An Efficient Approach to Manage Small Files in Distributed File Systems

Aakash Patil, Ganesh Sagare, Kunal Saraf

(BE in Computer Engineering, Sandip Institute of Engineering and Management, Nashik.)

Prof. Sujit. A. Ahirrao

Assistant Professor, Department of Computer Engineering

---***---

Abstract: Nowadays, to manage excessive number of small files is became a challenge in Distributed File System. Currently, the combined block storage technique is used to store the files this technique is used in existing system such as Extfs and Xfs. This technique is liable to inefficiency when accessing files randomly. We present the proposed system to manage small files which is based on simple metadata and storage architecture.

Our system focuses on replacing the existing system drawbacks in Data servers that used to store excessive number of small files and retrieval of files in a better way. We designed new metadata structure which will decrease the size of original metadata that will help to increase the speed of file accessing.

Keywords:

Information System , Information Storage And Retrieval. Indexing Methods, Content Analysis Computing Methodologies, Documents Processing, Various types of files.

1.Introduction:

We know that Metadata consist of data related data that means in file system metadata contains the information which is helpful to search the files in file systems for eg. Address of the file, size of the file, modified date of updated information etc.

Nowadays, Everyone is using social networking and e-commerce websites for communication and purchasing purpose by considering the usage of the websites which required to store the data which is small in size then there is the difficulty in storing and retrieving the files which are smaller in size and the number of this files are bulk because of many users are frequently uploading or modifying the data in the storage space.

So, the managing this small files is became a problem in distributed file system because of the metadata generated by the files is bigger in size. In some cases the files are rarely modified or updated and the size of this file is in between 1kb's to 10kb's such as pictures, text etc. uploaded on social networking and e-commerce websites in daily or timely basis. Distributed file system is based on storing and accessing files based on simple client-server architecture. In distributed file system all data is copied and placed on the different data servers and the information about the data is stored in which are then connected in network.

A client or user searches the file using metadata server other than the using the actual location of that file the same process is used in existing system, client request the file which is stored in a distributed file system by using two phases.

1.Client sends the query containing about the data needed to the metadata server and gets the IP address of data server which stores the target file.

2.In next phase connection between data server and user is established and granted for fetching the data file.

Why we are shrinking the size of metadata ?

In our proposed system the main reason behind shrinking the size of metadata is, in DFS when we are storing the file, the size of its metadata is big in size because of it contains every attributes as discussed earlier. Because of these the accessing speed of a particular file takes more time. In our system the metadata will contain only two things that are size of the file and physical address of that file so that accessing speed can be increased.

2.System Architecture:

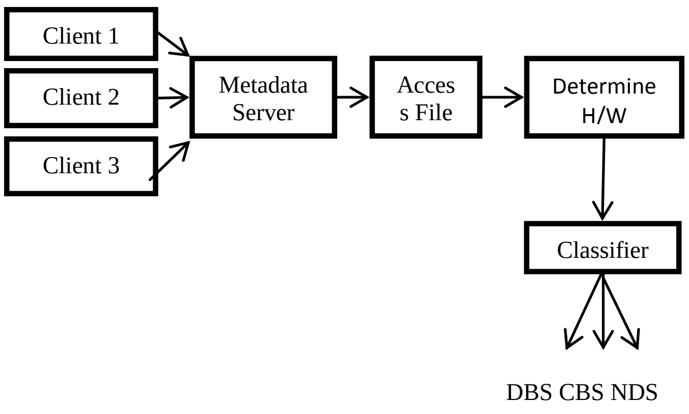

Fig 1. System Architecture

The system architecture defines the flow of accessing the files from the Distributed File System.

- Clients are the actual users who query the required file to obtain the contents of data.
- Metadata server contains the metadata (contains file attributes) of the files.
- Access File is the file which is requested by the client.
- The component determine hardware configuration detects the system hardware configuration of the client side system that will decide the client can access or cant access the data from the distributed file system.
- Classifiers classifies the data types in three ways:
- DBS(Divided Block Storage)
- CBS(Combined Block Storage)
- NDS(NoSQL Database System)

In existing system the classifiers are used to locate the files in three different locations as shown in Fig 1. DBS contains the large files, CBS contains the small files and NDS contains the byte level files so because of the three approach the time complexity is increased so in proposed system we are combining these three techniques in a single storage architecture that will help to increase the performance of the accessing time and reduces the time complexity.

3.Literature Survey:

Granrt Mackey,Saba Sehrish,Jung Wanvg (Granrt Mackey, 2009)***:*** In this paper it is given about to improve metadata management for small files in HDFS. This scheme is based on the assumption that each client is assigned quota in file system for the SPACE as well AS NUMBER OF FILES. the compression method "harballing" provides by hadoop is used.

Qinqin He,Zhanhuai Li,Bo Wang,Huifeng Wang,Jian Sun: (Qinqin He, 2011) In this paper the scientist has given about how to enhance system's performance and how to optimize system under the different configuration the future work.

Randolph Y Wang,Thomas E Anderson: (Randolph Y Wang, 1993) In 1993 generation of file system an inadequate in facing challenges of wide area networks and massive storage. XFS is a prototype file system developed to explore the issues brought about by these technology advances. It organizes hosts into a hierarchical structure so, locally within the cluster of workstation can be better exploited. XFS achieve better performance and ability then current generation network file system runs in wide area..

S. Anjanadevi,D. Vijaykumar. Dr. K. G.Shrinivasan: (S. Anjanadevi, 2014) Cloud computing is an emerging computing model wherein the tasks are associated to software, combination of connection and service accessed over network.

Xian Tao, Liang Alei (Xian Tao, 2014)***:*** small file access management based on GlusterFS is a strategy to optimize small files reading and writing performance on traditional distributed file system.

Tao Wang, Shilong Yao, Lian Xiong, Xin gu (Tao Wang, 2015)***:*** HDFS,DFS are adopted to support cloud storage and are designed for optimizing large file access but unfortunately the problem of massive small files is neglected and seriously restricts the performance of DFS. To improve and even solve the small files problem in this research user task access is defined. The co-relation among the access task, application and access fie are constructed by improving PLSA and research object is transformed from file level to task level

Songling Fu, Liagang He, Chenlin Huang and Keni Li: (Songling Fu, 2015)The processing of massive number of small files is challenge in the design of distributed file system currently the block-storage is used it causes inefficiency when accessing small files. iflatLFS is used to manage small files which are based on metadata scheme and flat storage architecture.

4.Mathematical Model:

Figure 2 shows the Mathematical model
Where,
Let the system is decided by,
S= {D,CS,AF,MS,C}

D: Data (Text,Image,Video)

CS: Client Search*:* Request for required data such as text, images, video audio Retrieve the data from server.

MS: Metadata Server : Queries locally for id of data block IP of all of data server Retrieve id of data block IP address of data server to client.

AF: Access File: Files which are requested by client such as text, multimedia files

C: Classifier: Classifies the file into different data blocks (Combined block, divided block, No SQL block)

DB: Database: Contains different type of data blocks (Combined block, divided block, No SQL block)

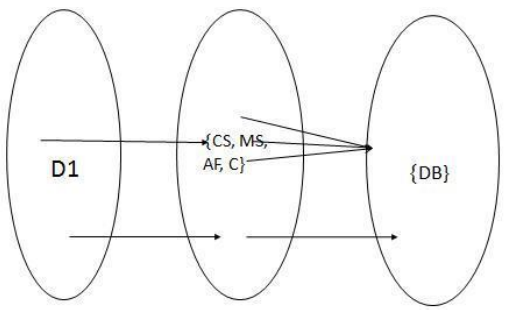

Fig 2. Mathematical Model

5.Methodologies/Algorithm:

Reading local metadata to retrieve the logical address of the target file data in the corresponding data block file. This phase includes three steps:

Phase1:

1. T1:ReadIFInode: Reading the inode of index file.

2. T2:ReadIFData: Reading index data from the index file.

3. TQueryLA: Querying the corresponding index item from the index data to get the logical address of the file.

Phase 2:

Reading file data. This phase includes 4 steps:

T3: ReadDBFInode:Reading the inode of the data block file.

T4: ReadAddressBlock: Reading the corresponding address block from the disk if the logical address is beyond the size of 12 disk blocks.

TQueryPA: Querying the physical address of the target file data from the inode or the address block.

T5:AccessData: Accessing file data using a disk operation.

6.Performance Evaluation:

Firstly, we evaluate the performance of our implementation in a typical DFS environment on a generic system, which has a 2.33 GHz Intel processor with four cores and 4 MB L2 cache, 4 GB of physical memory and a 1 TB SATA disk.

Since the objective of the experiments is to evaluate how well the implementation works the data storage layout in the data servers and the accessing patterns to these stored data were generated in the experiments to DFS context.

It can be seen from Figure that implemented system delivers the higher performance than the ordinary and generally used file accessing that is used in sending and receiving the files from distributed file system.

Figure also show that the performance of our implementation based DFS is better than that of traditional DFS in all ratios. In the experiments, the performance of traditional DFS. This result suggests that implemented project can always deliver better performance than traditional DFS.

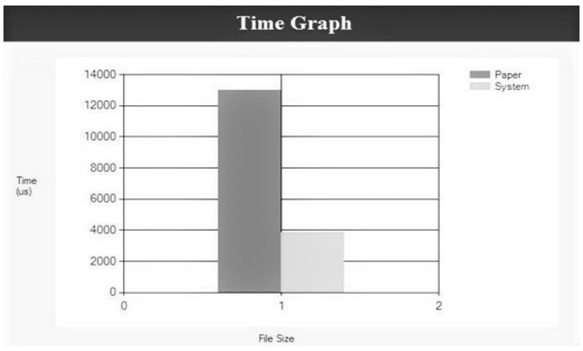

Fig 3. Comparison of performance

Nowdays, there are three types of large-scale distributed data storage systems: the divided-block-storage DFSes, the combined-block-storage DFSes and the NoSQL database systems. The divided-block-storage DFSes, such as GFS and HDFS, are usually used to store big files. But these DFSes cannot deliver the ideal performance when handling small files. The main aim of designing combined-block-storage DFSes, is to solve the problem of accessing massive numbers of small files efficiently. Furthermore, the NoSQL database systems are mainly designed for storing the data of tiny size.
From the Figure 3 our system can improve the performance of accessing massive numbers of small files with the KB-level size in the combined block storage DFSes. For the files with more than kb level or bigger size the another file storage system such as Divided Block Storage can achieve better performance.

7.Conclusion:

When developing efficient distributed file systems, one of the challenges is to optimize the storage and access of massive numbers of small files for Internet based applications. Previous work mainly focuses on reducing the problems in traditional files systems, which generate too much metadata and causes lack of file access performance on data servers. We focus on optimizing the performance of data servers in accessing massive numbers of small files and present a proposed system which directly accesses raw disks and adopts a simple metadata scheme and a flat storage architecture to manage massive numbers of small files. New metadata generated by our system consume only a fraction of total space used by the original metadata based on traditional file systems.

In this, each file access needs only one disk operation except when updating files, which rarely happens. Thus the performance of data servers and the whole DFS can be improved greatly. This paper finally proposes a hybrid storage system to integrate different storage systems, each of which represents a better solution for different ranges of data sizes.

8. References:

[1].***Grant Mackey, Saba Sehrish, Jun Wang***, (Granrt Mackey, 2009) "Improving metadata management for small files in HDFS," in Proc. IEEE Int. Conf. Cluster Computer. Workshops, New Orleans, LA, USA, Sep. 2009, pp.

[2].***Qinqin He*** (Qinqin He, 2011) Department of Computer Science Northwestern Polytechnic University, Xi'an China luluhe8848@hotmail.com Research On Cloud Storage environment File System Performance Optimization.

[3].***Randolph Y. Wang and Thomas E. Anderson*** { rywang,tea} @cs.berkeley.edu Computer Science Division University of California Berkeley, CA 94720
(Randolph Y Wang, 1993) XFS: A Wide Area Mass Storage File System

[4]. ***S. Anjanadevi, D. Vijayakumar, Dr. K .G. Srinivasagan*** (S. Anjanadevi, 2014) PG Scholar, Assistant Professor, Professor & Head Department of Computer Science and Engineering – PG National Engineering College (Autonomous), Kovilpatti, India An Efficient Dynamic Indexing andMetadata Based Storage in Cloud Environment

[5]. ***Xie Tao, Liang Ale*** (Xian Tao, 2014) Small File Access Optimization Based on GlusterFS ***i*** School of Software Engineering Shanghai Jiao Tong University Shanghai, China Foxterran@163.com, liangalei@sjtu.edu.cn

[6]. ***Tao Wang, Shihong Yao, Zhengquan Xu*, Lian Xiong, Xin Gu, Xiping Yang*** State Key Laboratory for Information Engineering in Surveying, Mapping and Remote Sensin Wuhan University Wuhan, China wangtao.mac@gmail.com (Tao Wang, 2015)
An effective strategy for improving small file problem in distributed file system.

[7]. ***Songling Fu, Liagang He, Chenlin Huang and Keni Li***: (Songling Fu, 2015) The processing of massive number of small files is challenge in the design of distributed file system.