No-SQL Column-Oriented Database - Apache HBase

Jaynil Savani
Master of Applied Computer science
Concordia University
Montreal, QC, Canada
jaynilsavani1998@gmail.com

Manthan Moradiya
Master of Applied Computer science
Concordia University
Montreal, QC, Canada
manthan.p.moradiya@gmail.com

YashKumar Vaghani
Master of Applied Computer science
Concordia University
Montreal, QC, Canada
yashvaghani1997@gmail.com

VickyKumar Patel
Master of Applied Computer science
Concordia University
Montreal, QC, Canada
patelvicky1995@gmail.com

ABSTRACT

As technology is growing nowadays, more and more users are accessing and dealing with data of any type. Storing and maintaining these data is very costly and hard to achieve in the relational data model, and it's challenging to handle data in a distributed system with a relational database. Therefore, many organizations prefer non-relational databases like MongoDB, Apache Cassandra, Couchbase, and Apache HBase. Moreover, the non-relational data model can operate a massive amount of unstructured data. In this project, We are using Apache Hbase to implement distributed system concepts.

1 INTRODUCTION

Apache HBase is a database management system designed in 2007 by Powerset, a Microsoft System. Apache HBase totally depends on Google's BigTable. It's an entirely non-relational database. It can store a large amount of a data in tabular format for swift read and write operation. Apache HBase is built over HDFS (Hadoop Distributed File System). HBase uses the Map-Reduce framework to retrieve and store data, and it is a fully column-family-oriented database. HBase depends on ZooKeeper, which is used for coordination between HBase master and region servers. Nowadays, many well-known organizations are using Apache HBase as their main data storage, like Pinterest, Tumblr, Adobe, Yahoo, Awin, HubSpot, SendGrid, etc.

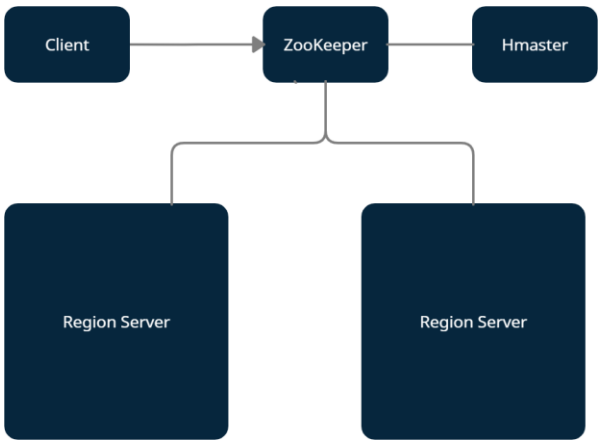

Figure 1: Apache HBase Architecture diagram

There are two main components of Apache HBase: Hmaster and Region Server. Where Hmaster provides administrative services in cluster. It is responsible for load balancing, region assignment, and other data related operations. Region server in HBase contains four elements:

- WAL
- BlockCache
- MemStore

- Hfiles

WAL is a write ahead log file that stores new data which is not persistent to the permanent storage. BlockCache stores frequent read data so that it provides faster access to this data. MemStore stores all the new information which the client writes in the database. HFile is actual storage where data is stored in a (key, value) pair.

1.1 Hadoop Distributed File System

Hadoop Distributed File System is based on a google file system. It uses master-slave architecture, and it's highly fault-tolerant. HDFS replication factor is three, so users get three copies of the same data whenever the user stores data in HDFS. In HDFS master is known as name node and slave is considered as a data node. Data are stored in a distributed pattern over all the data nodes. Name node keeps the information about what data are stored in which data node.

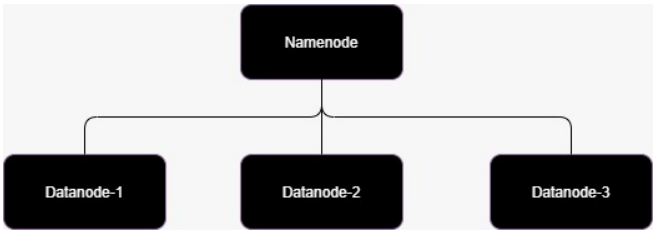

Figure 2: HDFS Architecture diagram

1.2 Map-Reduce Framework

Map-reduce is a data processing programming model. It removes bad data and retains necessary information. It contains two phases: map and reduces phase. In which map phase covers one form of input to another form like tuples(key, value). Reduce phase performs shuffle and reduce operations. Reduce phase takes input from map phase and apply user define a function on this input.

1.3 ZooKeeper

Zookeeper is a distributed application that is required for coordination between master and region servers. Hmaster and region servers are registered themselves with Zookeeper. Users have to access the quorum manager of Zookeeper to connect Hmaster and region servers. It provides services like managing/precuring configuration, distributed synchronization, naming, group services, and job scheduling. It provides five consistency guarantees like sequential, atomicity, single system image, reliability, timeliness. Whenever any node fails, quorum manager of Zookeeper will take action to repair the failed nodes.

2 IMPLEMENTATION

We have used GCP (Google cloud platform) to set up virtual machines with the Hadoop file system. We have implemented three region servers and one master node to set up a distributed system with the help of a zookeeper. We have studied the different distribution system concepts like Naming, Replication, Fault tolerance, Consistency, Atomicity, etc. We have taken data of size more than 1 GB. Then, we distribute data equally to six region servers with the help of the Map-Reduce framework. Therefore, these data can be used to perform some operations like retrieve data, update data, etc. Apache HBase provides easy queries to perform these tasks on data in the distributed system.

2.1 Google Cloud Platform

In order to set up Apache HBase, it is required to use Hadoop. To implement Hadoop, one needs high-performance machines. There are many cloud management tools are available in the market, such as Microsoft Azure, Amazon Web Service(AWS), Google Cloud Platform(GCP), etc. Here, we use GCP as an IaaS (Infrastructure as a Service) to set up our project as it provides a wide range of services such as

- Management and Developer Tools
- Identity & Security
- Storage and Databases
- Networking
- Compute
- Machine Learning
- Big Data

Here, we use the Compute Engine Service of GCP to set up our Apache HBase project, as GCP provides a scalable range of computing options that you can tailor to match your needs. It also provides highly customizable VMs, too. And provides the option to deploy your code directly or via containers.

In this project, we created four VM instances (1- NameNode, 3- DataNode) using compute engine services. The specifications of NameNode are four core CPUs, 8 GB ram, 50 GB Balanced Disk Space, Linux centos 7 Operating System. At the same time, DataNode specifications are 2 core CPU, 4 GB ram, 50 GB Balanced Disk Space, Linux centos 7 Operating System.

We have installed Hadoop and Apache HBase in all the datanode and namenode. As we mentioned earlier, Apache

HBase requires ZooKeeper; thus, we set up a quorum peer in two datanode and namenode.

2.2 Naming

Naming is required to identify entities in a distributed system. Name is a string of characters that is used to refer to an entity. Naming works with various techniques like Broadcasting and Forward points. In this project, ZooKeeper will provide this name resolution facility.

In Apache HBase, whenever a client requests data. That request will be processed in ZooKeeper. Then after ZooKeeper will decide which master or region server should process this request. Moreover, Hamster can contact to region server through ZooKeeper.

2.3 Replication

Replication in the distributed system ensures that data is available without any interruption. It is vital in a distributed system to solve the problem of failure of its components. The replication factor is an important aspect of replication in the distributed system. Which can be defined as the number of times the system makes a copy of specific data. Replication is very important in case of node failures because to set up Hadoop we are using commodity hardware, and such a system can be failed at any time.

In our project, Apache HBase uses HDFS replication. By default, the replication factor of HDFS is 3. It means whenever the user stores the data in any node, HDFS will make one copy in local and the other two copies in second and ternary nodes, respectively.

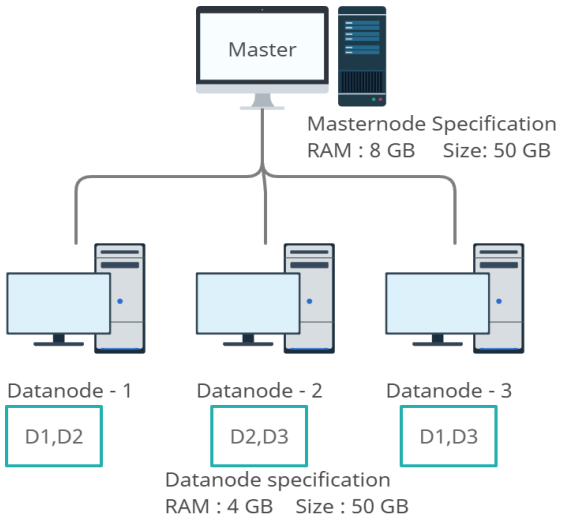

Figure 3: Replication in HDFS

For instance, we have six file blocks to store then three copies of each file block will be saved in data nodes so that 6 * 3 = 18 replicas will be generated for backup. 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

Distributed system's main purpose is to make such a system that it can automatically repair from failure without affecting the whole system. Fault tolerance can be defined as a system that works uninterruptedly in the case of failure of one or more components. It can be useful to achieve high availability in the system. High availability means the system should be available at any point in time.

In this project, we have achieved fault tolerance through replication. Apache HBase uses ZooKeeper, and it is responsible for getting a fault-tolerant system via a leader election algorithm.

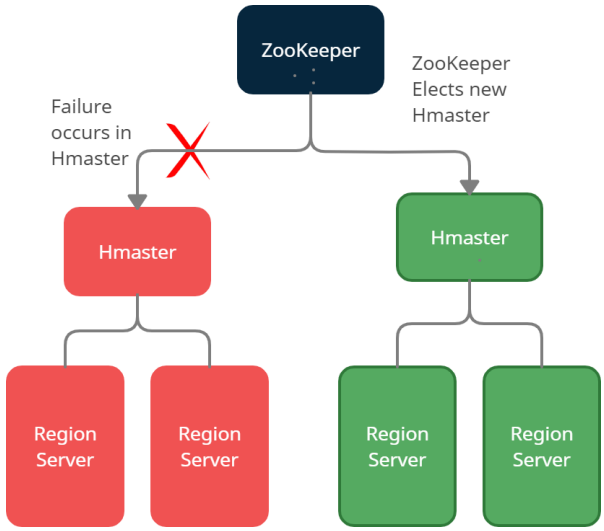

Figure 4: Fault-Tolerant System

Whenever any master node goes down and the system is interrupted then, ZooKeeper will run this algorithm to select a new leader or master node. In the above diagram, it is clearly described that one master node is failed due to some reason and ZooKeeper elects a new master node to achieve fault tolerance.

2.5 Consistency

Consistency in the distributed system means all the nodes contain the same data at any time. There are two types of

Consistency models in a distributed systems: the data-centric consistency model and the client-centric consistency model.

In Apache HBase, ZooKeeper provides some consistency guarantees like sequential consistency and atomicity. Sequential consistency, referred to as client updates, proceed in the same order as the client has sent. Atomicity is described as all the updates are successfully applied, or no updates are applied to the nodes. This means if any failure occurs during updates, the whole update process will be failed. So that, partial updates are not allowed in Apache HBase.

2.6 HBase Shell

Apache HBase provides an Hbase shell by which users can access Hbase facilities. It supports mainly two types of command definitions: Data Manipulation Language (DML) and Data Definition Language (DDL). DDL supports commands which can be used to perform some table-related operations in Hbase, such as create, alter, enable, disable, list. Whereas DML supports data manipulation commands on the table in Hbase like scan, put, delete, deleteall, get, count. For instance, if any user wants to see the list of available tables in Hbase then the user has to write the **'list'** command in the Hbase shell.

3 Benefits of Apache HBase

There are many advantages of Apache HBase, such as:

- Can handle big amount of data
- Powerful consistency model
- Automatically splitting of regions.
- Auto recovery
- Random & Quick read/write access
- Can be used with Java APIs
- Highly Reliable
- Integrated with Hadoop and Map-Reduce

4 CONCLUSION

To put it into a nutshell, Apache HBase is capable of operating a huge size of datasets with ease. It stores all the data on HDFS. Moreover, it uses ZooKeeper to reinforce distributed system's concepts such as replication, naming, etc. Additionally, it provides easy support to access column-oriented datasets.

5 REFERENCES

[1] Replication in HDFS https://www.geeksforgeeks.org/hadoop-file-blocks-and-replication-factor/

[2] Google Cloud Platform(GCP) https://www.edureka.co/blog/what-is-google-cloud-platform/

[3] Apache HBase https://hbase.apache.org/

[4] Apache HBase Region Server https://data-flair.training/blogs/hbase-architecture/

[5] Apache HBase Shell https://www.tutorialspoint.com/hbase/hbase_shell.htm