Industrial Science
Vol.1,Issue.6/Aug. 2014 **ISSN** : 2347-5420

Research Papers

HADOOP DISTRIBUTED FILE SYSTEM WITH CACHE TECHNOLOGY

Archana S. Kakade[1] **and Suhas Raut**[2]

[1]Student, NK Orchid College of Engineering & Technology, Solapur.
[2]Project Guide, NK Orchid College of Engineering & Technology, Solapur.

Abstract

Today's date disk capacity is more advanced. But it lacks in disk access time. As a result, system with disk based storage are finding difficult to cope up with the performance demands of large cluster based systems. Hadoop is an open source framework for big data. Hadoop supports applications that run on large clusters. In an attempt to eliminate disk access, this paper presents the design of caching mechanism based on Primary memory and integrate it with Hadoop. Most frequently data have been available in primary memory & hence process it much more quickly. This paper describes the system architecture that aims to provide a cache system to HDFS, we can avoid unnecessary trips HDD to fetch data and thus avoid delay.

KEYWORDS:

Big Data, Hadoop, Hadoop Distributed File System (HDFS), Cache system, Primary memory.

I.INTRODUCTION

Hadoop's file system is named as Hadoop Distributed File System. The Hadoop Distributed File System [Apache.HDFS] is a distributed file system and is designed to run on commodity hardware. It has many similarities with existing distributed file systems. However, the differences from other distributed file systems are significant. HDFS is highly fault-tolerant and is designed to be deployed on low-cost hardware. HDFS provides high throughput access to application data and is suitable for applications that have large data sets. HDFS has master/slave architecture. An HDFS cluster consists of a single NameNode, a master server that manages the file system namespace and regulates access to files by clients. In addition, there are a number of DataNodes, usually one per node in the cluster, which manage storage attached to the nodes that they run on. In HDFS file reading may contain several interaction of connecting NameNode and DataNode, which dramatically decreases the access performance. Cache System is absent in existing HDFS. Frequently accessed data is used to store in the cache memory & hence process it much more quickly. Cache memory acts as a high speed buffer and is used to store very active data since the cache memory is faster than disk access. The processing speed is increased by making the data needed available in cache. By providing cache system to HDFS, we can avoid unnecessary trips to HDD to fetch data and thus avoid delay.

II. RELATED WORK

There has been a humble amount of work on in memory storage and caches to improve job performance: such as RAMCloud [John K. Ousterhout et al]. John Ousterhout et al proposed approach to datacenter storage called RAMCloud. In RAMCloud information is kept entirely in DRAM. These are large-scale systems and are created by aggregating the main memories of thousands of commodity servers. RAMClouds provides durable and available storage with large throughput of disk-based systems and lower access latency. In RAMCloud there are two important aspects: (1) their extremely low latency and (2) their ability to aggregate the resources of large numbers of commodity servers. Together, these allow RAMClouds to scale to meet the needs of the largest Web applications. But RAMCloud has numerous challenging issues which must be addressed before a practical RAMCloud can be constructed.

III. PROPOSED SYSTEM

HDFS's User will make a request for file. This request goes from Cache. When request is received by cache system it follows following steps:

Cache system checks whether requested file is available in cache local memory or not.
If requested file is available then request is fulfilled by cache. Hence here we can avoid disk access to fetch a file.
Else client communicates with DataNodes to check whether requested file is present in their local memory. If file is available then request is fulfilled.

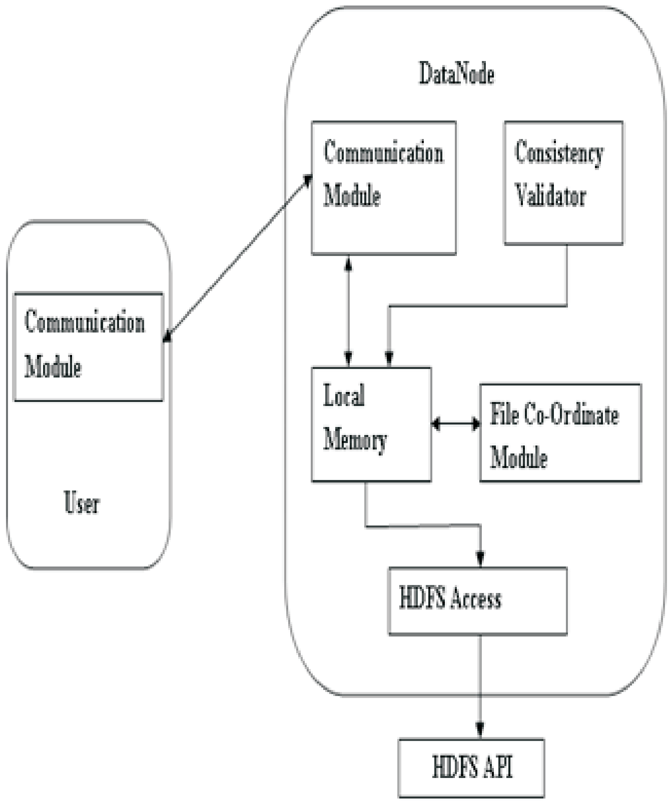

Fig.1: Architecture

Else if file is not available in cache local memory then file is fetched from disk by using HDFS API.

Consistency Validator is used to keep files present in cache local memory consistence with disk.

COMMUNICATION MODULE:

In the proposed system client communicates with HDFS in the same way as it communicates with HDFS without cache system.

The communication protocols build on the TCP/IP protocol. HDFS clients connect to a Transmission Control Protocol (TCP) port opened on the NameNode, and then communicate with the name node using a proprietary Remote Procedure Call (RPC)-based protocol. Data nodes talk to the NameNode using a proprietary block-based protocol. DataNodes continuously loop, asking the NameNode for instructions. Each DataNode maintains an open server socket so that client code or other DataNodes can read or write data. The host or port for this server socket is known by the NameNode, which provides the information to interested clients or other DataNodes.

CONSISTENCY VALIDATOR:

When a client requests for data, consistency validation is performed on request information. Consistency Validator retrieves the day of the month and time of last change of data. And it compares that information with current date and time. If the current date and time is greater than last modification time than the client is admitted to read data. Else client is not allowed to take data.

File Co-ordinate Module:

This module maintains a list of files which are currently presents in local memory.

Local Memory:

Local memory contains frequently requested files.

HDFS Access Module:

If requested file is not present in local memory then the requested file will be fetch from disk.

IV. EXPECTED PERFORMANCE IMPROVEMENT

In existing HDFS procedure of result computing and sending it to user need to be completed in 3-4 seconds while resource scheduling procedures must be completed in milliseconds. This timing needs to be reduced for better performance which our system is expected to do. We expect around 10% performance improvement.

V. CONCLUSION

Apache Hadoop is an open-source software framework for storage and large-scale processing of data on clusters. Hadoop is an Apache top-level project being built and used by a global community. The HDFS is able to handle massive data storage, but HDFS lacks consideration of real time data access. Cache memory is random access memory that a computer microprocessor can access more quickly than it can access disk. We can avoid unnecessary trips hard disk to fetch data, by providing a cache system to HDFS, and thus evade delay. So when request for file is arrived, proposed system looks in the cache and it finds the data there (from a previous reading of data), it does not have to do the more time-consuming reading of data from disk.

REFERENCES

1.[Apache.Hadoop] Apache Hadoop. Available at http://hadoop.apache.org.
2.[Apache.HDFS] Apache Hadoop Distributed File System. Available at http://hadoop.apache.org/hdfs.
3.[Apache.HDFS] Scalability of Hadoop Distributed File System.
4.[Borthakur et al] D. Borthakur (2011) et al. "Apache Hadoop goes real-time at Facebook", In Proceedings of the 2011 International Conference on Management of Data (SIGMOD'11), New York, 2011.

5.[Dean et al] J. Dean and S. Ghemawat (2004), “Mapreduce: Simplified Data Processing on Large Clusters”. In Proceeding of the 6th Conference on Symposium on operating Systems Design and Implementation (OSDI'04), Berkeley, CA, USA, 2004, pp.137-150.
6.[Ghemawat et al] S. Ghemawat, H. Gobioff and S. T. Leung (2003), “Google File System”, In Proc. of the Nineteenth ACM Symposium on Operating Systems Principles (SOSP'03), Lake George New York, 2003, pp.29-43.
7.[John K. Ousterhout et al] John K. Ousterhout et al “The Case for RAMClouds: Scalable High-Performance Storage Entirely in DRAM” Department of Computer Science Stanford University.
8.[Shafer et al] J. Shafer and S Rixner (2010), "The Hadoop distributed file system: balancing portability and performance”, In 2010 IEEE International Symposium on Performance Analysis of System and Software (ISPASS2010), White Plains, NY, March 2010. Pp.122-133.
9.[Zhang et al] Jing Zhang, Gongqing Wu, Xuegang Hu, Xindong Wu (2012), “A Distributed Cache for Hadoop Distributed File System in Real-time Cloud Services”. In 2012 ACM/IEEE 13th International Conference on Grid Computing.
10.[Zhang et al] S. Zhang, J. Han, Z. Liu, K. Wang (2009), “Accelerating MapReduce with Distributed Memory Cache”, In 15th International Conference on Parallel and Distributed Systems (ICPADS09), Shenzhen, 2009, pp.472-478.