VOL. 10, NO. 16, SEPTEMBER 2015 ISSN 1819-6608
ARPN Journal of Engineering and Applied Sciences

www.arpnjournals.com

A TAXONOMY AND COMPARISON OF HADOOP DISTRIBUTED FILE SYSTEM WITH CASSANDRA FILE SYSTEM

Kalpana Dwivedi and Sanjay Kumar Dubey
Department of Computer Science Engineering, Amity School of Engineering and Technology, Amity University Uttar Pradesh, Noida, (U.P.), India
E-Mail: skdubey1@amity.edu

ABSTRACT

As we know data and information is exponentially increasing in current era therefore the technology like Hadoop, Cassandra File System, HBase etc became the hot technology and preferred choice among the IT professionals and business communities. Hadoop Distributed File System and Cassandra File System is rapidly growing and proving themselves to be cutting edge technology in dealing with huge amount of structured and unstructured data. Both HDFS and CFS are open source software which comes under umbrella of Apache. Both technologies have large customer base which is exponentially growing and have certain pros and cons .Since both the file system are very popular and extensively been used in the areas of handling big data hence it is worth to do a comparison between both the technologies and helping the intended reader in selecting appropriate file system which efficiently meets the requirement of the customer. Paper covers about HDFS and CFS and then provides the comparative analysis of features provided by both the file systems.

Keywords: hadoop, cassandra, database, file system.

INTRODUCTION

Hadoop is an open source Apache project framework for big scale computation and data processing of huge data sets [1, 2]. Hadoop is a framework of tools consists of pig, chukwa, HBase, avro, hive, mapreduce, pig, hdfs, and zookeeper [3]. HDFS and Mapreduce are the storage and processing component of Hadoop. Both are the core components of Hadoop. Hadoop distributed file system is designed for processing, storing and analyzing huge amount of structured, unstructured and semi structured data. It provides fault tolerance, scalability, availability and reliable fast access to the information [4]. MapReduce is a massively parallel, scalable, processing framework [5]. The design of HDFS is not specific to one single machine rather than file system that is being used by Hadoop but it will be distributed into a number of machines like slave machines. HDFS is based on Google file system.

Apache Cassandra is an open source tuneably consistent, highly available, fault tolerant, distributed, and elastically scalable, decentralized, column oriented database [7]. Cassandra uses amazon dynamo scheme for data clustering and distribution and Big Table data model and is also non relational system [8]. Cassandra benefits are high availability, fully distributed, multiregional replication support (bi-directional), scalability, and write performance, simple to install and operate. The Cassandra file system has a replication facility, master less and decentralized [9]. Cassandra offers robust support for clusters spanning multiple data centers with asynchronous masterless replication allowing low latency operations for all clients.

WORKING OF HADOOP DISTRIBUTED FILE SYSTEM

Master/slave architecture is followed by HDFS. Daemon services of HDFS are NameNode and DataNode. NameNode is also known as the job tracker. It is the master of the system and maintains and manages the blocks which are present on the Datanode [10]. Data node is also known as task tracker The File system namespaces are managed by the master server that is the Namenode in a HDFS cluster and also stored on data nodes keeping the physical file metadata information [11]. There are DataNodes that can be one per node in the cluster. File breaks into one or more blocks and then stores in a set of Data Nodes whenever user attempts to store very huge size of file [12, 13]. NameNode stores the metadata information and assist client to perform various operations. Block replication, deletion and creation upon instruction from the Namenode are performed by the data node [14].

WORKING OF CASSANDRA FILE SYSTEM

Inspite of using sharded design a peer-to-peer distributed “ring” architecture is used by Cassandra file system which is much easier to setup, elegant and maintain [15]. Cassandra file system is modelled as two column families along with Keyspace [16]. Two primary HDFS services are represented by two column families [17]. The Inode column family is replaced by HDFS NameNode service and sblocks is replaced with DataNode service [18]. NameNode which tracks each files block locations and metadata and DataNode service stores the file blocks [19]. A dynamic composite type comparator is used by the ‘inode’ column family which contains about file metadata

information. Blocks are ordered sequentially by making use of Time UUID which helps in supporting HDFS [20].

TAXONOMY OF HDFS AND CFS

Motivation behind comparison is to explore various features of Hadoop Distributed File System and Cassandra File System. A comparison and useful artifacts that would help users or customers to choose the effective Distributed File System that best suits the client requirement and performs better secured and fault tolerant one[21].

Architecture

Architecture of both HDFS and CFS are different. HDFS is implemented using Master/Slave architecture where NameNode works as Master and DataNode work as worker node whereas Cassandra is designed with the understanding that system/hardware failure can and do occur and CFS follows peer to peer, distributed architecture where all nodes are same. Data is partitioned among all nodes in the cluster [22, 23].

Data storage model

HDFS maintains the data directly in file system. Large files are broken into small blocks and replicated on many data nodes [24].Cassandra gives post relational database solution. By post relational means it does not offers all the features of traditional databases but follows keyspace column family to store the data and introduces primary and secondary indexes for high availability of data

Read and write design

In HDFS, since NameNode works like master node and keeps all metadata information hence all read/write operation is performed via NameNode. It internally introduces map/reduce process to achieve high performance. CFS, since implements peer to peer architecture.

Fault tolerance

Master/Slave architecture of HDFS achieves high performance but vulnerable to failure when master node is down. In CFS cluster all node in the cluster are same and capable to handle the request. This is the most important feature of CFS which provides edge over HDFS [25].

Area of utilization

HDFS is more powerful solution for read intensive database for business intelligence system [26] while CFS is more suitable for real time database for online/transactional applications.

Mode of accessing data

Hadoop provides HDFS client which uses Map/Reduce component to fetch the data from HDFS [27]. Cassandra File system provides Command Line Interface and Cassandra query language tools to access the data stored in its column family [28].

Schema

In HDFS data is persisted onto data node machines in the form of document. [29].From Google big table Schema is mirrored which is a row oriented column structure used in Cassandra. Cassandra uses keyspace in which there is a column family that is more flexible and dynamic

Communication

Remote procedure call is used by many of distributed file system as the method of the communication. Two communication protocols are used for RPC are UDP and TCP.

Data partitioning

In HDFS, data partitioning is done by Namenode. HDFS configuration is based on the replication factor. Cassandra file system has the ring structure in which every node may act as master node for performing data partitioning. Based on the row key column family data is partitiones across the nodes.

Consistency and replication

Many of the distributed file system use checksum to provide consistency [30]. In distributed file system replication and caching play a very important role [31, 32]?

Load balancing

Adding or removing servers is the ability to auto-balance the system. Tools must be provided to store them on other servers, to recover lost data. In HDFS, if a node is unbalanced, replicas are moved from it to another one by communicating directly with the DataNodes [33, 34].

Security

A dedicated security mechanism in the architecture of Hadoop is not employed. In Cassandra based on internally controlled login accounts/passwords gives client-to-node encryption and also for the open source community it gives features object permission management security [35].

Naming

In HDFS file system, NameNode is the centerpiece which tracks the information of all files in the file system [36]. Unlike HDFS Cassandra file system have inode column family which uses a dynamic composite type

comparator and stores the metadata information about a file [37].

Client access interface

HDFS connect via HDFS client only in the HDFS consumer application. Checksum is implemented by the HDFS client software [38]. In Cassandra command line interfaces are used to access the Cassandra database. Schema and Cassandra specific language are used for performing any database operation and to access the database in Command Line Interface while to access the Cassandra database traditional SQL language is used in Cassandra Query Language [39].

Indexing

HDFS does not support indexing hence limiting its performance. To achieve Hadoop distributed indexing Apache solr or Terrier with Hadoop can be configured for optimizing the performance. Cassandra supports secondary indexes of the type keys and indexing is easy [40].

Throughput and latency

HDFS reduces the write latency due to less I/O bottleneck for large number of data nodes [41]. When Cassandra operations are performed in parallel, excellent throughput and latency are achieved by Cassandra. Throughput affect negatively if Cassandra does not lock the fast write request path.

Locking system

HDFS includes a locking mechanism that allows clients can lock directories and files in the file system. This indicates that the Name Node act as lock manager on behalf of its clients [42]. In Cassandra instead of using locking mechanisms or ACID transactions with rollback, it offers atomic, isolated, atomic, eventual/tunable and durable transactions.

Data persistence

In HDFS data is directly written to the data node in single operation [43]. Cassandra, Memtable is-called as in-memory structure in which data is first written to it and then persisted in SSTable to the disk after the Memtable is full.

OVERALL COMPARISON OF HDFS AND CFS

The Table-1 shows the various comparison between Hadoop distributed file system and Cassandra file system.

Table-1. Analysis of HDFS and CFS.

S. No.	Parameters	Analysis of Hadoop distributed file system	Analysis of Cassandra file system
1	Architecture	Master/Slave NameNode works as Master and data node as worker node.	Peer to Peer distributed architecture where all nodes are same.
2	Data storage model	HDFS stores the data directly in file system.	Cassandra gives post relational database solution.
3	Read and Write Design	Write once read many access models.	Read and write any where model.
4	Fault tolerance	Failure as norm.	No single point of failure.
5	Area of utilization	Batch-oriente danalytical solutions.	Real time online transactional processing.
6	Mode of accessing data	Map/Reduce for read/write operations.	Cassandra query language and Command line interface tools.
7	Storage schema	Physical file system schema.	Combines schema from Google big table and Amazon Dynamo.
8	Communication	RPC/TCP and UDP	Gossip protocol
9	Data Partitioning	NameNode breaks the data file into smaller chunks and distributes them across the data nodes.	On the basis of row key in the column family data is breaks across the nodes.
10	Consistency	For each block of the file it computes the checksums and in the same HDFS namespace stores these checksums in a separate hidden file.	In Write Consistency, before returning an acknowledgement to the client application how many replicas the write must succeed In Read Consistency, before a result is returned to the client application how

			many replicas must respond
11	Load balancing	For each data node the usage of the server different from the usage of the cluster by no more than the threshold value. In HDFS replicas are moved from it to another one respecting the placement policy if a node is unbalanced.	The data does not automatically get shared across new nodes equally when adding new nodes to the cluster and share load proportionately which makes completely unbalanced. By using the node tool move command we need to shift the token range and must be calculated in a way that involves sharing of data equally.
12	Security	No dedicated security mechanism. POSIX-based file permissions for users and groups with optional LDAP integration. Security features in HDFS are Authentication, service level authorization for web consoles and data confidentiality. Data encryption and role based authentication is also available.	Internal authentication, managing object permissions using internal authorization, client to node encryption, node to node encryption, Kerberos authentication, transparent data encryption, data auditing.
13	Naming	Central metadata server	Cassandra comes up with 'inode' column family to store meta data information.
14	Client Access Interface	A code library and HDFS client software.	Command line interface and Cassandra query language.
15	Indexing	Indexing is difficult. To achieve Hadoop distributed indexing we can configure Apache solr or Terrier with Hadoop.	Cassandra supports secondary indexes of the type keys. Indexing is easy.
16	Transaction rates	High	High
17	Throughput and latency	Reading a chunk of data can range from tens of milliseconds in the best case and hundreds of milliseconds in the worst case. Reduces write latency because of large number of data nodes.	Unlike most databases, Cassandra achieves excellent throughput and latency.
18	File Locking mechanism	HDFS give locks on objects to clients.	Provides isolated, atomic, isolated, transactions with eventual/tunable consistency, durable.
19	Replication	The replication factor can be specified At file creation time the replication factor can be specified. All decisions are made by Name node regarding replication of blocks which periodically in the cluster receives a block report and a Heartbeat from each of the Data Nodes and is functioning properly implies after receiving the Receipt of a Heartbeat.	Using replica placement strategy creates a keyspace based on the row key stores copies of each row.
20	Data persistence	Data is directly written to data node.	Data is first written in memory structure called mem-table and when it is full written to SStable and then to disk.

ANALYSIS

On the basis of feature extraction the following comparison on both Distributed File systems can help to choose an appropriate distributed file system according to the customer requirements.

Hadoop distributed file system

Suitable for application that has large data sets, very large files, fault tolerant which ensures that in case of any failures hardware or software there is a way to overcome it by the use of replication factor, cost effective as it works on commodity hardware and write once read many access model. It does not support updates and cannot read or modify the content. It is designed more for batch processing. Scenario where HDFS cannot be used like low latency accesses where you have millions of small files, parallel write or arbitrary read.

Cassandra file system

It is used in Big Data, flexible sparse wide column requirements, very high velocity random reads and writes. No multiple secondary index needs is allowed. Cassandra is not used in searching column data, dynamic Queries on Columns, secondary indexes, relational data transactional primary and financial records, stringent security and authorization of data.

CONCLUSIONS

This paper tried to extract various features of Hadoop Distributed File System, Cassandra File System and then comparative analysis of their characteristics.

HDFS and CFS both are fast growing technologies which deal in storage and management of BIG data. HDFS follow Master/Slave architecture while CFS implements peer to peer architecture. CFS peer to peer architecture makes it no single point of failure. HDFS cluster can be scaled horizontally by adding more nodes to the cluster or vertically by upgrading the hardware and software configuration. CFS combines highly dynamic and scalable Google's BIG table and Amazon's Dynamo DB. CFS either uses its query language or command line interface to access the data stored in its database.

Finally concluded the research with comparative study of both the file system on several parameters like architecture, protocols for communication, area of utilization, Data storage schema, fault tolerance etc.

REFERENCES

[1] B. Dong, J. Qiu, et al. 2010. A novel approach to improving the efficiency of storing and accessing small files on Hadoop: a case study by 20PowerPoint files. In 2010 IEEE International Conference on Service Computing (SCC), Miami. pp. 65-72.

[2] H. Zhang, Y. Han, F. Chen and J. Wen. 2011. A Novel Approach in Improving I/O Performance of Small Meteorological Files on HDFS. Applied Mechanics and Material. 117-199: 1759-1765.

[3] F. Chang, J, Dean, et al. 2008. Bigtable: A Distributed Storage System for Structured Data. ACM Transactions on Computer Systems. 26(2): 205-218.

[4] Kalpana Dwivedi, Sanjay Kumar Dubey. 2014. Analytical Review on Hadoop Distributed File System. In: 2014 IEEE 5th International Conference-Confluence The Next Generation Information Technology Summit (Confluence).

[5] J. Xie, S. Yin, et al. 2010. Improving MapReduce performance through data placement in heterogeneous Hadoop clusters", In: 2010 IEEE International Symposium on Parallel and Distributed file system.

[6] J. Shafer and S Rixner. 2010. The Hadoop distributed file system: balancing portability and performance. In 2010 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS2010), White Plains, NY. pp. 122-133.

[7] E. Dede, B. Sendir, P. Kuzlu, J. Hartog and M. Govindaraju. 2013. An Evaluation of Cassandra for Hadoop. In: Proceedings of the 2013 IEEE Sixth International Conference on Cloud Computing, CLOUD '13, pp. 494–501, Washington, DC, USA. IEEE Computer Society.

[8] A. Lakshman and P. Malik. 2009. Cassandra: structured storage system on p2p network. In: Proceedings of the 28th ACM symposium on Principles of distributed computing, PODC '09, pp. 5-5, New York, NY, USA, ACM.

[9] Adam Silberstein, Brian F. Cooper, Utkarsh Srivastava. 2008. Erik Vee, Ramana Yerneni, and Raghu Ramakrishnan, Efficient bulk insertion into a distributed ordered table. In: Proceedings of the 2008 ACM SIGMOD International conference on Management of Data.

[10] Adam Silberstein, Brian F. Cooper, Utkarsh Srivastava, Erik Vee, Ramana Yerneni and Raghu Ramakrishnan. 2008. Efficient bulk insertion into a distributed ordered table. In: Proceedings of the 2008 ACM SIGMOD International conference on Management of Data.

[11] J. Han, H. H E and G. Le. 2011. Survey on NoSQL Database. Piscataway, NJ, USA: IEEE. pp. 363-3.

[12] Y. Zhu, P. Yu and J. Wang. 2013. RECODS: replica consistency-on-demand store. In 2013 IEEE 29th International Conference on Data Engineering (ICDE). pp. 1360-1363.

[13] R. Al-Ekram and R. Holt. 2010. Multi-consistency data replication. In: 2010 IEEE 16th International Conference on Parallel and Distributed Systems (ICPADS). pp. 568-577.

[14] O. Rode, Avi Teperman. 2003. zFS A Scalable Distributed File System Using Object Disks. Proceedings of the 20th IEEE/11th NASA Goddard Conference on Mass Storage Systems and Technologies (MSS'03). pp. 207-219.

[15] J. Xie, S. Yin, *et al*. 2010. Improving MapReduce performance through data placement in hetergeneous Hadoop clusters. In: 2010 IEEE International Symposium on Parallel and Distributed file system.

[16] Chen Feng, Yongqiang Zou, Zhiwei Xu. 2011. CCIndex for Cassandra: A Novel Scheme for Multidimensional Range Queries in Cassandra. In: Seventh International Conference on Semantics, Knowledge and Grids.

[17] F. Chang, Jeffrey Dean, Sanjay Ghemawat, Wilson C. Hsieh, Deborah A. Wallach Mike Burrows, Tushar Chandra, Andrew Fikes, Robert E. Gruber. 2006. Bigtable: A Distributed Storage System for Structured Data, Proceedings of 7th Symposium on Operating System Design and Implementation (OSDI). pp. 205-218..

[18] S. A. Weil, Scott A. Brandt, Ethan L. Miller, Darrell D. E. Long. 2006. Ceph: A Scalable, High-Performance Distributed File System, Procceding of OSDI '06 Proceedings of the 7th symposium on Operating systems design and implementation. pp. 307-320.

[19] Fay Chang, Jeffrey Dean, Sanjay Ghemawat, Wilson C. Hsieh, Deborah A. Wallach, Mike Burrows, Tushar Chandra, Andrew Fikes, and Robert E. Gruber. 2006. Bigtable: a distributed storage system for structured data. In: 7th USENIX Symposium on Operating Systems Design and Implementation.

[20] Giuseppe DeCandia, Deniz Hastorun, Madan Jampani, Gunavardhan Kakulapati, Avinash Lakshman, Alex Pilchin, Swaminathan Sivasubramanian, Peter Vosshall and Werner Vogels. 2007. Dynamo:amazon's highly available key-value store. In: Proceedings of 21st ACM SIGOPS symposium on Operating systems principles.

[21] Tran Doan Thanh. A Taxonomy and Survey on Distributed File Systems. Fourth International Conference on Networked Computing.

[22] S. Ghemawat, H. Gobioff, and S.-T. Leung. 2003. The google file system. In: SOSP '03: Proceedings of the Nineteenth ACM Symposium on Operating Systems Principles, pp. 29-43, New York, NY, USA, ACM.

[23] J. M. Hellerstein, M. Stonebraker and J. Hamilton. 2007. Architecture of a database system. Foundations and Trends in Databases. 1(2):141-259.

[24] Guoxi Wang, Jianfeng Tang. 2012. The NoSQL Principles and Basic Application of Cassandra Model. In: proceedings of International Conference on Computer Science and Service System 2012.

[25] F. B. Schneider. 1990. Implementing fault-tolerant services using the state machine approach: a tutorial. ACM Computer Survey. 22(4): 299-319.

[26] F. Azzedin. 2013. Towards a scalable HDFS architecture. In: IEEE International Conference on Collaboration Technologies and Systems (CTS). pp. 155-161.

[27] Jing Zhang1, Gongqing Wu1, Xuegang Hu1, Xindong Wu1. 2012. A Distributed Cache for Hadoop Distributed File System in Real-time Cloud Services Proceedings of the 2012 ACM/IEEE 13th International Conference on Grid Computing. pp. 12-21.

[28] V. Bala, E. Duesterwald and S. Banerjia. 2000. Dynamo: a transparent dynamic optimization system. In ACM SIGPLAN Notices. 35: 1-12.

[29] D. Abadi. 2012. Consistency tradeoffs in modern distributed database system design: CAP is only part of the story. Computer. 45(2): 37-42.

[30] M. Shapiro, N. Preguic¸a, C. Baquero and M. Zawirski. 2011. A comprehensive study of convergent and commutative replicated data types.

[31] S. B. Davidson, H. Garcia-Molina, and D. Skeen. 1985. Consistency in a partitioned network: a survey. ACM Computer Survey. 17(3): 341-370.

[32] H. Yu and A. Vahdat. 2001. The costs and limits of availability for replicated services. In ACM SIGOPS Operating Systems Review, vol. 35, 2001, pp. 29-42.

[33] Thanh Cuong Nguyen. Wenfeng Shen, Jiwei Jiang, Weimin Xu. 2013. A novel data encryption in HDFS. In: IEEE Internatonal Conference on and IEEE Cyber, Physical and Social Computing on Green Computing and Communications (GreenCom). pp. 1-10.

[34] Shaochun Wu, Guobing Zou, Honghao Zhu, Xiang Shuai, Liang Chen, Bofeng Zhang. 2013. The Dynamically Efficient Mechanism of HDFS Data Prefetching. In: IEEE and Internet of things (iThings/CPSCom), IEEE International Conference on and IEEE Cyber, Physical and Social COMPUTING on Green Computing and Communications (GreenCom). pp. 1-10.

[35] Kai Fan, Dayang Zhang, Hui Li, Yintang Yang. 2013. An adaptive feedback load balancing algorithm in HDFS, in 2013 5th IEEE international Conference on Intelligent Networking and collaborative Systems (INCoS). p. 28.

[36] Zhen Ye, Shanping Li. 2011. A Request Skew Aware Hetergeneous Distributed Storage System Based on Cassandra. International Conference on Computer and Management - CAMAN, DOI: 10.1109/CAMAN.2011.5778745.

[37] P. Malik, Avinash Lakshman. 2010. Cassandra - a decentralized structured storage, ACM SIGOPS Operating System Review. 44(2): 35-40.

[38] K. Karun. A and Chitharanjan. K. 2013. Locality Sensitive Hashing based Incremental Clustering for creating affinity groups in Hadoop-HDFS-an infrastructure extensión. In IEEE International Conference on Circuits, Power and Computing Technologies (ICCPCT 2013). p. 1243.

[39] Ymir Vigfusson, Adam Silberstein, Brian F. Cooper, Rodrigo Fonseca. 2009. Adaptively parallelizing distributed range queries. In Proc. VLDB Endow. 2: 682-693. VLDB Endowment.

[40] Yongqiang Zou, Jia Liu, Shicai Wang, Li Zha, and Zhiwei Xu. 2010. CCIndex: a Complemental Clustering Index on Distributed Ordered Tables for Multi-dimensional Range Queries. In: 7th IFIP International Conference on Network and Parallel Computing.

[41] Y. Gao and S. Zheng. 2011. A Metadata Access Strategy of Learning Resources Based on HDFS. In: proceeding International Conference on Image Analysis and Signal Processing (IASP). pp. 620-622.

[42] G. Zhang, Chuanjie Xie, Lei Shi, Yunyan Du. 2012. A tile-based scalable raster data management system based on HDFS. Geoinformatics. pp. 1-4.

[43] Zhendong Cheng, Alain Roy, Ning Zhang, Gang Guan, You Men, Depei Qian. 2012. ERMS: An Elastic Replication Management System for HDFS, In: proceedings of the IEEE International Conference on Cluster Computing Workshops. pp. 1-10.