International Journal of Engineering Trends and Applications (IJETA) – Volume 3 Issue 3, May-Jun 2016

RESEARCH ARTICLE OPEN ACCESS

Hadoop: Addressing Human Bottlenecks in Big Data

Harinderjit Kaur, Karambir Kaur, Surbhi

Department of Computer Science and Engineering

Punjab Institute of Technology

Kapurthala – India

ABSTRACT

As per the records, usually 2.5 quintillion of data is created per day- so much that 90% of the data in the world has been created in the last two years alone. The volume of data is rising tremendously. Different organisations have generated big and big data for years, but struggle to use it effectively. Moreover, data size is increasing at exponential rates with the advent of penetrating devices like android phones, social networking sites like LinkedIn, Facebook etc. and other sources like Google +, Data Sensor devices etc. All this plethora of data is termed as Big Data. There is a need to manage and process this Big data in a suitable manner to produce meaningful information. Traditional techniques of managing data have fall short to analyse this data. Due to its different nature of Big Data, various file system architectures are used to store it.

Big Data is a challenging task as it involves large distributed file systems which should be fault tolerant, scalable and flexible. The Apache Hadoop provides open source software for reliable, scalable and distributed computing. Map Reduce technique is used for efficient processing of Big Data. This paper gives a brief overview of Big Data, Hadoop Map Reduce and Hadoop Distributed File System along with its architecture.

Keywords:- Big Data; Hadoop; Map reduce; Hdfs (Hadoop Distributed File System)

I. INTRODUCTION

Big data analytics is the area where advanced analytic techniques operate on big data sets. It is really about two things, Big data and Analytics and how the two have teamed up to create one of the most profound trends in business intelligence (BI) [12]. The issue with Big Data is that they use NoSQL and has no Data Description Language (DDL) and it supports transaction processing. Also, web-scale data is heterogeneous and it is not universal. For analysis of Big Data, database cleaning and integration is much challenging than the traditional mining approaches. Parallel processing and distributed computing is becoming a standard procedure which are nearly non-existent in RDBMS. The foremost challenge for researchers and academicians systems is that the large datasets needs special processing systems. Hadoop is one of the technology used for this purpose. Hadoop, which is an open-source implementation of Google MapReduce, including a distributed file system, provides to the application programmer the abstraction of the map and the reduce. With Hadoop, it is easier for organisations to get a grip on the large volumes of data being generated each day.

II. BIG DATA

The term "big data" is used for massive data sets whose size is beyond the ability of traditionally used software tools to store, manage, and process the data within a certain bounded time. Big data sizes are continuously rising ranging from terabytes to many petabytes of data in a single data set. Challenges include capture, storage, search, sharing, analytics and visualizing. Examples of big data mostly includes sensor networks, web logs, satellite and geo-spatial data, data from social networking sites, internet text and documents, internet search indexing, call detail records, astronomy, video archives, and large-scale ecommerce. Big data impacts include walmart handles more than 1 million customer transactions per hour, which is imported into databases having more than 2.5 petabytes of data – the equivalent of 165 times the information contained in all the books in the US library of congress.

Big Data can be defined as [9]: "The tools or techniques for describing the new generation of technologies and architectures that are designed to economically extract value from very large volumes of a wide variety of data, by enabling high-velocity capture, discovery or analysis". Doug Laney describes the definition of "Big Data" in terms of its attributes by 3 V's: Volume, Variety and Velocity in 2011. Later in 2012, IBM describes two more V's as Value and Veracity, thus making 5 V's of Big Data. Then in 2013, one more V was proposed as Variability to make 6 V's of Big Data .These 6 V's are now listed as: Volume, Variety, Velocity, Value, Veracity and Variability.

Hadoop supports the running of application on Big Data, and addresses three main challenges (3V) created by Big Data-

- Volume: Large volume of data is main challenge of storage. Hadoop provides framework to process, store and analyse large data sets to address volume of data. Data volumes are expected to grow 60 times by 2020.
- Velocity: Hadoop handles furious rate of incoming data from very large system.
- Variety: Hadoop handles different types of structured and unstructured data such as text, audio, videos, log files and many more.

A. What is Big Data Problem?

Big Data has popularised because there is high use of data intensive technologies. The main difficulty of big data is the working with its traditional relational databases and desktop statistics packages, requiring instead "massively parallel software running on tens, hundreds, or even thousands of servers". Different challenges faced in large data management include – scalability, accessibility, unstructured data, real time analytics, fault tolerance and many more. Moreover the variations in the amount of data stored in different sectors, the types of data generated and stored—i.e., whether the data is structured, semi-structured or quasi-structured—also differ from industry to industry[4].

B. Big Data Techniques and Technologies

Big data needs effective technologies to efficiently process massive amount of data within tolerable bounded times. A wide variety of techniques and technologies has been developed and adapted to aggregate, manipulate, analyze, and visualize big data. There are the different technologies (like Hadoop, Map Reduce, Apache Hive, No SQL and HPCC) which use almost same approach i.e. to distribute the data among various local servers and reduce the load of the master server to avoid the traffic. The technique discussed in this paper is Hadoop.

III. HADOOP

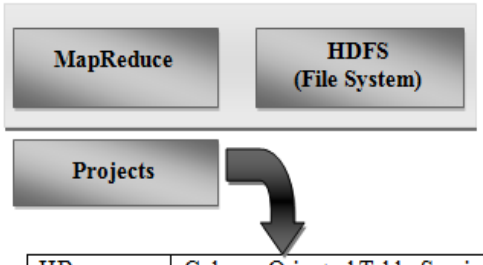
Hadoop is an open-source software framework for storing and processing big data in a distributed fashion on large clusters of commodity hardware. A common set of services is provided by a whole large set of softwares that work together. The creator of Hadoop and apache license is Dough cutting. Hadoop is inspired by Google File System and Google Mapreduce level and is a top level project [3]. Importantly, it accomplishes two main jobs: large data storage and faster processing. Open-source software: Open source software differs from commercial software due to the broad and open network of developers that create and manage the programs. Traditionally, it's free to download, use and contribute to, though more and more commercial versions of Hadoop are becoming available. Framework: It means everything you need to develop and run your software applications is provided – programs, tool sets, connections, etc.

- *Massive storage*
  The Hadoop framework can store large amount of data by splitting the data into blocks and storing it on clusters of lower-cost commodity hardware.
- *Distributed*
  Data is divided and stored across multiple nodes, and computations can be run in parallel across multiple connected machines.
- *Faster processing*
  Hadoop provides faster processing of huge data sets in parallel fashion across clusters of tightly connected low-cost computers for quick results [1].

Hadoop's existence originates from Google File System (GFS) and MapReduce which become apache HDFS and Apache Mapreduce respectively [13].

The Hadoop "brand" contains many different tools. ***Following two are core parts of Hadoop:***

- Hadoop Distributed File System (HDFS) is a virtual distributed file system that works like any other file system except that when the file is moved on HDFS, this file got split into many small files, each of those files is replicated and stored on 3 servers for fault tolerance constraints.
- Hadoop MapReduce is a technique to split every job into smaller jobs which are sent to many small servers, allowing a truly scalable use of CPU power [1].



HBase	Column Oriented Table Service
Hive	Data warehouse Infrastructure
Pig	Dataflow language
ZooKeeper	Distributed Coordination Service
Chukwa	Data Aggregation System
Avro	Data Serialization System

Figure 1. Components of hadoop

A. *Need of Hadoop*

The challenges and complexity of modern data rising needs is outdating the computing power of traditional systems. With its ability to integrate data from different sources, Hadoop can handle massive data sets which can be structured or unstructured in a distributed way.

- The biggest reason to use Hadoop is that before Hadoop, data storage was expensive.
- Hadoop moreover, lets you store as much data as you want in whatever form you need, simply by adding more servers to a Hadoop cluster. Each new server (which can be commodity x86 machines) adds more storage and more processing power to the whole cluster. This makes data storage with Hadoop far less costly than prior methods of data storage [6].

Hadoop is best suited for:

- Processing unstructured data
- Complex parallel information processing
- Large Data Sets/Files
- Critical fault tolerant data processing
- Queries that cannot be expressed by SQL
- Data processing Jobs needs to be faster [8]

B. *Comparison of three major Hadoop Distributions*

Table below shows comparison on three major Hadoop distributions (1) Amazon Hadoop Distribution, (2) MapR Hadoop Distribution and (3) Cloudera Hadoop Distribution based on four broad parameters [8]:

TABLE I. COMPARISON OF THREE MAJOR HADOOP DISTRIBUTIONS

Parameters	**Amazon Hadoop Distribution (ver. 1.0.3)**	**MapR Hadoop Distribution (ver. 0.20.205)**	**Cloudera Hadoop Distribution (ver. 2.0.0)**
Technical Features	• Tightly integrated with otherAWS services. • Job scheduling support for Java and Hive. • Support for HBase and S3 storage.	• Customized to support NFS file system. • Tightly integrated with other AWS services. • Job scheduling support for Java,streaming and Hive.	• Supports Oozie Job scheduler and Zookeeper for management. • It is not tightly integerated with AWS services.

Deployment	• Deployment managed through AWs toolkit/console. • Basic configuration management and performance tuning is supported through AWS,EMR and management console.	• Deployment manged through AWS toolkit. • Basic configuration management and performance tuning is supported through AWS,EMR and management console.	• Deployment with whirr toolkit. • Needs separate deployment of Hadoop components like HDFS,Hive and HBase.
Maintenance	• Easy to maintain as cluster is managed through AWS management console and AWS toolkit.	• Easy to maintain as cluster is managed through AWS management console and AWS toolkit.	• Cloudera Hadoop is managed through Cloudera manager. • The maintenance and upgrade requires efforts.
Cost	• Open source distribution • AWS EC2 and other AWS service costs apply.	• MapR is a proprietary distribution. • Billing is done through AWS on hourly basis.	• Can be implemented on any cloud • Costs are applicable based on components and tools adopted.

C. HDFS (Hadoop Distributed File System)

HDFS is the file system component of Hadoop[5]. HDFS uses clustered storage architecture which is fault tolerant. Hadoop provides a distributed file system(HDFS) that can store data across thousands of servers, and a means of running work across those machines, running the work near the data. Large data is splitted into parts which are managed by different data nodes in the hadoop cluster. HDFS stores all the file system metadata on single Name Node. HDFS uses replication of data stored on Data Node to provide reliability instead of using data protection mechanism such as RAID.

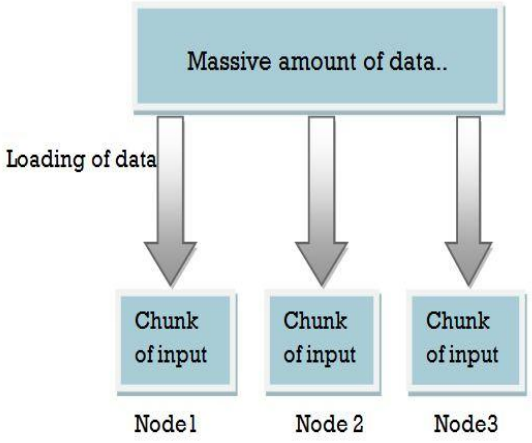


Figure 2. Distribution of data at load time

The slave Data Nodes store multiple copies of the application data.

D. MapReduce Framework

MapReduce follows Parallel and Distributed processing. Its functionality is flexible and uses high level programming language like Phython, Java. A MapReduce program is composed of a Map( ) procedure that performs sorting and filtering and a Reduce( ) procedure that performs a summary

operation. MapReduce follows programming model for processing large datasets. The Map( ) function takes an input key/value pair and produces a list of intermediate key/value pairs[1].

Map (input_key,input_value)>list(output_key,intermediate_value)

Reduce (output_key,list(intermediate_value))>list(output_value)

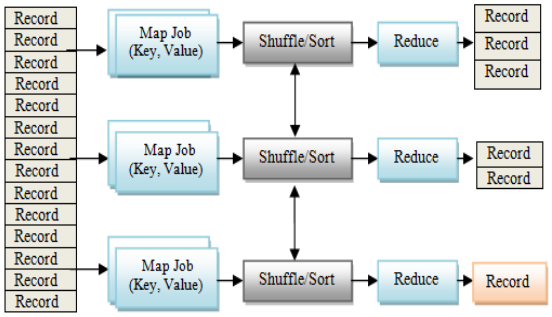


Figure 3. MapReduce working

1) Map Reduce Components

a) Name Node: manages HDFS metadata, no direct dealing with files

b) Data Node: stores HDFS data into blocks– default replication level for each block is 3

c) Job Tracker: schedules, monitors and allocates task execution on slaves (Task Trackers)

d) Task Tracker: runs Map Reduce functions. Hadoop MapReduce uses mapper and reducer functions.

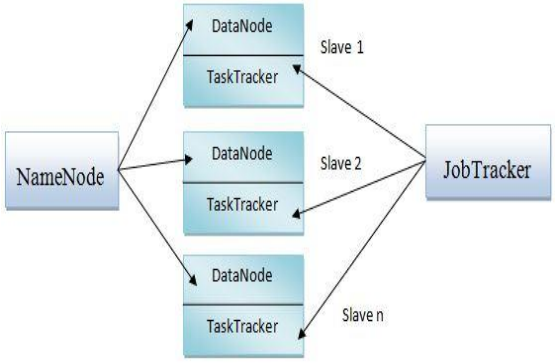


Figure 4. Map reduce working by master slave

2) Map Reduce Techniques

a) Prepare the Map( ) input: the "MapReduce system" designates Map processors, assigns the K1 input key value each processor would work on, and provides that processor with all the input data associated with that key value.

b) Run the Map( ) code: For each K1 key value, Map() is run exactly once generating output organized by key values K2.

c) *"Shuffle" Map output to the Reduce processors*: the MapReduce system designates Reduce processors, assigns the K2 key value each processor should work on, and provides that processor with all the Map-generated data associated with that key value.

d) Run the Reduce( ) code: For each K2 key value produced by the Map step, Reduce() is run exactly once.

e) Produce the final result: the MapReduce system combines all the Reduce output, and sorts it by K2 to produce the final result [1].

3) Programming Model [10]

The model takes a set of input key/value pairs, and produces a set of output key/value pairs. The computation of Mapreduce library is expressed as two functions: map and reduce. Map, written by the user, takes an input pair and produces a set of intermediate key/value pairs. The MapReduce library groups together all intermediate values associated with the same intermediate key I and passes them to the reduce function. The reduce function, also written by the user, accepts an intermediate key I and a set of values for that key. A possibly smaller set of values is produced after merging these values together. The intermediate values are forwarded to the user's reduce function via an iterator. This makes us allow to handle lists of values that are too large to fit in memory.

Example Consider the problem of counting the number of occurrences of each word in a large collection of documents. The high-level structure would look like this [14]:

```
mapper (filename, file-contents):
for each word in file-contents:
emit (word, 1)

reducer (word, values):
sum = 0
for each value in values:
sum = sum + value
```

emit (word, sum)

The map function emits each word plus an associated count of occurrences (just 1 in this simple example). The reduce function sums together all counts emitted for a particular word.

4) Comparison of MapReduce with RDBMS [11]

In many ways, MapReduce can be seen as a complement to a Rational Database Management System (RDBMS).

Table 2: RDBMS compared to MapReduce

Parameters	Traditional RDBMS	MapReduce
Data Size	Gigabytes	Petabytes
Updates	Read and write many times	Write once,read many times
Structure	Static schema	Dynamic schema
Integrity	High	Low

IV. HADOOP SYSTEM ARCHITECTURE

The system architecture consists of hadoop architecture, hadoop multi-node cluster architecture, architecture of HDFS and implementation of Map Reduce programming model [1].

A. Hadoop Cluster High Level Architecture

Hadoop cluster consists of a single master and multiple slaves or "worker nodes". The JobTracker schedules the job within Hadoop and allocates MapReduce tasks to specific nodes in the cluster, specially the nodes that have the data.

A TaskTracker node in the cluster is given three tasks-

- Map
- Reduce
- Shuffle

by a JobTracker. The master node is having a NameNode, DataNode, JobTracker and TaskTracker. A slave or worker node acts both as a TaskTracker and DataNode. In bigger cluster, the HDFS is managed by a dedicated NameNode server to host the file system index, and a secondary NameNode is used which can generate image of the name node's memory structures [1].

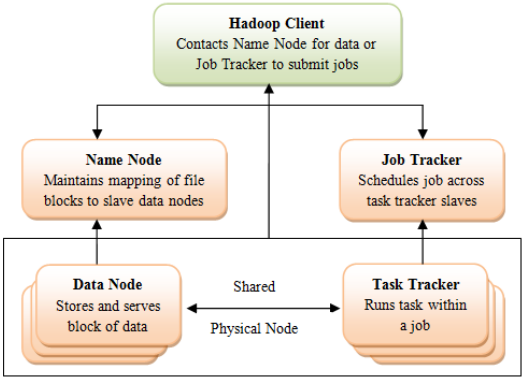


Figure 5. Hadoop high level architecture

B. HDFS Architecture

In a large cluster, directly attached storage is hosted by thousands of servers. By distributing storage and computation across many servers, the resource can grow with demand while remaining economical at every size.

1) NameNode

The file system namespace is managed by the master Name Node by keeping index of data location and regulates access to files by clients. Files and directories are represented on Name Node and it executes operations like opening, closing and renaming files and directories. NameNode itself doesn't store any data and is not responsible for any data flow through it. It only determines and keeps track of mapping of file blocks to Data Node, thus acting as a repository for all HDFS metadata. The NameNode provides the locations of data blocks containing the file. The client application then pipelines the data to the DataNode nominated by NameNode.

2) DataNode

The blocks of file are stored by the Data Nodes as determined by the Name Node. Internally, data file to be stored is first split into one or more blocks. Data Nodes are responsible for creating, deleting and replicating blocks of file after being instructed by the Name Node. DataNodes send *heartbeats* to the NameNode to confirm that the DataNode is operating and the block replicas it hosts are available.

The NameNode does not directly call DataNodes. It uses responses to heartbeats to send instructions to the DataNodes. The instructions include functions to:

- remove local block replicas;

- replicate blocks to other nodes;
- re-register or to shut down the node;
- send an immediate block report.

3) *HDFS Client*

When a file is read by an application, the HDFS client first asks the NameNode for the list of DataNodes that host replicas of the blocks of the file. It then messages a DataNode directly and requests for the transfer of the needed block. When a client writes, it first asks the NameNode to choose DataNodes to host replicas of the first block of the file. The client manages a pipeline from node-to-node and sends the data. When the first block is filled, the client requests new DataNodes to be chosen to host replicas of the next block [5].

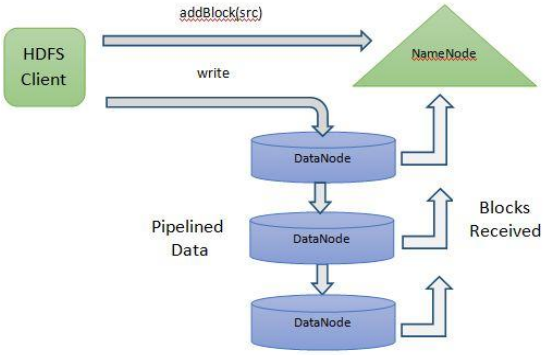


Figure 6. *An HDFS client*

V. FAULT TOLERANCE

A. *Replica Placement*

The placement of replica is crucial to HDFS performance and reliability. HDFS acts as a self-healing system. As depicted in the figure suppose the second data node fails, we still have two other DataNodes which have required data's replicas. If a DataNode goes down, then the *heartbeat* from DataNode to NameNode will stop and after ten minutes NameNode will consider that DataNode to be dead and all the blocks that were stored on that DataNode will be rereplicated and distributed evenly on other living DataNodes [2].

B. *Rack Awareness Policy*

An HDFS file consists of blocks. Whenever the data is to be stored on a new block, the NameNode allocates a block with a unique block ID and determines a list of DataNodes to host replicas of the block. Data is then pipelined from the client to the DataNodes. As shown in the figure, nodes are distributed across multiple racks.They share a switch connected by one or more core switches. The NameNode, acting as a hub maintains the metadata that helps in resolving the rack location of each DataNode. The main aim of rack aware replica policy is to improve availability and reliability of data along with network bandwidth utilization. The default HDFS rack aware replica policy is as follows:

• DataNode should not contain more than one replica of any block of file.

• Rack should not contain more than two replicas of the same block, provided there are sufficient numbers of racks on the cluster [2].

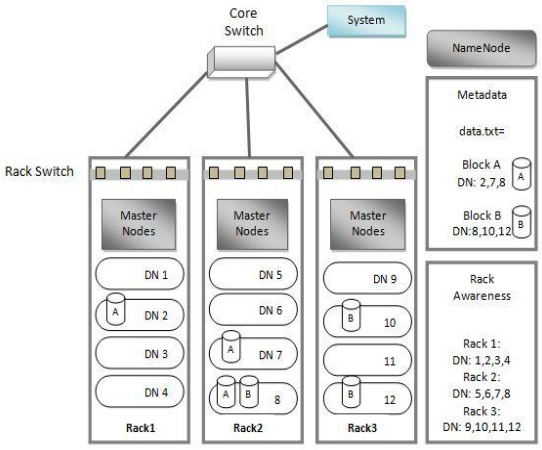


Figure 7. Rack awareness

VI. HADOOP INSTALLATION ON SINGLE NODE

Hadoop requires a working Java 1.5+ (aka Java 5) installation. However, using Java 1.6 is recommended for running Hadoop. Refer the following tutorial for installation: http://www.michael-noll.com/tutorials/running-hadoop-on-ubuntu-linux-single-node-cluster/. This tutorial has been tested with the following software versions:

- Ubuntu Linux 10.04LTS
- Hadoop 1.0.3. released May 2012

VII. CONCLUSION

In this work, Hadoop data cluster, HDFS and Map Reduce programming framework has been explored to provide solution to big data problem. This paper discussed an architecture using Hadoop HDFS distributed data storage and MapReduce distributed data processing over a cluster of commodity servers. Being distributed, reliable (both in terms of computation and data), scalable, fault tolerant and powerful, Hadoop is now widely used by Yahoo!, eBay, Amazon, IBM, Facebook and Twitter for massive storage of any kind of data with enormous processing power. MapReduce framework of Hadoop makes the job of programmers easy as they need not to worry about the location of data file, management of failures, and how to break computations into pieces as all the programs written are scaled automatically by Hadoop. MapReduce can be explored to handle different problems related to text processing at scales that would have been unthinkable a few years ago. The main goal of our paper was to make a survey of hadoop components and its different distributions.

REFERENCES

[1] R.Saranya, V.P.Muthukumar, N.Mary, "BIG DATA IN CLOUD COMPUTING", International Journal of Current Research in Computer Science and Technology (IJCRCST) Volume 1, Issue 1(Dec'2014)

[2] Puneet Singh Duggal, Sanchita Paul, "Big Data Analysis: Challenges and Solutions", International Conference on Cloud, Big Data and Trust 2013, Nov 13-15, RGPV

[3] Kalpana Dwivedi, Sanjay Kumar Dubey, "Analytical Review on Hadoop Distributed File System", 2014 IEEE.

[4] Aditya B. Patel, Manashvi Birla, Ushma Nair, "Addressing Big Data Problem Using Hadoop and Map Reduce" ,2012 Nirma University International Conference on Engineering, Nuicone-2012, 06-08December, 2012.

[5] Konstantin Shvachko, Hairong Kuang, Sanjay Radia, Robert Chansler, "The Hadoop Distributed File System", 2010 IEEE

[6] http://readwrite.com/2013/05/29/the-real-reason-hadoop-is-such-a-big-deal-in-big-data

[7] http://www.experfy.com/blog/cloudera-vs-hortonworks-comparing-hadoop-distributions/

[8] http://blog.blazeclan.com/252/

[9] Punam Bedi, Vinita Jindal, Anjali Gautam, "Beginning with Big Data Simplified", 2014 IEEE

[10] J. Dean and S. Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters," Commun. ACM , vol. 51, no. 1, 2008, pp. 107–13

[11] T. White, Hadoop: the Definitive Guide, O'Reilly, 3rd ed., 2012

[12] Shankar Ganesh Manikandan, Siddart h Ravi, "Big Data Analysis using Apache Hadoop", 2014 IEEE

[13] Kamalpreet Singh, Ravinder Kaur, "Hadoop: Addressing Challenges of Big Data", 2014 IEEE

[14] E.Sivaraman, Dr.R.Manickachezian, "High Performance and Fault Tolerant Distributed File System for Big Data Storage and Processing using Hadoop", 2014 International Conference on Intelligent Computing Applications, 2014 IEEE DOI 10.1109/ICICA.2014.16