Project Phase 3

CSC 244: Database System Design

Hadoop Distributed File System with HBase

04/14/2017

Prof: Dr. Ying Jin

Name	Contribution (%)
JAY BIBODI	50
JAIDIPKUMAR PATEL	50

ABSTRACT

“The Hadoop distributed file system (HDFS) is a distributed, scalable and portable file system written in java for the Hadoop framework” [9].

“HBase is distributed database developed as part of Apache Software Foundation's Apache Hadoop project and runs on top of HDFS (Hadoop Distributed File System), designed to provide quick random access to huge amounts of structured data” [10].

Hadoop, used for distributed file storage, provides a high-throughput access to application data. As the HDFS part of this paper, this project will demonstrate the features including performing HDFS operations, architecture and goals of Hadoop. This paper will also discuss the scenarios where Hadoop is best suited, its interaction with the application in terms of data access and performance and also HDFS’ pros and cons.

HBase is column-oriented key-value database and is been well-liked due to its parentage from Hadoop and HDFS. The HBase part of this paper will exhibit its features, storage mechanism used, possible applications of HBase; also understand the difference between column-oriented(HBase) and row-oriented databases. Furthermore, it will highlight the difference between HBase and HDFS.

TABLE OF CONTENTS

Overview

It is important to know few terms which will help us to get the general idea of Hadoop distributed file systems and HBase. Set of machine in a single local area network is called as cluster. "Commodity hardware is hardware which are cheap, easily obtained and changed [8]". File system is the way in which we store file and directories on the computers.

There were two issues before Hadoop came into market due to growth of data. These issues were storing and processing the huge amount of data or big data.

To address the issue listed above Apache Hadoop developed an open source framework. Hadoop is not a single product, but a collection of software application which consists of many different modules used to store and process big data. Generally, big data is a huge, wide variety of dataset which changes rapidly and scales from terabytes to petabytes. One of the important component in Hadoop architecture is Hadoop distributed file system abbreviated as HDFS used to store the big data.

1. Introduction of Hadoop Distributed File System

HDFS is a file system specially designed for storing huge dataset on cluster of commodity hardware and with streaming access pattern. File system on various nodes in the LAN are linked together, so, as for the users it's just a single file system. HDFS works with principle of "write once, read always". However, we could retrieve the file, change the content and store that file as a new file in HDFS. Key features of HDFS are fault tolerance and high throughput. File replication method is used to avoid fault in the HDFS. Reading the data in parallel helps us to achieve high throughput in the HDFS.

2. Introduction of HBase

HBase is a NoSQL column-oriented distributed database built on top of Hadoop distributed file system. It can be scaled horizontally. HBase is an open source, distributed, versioned, non-relational database, modeled after Google Big table and supports random real time CRUD operations. All the data will be taken as key-value pair of byte-array. User can gain access to read and write into Hadoop distributed file system using HBase. Hadoop cannot handle high speed of random write and read and it is unable to change a file without rewriting it. HBase allows fast random read and writes in optimized manner.

3. Hadoop Distributed File System

3.1. Hadoop Architecture

Hadoop distributed file system (HDFS) has 3 core components

- Name Node
- Data Node
- Data Replication

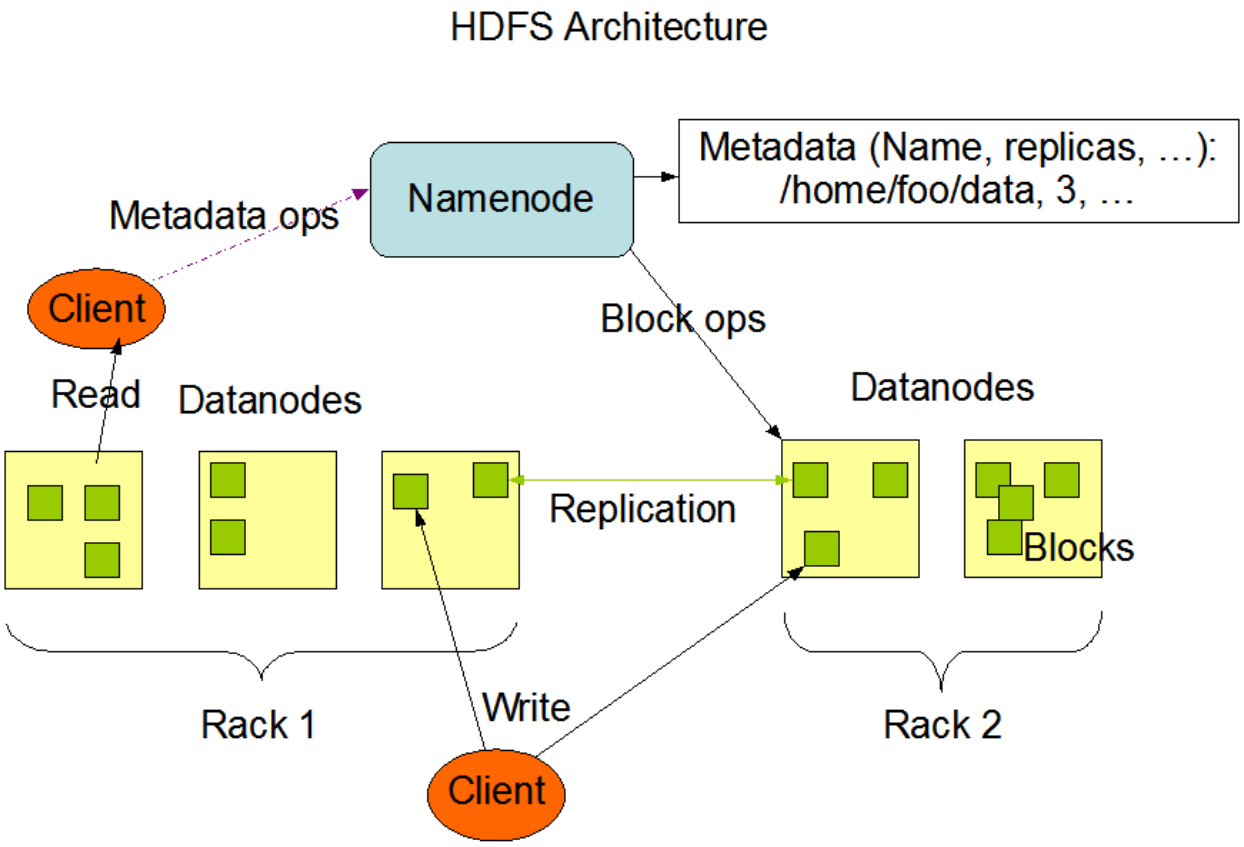

Fig 1. Hadoop Architecture [1]

HDFS is one of the core concepts of Hadoop. HDFS basically works on master and slave approach, Name node works as master and Data nodes work as a slaves.

A. Name Node

Name node is a one dedicated server which stores metadata of the file system, it will keep the details such as a number of blocks required to store the file, size and type of the file and also store the replicated nodes detail which is used in case of node failure or data loss.

Name node is responsible for allocating block of memory and Data node to store the file. When Name node store any file at any node, it also copies same file data to two other nodes for backup. Name node takes block report and heartbeat details from active Data node for every short period

of time. If at any point failure of Data node found, it will assume that particular data node is not active and failure has occurred, the name node immediately checks its replicated node and the nearest is allocated as the original Data node and will keep maintaining two copies for backup.

Name node is also responsible for serving all the client request. For example, when a client needs to store any file it will request Name node, and in response to that request, Name node provides the Data node information to the client where it can store the file. Name node plays a key role in HDFS architecture also serves as a Single point of failure in HDFS as it takes care of major request/response call, maintain metadata and handle the failure of Data node.

B. Data Node

There is a cluster of nodes available to store the application data in the HDFS which is known as Data node. Data node establish a connection with Name node and takes instructions from it to perform certain operations such as the read and write operation on file system to serve the client request. Data node periodically gives the block report (every tenth heartbeat) and heartbeat (every three seconds) information to the Name node to inform about its status. Block report helps the Name node to maintain metadata, also Name node can check the status of the data node which helps it to keep maintain the copies of the blocks.

C. Data Replication

To enable the Fault Tolerance feature in the HDFS, HDFS replicates the file blocks. Name node is responsible for taking care file blocks replication. Name node maintains the metadata about the block replication and also decision concerning block replication has been taken by Name node only. “The placement of replicas is critical to HDFS reliability and performance. Optimizing replica placement distinguishes HDFS from most other distributed file systems. The purpose of a

rack-aware replica placement policy is to improve data reliability, availability, and network bandwidth utilization [1].”

3.2. Hadoop Operations and Goals

A. Operations

- Format and Start HDFS [2]

  At the beginning we have to format the configured HDFS file system.

  $ hadoop namenode – format

  Then after formatting the HDFS successfully, start the HDFS by the following command

  $ start -dfs.sh

- List Files in HDFS [2]

  This operation helps us to find list of files in a directory

  $ $HADOOP_HOME**/bin/hadoop fs -ls <args>**

- Insert Data into HDFS [2]

  First, we assume that data in .txt file format, to insert data first we need to create input directory by the following command.

  $ $HADOOP_HOME**/bin/hadoop fs -mkdir/user/input**

  Secondly, Transfer and store data file from local system to the Hadoop file system using put command.

  $ $HADOOP_HOME**/bin/hadoop fs -put /home/file.txt /user/input**

- Retrieval from HDFS [2]

  Assume that we have HDFS file called outfile.

  Using cat command, we can view the data.

  $ $HADOOP_HOME**/bin/hadoop fs -cat /user/output/outfile**

You can always get the HDFS file to the local file using get command.

$ $HADOOP_HOME**/bin/hadoop fs -get /user/output/ /home/hadoop_tp/**

- Shut down the HDFS [2]

  $ stop-dfs.sh

B. GOALS

- Hardware Failure [1]

  HDFS is a cluster of hundreds or thousands of nodes, there is a possibility of the node failure. Therefore, quick detection of the fault and automatic recovery from the failure is a prime goal of HDFS.

- Streaming Data Access [2]

  HDFS need streaming data access as it is not designed for general purpose applications, it is designed for batch processing. It implies the principle such as write once, read a number of times but do not change the data.

- Large Data Sets [2]

  HDFS Designed to work with several data with the terabytes and or petabytes size.

- Hardware at data [3]

  To improve the efficiency of the task, computation should take place near the data. If this feature implies on huge data, it reduces the network congestion and increases the throughput of the system.

3.3. Pros and Cons

A. Pros

- Data Source Range [4]

  There are various sources available for data collection which are in structured or unstructured form. The sources can be social media, weather data, email conversation. To convert a variety of data into one single form is a tedious task. Hadoop helps us to derive valuable information from a variety of data sources into a single form, which saves data formatting time.

- Cost effective [4]

  A company size is big or small they had to spend a significant amount of their benefit into storing large amounts of data. Sometimes it is possible that companies had to delete some of their data to make space for storing new data. in this case, it is highly possible that important data was lost. Hadoop is useful in those cases; it can solve the data storage problem for companies. Hadoop provides huge data storage capacity which helps the companies to study the data and can take decision for company's benefit.

- Speed [4]

  Hadoop enables the organization to get the work done at a faster rate with its data storage needs. As Hadoop used a tool to processed the data which are located on same servers as the data, which enables the data processing rate faster. In fact, you can process several terabytes or petabytes data within minutes.

- Multiple copies [4]

  Hadoop architecture designed in such a way that it automatically duplicates the blocks of data that are stored in data node. Which adds the features of fault tolerance and high availability of data.

B. Cons

- Security measures [4]

  By default, Hadoop disabled security measures. Therefore, it increases the responsibility of data analytics person to take care of company's sensitive data and its security.

- Data concerns [4]

  Hadoop provides the facility to store and process huge data i.e. terabytes or petabytes. Therefore, Hadoop can only useful for big size organization who deals with the large size data, for the small organization it might be not a good fit.

- Difficult Integration

  Hadoop is not designed and optimized for ease of use, it might be difficult to installing and integrating Hadoop with existing database.

- Lack of interactive access

  Hadoop process data in the batch oriented MapReduce, which restricts it to serve as interactive query processing on random data.

4. HBASE

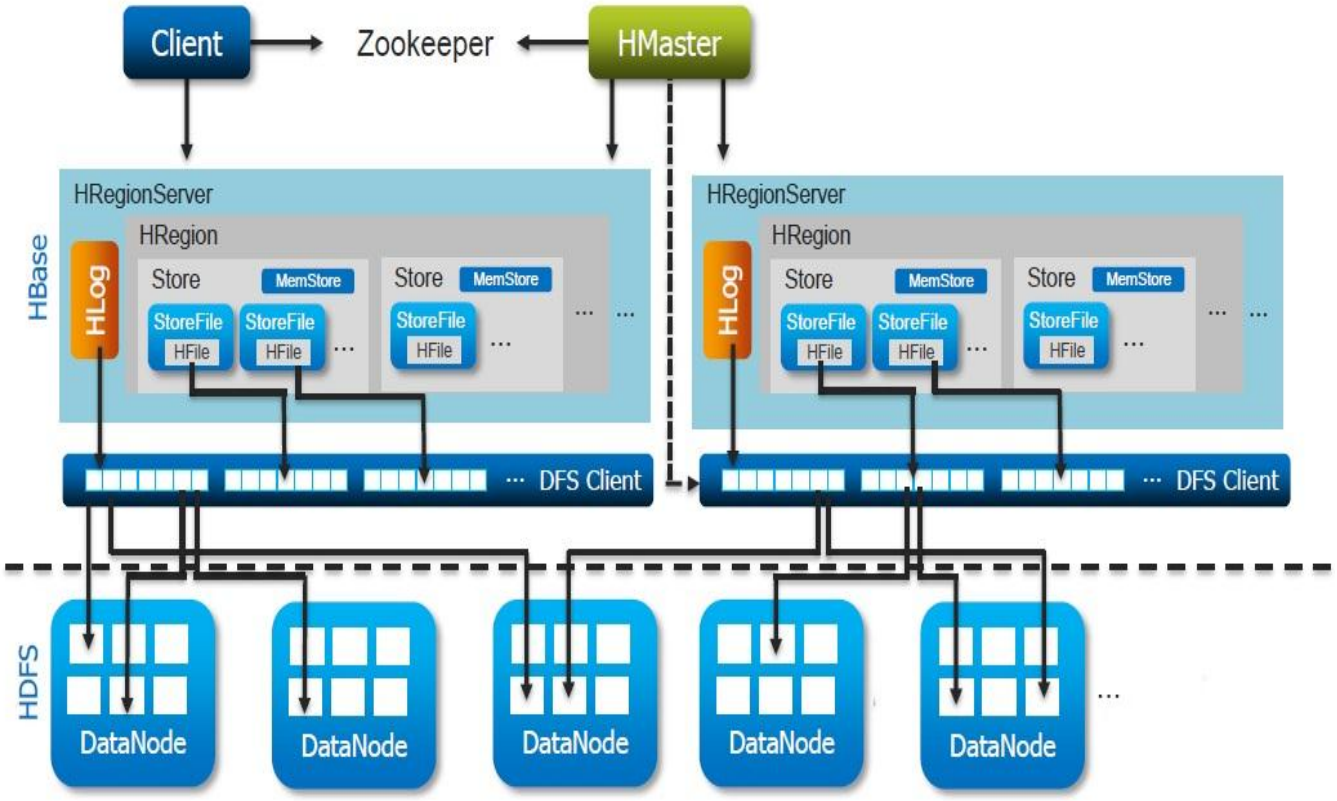

Fig 2. HBase Architecture [5]

4.1 HBase Architecture

HBase architecture has 3 main components

- HMaster
- Zookeeper
- Region server

A. HMaster

When HBase starts HMaster will assign the region to the Hregion server with the help of zookeeper. It is also responsible for managing and coordinating activities of schema design. HMaster also takes care of load balancing of region server. HMaster has the details of metadata. To generalize HMaster in the simpler way we can call it Team leader.

B. Zookeeper

Zookeeper provides services such as naming, synchronization and also maintain configuration information. It provides a communication path between the client and regional servers. Zookeeper has temporary nodes which are used by master servers to discover available servers. We can generalize it as Project Leader.

C. Region Server

Region server has regions which were used for read – write request handling. It also communicates with the client, decide the size of the region and also handle data-related operations.

- MemStore: “It stores all the incoming data before committing it to the disk or permanent memory. There is one Memstore for each column family in a region [6]”.
- HFile: “It stores the actual cells on the disk [6]”.
- WAL: “It stores the new data that hasn’t been committed to the permanent storage [6]”. WAL is stored in HDFS so that if any time one of region server go down we can rebuild everything using the log file.

The flow of this architecture works as, the client will communicate with Zookeeper and Zookeeper get updated data from HMaster, as there is no direct mapping between Hregion

server and data node. Then after client comes to know in which Hregion server it should follow. Once the client knows the region it will contacts Hregion server directly.

4.2 Storage Mechanism

HBase is designed to work with huge data, it follows the column-oriented database approach in which tables sorted by row. only column families define as table schema, column family can have any number of columns with their respective key value pairs. This kind approach more suitable for online analytical processing (OLAP).

Rowid	Column family			Column family		
1	Col1	Col2	Col3	Col1	Col2	Col3
2						
3						

4.3 Application of Hbase

- **Adobe:** “Adobe use it to write data to HBase and run MapReduce jobs to process then store it back to Hbase or external systems. They have 30 nodes running HDFS, out of which 5 to14 nodes used for production and development [7]”.
- **Facebook:** “Facebook use Hbase for Messages Infrastructures [7]”.
- **Yahoo:** “It uses Hbase to store document fingerprint for detecting near-duplications [7]”.
- **Twitter:** "HBase provides distributed, read/write a backup of all MySQL tables in Twitter's production backend, allowing engineers to run MapReduce jobs over the data while maintaining the ability to apply periodic row updates [7]."

4.4 Difference between Column-oriented and Row-Oriented Databases

Column-oriented	Row-Oriented
More used with Analytical Processing	More used with Transactional Processing
Only relevant columns are required	All the columns are required
Strong Compression due to single datatype and range of values.	Poor compression due to multiple types in the row.
Efficient and used when working with the Huge dataset.	Efficient and used when working with a small dataset.
Provides Fast Access to data	Less Fast than Column Oriented Database.

4.5. Difference between Hbase and RDBMS

Hbase	RDBMS
It defines Column Families. It is schema-less.	It has a schema which describes the structure of tables.
It is Horizontally Scalable.	It is Hard to Scale.
Not Transactional.	It is Transactional.
It is good for both semi-structured as well as structured data.	It is well suited for Structured Data.

5. Difference between HDFS and HBase

HDFS	HBase
It is a Distributed file system suitable for storing huge data.	It is a Database run on top of HDFS.
Not Supports Fast record lookups.	It supports fast lookups for large tables.
It provides sequential access of data.	It uses key, value pair and provides random access.
High latency batch processing.	It provides the Low latency (Random access).

6. Summary

This Report gives a basic idea of the HDFS, HDFS architecture and details about its main components such as Name node, Data node, Data Replication. Also basic Hdfs operations, Goals, Pros and Cons of HDFS. It also explains the HBase, HBase architecture, storage mechanism of HBase, which companies uses HBase and for what purpose. Finally, we sum up the report with the difference between HDFS and HBase.

7. References

[1] https://hadoop.apache.org/docs/r1.2.1/hdfs_design.html

[2] https://www.tutorialspoint.com/hadoop/hadoop_hdfs_operations.htm

[3] https://www.tutorialspoint.com/hadoop/hadoop_hdfs_overview.htm

[4] http://www.knowledgehut.com/blog/bigdata-hadoop/top-pros-and-cons-of-hadoop

[5] https://www.edureka.co/blog/overview-of-hbase-storage-architecture/

[6] https://www.edureka.co/blog/hbase-architecture/

[7] http://hbase.apache.org/poweredbyhbase.html

[8] http://www.webopedia.com/TERM/C/commodity_hardware.html

[9] https://en.wikipedia.org/wiki/Apache_Hadoop

[10] https://en.wikipedia.org/wiki/Apache_HBase