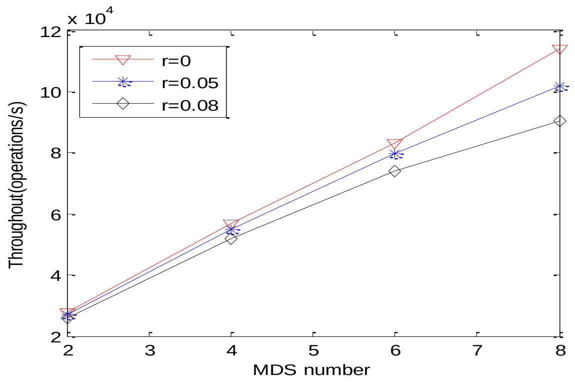


Figure 7. Access Performance Under Different Redirection Rates

Secondly, DPRD and DPFN in making directories, creating files and reading metadata was compared, and the test result was shown in Figure 8. It can be seen from the picture that DPRD is superior to DPFN in performance and expandability, it result from the bottleneck, meaning there is directory path index server in DPFN. As for DPRD, reading metadata and creating files can be finished at one network access. Therefore, the two have similar performances, but when creating files, the directory files of directory path should be equipped with write lock, while for the reading of metadata should be equipped with read lock, and its concurrent ability is higher than that of file creation. Therefore, it is superior to the creating of files in performance. However, making directory requires two network accesses, namely, one is to modify the directory file of the directory path, and the other is to create the directory file. Therefore, it is lower than file creation and metadata reading in performance.

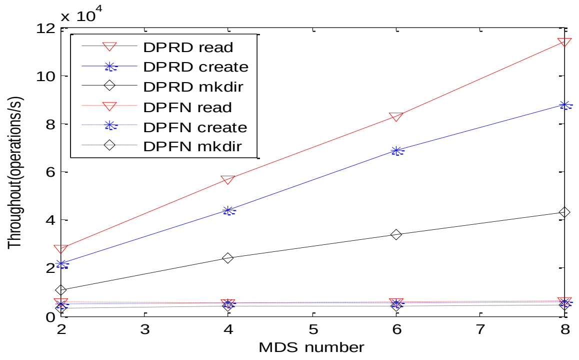


Figure 8. Performance Comparison for DPRD And DPFN

In the end, the performance of renaming directory operation under such condition in which the directory contains 1, 2, 4, 8 and 16 files was tested, and the result is shown in Figure 9. It can be discovered that the response delay of renaming directory operation is only related to the number of sub-directory under the renamed directory, which it has nothing to do with the number of files under the renamed directory. It is mainly because on one hand, the metadata of files is not moved when the directory is renamed in DPRD, and it is carried out when the load is light, while on the other hand, directory renaming will lead to the changes in all the directory paths, which requires redirection. It has also been discovered in the test that the response delay of renaming directory has nothing to do with the number of MDS, for when renaming a directory, the number of redirection operation has nothing to do with the directory path.

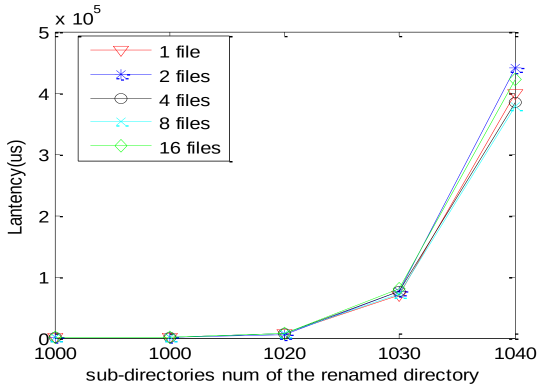


Figure 9. Latency of Renaming a Directory in MHS

7. Conclusions

Our distributed metadata management strategy DPRD is used to eliminate the following questions. Firstly, SUBTREE PARTITION always needs to traverse the directory hierarchy and at the same time metadata partitioning among MDS cluster is coarse-grained. DPRD adopts hash method for directory paths to avoid the directories traversal. Secondly, although a full pathname used to identify a file in HASH can directly and quickly locate the destination MDS, it cannot efficiently handle some situations such as renaming a directory. DPRD adopts directory path redirection to avoid the renaming overhead of files metadata in a renamed directory. Thirdly, DPFN avoid also the renaming overhead of files metadata in a renamed directory by endowing the globally unique ID (DPID) for each directory path, but it uses a directory path index server to manage the mapping relationship between directory path and DPID, which become a bottleneck. DPRD adopts the metadata distribution based directory path to break through the bottleneck. Also, DPFN records entire directory path space, while DPRD only records the redirected directory path space, and the redirected directory space is distributed into N metadata server by the hash method of directory path, so each MDS stores nearly 1/N of the redirected directory space, which greatly shortens the searching space of each access. We have described our distributed metadata management strategy DPRD and given the survey of the performance comparisons, and the performance results are encouraging.

References

[1] S. V. Patil, G. Gibson, S. Lang and M. Polte, “Giga+: scalable directories for shared file systems”, In PDSW '07: Proceedings of the 2nd international workshop on Petascale data storage, **(2007)**, pp. 26-29.
[2] Y. Hua, Y. Zhu, H. Jiang, D. Feng and L. Tian, “Scalable and adaptive metadata management in ultra large-scale file systems”, In ICDCS '08:Proceedings of the 2008 The 28th International Conference on Distributed Computing Systems, pages 403-410, Washington, DC, USA, **(2008)**.
[3] J. Xing, J. Xiong, N. Sun and J. Ma, “Adaptive and scalable metadata management to support a trillion files”, Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis, Portland, Oregon, **(2009)**.
[4] Facebook Website.www.facebook.com.
[5] S. Ghemawat, H. Gobioff and S.-T. Leung, “The Google file system”, In SOSP, **(2003)**.
[6] S. M. Kistler, J. J. Kumar, P. Okasaki, M. E. Siegel and E. H Steere, “Coda: a highly available file system for a distributed workstation environment”, IEEE Transactions on Computers, vol. 39, no. 4, **(1990)**, pp. 447–459.
[7] S. A. Weil, K. T. Pollack, S. A. Brandt and E. L. Miller, “Dynamic metadata management for petabyte-scale file systems”, In: Proceedings of IEEE/ACM SC2004 Conference, **(2004)**, pp. 523–534.
[8] O. Rodeh and A. Teperman, “zFS – a scalable distributed file system using object disks”, In: Proceedings of the 20th IEEE/11th NASA Goddard Conference on Mass Storage Systems and Technologies, **(2003)**, pp. 207–218.
[9] S. A. Brandt, E. L. Miller, D. D. E. Long and L. Xue, “Efficient metadata management in large distributed file systems”, In: IEEE Symposium on Mass Storage Systems, **(2003)**, pp. 290–298.
[10] Z. Liu and X. M. Zhou, “A metadata management method based on directory path”, Journal of Software vol. 18, no. 2, **(2007)**, pp. 236–245.
[11] J. Wang, D. Feng, F. Wang and C. Lu, “MHS: A Distributed Metadata Management Strategy”, The Journal of Systems and Software, vol. 82, no. 12, **(2009)**, pp. 2004-2011.
[12] N. Agrawal, W. J. Bolosky and J. R. Douceur, “A five-year study of file-system metadata”, In: Proceedings of the 5th USENIX conference on File and Storage Technologies, **(2007)**, pp. 31-45.
[13] N. Agrawal, A. C. A. Dusseau and R. H. A. Dusseau, “Generating Realistic Impressions for File-System Benchmarking”, In: Proceedings of the 7th USENIX Conference on File and Storage Technologies. **(2009)**, pp. 125-138.