2018 14th International Conference on Computational Intelligence and Security (CIS)

An improved small file storage strategy in Ceph File System

Ya Fan[1], Yong Wang[1], Miao Ye[1,2*]
1. School of Computer Science and Information Security, Guilin University of Electronic Technology, Guangxi, P.R. China
2. Information Science and Technology, Guilin University of Technology, Guilin, Guangxi 541004, China
(E-mail: 549958504@qq.com; ywang@guet.edu.cn; ym@mail.xidian.edu.cn)

***Abstract*—As the double-write and backup strategies cause low efficiency of storing massive small files in Ceph FS (file system), we design a framework SFPS (Small File Process System) which adopted three technologies, including: k-means, Simhash and Analytic Hierarchy Process (AHP). The designed method can reduce the quantity and redundant data of massive small files by merging similar files after the deduplication with adaptive block skipping. Due to the associated files have high probability to be read in the next time, we also design a pre-fetching mechanism and metadata management module based on the high-efficiency database Redis to guarantee a high efficiency of read rate. The proposed scheme aims to achieve a better trade-off among the utilization of space of hard-disk and bandwidth resources, file access time, hard-disk I/O as well as the cluster performance in Ceph FS by eliminating duplicate copies of repeating data, merging similar small files, and introducing the cache module. Experimental results show that the method presented in this paper can not only effectively improve the utilization of bandwidth resources and space of device storage as well as the small file read rate, but also significantly reduce the amount of disk I/O generated by reading and writing files.**

Keywords-component; Ceph FS; Small files; Clustering; Deduplication; Caches

I. INTRODUCTION

Since the Ceph distributed file system has been launched, it has attracted great attention from major enterprises and scientific research fields by its high expansibility, reliability, high performance and decentralized functions [1]. Librados is usually adopted to interact with Ceph directly which includes a messaging layer protocol in the form of a library. The main steps to write data into Ceph FS with librados are as follows: 1. Map files to PG (Placement Group) groups through the Hash function. 2. Get the latest cluster map from mon node to calculate OSDs list through CRUSH algorithm. 3. Connect with main OSD node directly to write data. There are two typical advantages of storing massive small text files in Ceph FS. Firstly, the CRUSH [2] algorithm was used for addressing to avoid some shortages, such as single point of failure, lack of storage space, performance bottleneck and other problems caused by center nodes when storing massive small files. Secondly, each OSD has a journal to quickly respond to the write requests, which allow the file to firstly be written to journal and then get written to the Ceph FS at appropriate intervals.

II. RELATIVE WORK

We list some methods presented by many authors about how to improve the efficiency of storing small files and get a higher data deduplication rate as well as increase the read rate from file systems. Song Xiaodong [3] and Tie Li [4] provide a storage optimization scheme for massive small files on Hadoop platform. Both schemes use thermal storage algorithm to combine small files with high frequency in a period of time into large files for storage and use the mechanism to reduce the storage space of namenode and increase the file read rate. You Xiaorong [5] presents an optimized method to store massive small education files on Hadoop, which merges related small files to a big file and designs the cache mechanism to improve the efficiency of file access. Yan Fang [6] gives a data de-duplication method for OpenXML compound files based on object, which aims to improve the deduplication ratio by eliminating replicas of extracted atomic objects, but this method sacrifices metadata storage space, index-efficiency, and time complexity of computation. Peng Shuanghe [7] gives a method of deduplication based on Simhash to eliminate duplicate copies of repeating data based on paragraphs in Chinese text files, and get paragraphs with higher similarity chunked to increase the deduplication ratio. Wang Qingsong [8] gives a method of data deduplication with secondary index based on Simhash, which improves the efficiency of data deduplication by leveraging the primary index to locate the secondary index.

III. STATEMENT OF THE RESEARCH PROBLEM

Although Ceph gets rid of the performance bottle-neck caused by the center nodes while storing massive small files, there are still many problems need to be solved in two aspects mainly:

While using the interfaces of librados to write a file to Ceph FS under the premise of three replicas, due to the double-write and backup strategies, the file will produce 6 times of disk I/O to finish the whole writing procedure. Writing massive small files to Ceph FS directly will generate lots of disk I/O which could affect the overall performance of the cluster and reduce the efficiency of writing.

While reading a file from Ceph FS, as Ceph FS is built on Linux OS, it usually takes three disk I/O each time to read a file. When faced with large amount of file index, it's harder to get all the information to memory with one disk I/O, which lead to the inefficiency of reading.

978-1-7281-0169-9/18/$31.00 ©2018 IEEE
DOI 10.1109/CIS2018.2018.00116

According to the problems described above, storing massive small files to Ceph FS directly will affect the efficiency of file access. So as to optimizing the file access performance of storing massive small files in Ceph FS, this paper proposed a structure SFPS to integrate lots of mussy, massive and redundant small files to a few big files with little redundant data, which could increase the utilization of bandwidth resources and space of device and reduce the write I/O of disk. Besides, we also give a metadata management module and a cache mechanism, which could improve the efficiency of reading files and retrieval greatly.

IV. DESIGN AND IMPLEMENTATION OF SFPS

A. Structured Flowcharts of SFPS

To optimize the efficiency of storing massive small files in a distribute file system, a normal method to is to merge small files into a big file which could efficiently reduce the number of small files. But because of the large redundant data of merged files, the utilization of bandwidth resource and storage space is decreased, thus we present a framework SFPS to preprocess massive small files with three steps: clustering, deduplicating and merging, which could increase the information entropy of big files and the utilization of bandwidth resource and storage space. Though the smaller the chunk size is, the higher the deduplication ratio is, smaller chunk size will generate more metadata related which will waste the space of memory and decreases the file read rate, thus we design a structure to get a better trade-off between deduplication ratio and metadata, where we use two times of clusters and deduplications. We firstly use WFD (Whole file detection) to eliminate replica data after first clustering, and then we use the Hamming distance of Simhash digital fingerprint of file in first result to generate a matrix as the input of second cluster. After two clusters, small files in one cluster are greatly similar to each other, then we choose those files in a fixed value in the matrix to perform CDC operation. We use the AHP to determine the fixed value of k of k-means and the threshold of Hamming distance to divide files with remarkably similarity into chunks. All the metadata generated in SFPS is stored in Redis. The structure of SFPS is shown in Fig. 1:

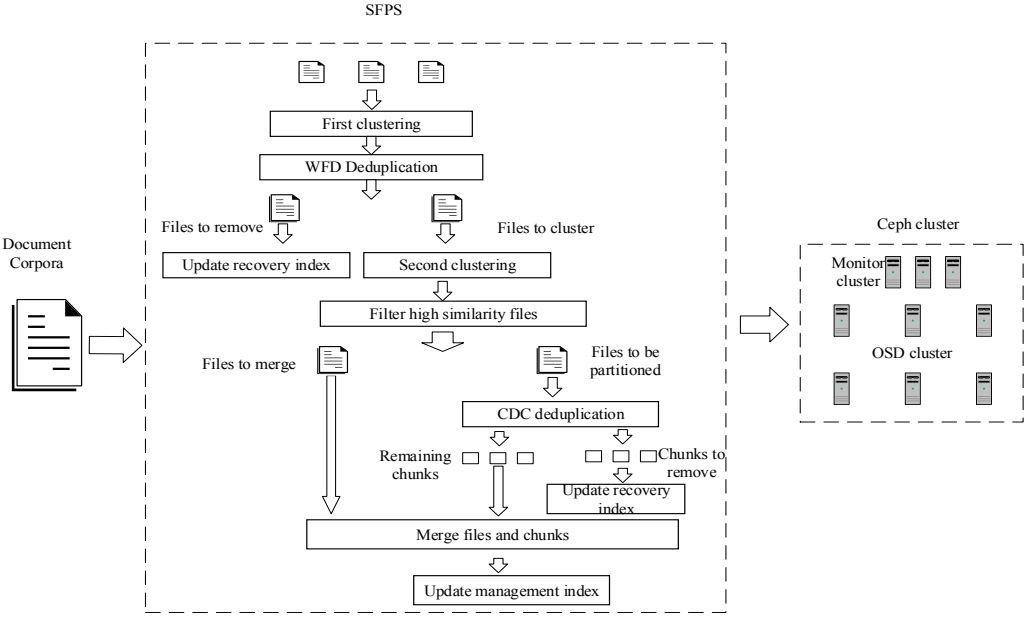

Figure 1. The frame diagram of SFPS

B. Process of Implement SFPS

In the first cluster, we transfer massive small files into a matrix which is used as the input matrix of first k-means. Then we use the WFD technology to eliminate files with same hash value generated by MD-5 in one class. After that we use AHP to determine the optimal clustering center of secondary cluster and the threshold value for partitioning.

After finishing the first cluster and deduplication, we calculate the hamming distance matrix as the input of secondary cluster. The hamming distance in the matrix is distributed in a range of 0~50. In order to increase convergence speed of secondary clustering and improve the accuracy of the clustering, z-score standardization is applied to the generated hamming distance matrix. We use 500 small files in one class as the experiment data to choose the clustering center and the threshold value for partitioning by AHP. The ratios of the result of different schemes to the result of CDC technology about the eliminated data, recover metadata and management metadata are applied as the attribute values of multi-attribute decision matrix. Due to the large number of combination schemes of hamming distance d and primary center of k-means k, selection method is used

to screen schemes to reduce the workload of evaluation, the matrix with linear transformation is shown in TABLE I. .

TABLE I. FINAL MULTI-ATTRIBUTE DECISION MATRIX

j / i	z1(y1)	z2(y2)	z3(y3)
1	0.987	0.012	0.015
2	0.959	0.032	0.049
3	0.960	0.042	0.060
4	0.941	0.048	0.070
5	1	0	0
6	0.993	0.013	0.016
7	0.983	0.017	0.028

Then we give a judgment matrix A, in terms of the relative importance rank value of attributes given by Saaty. The importance in A of the ratio of deduplication data, recovery metadata and management metadata is judged by the value of storage space where they are stored. In that recovery metadata and management metadata is stored on memory, but files are stored on hard-disk in our experimental environment, thus the weight of three attributes is given by 7, 2 and 1 respectively. The judgement matrix is shown in (1):

$$A = \begin{bmatrix} 1 & \frac{7}{2} & 7 \\ \frac{2}{7} & 1 & 2 \\ \frac{1}{7} & \frac{1}{2} & 1 \end{bmatrix} \quad (1)$$

CI (Consistence index) is introduced to measure the estimation consistency of the elements in matrix A, the formula is shown in (2).

$$CI = \frac{\lambda_{max}-n}{n-1} \quad (2)$$

Where λ_{max} denotes the largest eigenvalue of A, n denotes the quantity of targets. We figure out the eigenvalues of A are 3,0,0, the largest eigenvalue of A is 3, hence CI is 0. Then we can use the CR (consistence ratio) to determine whether the matrix A can be accepted, which can be calculated by the formula CR = CI/RI，where RI (random index) is the random index of matrix. We calculate that CR = 0 is small than 0.1, which means A passed the consistency test. The weight vector of A is calculated as

$$Aw=\lambda_{max}w \quad (3)$$

The computed result of W is [0.952,0.272,0.136] ^{T}. Then we use the weight sum method of multiple attributive decision to select a best scheme. The C_i is computed as (4)

$$C_i=\sum_{j=1}^{n} W_j Z_{ij} \quad (4)$$

Where W_j is the weight of attribute, Z_{ij} is the attribute. The weighted sum is shown in TABLE II.

TABLE II. WEIGHTED SUMS

i	1	2	3	4	5	6	7
Ci	0.945	0.928	0.934	0.918	0.952	0.951	0.944

As is shown in TABLE II. , plan 5 is best one. Therefore, 2 is assigned to the initial clustering center of the secondary cluster. After the secondary clustering, we choose files with the hamming distance less than 24 to be partitioned. After the two data deduplications by the technology of WFD and CDC, the remaining files and chunks in one class are merged into a big file, and all the management metadata related is stored in Redis database m-db.

C. *Reading Process*

File reading process is displayed in Fig. 2.

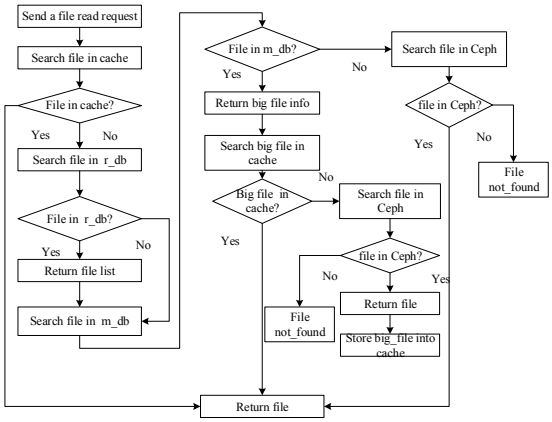

Figure 2. Process of reading file

V. EXPEREMENTS AND RESULTS

We use the small Chinese text files in a size of 10-20kb in our experimentation. Different number of small files are respectively written to Ceph FS to test the data deduplication ratio and the hard-disk I/O and read out to get the average access time.

A. *Data Deduplication Ratio*

The original Ceph FS, WFD, FSP (fixed-sized partition), CDC and FKMS presented in this paper were tested for the data deduplication ratio which reflects the utilization of the space of device in Ceph and the bandwidth resources.

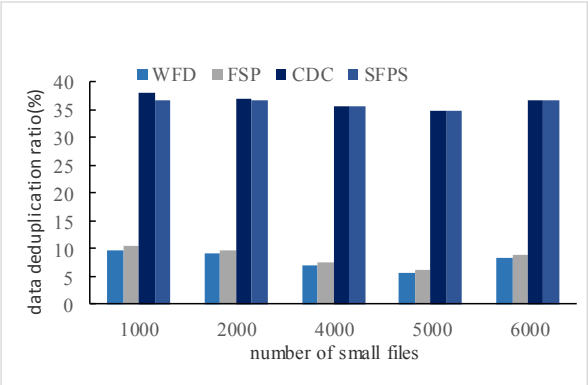

Figure 3. Data deduplication ratio

It can be seen from Fig. 3 that the data deduplication ratio of WFD, FSP, CDC and SFPS is changed on an even keel. WFD and FSP has a similar dada deduplication ratio, but FSP is slightly better than WFD with a number around 8%. CDC and FKMS has a similar number of data deduplication ratio which comes up to 35%. FKMS is slightly lower than CDC for the reason we only choose those files with higher similarity to be partitioned, which aims to increase the utilization of storage space for storing metadata.

B. Disk I/O

We get the quantity of disk I/O generated by writing massive files to Ceph FS directly and writing the big file processed with C&M (clustering and merging) used in [9] and SFPS presented in this paper.

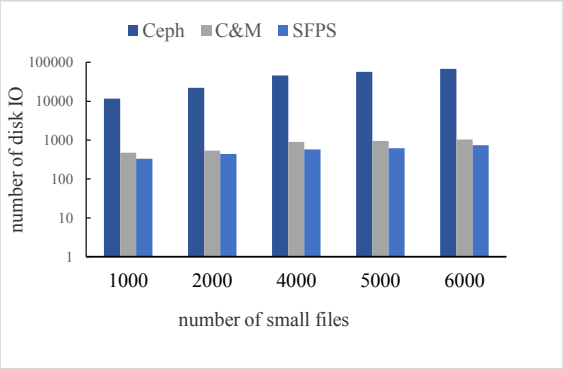

Figure 4. Disk I/O

In Fig.4, the disk I/O generated by uploading merged files in C&M and SFPS is significantly less than uploading massive small files directly in Ceph FS, which constitute 1% of Ceph. The disk I/O of SFPS is always small than the two other methods, and SFPS is account for 70% of C&M.

C. File Access Time

In this paper, we use python interface of Ceph librados to read small files. The average time of file access of original system and our method in this paper is tested. the test results are shown in Fig. 5

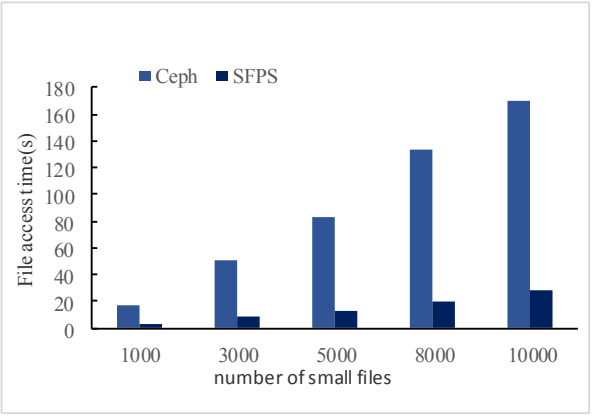

Figure 5. Fils access time

As can be seen from Fig.5, The read rate of SFPS is significantly higher than that in original system even in the worst case. For the same number of small files, the average reading time of FKMS is only about 16% of the original system.

VI. CONCLUDING

In this dissertation, we have firstly described the problem existed in distributed file system when in face of storing massive small file, then we analyzed certain method presented by other authors for improving the efficiency of storing massive small files. Finally, we give a structure SFPS to dispose the massive small files to reduce the quantity of files and the redundant data in big files which merges small files in one class. The experimental results show that the scheme we designed can efficiently decrease the amount of disk I/O and the average read time, as well as the occupancy rate of hard disk of Ceph storage cluster. In the future, we will focus on how to improve the efficiency of writing files and the mechanism to eliminate of cache in Redis.

ACKNOWLEDGMENT

This work is partly supported by the National Natural Science Foundation of China (Nos.61662018, 61831013), Project Funded by China Postdoctoral Science Foundation (No.2016M602922XB), Scientific Research and Technology Development Project of Guangxi (No. 1598019-2).

REFERENCES

[1] A. Weil, Sage & Brandt, Scott & Miller, Ethan & Long, Darrell & Maltzahn, Carlos. (2006). Ceph: A Scalable, High-Performance Distributed File System. OSDI. 307-320.

[2] S. A. Weil, S. A. Brandt, E. L. Miller and C. Maltzahn, "CRUSH: Controlled, Scalable, Decentralized Placement of Replicated Data," SC 2006 Conference, Proceedings of the ACM/IEEE, Tampa, FL, 2006, pp. 31-31.doi: 10.1109/SC.2006.19.

[3] Song Xiaodong. Study on performance of storing small files in Hadoop distributed file system [D]. Beijing jiaotong university,2017.

[4] Tie Li, Yan Cairong, Huang Yongfeng, et al. Small file access optimization method for Hadoop distributed file system [J]. Computer application, 2014, 34(11):3091-3095.

[5] You Xiaorong, Cao sheng. Storage Reserch of Small Files in Massive Education Resource [J]. Computer science, 2015, 42(10):76-80.

[6] Yan Fang, Li Yuanzhang, Zhang Quanxin, Tan Yu'an. Object-Based Data De-Duplication Method for OpenXML Compound Files[J]. Journal of Computer Reserch and Development,2015,52(07):1546-1557.

[7] Peng Shuang-he, Tuergong.MAITISABIER, Zhou Qiaofeng. Inverstigation on Deduplication Technique of Chinese Text with Simhash [J]. Computer Technology and Development, 2007,27(11):137-140+145.

[8] Wang Qingsong, Ge hui. Secondary Index Deduplication Algorithm Based on Similar Clustering [J]. Journal of Chinese Computer System,2017,38(12):2797-2801.

[9] zhang Bitao. Efficient Method for Storing Massive Small Files in Ceph [C]// academic annual conference of China communications society. 2014.