International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 04 Issue: 10 | Oct -2017 www.irjet.net p-ISSN: 2395-0072

Hadoop Distributed FileSystem: Metadata Management

Piyush P Deshpande

YCCE, RTMNU Nagpur University,
Nagpur, Maharashtra 441110

Abstract - *A Hadoop Distributed File System (HDFS) is designed to store very large data sets reliably and to stream those data sets at high bandwidth to user applications. The present Hadoop relies on secondary namenode for failover which slows down the performance of the system. Hadoop system's scalability depends on the vertical scalability of namenode server. As the namespace of Hadoop distributed file system grows, it demands additional memory to cache.*

Key Words: Hadoop, HDFS, Distributed File System, Metadata Management, File Management System

1. INTRODUCTION

With the rapid development of Internet, the amount of data is growing exponentially, and the large-scale data storage and processing has become a problem. Cloud computing is one of the most popular solutions to meet the demand. Cloud computing provides decreased cost of hardware resource and increased equipment utilization.

1.1 INTRODUCTION TO HADOOP

Hadoop provides a distributed file system and a framework for the analysis and transformation of very large data sets using the Map Reduce paradigm. An important characteristic of Hadoop is the partitioning of data and computation across many (thousands) of hosts.

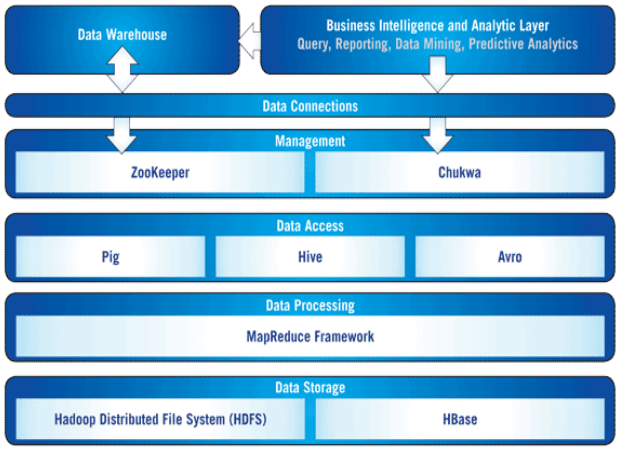

Figure 1: Hadoop System

Table No 1: Components of Hadoop

HDFS	Distributed file system Subject of this paper!
MapReduce	Distributed computation framework
HBase	Column-oriented table service
Pig	Dataflow language and parallel execution framework
Hive	Data warehouse infrastructure
ZooKeeper	Distributed coordination service
Chukwa	System for collecting management data
Avro	Data serialization system

The table 1 shows the components of Hadoop in general which are widely used in Software Industries. Hadoop is an Apache project; all components are available via the Apache open source license.

1.2 HADOOP DISTRIBUTED FILE SYSTEM

The Hadoop Distributed File System (HDFS) [7] is a distributed file system designed to run on commodity hardware.

- Highly Fault-Tolerant
- Easily deployable on low-cost hardware
- High throughput access to application data.

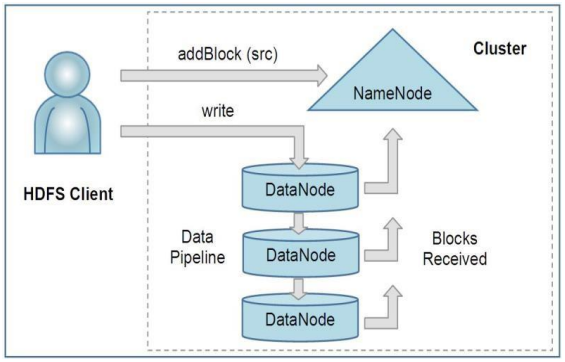

Figure -2: HDFS Architecture

HDFS stores file system metadata and application data separately. HDFS architecture consists of NameNode, DataNode, and HDFS Client. A HDFS Cluster may consist of thousands of DataNode and tens of thousands of HDFS clients

per cluster, as each DataNode may execute multiple application tasks concurrently. The above figure 2 shows the Hadoop Distributed File System Architecture.

Current limitations of Hadoop File system:

I) Scalability: The entire file system metadata is managed and maintained by NameNode in memory. Metadata has space limited by the physical memory available on the node. Some of the storage issues arise as follows:

- Scaling storage
- Scaling the namespace

II) Isolation: No isolation for a multi-tenant environment. An experimental client application that puts high load on the central name node can impact a production application.

III) Availability: While the design does not prevent building a failover mechanism, when a failure occurs the entire namespace and hence the entire cluster is down.

2) Metadata Management Techniques

To distribute metadata among multiple servers some techniques are used like Sub-Tree Partitioning, Hashing technique and Consistent Hashing

2.1) Subtree partioning- The Sub Tree Partitioning [4][6] is used in Ceph file system and Coda file system. The key design idea is that initially, the partition is performed by hashing directories near the root of the hierarchy, and when a server becomes heavily loaded, this busy server automatically migrate some subdirectories to other servers with fewer loads thus improving performance.

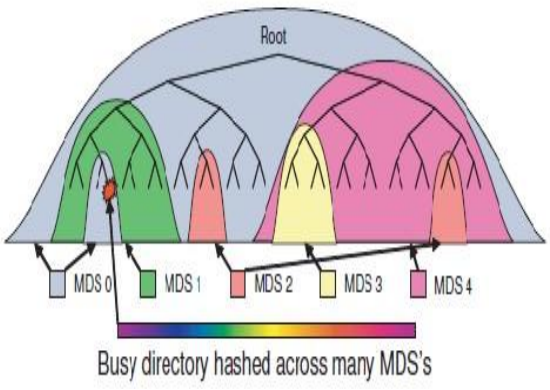

Fig -3: Sub-tree partitioning

- In sub tree partitioning, namespace is divided into many directory sub trees, each of which is managed by individual metadata servers.

2.2) Hashing Technique-Hashing technique [10] is used in Lustre, zFs file system. Hashing technique uses a hash function on the path name to get metadata location. In this scheme, metadata can be distributed uniformly among cluster, but the directory locality feature is lost, and if the path is renamed, some metadata have to migrate. Following mapping approach can be witnessed in this technique

- Balance of metadata workloads
- Faster metadata lookup operations

2.3) Consistent Hashing: **Consistent hashing** [9] is proposed hash method used in Amazon Dynamo. In basic consistent hashing, the output range of the hash function is treated as a ring. Pecularity of this technique are as follows:

- The addition and removal of a node only affects its neighboring nodes and not the entire ecosystem.
- Instead of mapping a physical node to a single point in the ring, each physical node is assigned to multiple positions.
- Virtual node distributes workloads uniformly.

3) System Architecture and Design Issues

Hadoop System basic architecture involves four components: Client, NameNodes, NameNode Manager and Database. Client exposes interfaces to access metadata. NameNode is responsible for managing metadata and dealing with metadata request from Client.

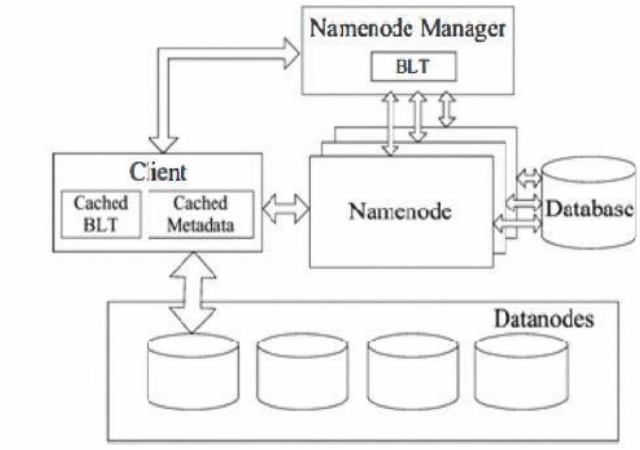

Figure 4: Metadata Management System

The Directory metadata includes a hierarchical namespaces and directory attributes.

3.1) Metadata Format

Generally, a metadata is a tuple as below:

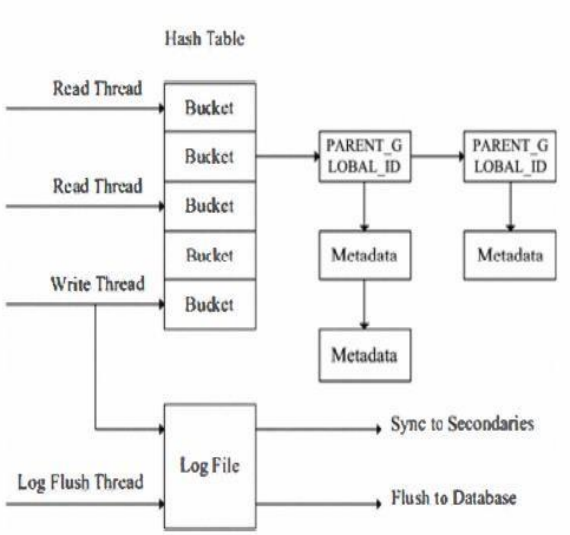

Figure 5: Metadata Tuple

- GLOBAL_ID acts as the global unique identifier that is invariable once the path is formed.
- USER_ID is the identifier of user that created the path.
- PARENT_GLOBAL_ID is the GLOBAL_ID of the parent directory of the path.
- OTHER_META saves other information, such as permission, update time and last access time.
- BLOCK_PTR is the pointer to the file data blocks.
- The updated metadata in NameNode is persisted into database periodically.
- NameNode Manager acts a router which routes for Client to get the target NameNode.

3.2) Partitioning of Metadata

Consistent hashing ring is divided in equal size Q parts and each part is called "bucket". Mapping path from metadata to bucket is like consistent hashing ,first hash USER_ID and PARENT_GLOBAL_ID of the path to yield its position p. Going clockwise to find 1st bucket having position larger than p.

3.3) Accessing Metadata

To organize namespace hierarchy they have adopt hash table. The figure 6 shows the NameNode data structure. For example we want to access to path /A/B/C/filename

- Client gets user_id and global_id of path as Parent_Global_ID
- Computes the consistent hashing result
- Then client see the cached BLT to find out bucket_id and NameNode I.
- Sends the request to NameNode I in form of <bucket_id, user_id, parent_global_id, filename> then the NameNode I see its bucket array by bucket_id.

Figure 6: NameNode Data Structure

In the paper [9] they have also proposed the solution to metadata safety in memory. The solution is "Log Replication". As the metadata is periodically persisted into database, they have got the newest metadata by applying the latest log records on the last-persisted metadata in database.

4. CONCLUSION

We have seen the components and distribution of file system in brief. As compared to other file system HDFS offers a highly reliable and fault tolerable system. Owing to such features of reliability and data failure recovery mechanisms, Hadoop has taken over the other older data store systems which have become obsolete. The main drawback of HDFS was its single NameNode which handles all metadata operations. In this the drawback is overcome by introducing multiple NameNodes like a block chain cluster in the system. To handle multiple metadata servers we have compared three techniques Sub-Tree partitioning, hashing technique and consistent hashing. The consistent hashing technique uses a distributive approach of the metadata which is efficient compared to other techniques.BLT provides an effective metadata retrieval mechanism for Client to find the target NameNode. Metadata availability under cluster failure is guaranteed with Paxos algorithm of log replication. The paper also discusses about the crucial point of NameNode failure detection by heartbeat mechanism and also focuses on failure handling which includes metadata recovery by means of bucket-redistribution. In addition, system performance benefits from metadata caching and prefetching has also been discussed.

REFERENCES

[1] Kornack and P. Rakic, "Cell Proliferation without Neurogenesis in Adult Primate Neocortex," Science, vol. 294, Dec. 2001, pp. 2127-2130, doi:10.1126/science.1065467.

[2] M. Young, The Technical Writer's Handbook. Mill Valley, CA: University Science, 1989.

[3] R. Nicole, "Title of paper with only first word capitalized," J. Name Stand. Abbrev., in press.

[4] K. Elissa, "Title of paper if known," unpublished.

[5] Y.Zhu, H.Jiang, J.Wang, and F.Xian, "HBA: Distributed Metadata Management for Large Cluster-Based Storage Systems", IEEE Trans. Parallel and Distributed Systems, June 2008, vol.19, no.6, pp.750- 763.

[6] S. A. Weil, S. A. Brandt, E. L. Mille, et ai, "Ceph: A Scalable, High- Performance Distributed File System", In Proceedings of the 7th symposium on Operating systems design and implementation, 2006, pp. 307-320.

[7] http://hadoop.apache.org/docs/r0.20.0/hdfs_design.html

[8] Karger, D., Lehman, E., Leighton, T., Panigahy, R., Levine, M. , and Lewin, D, "Consistent hashing and random trees: distributed caching protocols for relieving hot spots on the World Wide Web", In Proceedings of the Twenty-Ninth Annual ACM Symposium on theory of Computing, ACM Press, New York, 1997, pp. 654-663.

[9] Bing Li, Yutao He, Ke Xu, "Distributed Metadata Management Scheme in Cloud Computing ", In Proceedings of IEEE in PCN&CAD CENTER, Beijing University of Post and Telecommunication, China, 2011. Sage A. Weil, Kristal T. Pollack, Scott A. Brandt, Ethan L.Miller ,"Dynamic Metadata Management for Petabyte-scale File Systems ", In Proceedings of IEEE University of California, 2004

[10] Harcharan Jit Singh V. P. Singh "High Scalability of HDFS using Distributed Namespace" International Journal of Computer Applications (0975 – 8887) Volume 52– No.17, August 2012