A DESIGN OF EFFICIENT METADATA CLUSTER IN LARGE DISTRIBUTED STORAGE SYSTEMS

LIN XIA[1], HAN-CONG DUAN[1], LIN LI[1], XIAO-WEN NIE[1]

[1]School of Computer Science & Engineering,
University of Electronic Science and Technology of China, Chengdu , China
E-MAIL: margaretxl@hotmail.com

Abstract:

Searching and managing information are becoming increasingly important for large distributed storage systems. This paper presents a metadata management mechanism named CBMM, which is based on the *Chord protocol* to organize the metadata server cluster. Utilizing the advantages of Chord protocol for distributing metadata across a cluster of metadata servers can highly improve the throughput, balance metadata distribution, and afford dynamic scalability of metadata servers.

Keywords:

Metadata management; distributed system; Chord; dynamic scalability

1. Introduction

As the amount of information accessed by users dramatically increases, how to manage the information efficiently becomes more and more important. In large distributed storage system, 50% to 80% of all file system accesses are to metadata[1], while the size of metadata is small compared to the overall storage capacity of such a system. Therefore, performance of the metadata server(MDS) cluster will be critical to overall system.

We have designed and implemented a metadata infrastructure--Chord-based metadata management architecture (CBMM). Chord is a distributed lookup protocol, which provides support for just one operation: distributed a key, it maps the key onto a node[3]. CBMM used Chord build up a DHT network to form a Metadata server cluster. In Modern distributed file systems, metadata and data are separated, and data is stored on devices that can be directly accessed through the network, while metadata is managed separately by MDS. Metadata server cluster manages the namespaces, directory hierarchy, the location information of data, but not involved in the storage and retrieval data. In CBMM, the DHT network achieved metadata distribution and fast location. Based on the Chord protocol, we designed some applications to support serving tens of hundreds of thousands of parallel accessed to metadata servers, at the same time, maintaining standard directory and file semantics, affording directory operation such as *ls*.

2. Related Work

In tradition systems, the core problem is to distribute metadata across metadata servers to provide metadata high availability and metadata servers dynamic scalability. At present，common techniques in existing distributed file systems include directory subtree partitioning and pure hashing. Directory subtree partitioning maintains a good directory structure, which has advantages in developing the locality of directory storage, but fails to balance the workload between the MDS and may easily result in hot spots; Pure hashing uses hashing to widely distribute the namespace among the metadata servers[5]. The hash function tends to balance load, make the metadata distributed more evenly, but the hash function changes would cause a large number of metadata migration. This is not good for metadata server cluster dynamic expansion. In addition, the storage systems research center in University of California, Santa Cruz has developed Lazy Hybrid (LH) and dynamic subtree partitioning metadata management. LH uses a hash of the the file's full path name and Metadata Look-up Table to distribute the metadata, use a unique dual-entry access control list (ACL) structure[6], and create file metadata and path permissions at the same time, thus allows direct access to file metadata without involving all of the MDS storing directories along the path. But like the pure hashing, hash function changes will lead to a number of metadata migration. Dynamic subtree partitioning[7] approach treats the file system as a hierarchy, partitions the file system by delegating authority for subtrees of the hierarchy to different metadata servers. It's more flexible than Directory subtree partitioning. Moreover, it is employed to adapt to a changing file system and workload. However, like the Directory subtree partitioning, traversing the directory hierarchy tree is costly, and re-authorizing a subtree needs to move a lot of metadata.

Being effective in some places though, solutions above are generally suffering a common problem--metadata migration, caused by metadata server dynamic expansion. CBMM solved the problem effectively, achieved dynamic expansion for MDS.

3. CBMM

The CBMM is a metadata cluster architecture utilizing Chord protocol for storing file metadata fully distributed, providing operations like publish, read, lookup, ls and so on. Chord protocol is responsible for locating and looking up metadata. The merit of Chord effectively avoid servers interrupt thanks to one MDS failure, meaning that node join and leave will not affect the normal services and hash function, so that provide a high scalability for metadata management.

3.1. System Architecture

In our prototype, we treat the metadata in two ways: file metadata and directory metadata. File metadata referred to common file metadata, while directory metadata is specifically responsible for recording the information of file and subdirectory contained in this directory. The purpose of separating the directory information from file metadata is to providing a traditional directory structure. CBMM consists of three modules: DHT routing module, control module and metadata storage module.

DHT module mainly contains three partial functions. First, monitoring the other nodes join and quit, update own finger table in time, successor and predecessor, to maintain the astringency of Chord ring. Second, monitoring key queries from client, to take lookup algorithm and then locate this node basing on routing protocols and node ID. Third, store this node's metadata on its neighboring nodes synchronously. When the node is offline, the neighbor node should be able to function immediately.

Control module is responsible for initiating the server, opening the DHT module to listen port and other services after starting the chord. For example, it monitors client's requests like publish, download, ls and so on. Control module monitors the occurrence of various matters under the Epoll mechanism, then calls different event handlers to complete the business processes which interact with clients.

The work of metadata storage module is to write and read metadata from memory and disk。

3.2. Metadata Distribution And Organization

The Chord protocol uses a variant of consistent hashing[5] to assign keys to Chord nodes. In CBMM the Chord node is Metadata server, the key is the pathname of the file or directory, each metadata server has a unique node ID which produced by the SHA-1 hash its IP address, while each file or directory has a unique file ID which derived from the SHA-1 hash of its full pathname. The process of map keys to nodes is just the match process: Match file ID to node ID, so that CBMM distribute the metadata across the metadata server cluster. In addition, CBMM treat the metadata of file and directory at the same, their difference is only what is contained in a different. Therefore, all metadata can be averagely distributed in the metadata server.

As the result of CBMM treat the metadata of file and directory at the same and use the same hash, a file and its parent directory along the path have different file ID, so they may be assigned to different metadata servers. Figure 1 shows an example of constructing a hierarchical directory separate file /home/xl/a..c and it’s directory /home/xl.

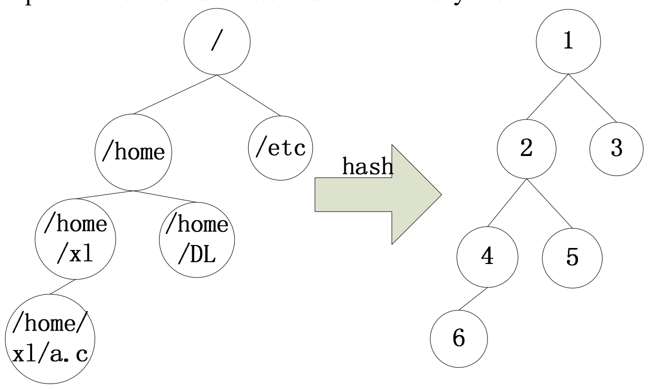

Figure 1. Metadata distribution and organization

3.3. Metadata Access

To access metadata, firstly, clients hash the pathname of the file to produce a hash value which is the file ID, then client send a message to a well know node to find which node contains the file ID. This message transferred between Chord’s nodes, and will finally arrive the node which is right node to store the file, that indicating which metadata server contains the metadata for that file. The client then contacts the appropriate metadata server to access the metadata. In client will develop query cache (in CBMM’s future work) to achieve directly lookup，so most metadata requests will require a single message to a single metadata server. The whole process as depicted in Figure 2.

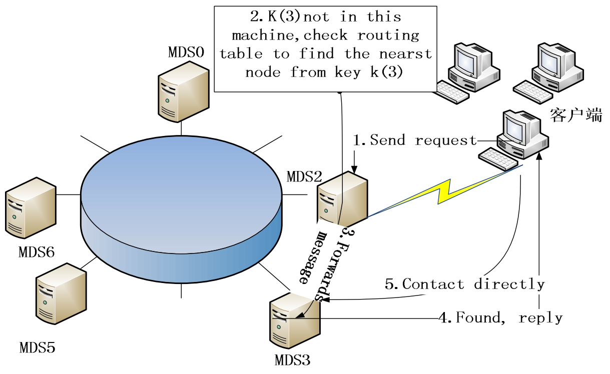

Figure 2. The process of metadata access

4. Scalability

CBMM takes two ways to assure the scalability of system.

4.1. Node join and leave

The Chord protocol is a scalable protocol for lookup in a dynamic peer-to-peer system with frequent node arrivals and departures. Unlike traditional hash algorithm, add or remove MDS will change the output range and map values of hash function， so need to rebuild Hash function, all of this lead to the location of metadata changed and metadata migration. However, CBMM can effectively avoid such a situation. In CBMM's prototype, three kinds of situations may occur:

● When a node joins, after a series of routing operation and sending some messages, the new node will build up their own finger table and inform other nodes update their finger table, moreover, receive the metadata it responsible for from it's successor node, those metadata is small compare to whole system.

●When a node leaves, this node will send message to system, the other node in the ring also will take appropriate action to respond this change, it's successor need to replace the leave node to manage it's metadata.

● When a node disrupts abnormally, heartbeat detection system will find the node fails, then turn into node leave steps.

4.2. Metadata Backup

Metadata will be simultaneously backed up on a successor node when build or update metadata. If there is failure of multiple nodes at the same time, CBMM can be extended to store multiple successor pointer in finger table, and take redundant storage strategy to increase reliability and stability, so as to provide a guarantee for node failure.

4. Conclusions

This paper presents that Chord protocol should be applied to metadata server cluster, make full use of the powerful primitive offered by Chord. Together with improving system performance, balancing the distribution of metadata, achieving load balance and MDS dynamic scalability, reducing metadata migration, the system is also fully decentralized. Because of this decentralization, no node is more important than any other, which makes a single node failure unnoticeable

Acknowledgements

This paper is supported by the Youth Science and Technology Foundation of University of Electronic Science and Technology of China under Grant Number L08010601JX0808, and the ZTE high school cooperation projects (Mass Distributed File System Simulation Software).

References

[1] J. K. Ousterhout, H. D. Costa, D. Harrison, J. A. Kunze,M. Kupfer, and J. G. Thompson. A trace-driven analysis of the Unix 4.2 BSD file system. In Proceedings of the 10th ACM Symposium on Operating Systems Principles (SOSP ' 85), pages 15–24, Dec. 1985.

[2] ssrc, Scalable Metadata Management. http://www.ssrc.ucsc.edu/proj/mdmgmt.html.

[3] Ion Stoica, Robert Mrris, David Karger, M. Frans Kaashoek, Hari Balakrishnan. Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications. In Proc. ACM SIGCOMM'01(August,2001, San Diego, California, USA)

[4] Scott A. Brandt, Ethan L. Miller, Darrell D.E. Long, Lan Xue, "Efficient Metadata Management in Large Distributed Storage Systems", Proceedings of the 20th IEEE / 11th NASA Goddard Conference on Mass Storage Systems and Technologies , CA, April 2003.

[5] P. F. Corbett and D. G. Feitelso "The Vesta parallel file system. ACM Transactions on Computer Systems", 14(3):225-264, 1996.

[6] Sage A. Weil, Kristal T. Pollack, Scott A. Brandt, Ethan L. Miller, "Dynamic Metadata Management for Petabyte-scale File Systems", Proceedings of the 2004 ACM/IEEE Conference on Supercomputing, 2004.

[7] KARGER, D., LEHMAN, E., LEIGHTON, F., LEVINE, M., LEWIN, D., AND PANIGRAHY, R, "Consistent hashing and random trees: Distributed caching protocols for relieving hot spots on the World Wide Web". Proceedings of the 29th Annual ACM Symposium on Theory of Computing (El Paso, TX, May 1997), pp. 654–663.