© 2016 IJSRSET | Volume 2 | Issue 2 | Print ISSN : 2395-1990 | Online ISSN : 2394-4099
Themed Section: Engineering and Technology

A Hadoop Framework Require to Process Bigdata very Easily and Efficiently

Harin C Naik, Divyesh Joshi
Department of Computer Science and Engineering, Parul Institute of Engineering and Technology/GTU, Vadodara, Gujarat, India

ABSTRACT

Main aim of invention of Hadoop is to process of big data very efficiently. Nowadays, web is generating lots of information on a daily basis, and it is highly require and difficult to manage billion of pages of content. This paper will clearly describe the evolution of hadoop, its need and uses. Detail study of hadoop framework and its concepts to open source software to support distributed computing. Hadoop also includes a Distributed File System (HDFS), which manages distributed data on different node and Map-Reduce for programming paradigm.

Keywords: Hadoop, HDFS, Map-Reduce, Bigdata, Data mining, Apache, Distributed Computing

I. INTRODUCTION

Apache Hadoop™ was born out of a need to process an avalanche of big data. The web was generating more and more information on a daily basis, and it was becoming very difficult to index over one billion pages of content. In order to cope, Google invented a new style of data processing known as MapReduce. A year after Google published a white paper describing the MapReduce framework, Doug Cutting and Mike Cafarella, inspired by the white paper, created Hadoop to apply these concepts to an open-source software framework to support distribution for the Nutch search engine project. Given the original case, Hadoop was designed with a simple write-once storage infrastructure.

Hadoop has moved far beyond its beginnings in web indexing and is now used in many industries for a huge variety of tasks that all share the common theme of lots of variety, volume and velocity of data – both structured and unstructured. It is now widely used across industries, including finance, media and entertainment, government, healthcare, information services, retail, and other industries with big data requirements but the limitations of the original storage infrastructure remain. Hadoop is increasingly becoming the go-to framework for large-scale, data-intensive deployments. Hadoop is built to process large amounts of data from terabytes to petabytes and beyond. With this much data, it's unlikely that it would fit on a single computer's hard drive, much less in memory. The beauty of Hadoop is that it is designed to efficiently process huge amounts of data by connecting many commodity computers together to work in parallel. Using the MapReduce model, Hadoop can take a query over a dataset, divide it, and run it in parallel over multiple nodes. Distributing the computation solves the problem of having data that's too large to fit onto a single machine. [6]

II. METHODS AND MATERIAL

1. How Hadoop is different from relational databases

Hadoop can handle data in a very fluid way.

Hadoop is more than just a faster, cheaper database and analytics tool. Unlike databases, Hadoop doesn't insist that you structure your data. Data may be unstructured and schemaless. Users can dump their data into the framework without needing to reformat it. By contrast, relational databases require that data be structured and schemas be defined before storing the data.

Hadoop has a simplified programming model.

Hadoop's simplified programming model allows users to quickly write and test software in distributed systems.

Performing computation on large volumes of data has been done before, usually in a distributed setting but writing software for distributed systems is notoriously hard. By trading away some programming flexibility, Hadoop makes it much easier to write distributed programs. [2]

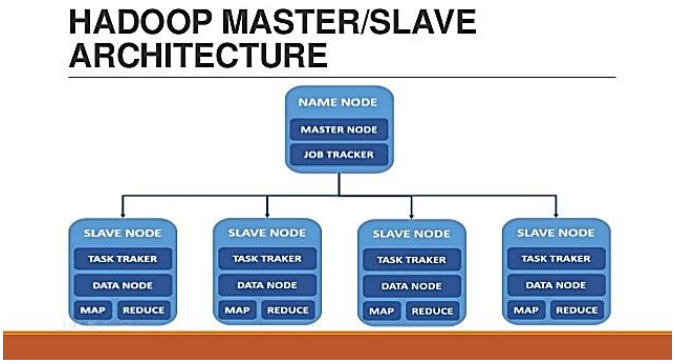

Figure 1 : Hadoop Master/Slave Architecture

Because Hadoop accepts practically any kind of data, it stores information in far more diverse formats than what is typically found in the tidy rows and columns of a traditional database. Some good examples are machine-generated data and log data, written out in storage formats including JSON, Avro and ORC. The majority of data preparation work in Hadoop is currently being done by writing code in scripting languages like Hive, Pig or Python. [5]

Hadoop is Easy to Administer

Alternative high performance computing (HPC) systems allow programs to run on large collections of computers, but they typically require rigid program configuration and generally require that data be stored on a separate storage area network (SAN) system. Schedulers on HPC clusters require careful administration and since program execution is sensitive to node failure, administration of a Hadoop cluster is much easier.

Hadoop invisibly handles job control issues such as node failure. If a node fails, Hadoop makes sure the computations are run on other nodes and that data stored on that node are recovered from other nodes.

Hadoop is agile

Relational databases are good at storing and processing data sets with predefined and rigid data models. For unstructured data, relational databases lack the agility and scalability that is needed. Apache Hadoop makes it possible to cheaply process and analyze huge amounts of both structured and unstructured data together, and to process data without defining all structure ahead of time.

2. Hadoop Distributed File System (HDFS)

Hadoop File System was developed using distributed file system design. It is run on commodity hardware. Unlike other distributed systems, HDFS is highly fault tolerant and designed using low-cost hardware.

HDFS holds very large amount of data and provides easier access. To store such huge data, the files are stored across multiple machines. These files are stored in redundant fashion to rescue the system from possible data losses in case of failure. HDFS also makes applications available to parallel processing.

Features of HDFS

- It is suitable for the distributed storage and processing.
- Hadoop provides a command interface to interact with HDFS.
- The built-in servers of namenode and datanode help users to easily check the status of cluster.
- Streaming access to file system data.
- HDFS provides file permissions and authentication.

HDFS Architecture

1) Namenode

The namenode is the commodity hardware that contains the GNU/Linux operating system and the namenode software. It is a software that can be run on commodity hardware. The system having the namenode acts as the master server and it does the following tasks:

- Manages the file system namespace.
- Regulates client's access to files.
- It also executes file system operations such as renaming, closing, and opening files and directories.

2) Datanode

The datanode is a commodity hardware having the GNU/Linux operating system and datanode software. For every node (Commodity hardware/System) in a cluster, there will be a datanode. These nodes manage the data storage of their system.

- Datanodes perform read-write operations on the file systems, as per client request.
- They also perform operations such as block creation, deletion, and replication according to the instructions of the namenode.

3) Block

Generally the user data is stored in the files of HDFS. The file in a file system will be divided into one or more segments and/or stored in individual data nodes. These file segments are called as blocks. In other words, the minimum amount of data that HDFS can read or write is called a Block. The default block size is 64MB, but it can be increased as per the need to change in HDFS configuration.

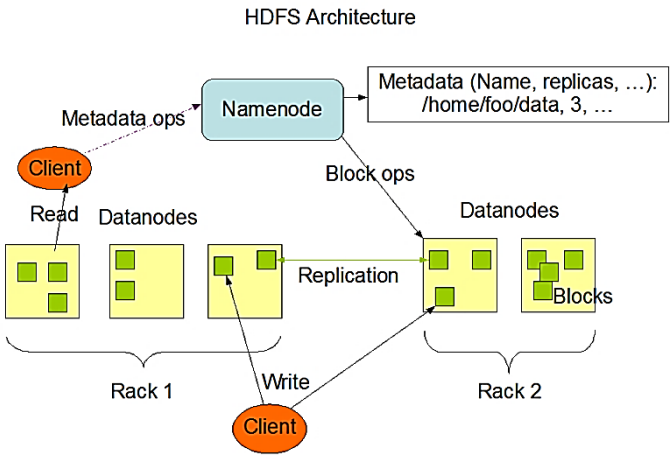

Figure 2: Architecture of HDFS

3. MAPREDUCE

MapReduce is a processing technique and a program model for distributed computing based on java. The MapReduce algorithm contains two important tasks, namely Map and Reduce. Map takes a set of data and converts it into another set of data, where individual elements are broken down into tuples (key/value pairs). Secondly, reduce task, which takes the output from a map as an input and combines those data tuples into a smaller set of tuples. As the sequence of the name MapReduce implies, the reduce task is always performed after the map job. [3]

The major advantage of MapReduce is that it is easy to scale data processing over multiple computing nodes. Under the MapReduce model, the data processing primitives are called mappers and reducers. Decomposing a data processing application into mappers and reducers is sometimes nontrivial. But, once we write an application in the MapReduce form, scaling the application to run over hundreds, thousands, or even tens of thousands of machines in a cluster is merely a configuration change. This simple scalability is what has attracted many programmers to use the MapReduce model.

The Algorithm

- Generally MapReduce paradigm is based on sending the computer to where the data resides!

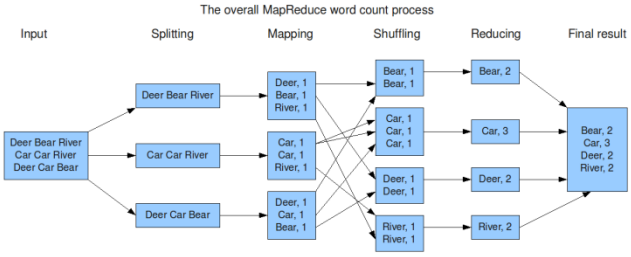

Figure 3 : Architecture of MapReduce

- MapReduce program executes in three stages, namely map stage, shuffle stage, and reduce stage.
  - **Map stage**: The map or mapper's job is to process the input data. Generally the input data is in the form of file or directory and is stored in the Hadoop file system (HDFS). The input file is passed to the mapper function line by line. The mapper processes the data and creates several small chunks of data.
  - **Reduce stage**: This stage is the combination of the **Shuffle** stage and the **Reduce** stage. The Reducer's job is to process the data that comes from the mapper. After processing, it produces a new set of output, which will be stored in the HDFS.
- During a MapReduce job, Hadoop sends the Map and Reduce tasks to the appropriate servers in the cluster.
- The framework manages all the details of data-passing such as issuing tasks, verifying task completion, and copying data around the cluster between the nodes.

- Most of the computing takes place on nodes with data on local disks that reduces the network traffic.

After completion of the given tasks, the cluster collects and reduces the data to form an appropriate result, and sends it back to the Hadoop server.

III. CONCLUSION

There are many ways to massively process data in parallel outside of a relational database. Hadoop follows the MapReduce programming paradigm to write programs to process the data. While Hadoop provides flexibility, the solution is very specific and requires software development skills to build a query to access the data.

IV. REFERENCES

[1] Guanghui Xu, Feng Xu, Hongu Ma. "Deploying and searching Hadoop in virtual machines" International Conference on Automation and Logistics, China, August 2012 IEEE Conferences.

[2] First Author and Second Author. 2002. International Journal of Scientific Research in Science, Engineering and Technology. (Nov 2002), ISSN NO:XXXX-XXXX DOI:10.251XXXXX

[3] Zhuong Zhang, Ludmila Cherkasova, Boon Thau Loo. "Getting more for less in optimized Map-Reduce Workflows" , IEEE 2013, pp 93-100

[4] Hai-Gaung li, Gong-Qing Wu, Xue-Gang Hu , Jing Zang, Lian Li, Xindong Wu. " K-Means Clustering with Bagging and Map-Reduce", 44th Hawaii International Conference on System Sciences, IEEE 2011 , pp 1-8.

[5] S.Ghemawat, H.Gobioff, S.Leung. " The Google file system",In Proc. Of ACM Symposium on Operating Systems principles,Lake George ,NY,Oct 2003,pp29-43.

[6] Apache Hadoop. http://hadoop.apache.org/

[7] Description if Single Node Cluster Setup at: http://www.michael-noll.com/tutorials/running-hadoop-on-ubuntu-linux-single-node-cluster/ visited on 21st January, 2012

[8] Description of Multi Node Cluster Setup at: http://www.michael-noll.com/tutorials/running-hadoop-on-ubuntu-linux-multi-node-cluster/ visited on 21st January, 2012

[9] Makho Ngazimbi ,PhD, "Data Clustering Using Map-Reduce" ,Boise State University ,March 2009

[10] Prajesh P Anchalia, Anjan K Koundinya, Srinath N K." Map-Reduce design of k-Means Clustering Algorithm", IEEE 2013.