A Review Paper on Hadoop

Payal Gothi[1] Riddhi Kamat[2]
[1,2]Information Technology
[1,2]Atharva College of Engineering, Mumbai

Abstract— The term BIG DATA is used to describe the collection of complex and large data sets such that it is difficult to store, process and analyze this kind of data using conventional database management tools and traditional databases management systems. Earlier RDBMS systems tried to handle this unstructured and semistructred large chunk of data but couldn't handle the same. So hadoop came into existence. In this paper we present Hadoop and its core concepts which are HDFS and MapReduce. Hadoop is one of the few frameworks that support storing of unstructured data like video files, audio files, image files etc. along with storing the normal structured data. The Hadoop Distributed File System (HDFS) is a distributed file system designed to run on commodity hardware. MapReduce in Hadoop is a framework for writing applications that process large amounts of structured and unstructured data in parallel across a cluster of thousands of machines, in a reliable and fault tolerant manner.

Key words: Big Data, Hadoop, HDFS, MapReduce

I. INTRODUCTION

Apache Hadoop is an open source framework for distributed storage and processing of large sets of data on commodity hardware. Hadoop enables businesses to quickly gain insight from massive amounts of structured and unstructured data. It is a set of algorithms (an open source software framework written in Java) for distributed storage and distributed processing of very large data sets (Big Data) on computer clusters built from commodity hardware. All the modules in Hadoop are designed with a fundamental assumption that hardware failures (of individual machines, or racks of machines) are commonplace and thus should be automatically handled in software by the framework. Apache Hadoop is 100% open source, and pioneered a fundamentally new way of storing and processing data. Instead of relying on expensive, proprietary hardware and different systems to store and process data, Hadoop enables distributed parallel processing of huge amounts of data across inexpensive, industry-standard servers that both store and process the data, and can scale without limits. With Hadoop, no data is too big. And in today's hyper-connected world where more and more data is being created every day, Hadoop's breakthrough advantages mean that businesses and organizations can now find value in data that was recently considered useless.

This paper is organized as follows. In Section 2, we discuss the key features of Hadoop. In Section 3, HDFS is described. It is a subproject of Apache Hadoop. In Section 4, MapReduce is presented. It is a software framework for distributed processing of large data sets on compute clusters of commodity hardware. In Section 5, conclusions are drawn. Section 6 provides the references.

II. KEY FEATURES

Hadoop is a highly scalable storage platform, because it can store and distribute very large data sets across hundreds of inexpensive servers that operate in parallel. Unlike traditional relational database systems (RDBMS) that can't scale to process large amounts of data, Hadoop enables businesses to run applications on thousands of nodes involving thousands of terabytes of data. It also offers a cost effective storage solution for businesses' exploding data sets. The problem with traditional relational database management systems is that it is extremely cost prohibitive to scale to such a degree in order to process massive volumes of data. In an effort to reduce costs, many companies in the past would have had to down-sample data and classify it based on certain assumptions as to which data was the most valuable. The raw data would be deleted, as it would be too cost-prohibitive to keep. While this approach may have worked in the short term, this meant that when business priorities changed, the complete raw data set was not available, as it was too expensive to store. Hadoop, on the other hand, is designed as a scale-out architecture that can affordably store all of a company's data for later use. The cost savings are staggering: instead of costing thousands to tens of thousands of pounds per terabyte, Hadoop offers computing and storage capabilities for hundreds of pounds per terabyte.

Hadoop enables businesses to easily access new data sources and tap into different types of data (both structured and unstructured) to generate value from that data. This means businesses can use Hadoop to derive valuable business insights from data sources such as social media, email conversations or clickstream data. In addition, Hadoop can be used for a wide variety of purposes, such as log processing, recommendation systems and data warehousing. Hadoop's unique storage method is based on a distributed file system that basically 'maps' data wherever it is located on a cluster. The tools for data processing are often on the same servers where the data is located, resulting in much faster data processing. If you're dealing with large volumes of unstructured data, Hadoop is able to efficiently process terabytes of data in just minutes, and petabytes in hours.

A key advantage of using Hadoop is its fault tolerance. When data is sent to an individual node, that data is also replicated to other nodes in the cluster, which means that in the event of failure, there is another copy available for use.

III. HADOOP DISTRIBUTED FILE SYSTEMS (HDFS)

The Hadoop Distributed File System (HDFS) is a distributed file system 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 relaxes a few POSIX requirements to enable streaming access to file system data. HDFS was originally built as infrastructure for the Apache Nutch web search engine project. HDFS is now an Apache Hadoop subproject. Section 3.1 describes the architecture of HDFS.

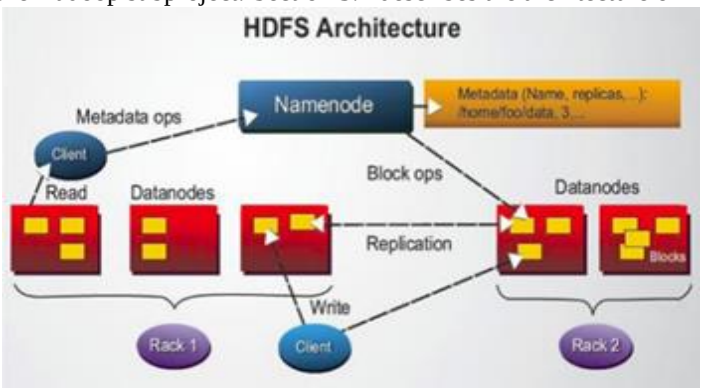

Fig. 1: Section 3.1 Architecture of HDFS

HDFS has a 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. HDFS exposes a file system namespace and allows user data to be stored in files. Internally, a file is split into one or more blocks and these blocks are stored in a set of DataNodes. The NameNode executes file system namespace operations like opening, closing, and renaming files and directories. It also determines the mapping of blocks to DataNodes. The DataNodes are responsible for serving read and write requests from the file system's clients. The DataNodes also perform block creation, deletion, and replication upon instruction from the NameNode.

The NameNode and DataNode are pieces of software designed to run on commodity machines. These machines typically run a GNU/Linux operating system (OS). HDFS is built using the Java language; any machine that supports Java can run the NameNode or the DataNode software. Usage of the highly portable Java language means that HDFS can be deployed on a wide range of machines. A typical deployment has a dedicated machine that runs only the NameNode software. Each of the other machines in the cluster runs one instance of the DataNode software. The architecture does not preclude running multiple DataNodes on the same machine but in a real deployment that is rarely the case. The existence of a single NameNode in a cluster greatly simplifies the architecture of the system. The NameNode is the arbitrator and repository for all HDFS metadata. The system is designed in such a way that user data never flows through the NameNode. The NameNode maintains the file system namespace. Any change to the file system name spaceorits properties is recorded by the NameNode. An application can specify the number of replicas of a file that should be maintained by HDFS. The number of copies of a file is called the replication factor of that file. This information is stored by the NameNode.

HDFS is designed to reliably store very large files across machines in a large cluster. It stores each file as a sequence of blocks; all blocks in a file except the last block are the same size. The blocks of a file are replicated for fault tolerance. The block size and replication factor are configurable per file. An application can specify the number of replicas of a file. The replication factor can be specified at file creation time and can be changed later. Files in HDFS are write-once and have strictly one writer at any time. The NameNode makes all decisions regarding replication of blocks. It periodically receives a Heartbeat and a Block report from each of the DataNodes in the cluster. Receipt of a Heartbeat implies that the DataNode is functioning properly. A Block report contains a list of all blocks on a DataNode.

Each DataNode sends a Heartbeat message to the NameNode periodically. A network partition can cause a subset of DataNodes to lose connectivity with the NameNode. The NameNode detects this condition by the absence of a Heartbeat message. The NameNode marks DataNodes without recent Heartbeats as dead and does not forward any new IO requests to them. Any data that was registered to a dead DataNode is not available to HDFS anymore. DataNode death may cause the replication factor of some blocks to fall below their specified value. The NameNode constantly tracks which blocks need to be replicated and initiates replication whenever necessary. The necessity for re-replication may arise due to many reasons: a DataNode may become unavailable, a replica may become corrupted, a hard disk on a DataNode may fail, or the replication factor of a file may be increased.

The HDFS architecture is compatible with data rebalancing schemes. A scheme might automatically move data from one DataNode to another if the free space on a DataNode falls below a certain threshold. In the event of a sudden high demand for a particular file, a scheme might dynamically create additional replicas and rebalance other data in the cluster. These types of data rebalancing schemes are not yet implemented.

It is possible that a block of data fetched from a DataNode arrives corrupted. This corruption can occur because of faults in a storage device, network faults, or buggy software. The HDFS client software implements checksum checking on the contents of HDFS files. When a client creates an HDFS file, it computes a checksum of each block of the file and stores these checksums in a separate hidden file in the same HDFS namespace. When a client retrieves file contents it verifies that the data

it received from each DataNode matches the checksum stored in the associated checksum file. If not, then the client can opt to retrieve that block from another DataNode that has a replica of that block. 0

IV. MAPREDUCE

MapReduce is useful for many applications, such as web access log stats, machine learning and statistical machine translation. Map reduce works in two steps. A MapReduce job splits a large data set into independent chunks and organizes them into key, value pairs for parallel processing. This parallel processing improves the speed and reliability of the cluster, returning solutions more quickly and with more reliability. The Map function divides the input into ranges by the InputFormat and creates a map task for each range in the input. The JobTracker distributes those tasks to the worker nodes. The output of each map task is partitioned into a group of key-value pairs for each reduce. The Reduce function then collects the various results and combines them to answer the larger problem the master node was trying to solve. Each reduce pulls the relevant partition from the machines where the maps executed, then writes its output back into HDFS. Thus, the reduce is able to collect the data from all of the maps for the keys it is responsible for and combine them to solve the problem.

A MapReduce job splits a large data set into independent chunks and organizes them into key, value pairs for parallel processing. This parallel processing improves the speed and reliability of the cluster, returning solutions more quickly and with more reliability. The Map function divides the input into ranges by the Input Format and creates a map task for each range in the input. The JobTracker distributes those tasks to the worker nodes. The output of each map task is partitioned into a group of key-value pairs for each reduce. The Reduce function then collects the various results and combines them to answer the larger problem the master node was trying to solve. Each reduce pulls the relevant partition from the machines where the maps executed, then writes its output back into HDFS. Thus, the reduce is able to collect the data from all of the maps for the keys it is responsible for and combine them to solve the problem.

V. CONCLUSION

For big data to deliver on the promise of its vast potential, however, technology must be in place to enable organizations to capture and store massive amounts of unstructured data in its native format. That’s where Hadoop has become one of the enabling data processing technologies for big data analytics. Hadoop allows companies to store and manage far larger volumes of structured and unstructured data than can be managed affordably by today's relational database management systems.

REFERENCES

[1] https://developer.yahoo.com/hadoop/tutorial/module2.html
[2] https://hadoop.apache.org/docs/r1.2.1/hdfs_design.html
[3] http://www.tutorialspoint.com/hadoop/hadoop_mapreduce.html
[4] https://www.elastic.co/guide/en/elasticsearch/hadoop/.../features.html
[5] www.sas.com/en_us/insights/big-data/hadoop.html