HDFS distributed metadata management research

An-ping Xiong & Jin-yong Ma

College of Computer Science & Technology, Chongqing University of Posts & Telecommunications, Chongqing 400065, China

Keyword: hdfs; Metadata Management; Consistency; Two-phase commit protocol

Abstract. As it can provide high throughput data access, and easy to deploy on a cheap machine ,Distributed File System HDFS is used widely.All HDFS file access requests are required by the metadata server node .so the single metadata server nodes often become the bottleneck of the whole system problem, this paper design a multi-node distributed metadata management solution in the directory Hash-based metadata storage strategy, proposed a protocol based on two-phase commit to ensure distributed metadata consistency. Experimental results show that the program can improve HDFS file system metadata management efficiency.

1. INTRODUCTION

In recent years, most of the mass storage system uses a distributed file system, a common practice is that metadata access seperate from file data access (1), assume metadata service by independent metadata service system. According to the survey, in a distributed file system, about 50% -80% of the access operation objects are metadata (2), the throughput and access metadata management system response time greatly influence the Distributed File System performance. To solve the bottleneck problem of metadata server access exists,multivariate metadata server solutions currently are often used , but the metadata in different ways according to their organization, there will be metadata consistency problem.

2. BACKGROUND AND RELATED WORK

According to the HDFS (Hadoop Distributed File System) Namenode (metadata node) single point bottleneck problem, there are many research institutions, enterprises put forward the corresponding solutions. (1) HDFS Federation (4) is the latest released version of Hadoop Hadoop-0.23.0 that it solve the HDFS single point of failure, through the use of multiple independent Namenode to make HDFS namespace services can scale horizontally, in the architecture all Namenodes are the alliance,they are independent and does not require coordination between them. This method is based on the original single point of failure to solve the problem, but the loss of a unified cluster name space management, a single failure can cause Namenode corresponding data inaccessible, and it does not support independence unified namespace;(2) Facebook gives increased Avatarnode (5) of the program, in order to solve the original architecture Namenode restart recovery time can not meet long service 24 * 7 online applications and services, Avatarnode provide hot standby mechanism, which is to ensure Primary Avatarnode and Standby Avatarnode two nodes running Namenode code while one external services, another node at any time make a handover preparation, and strive to control failover time to seconds. In this architecture an additional NFS (Net File System) server is shared as metadata intermediaries between a Primary Avatarnode (master) and Standby Avatarnode (hot standby node) to ensure consistency between the data ,in the cluster Datanodes (data nodes) need to report a heartbeat message to the Primary Avatarnode and Standby Avatarnode, without distinguishing between primary and secondary. The program implementation mechanism is relatively complex, limiting its application and promotion; NFS server is a single point, NFS has become a performance bottleneck in the system; only one backup node, no large-scale expansion.(3) Taobao's TBFS (3) (TaoBao File System) is a solution for a single point in HDFS Namenode node bottlenecks, TBFS achieve horizontal expansion Namenode node through stateless manner.In order to achieve a stateless Namenode node the key

part of the data structure retained in memory before Namenode node needs to be moved to a third party, and persistence saved. The part of the state information stored in the database, a large number of database operations can cause system performance bottlenecks; unrealized distributed Namenode services.

3. THE MULTI-NODE DISTRIBUTED METADATA MANAGEMENT SOLUTION

To solve the problem of the traditional metadata server bottlenecks existed HDFS systems, on this paper distributed metadata management solutions adopt multivariate metadata server and the directory structure of the Hash-based metadata storage strategy, and on this basis, given the two-phase protocol to ensure metadata consistency ,it make HDFS to provide an effective metadata management solutions.

3.1Multivariate metadata server architecture

Figure 1 shows a master/slave cluster architecture used in this paper, which is a HDFS cluster has multiple Namenode, a Secondary Namenode and a certain number of Datanode.

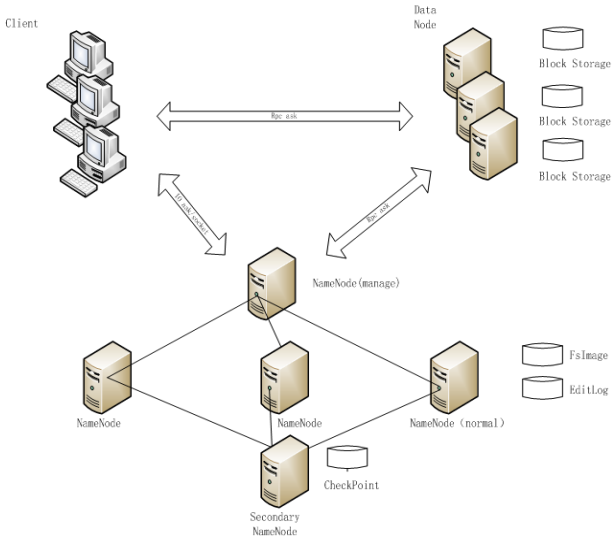

Figure 1 HDFS multivariate metadata server architecture

Namenode is the center of HDFS cluster architecture. First, each node only save part of the metadata information, namespaces of metadata distributed to each metadata node of the entire system ; metadata information is stored in two main documents, as the image file fsimage and log files editlog, fsimage storage file system namespace, editlog lasting record every change to the file system metadata; node is also responsible for processing requests from clients unified and data nodes metadata operations. These nodes have a management node, the others are ordinary nodes; management node in addition to a common node functions but also have to monitor the role of other nodes, including timely gathering ordinary node's operation by heartbeat , save a copy of the entire file system metadata.

SecondaryNamenode has two roles, one is image backup, the second is regularly merging logs and image. Two processes at the same time,as it calls checkpoint, the role of image backup: backup fsimage on the Namenode node, preventing fsimage file unusable due to Namenode failures ; log and image merge regularly role: regularly mergering Namenode log and fsimage prevent edlitLog files too large to affect the efficiency of loading editLog.

Datanode is responsible for splitting files into small files block to store in the local file system . Client read and write file blocks , first acquires position information of the data block by Namenode, then reading and writing with Datanode subsequent interactive communication. Meanwhile Datanode require regular reporting information and data blocks status by heartbeat to Namenode.

3.2 Hash directory-based metadata storage strategy

Depending on the property division, the metadata is divided into two kinds, namely the directory attribute metadata and file metadata (7), directory metadata includes the directory name, directory

ID, access permission,parent directories, subdirectories, access time, modification time , the height of the global directory, contains tags and other attributes. Each directory when you create will be assigned a globally unique identifier for metadata management system; file metadata contains the file name, file identification, directory mapping DataNode node, block number, block number of copies and other information;

For multivariate data server architecture, metadata organization and division on the server is an important factor in determining the performance and scalability of data services, and the more popular methods are directory subtree partitioning algorithm (6) and the hash algorithm (9) two categories .

To solve the problem of metadata local hot spots and ensure the integrity of the local directory, this paper complex the advantages of directory subtree and hash algorithm, Reference literature(10) two-stage mapping mechanism based on catalog division and consistency Hash algorithm to manage the namespace method used in the system in CLover, this paper adopt the method hash-mapping based on sub-tree directory to store distribution metadata. every directory set a unique identifier, in order to ensure the uniqueness of identifiers, avoid Hash-conflict, specifically this article firstly make directory convert 32 unique identifier with the MD5 way, then hash the identifier, lastly directory metadata and file metadata mapped are distributed to each node, for example, shown in Figure 4, the tree after hash map, directory A and the file file1 and file2 metadata in directory A can be distributed to the node Namenode0, distribution directory B on Namenode1, directory /B/C and file3 filesand file4 metadata in that directory are distributed to the node Namenode2;

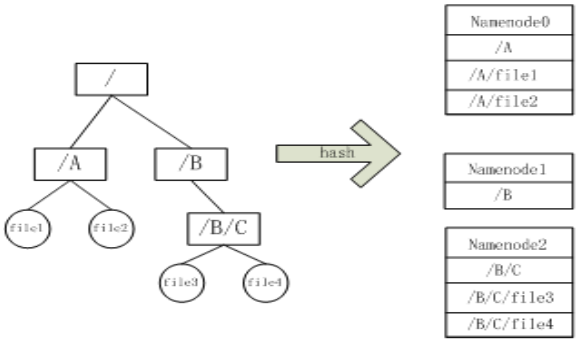

Figure 2: Hash-based directory mapping table

As can be seen from Figure 2 file metadata and its parent directory is on the same metadata server, but between metadata catalog may not be on the same server, then some metadata operations may involve a different directory, we define each directory is an object,there are mainly two types involving in catalog metadata operations , one type of operation only involves an object, such as getattr, setattr, readdir operation and so on . Another operation involves multiple objects, such as createdir, mkdir, rmdir, etc. need to involve two objects: to be created /deleted file or directory itself and its parent directory, we know that in HDFS, the file metadata and catalog metadata in the name of the node is represented by inode, inode record a variety of attributes, including permissions, modification and access times, naming space and disk space allocated, etc., to create a directory C under directory B , for example, this time involving to modify multiple metadata, specific algorithm steps are as follows:

(1) According to Hash rule, through the catalog /B/C and /B corresponding unique identifier to find the corresponding metadata server Namenode2, Namenode1;

(2) allocated a free inode to the newly created directory on Namenode2, and modify the inode bitmap;

(3) allocate storage space for the newly created directory and modify the free block bitmap, inode contents of the new catalog is filled;

(4) modify the parent directory inode on Namenode1 , including nlink count, modification time and access time;

3.3 Consistency policy based on the two-phase commit protocol

Distributed metadata management need multiple Namenode work, metadata storage strategy adopted the hash-based directory, once there is a server crash in the metadata change process, it may appear metadata inconsistencies between multiple Namenodes, as previously description, create the directory /C at /B directory, involves modifying corresponding inode information on Namenode1 and Namenode2 , if in the revision process, the emergence of a server crash, will cause the two metadata service node corresponding inode information inconsistent. Requiring the use of atomic commit protocol to ensure metadata consistency.

For the reduction of metadata processing protocols and improve metadata processing performance considerations, this paper assumes that the distributed metadata operations involving up to two metadata servers,the hypothesis consistent with the vast majority of cases.

two-phase commit protocol (12) (8) (11) to ensure the consistency of metadata operations, in distributed metadata operation only involves two Namenode, which we regard the Namenode received from the client request as a coordinator, another Namenode participated in the operation as participants. A case study in metadata operations Mkdir, create a directory /C under /B directory , you need to create inode on the namenode directory /B/C corresponded, fill related content in the inode, at the same time need to modify related information containing nlink count, modification time and access times on the parent directory /B corresponding metadata server Namenode1 , this paper based on the two-phase commit protocol metadata consistency processing flow shown in Figure 3;

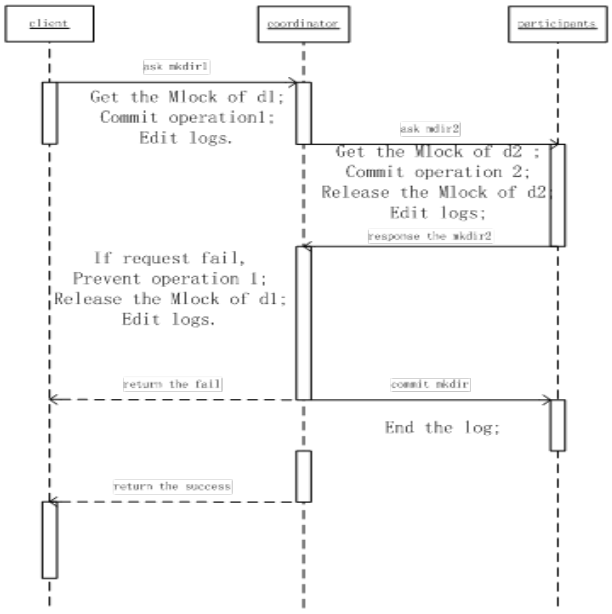

Figure 3 mkdir two-phase commit protocol processing

In this process protocol, assuming Namenode implementation is non-blocking, each node involves two steps in the whole process , the entire metadata operation involving multiple NameNode is regarded as an atomic transaction. Specifically coordinator and participant node process is as follows:

(1) The first is the voting phase, the coordinator check the status of the participants, the participants sent in response to a request coordinator, if the participants is in a failed state, revocation mkdir affairs, writes "Transaction mkdir undo" record in the log ; if participants is in the normal state, the coordinator writes the log "Transaction mkdir start" record in the log, process flow proceeds to the implementation phase;(2) In the implementation phase, first,according to the message received by participants retruned coordinate makes a decision on whether to commit or revoke the global mkdir transaction, if the participants back to the "Proposal submitted" message, the coordinator writes "Transaction mkdir Submit" record in the log , and publish execute transactions mkdir command,participants perform related mkdir operations, if the participants back

to "propose to remove" message, the coordinator writes "Transaction mkdir undo" record in the log and undo the transaction, participants write "cancel transaction mkdir" record in the log.

In order to ensure consistency of metadata, the metadata server using distributed logging technology, on each node metadata modification logs are recorded and stored in the local file, called the log file, each sub-operation log records of a distributed operation are located in the log file of each node , on the recovery time ,it needs that related metadata server consult every sub-operations in accordance with their logging.

4. EXPERIMENTAL DESIGN

4.1 Experimental environment

In this paper, the experimental environment builds a seven nodes in a distributed system, the configuration of each node as follows: 6-core Intel Xeon CPU 2.4GHz, 8G memory, 10T hard drive. Gigabit fiber network environment. Two machines for Namenode, 1 station for SecondNamenode, the remaining four machines all make DataNode, the operating system installed on each node for Cent OS5.4, Hadoop version 2.0, JDK version 1.6.0_35.

4.2 Experimental Methods

The main experimental validation data in a multi-server environment, whether metadata processing request performance can improve. This article metadata processing performance test used SliveTest, SliveTest located the test package of hadoop, build rpc requests through a large map, detect Namenode performance, specific approach: the use of multi-threaded SliveMappe, in a multi-threaded multi-map simulated client sends requests,each client metadata operations 100 000 times to count the total time of the file system server processing takes these requests, and then calculate the number of requests processed per second, as the average processing capacity ,mkdir is an example.

4.3 Experimental results and analysis

The average processing capacity of system shown in Figure 4,

Figure 4 system average processing capacity

the experiment observations the metadata processing performance change when different number of clients do mkdir operations in the improved architecture and original architecture. As can be seen from the figure, when the number of clients to 7, HDFS metadata processing capacity limit is reached, further increase the number of clients will not significantly improve their ability to respond to the request. For improved architecture, along with the number of clients increases, the ability to enhance the metadata processing operations, the average number of requests per second that can be processed gradually increased. When the client does not reach the number 7, improved architecture in performance, while still below the HDFS, which is due to the improved architecture mkdir operating involved two objects, the need for distributed transaction processing, through a two-stage commit protocol to ensure consistency of the metadata, adds extra overhead synchronization between nodes, therefore enhance the finite element data processing performance. Subsequently, when the number of clients the advantage over 7, the load increased, improved concurrent processing architecture emerged, compensating for the loss of performance caused by a distributed transaction, the average request processing capacity is higher than the final HDFS, and with the

number of clients increases, larger performance gains.

In summary, At the request of the client data quantity is less, more efficient single node,when the volume of requests is increasing, distributed metadata management has a better performance, At this time the consume due to maintain the consistency of performance are negligible .

5. CONCLUSIONS

In this paper, the current system HDFS metadata exists single point of failure and a mount of client data requests cause the system to deal with excessive high latency problem, design a solution for multi-node management Namenode metadata strategy, Based on the metadata storage strategy under the Hash directory, gives a method based on the two phase commit protocol to ensure consistency of distributed metadata,Experiments show that, due to the use of multiple Namenode node architecture, name space of the whole system distribution storage to each node, when the number of client requests continuously increases,it can handle the request to each node parallel distributed processing, improved program performance, compared with the original metadata processing architecture has also been improved.

6. REFERENCES

(1) Wang J, Feng D, Wang F, et al. MHS: A distributed metadata management strategy[J]. Journal of Systems and Software, 2009, 82(12): 2004-2011.

(2) Ousterhout, J. K., Da Costa, H., Harrison, D., Kunze, J. A., Kupfer, M., & Thompson, J. G. (1985). A trace-driven analysis of the UNIX 4.2 BSD file system (Vol. 19, No. 5, pp. 15-24). ACM.

(3) Liang Liyin,Ali Hadoop cluster architecture and service system of [C]. Hadoop and big data technology conference (HBTC 2012), Beijing,2012

(4)Apache.HDFSFederation[OL] .http://hadoop.apache.org/docs/r2.0.3-alpha/hadoop-project-dist/hadoop-hdfs/Federation.html, 2013

(5) D. Borthakur. Hadoop avatarnode high availability[OL]. http://hadoopblog.blogspot.com /2010/02/hadoop-namenode-high-availability.html, 2013

(6) Zhou G, Lan Q, Chen J. A Dynamic Metadata Equipotent Subtree Partition Policy for Mass Storage System[C]//Frontier of Computer Science and Technology, 2007. FCST 2007. Japan-China Joint Workshop on. IEEE, 2007: 29-34.

(7)Ross R B, Thakur R. PVFS: A parallel file system for Linux clusters[C]//Proceedings of the 4th annual Linux showcase and conference. 2000: 391-430.

(8) Bernstein P A, Hadzilacos V, Goodman N. Concurrency control and recovery in database systems[M]. New York: Addison-wesley, 1987.

(9) Allasia W, Gallo F, Milanesio M, et al. Indexing and Retrieval of Multimedia Metadata on a Secure DHT[J]. Informatica (Slovenia), 2009, 33(1): 85-100.

(10) Zhou Jiang, Wang Weiping, Meng Dan, et al. The key technology of [J]. oriented distributed file system data analysis.Journal of Computer Research and Development, 2014, 51(2): 382-394.

(11) Özsu M T, Valduriez P. Principles of distributed database systems[M]. Springer Science & Business Media, 2011.

(12) Gray, J. (1981, September). The transaction concept: Virtues and limitations. In VLDB (Vol. 81, pp. 144-154).