Distributed System in Apache HBase

Arjun Jaysukhbhai Anghan
Master of Applied Computer Science
Montreal, QC, Canada
arjunanghan@gmail.com

Kary Sutariya
Master of Applied Computer Science
Montreal, QC, Canada
sutariyakary11@gmail.com

Sagar Sanghani
Master of Applied Computer Science
Montreal, QC, Canada
sasagarsa1998@gmail.com

ABSTRACT

Data is raw material to make money which is the driving force for finding out a better option for data processing. With Relational databases, the cost of processing is unbearable which is also slow. In another word, it can be said that relational databases can't handle an enormous amount of data because of their strict structure as well as redundancy issue is always there with relational data. Due to these major reasons, many MNCs are spending their resources behind the research work for getting a better database model. Hadoop and HBase are the results of these research works. HBase has been forged to handle large data. In this project, we have utilized Hadoop and HBase to implement a few distributed system concepts.

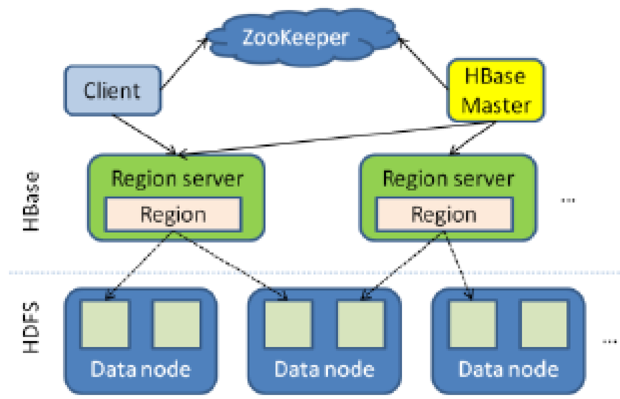

Figure 1: Apache HBase Architecture diagram [4]

1 INTRODUCTION

Apache Hbase is an open-source non-relational distributed column-oriented database. It is built over HDFS (Hadoop Distributed File System). It is a part of the Hadoop ecosystem. Hbase was inspired by Google's Big Table. HBase is a completely column-oriented database. It uses Map-Reduce framework to store and retrieve data. It has the capability to store data in tabular form and provide swift read and write operations. It also provides random access to a large amount of structured data. Hbase is also horizontally scalable. Apache HBase architecture has two essential components: HMaster and Region Server

- **HMaster** – HMaster along with the zookeeper form the master server in HBase. One of the main tasks of HMaster is to assign and manage regions of the region servers. It also balances the load between the regions to prevent underutilized and over-utilized servers. HMaster monitors the region servers and executes suitable steps in case of region server failures. Furthermore, it is also responsible for creating, delete and update of the tables and their metadata.
- **Region Server** – Data in Hbase is divided into regions in the region servers. Regions are allocated in a region server, each region has column of all the column family of the HBase table. Region server manages the read and write operations of the regions allocated under itself. The default size of a region is 256 Megabytes.

There are four components of a Region Server, which runs on an HDFS data node:

- **WAL** – WAL - Write ahead log is a file in Region server. It is used to store new data that has not been committed to the permanent storage. So in case of failure, WAL can be used for recovery.
- **BlockCache** – BlockCache stores frequently read data to provide quick access to the data. The least recently used data is removed from the BlockCache when it reaches its full capacity. It is also known as read cache.
- **MemStore** – MemStore holds the new data that has not been committed to the disk. It is also known as write cache.
- **Hfiles** – These files store the actual data as sorted KeyValues on disk.

1.1 Hadoop Distributed File System(HDFS)

Hadoop File System uses Hadoop Distributed File System at its base level and it uses commodity hardware which makes it unique and cheaper than other Distributed Systems. HDFS can handle Fault Tolerance even though the hardware cost for making the system is low. It can handle a huge amount of data with more accessibility towards each chunk of data present in the system. In order to store huge data, it is divided into multiple parts across multiple machines in a redundant manner so that if one of the machines fails, the system can retrieve data from another machine where the redundant data was stored. For the data to be accessed and modified, it requires parallel processing which can be made available using

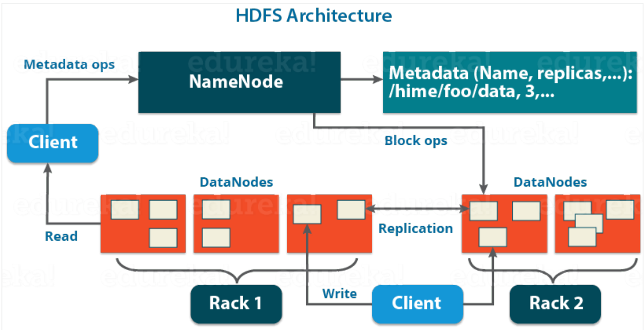

Figure 2: HDFS Architecture diagram [1]

HDFS. HDFS follows a Master-Slave architecture with two major elements; Namenode and Datanode.

- **Namenode** – It is also one of those commodity hardware with GNU/Linux OS and the NameNode software to implement the functionalities to be provided in the Distributed FIle System and it can be run on any hardware regardless of having high system specifications. Name Node acts as Master Server and performs the following tasks: It handles the File Namespace of the whole system and provides full information about each file present in the whole system. It can also control the accessibilities on files by different users. It supports normal system file operations like Renaming, Closing and opening the files and directories.
- **Datanode** – It stores the data in the form of a small part and can recognize frequently accessed data and stores it in the primary memory for faster and easier access while performing tasks on the data. Similarly, when the data is not accessed for a longer time, it gets deleted from that data node in order to save the storage when the storage capacity is touched. Data Node performs read-write operations on those data files as mentioned by the client. It can perform operations like block creation for storing data in block manner, deletion of the data from the DFS, and replication of data as instructed by the namenode.

1.2 Map-Reduce Framework

MapReduce is a software framework that consists of two phases, Map and Reduce. Map tasks address splitting and mapping data, while Reduce tasks handle rearranging and reducing data. MapReduce programs can be written in a number of languages, including Java, Ruby, Python, and C++. They are parallel in nature, making them ideal for performing large-scale analytics by using multiple machines in a cluster.The input to each phase is a pair of key-value pairs, and a programmer needs to specify two functions: the map function and the reduce function.

1.3 ZooKeeper

Zookeeper is a distributed application that acts as a mediator for master and region servers to communicate with each other.[5] All the region servers and HMaster servers and its backup servers are registered with Zookeeper. Zookeeper provides services like configuration service, managing, naming, job scheduling, and distributed synchronization. It provides five consistency guarantees like sequential, atomicity, single system image, reliability, timeliness. If any node fails then quorum manager of Zookeeper will take action to repair the failed nodes. Users have to access the quorum manager of Zookeeper to connect Hmaster and region servers.

2 IMPLEMENTATION

In order to implement this project, we have used GCP (Google cloud platform) as Infrastructure as a service(iaas). In GCP, we have created four virtual machines. One machine acts as namenode and other three act as datanode(region servers). Firstly, in these machines, we have configured Hadoop. After that, we set up HBase[10] in all machines and load our dataset with help of the Map-reduce framework, which distributed this dataset equally across these region servers. We have taken the airline delay analysis dataset[9] with the size of 1.5 GB. Using HBase shell we perform different operations like retrieval and updating data to accomplish operations on data in a distributed system. We have learned three different distribution system concepts like Replication, Fault tolerance, and Consistency.

2.1 Google Cloud Platform(GCP)

Since Apache HBase runs on top of Hadoop, we must use Hadoop to set up Apache HBase. To implement Hadoop to its full potential, you need a cluster of nodes. There are many cloud management tools, for instance Microsoft Azure, Amazon Web Services (AWS), Google Cloud Platform (GCP), etc. Here, GCP is used as IaaS (Infrastructure as a Service) to set up the system. This is to provide a wide range of services, including:

- Virtual Machines
- Security and Identity Management
- Networking
- Storage Services
- Big Data

In order to implement this Apache Hbase system we have used compute engine service of GCP beacuse GCP provides a scalable range of computing options that you can tailor to match your needs.GCP also provides a features of highly customizable VMs in which you can create your own VM as per your requirements.Using this VMs you can deploy your system very easily.

2.2 Configuration

In this project, we used the Compute Engine service to create four VM instances (one NameNode, three DataNodes). The specifications of NameNode are four core vCPUs, 16 GB ram, 200 GB standard persistent disk, centos 8 Operating System. At the same time, DataNode specifications are 2 core vCPU, 4 GB ram, 100 GB standard persistent disk, centos 8 Operating System.[12][14][11] We have installed Hadoop and Apache HBase in all the data node and name node. [13] [8]

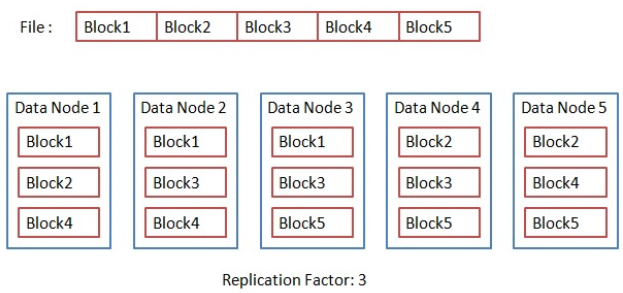

Figure 3: Replication in HDFS [3]

2.3 Replication

Replication in the distributed system means that we made multi copies of our data and it helps to ensure availability. This feature is very essential when it comes to handle fault tolerance. For implementing this feature we have to set the replication factor when we configure Hadoop. This is the number of times the system copies a certain piece of data. This feature is very essential because if any time there is a hardware failure situation occurs, we can even retrieve our data.
For replication in our project we are using HDFS replication, and the default replication factor of HDFS is 3.[2] It means that anytime a user saves data on any node, HDFS will make one local copy in that node and two copies in two different nodes.

For instance, in above figure 3 we have five file blocks to store in datanodes so for that for each file block, HDFS will create three copies and these three copies are stored in different datanodes. If any user wants to set a custom replication factor, then it can be changed from the hdfs-site.xml file.

2.4 Fault Tolerance

Fault Tolerance is the paramount parameter for all distributed systems. HBase has a special structure for fault tolerance. For this, we must understand three terms:

(1) **Region**: Chunks of tables that are spread all over the region servers of one data node. (scalability)
(2) **Region server**: It performs actual operations namely read, write, etc.
(3) **Master server**: Assign Regions to Region servers with the help of Zookeeper. (Load balancing)

The master server is the one who handles administrative operations. It assigns Regions to Region servers. It also makes replicas of these regions and stores them on different Region servers.[6] In case, if a Region server goes down, we can get data from another Region server. This shows the implementation of fault tolerance in HBase.

2.5 Consistency

When data is same across all the nodes at a given time, then the data is said to be consistent. Principally, data-centric consistency model and client-centric consistency model are the consistency models in distributed systems.
By default, Apache Hbase provides strong consistency, but it can

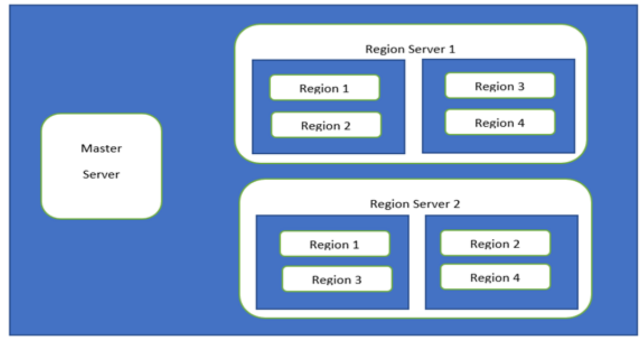

Figure 4: Fault-Tolerant System

be changed. Strong consistency means that if data is updated in one of the nodes then that update is strongly propagated to all the nodes in the distributed system so that users connected to different nodes will get same updated data. Furthermore, Zookeeper delivers consistency guarantees like atomicity and sequential consistency. Atomicity means that all updates are applied to all the nodes, or no updates are applied to any node. It essentially means that partial updates are not allowed. If any failure occurs during an update then the whole update process is failed. Sequential consistency means that operations in system will have a total order.

2.6 HBase Shell

Apache HBase provides a shell in which we can write a query and using that we can perform CRUD operations. It mainly provides two types of command one is Data Definition Language and the second is Data Manipulation Language. Using these commands we can perform different operations like insert data, retrieve data, update data, delete data, drop entire table, count and truncate, etc[7]. Below are the basic Hbase shell command list:

(1) **create**: Creates a table.
(2) **delete**: Delete a cell value in table.
(3) **get**: Retrieve data from a particular row of the table.
(4) **put**: Put a cell value in a particular table.
(5) **scan**: Display data in the table, limit can be used to show first n rows.
(6) **list**: Lists all the tables in the databases.

3 BENEFITS OF APACHE HBASE

- Can handle big data
- Auto recovery
- Strong consistency
- Built on Hadoop technologies
- Active development community [15]
- Random, swift and consistent read/write access
- Can be used with Java APIs and REST APIs
- Automatic splitting of data
- Low Latency

4 DRAWBACKS OF APACHE HBASE

- Lacks a friendly, SQL-like query language

- Setup beyond a single-node development cluster can be difficult [15]
- Requires high memory machines

5 CONCLUSION

HBase is a powerful tool to manage a huge amount of data but it has its own limitation. All the concepts of a distributed system can be implemented because of Zookeeper. Without it, we can use HBase in just stand-alone mode which doesn't make more sense. Moreover, the non-relation database architecture of HBase makes it a better option for processing humongous data.

REFERENCES

[1] Ashish Bakshi. Hdfs architecture.
[2] dikshantmalidev. Replication factor.
[3] dikshantmalidev. Replication in hadoop.
[4] Xiaoming Gao. Hbase architecture.
[5] Zookeeper Org. Zookeeper: A distributed coordination service for distributed applications.
[6] Tutorials Point. Fault tolerance.
[7] Tutorials Point. Hbase shell.
[8] RAN. How to configure fully distributed hbase cluster on centos 7.
[9] Sherry and U.S. Department of Transportation (DOT). Airline delay analysis dataset, 2018.
[10] Siva. Hbase installation : Fully distributed mode.
[11] David Taylo. How to install hadoop with step by step configuration on linux ubuntu.
[12] techcert learn. Hadoop multi node cluster setup in google cloud platform.
[13] Jayati Tiwari. Hbase installation : Fully distributed mode.
[14] USAINTube. Creating hadoop 2-node cluster.
[15] Info World. Review: Hbase is massively scalable – and hugely complex.