International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 02 Issue: 05 | Aug-2015 www.irjet.net p-ISSN: 2395-0072

# A Survey on Big Data

**D.Prudhvi[1], D.Jaswitha[2], B. Mounika[3], Monika Bagal[4]**
[1 2 3 4] *B.Tech Final Year, CSE, Dadi Institute of Engineering & Technology ,Andhra Pradesh,INDIA*

---

**Abstract** – ***Hadoop [1], the open-source implementation of Google's MapReduce [6], has become enormously popular for big data analytics, especially among researchers. Due to Hadoop's popularity, it is natural to ask the question: How well is it working? To answer this question, we need to move beyond the conventional "cluster-centric" analysis that models each job as an independent black box and focuses on coarse global metrics such as resource utilization, usually in the context of a single cluster [2, 3, 4, 8]. Instead, we need to recognize that not all clusters are used in the same way, and that there is a rich ecosystem of tools, extensions, and libraries related to Hadoop that influence application characteristics, user behavior, and workflow structure.***

***Key Words:*** *Hadoop, MapReduce, HDFS, HBase, Sqoop, Oozie, Big data.*

## 1. INTRODUCTION

Hadoop is designed to run on a large number of machines that don't share any memory or disks. That means you can buy a whole bunch of commodity servers, slap them in a rack, and run the Hadoop[2] software on each one. When you want to load all of your organization's data into Hadoop, what the software does is bust that data into pieces that it then spreads across your different servers. There's no one place where you go to talk to all of your data; Hadoop keeps track of where the data resides. And becausethere are multiple copy stores, data stored on a server that goes offline or dies can be automatically replicated from a known good copy. In a centralized database system, you've got one big disk connected to four or eight or 16 big processors. But that is as much horsepower as you can bring to bear. In a Hadoop cluster, every one of those servers has two or four or eight CPUs. You can run your indexing job by sending your code to each of the dozens of servers in our cluster, and each server operates on its own little piece of the data. Results are then delivered back to you
in a unified whole. That's MapReduce you map the operation out to all of those servers and then you reduce the results back into a single result set. Architecturally, the reason you're able to deal with lots of data is because Hadoop spreads it out. And the reason you're able to ask complicated computational questions is because you've got all of these processors, working in parallel, harnessed together. Hadoop implements a computational paradigm named Map/Reduce, where the application is divided into many small fragments of work, each of which may be executed or re-executed on any node in the cluster.

**Big Data**, now a days this term becomes common in **IT** industry. As there is lot of data lies in the industry but there is nothing before big data comes into picture.

Why we need the Big Data ? As we know there is lot of data surrounds us but we can't make that data useful to us. Reason ? Reason is simple, that there is no power tool that can make out analysis or information from this huge amount of data. There is one example, that one team of scientist have some data with them and they want to do some analysis on them, So one well know vendor in market approach them. That vendor will take 15 years to make analysis on that huge data. Now see, after long 15 years, is there any relevance of that data or that data is of any use to that user. In the era where we can't wait for 5 sec to open Google page. How we think of a long time to make the analysis.

When we talk about Big Data[2], the first name comes in mind is "HADOOP" a well know product in the market of big data. Hadoop is Linux based product used by big player of market like Google, Yahoo etc.

**Name Node** is a type of master node, which is having the information or we can say that meta data about the all data node there is address(use to talk ), free space, data they store, active data node , passive data node, task tracker, job tracker and many other configuration such as replication of data..

**Data Node** is a type of slave node in the hadoop, which is used to save the data and there is task tracker in data node which is use to track on the ongoing job on the data node and the jobs which coming from name node.

**Hadoop Architecture**, is based on HDFS, which is hadoop distributed file system. In which data is equally (ideally) distributed on each node in the hadoop system. When we (client) want to fetch or add or modify or delete some data from hadoop, then hadoop system collect the data from each node of our interest and do the meaningful actions of our interest.

**Scheduler in Hadoop:** A scheduler plays a very important role in the big data processing . A fair scheduler, schedules the jobs in such a way that all the resources share by the command or by the system in equal amount without any over loading on one of the part of the system. Like scheduler, will take care of the resources on each data node. It helps to maintain the load on all the data node in the system.

How does scheduling help in the processing of big data?

Take one example, suppose we have ten data node in hadoop cluster, and our scheduler is not fair it cannot manage resources in a right manner. So what does he do that, he give work on 5 data node out of 10 data node in the cluster, and suppose it take around x amount of time to complete that command[4]. Now think that our scheduler is fair enough to distribute work on all the data node in our cluster, So according to our assumption it will take around x/2 amount of time to complete the whole process.

### 1.1. SOURCES OF BIG DATA:

• The New York Stock Exchange generates about one terabyte of new trade data per day[7].

• Face book hosts approximately 10 billion photos, taking up one petabyte of storage.

• Ancestry.com, the genealogy site, stores around 2.5 petabytes of data.

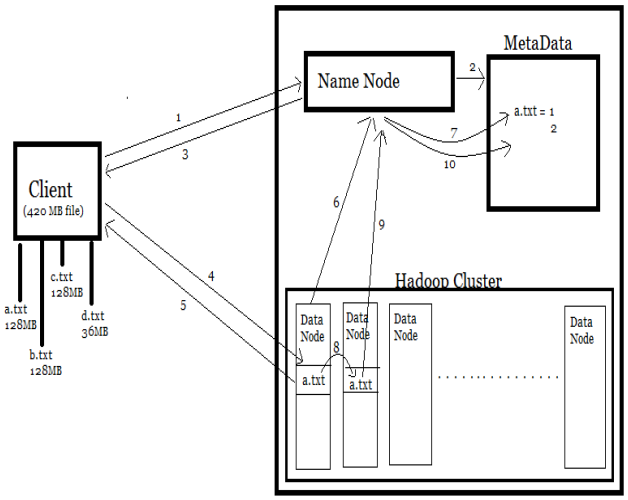
• The Internet Archive stores around 2 petabytes of data, and is growing at a rate of 20 terabytes per month.

• The Large Hadron Collider near Geneva, Switzerland, will produce about 15 petabytes of data per year.

### 1.2.COMPARISON TO RDBMS:

Relational database management systems (RDBMS) are often compared to Hadoop but in fact are more like the opposite. Both systems got different strengths. While RDBMS can hardly store more then 1 terabyte of data, using Hadoop just starts to make sense with several terabytes. Hadoop shines when reading the whole data is necessary while RDBMS are using indexes and caches for random access to access small portions of the data. Another big difference are constraints which are non-existent in Hadoop but impossible to think away from database systems.

**Cluster:** A cluster consists of several nodes organized into racks, each node running a TaskTracker for the MapReduce tasks and a DataNode for the distributed storage system. One special node, the Master-Node runs the JobTracker and the NameNode which are organizing the distribution of tasks and data.




### 2. THE CORE OF HADOOP: HDFS & MAPREDUCE

HDFS and MapReduce are robust. Servers in a Hadoop cluster can fail and not abort the computation process. Programming Hadoop at the MapReduce level is a case of working with the Java APIs, and manually loading data files into HDFS.

To store data, Hadoop utilizes its own distributed file system, HDFS, which makes data available to multiple computing nodes. A typical Hadoop usage pattern involves three stages:
• loading data into HDFS,
• MapReduce operations, and
• retrieving results from HDFS.

**HDFS:**
File systems that manage the storage across a network of machines are called distributed file systems. HDFS is Hadoop's flagship file system designed for storing very large files with streaming data access pattern, running on clusters of commodity hardware. On the other side, it provides high throughput (as it offers parallel processing) access to application data and is suitable for applications that have large data sets[4]. It is a fault-tolerant distributed file system designed to run on low-cost hardware.


### 3. HDFC Architecture :

1) HDFS stands for Hadoop Distributed File System

2) HDFS is designed to run on low-cost hardware

3) HDFS[5] is highly fault-tolerance(as it supports block replication)

4) HDFS was originally built as infrastructure for the **Apache Nutch** web search engine project

5) HDFS is now an Apache Hadoop sub project

6) An HDFS instance may consist of hundreds or thousands of server machines, each storing part of the file system's data.

7) HDFS is designed more for batch processing rather than interactive use by users. The emphasis is on **high throughput** of data access rather than **low latency** of data access

8) A typical file in HDFS is gigabytes to terabytes in size[8].

9) HDFS applications need a write-once-read-many access model for files. This assumption simplifies **data coherency issues** and enables high throughput data access.

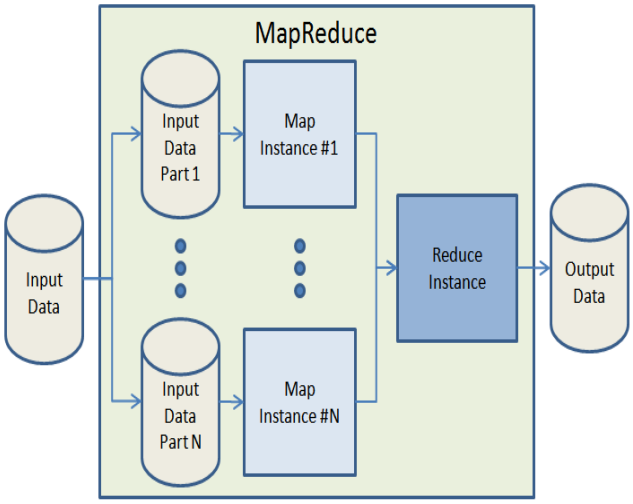
**MAP REDUCE :**

In Map Reduce the programmer writes two functions: a map function and a reduce function, each of which defines a mapping from one set of key-value pairs to another[5]. These functions are unaffected to the size of the data or the cluster that they are operating on, so they can be used unchanged for a small dataset and for a massive one. One more important thing to remember is, if you double the size of the input data, a job will run twice as slow. But if you also double the size of the cluster, a job will run as fast as the original one. This is not generally true of SQL queries. It is a programming model for processing large data sets with a parallel, distributed algorithm on a cluster. The data used in Map Reduce is semi structured and record oriented[6].

Map Reduce works by breaking the processing into two phases: **Map phase and Reduce phase.** Each phase has a key value pair as input and output, types of which are chosen by programmer[2]. The programmer also specifies two functions:

To take advantage of the parallel processing that Hadoop provides we need to express our query as a MapReduce job. After some local, small scale testing we can run it on a cluster of machines. Broadly MapReduce working can be broken down into three components:[7]

- Map Method
- Reduce Method
- Code to run(Driver Code).




**Apache Hadoop and Hadoop ecosystem :**

Although Hadoop is best known for MapReduce and its distributed filesystem (HDFS), the term is also used for a family of related projects that fall under the umbrella of infrastructure for distributed computing and large-scale data processing.

**Improving Programmability: Pig and Hive**

Working directly with Java APIs can be tedious and error prone. It also restricts usage of Hadoop to Java programmers. Hadoop offers two solutions for making Hadoop programming easier[7].

**Pig :**

A data flow language and execution environment for exploring very large datasets. Pig runs on HDFS and MapReduce clusters.

**Hive :**A distributed data warehouse . Hive manages data stored in HDFS and provides a query language based on SQL (and which is translated by the runtime engine to MapReduce jobs) for querying the data.

**Improving Data Access: HBase, Sqoop, and Flume**

At its heart, Hadoop is a batch-oriented system. Data are loaded into HDFS, processed, and then retrieved. This is somewhat of a computing throwback, and often, interactive and random access to data is required.

**HBase:**

A distributed , column-oriented database. HBase uses HDFS for its underlying storage, and supports both batch-style computations using MapReduce and point queries (random reads).

**Sqoop:** A tool for efficiently moving data between relational databases and HDFS[5].

**Flume:** Highly reliable, configurable streaming data collection

**Coordination and Workflow: Zookeeper and Oozie**

**ZooKeeper:** With a growing family of services running as part of a Hadoop cluster, there's a need for coordination and naming services.A distributed, highly available coordination service. ZooKeeper provides primitives such as distributed locks that can be used for building distributed applications.

**Oozie:** The Oozie component provides features to manage the workflow and dependen- cies, removing the need for developers to code custom solutions.

## 4. CONCLUSION

Nowadays, Companies need to process Multi Petabyte Datasets efficiently. The Data may not have strict schema for the large system. It has become Expensive to build reliability in each Application for processing petabytes of datasets. If there is problems of Nodes fail every day, some of the causes of failure may be. Failure is expected, rather than exceptional. The number of nodes in a cluster is not constant. So there is a Need for common infrastructure to have Efficient, reliable, Open Source Apache License. In this way we discussed about the basics of Big Data and Hadoop distributed file system.

## REFERENCES

[1] Hadoop: The Definitive Guide, Third Edition by Tom White.
[2] Big Data Now: 2012 Edition by O'Reilly Media, Inc.
[3] InternetEdureka: http://www.edureka.co/blog/introduction-to-apache-hadoop-hdfs/ (Retrieved 16-08-2015)
[4] HadoopTutorial: http://developer.yahoo.com/hadoop/tutorial/module1.html
[5] Tom white, Hadoop Definitive Guide,Third Edition,2012
[6] Tyson Condie, Neil Conway, Peter Alvaro, Joseph M. Hellerstein, Khaled Elmeleegy, and Russell Sears. MapReduce online. In NSDI, 2010.
[7] Y. Chen, S. Alspaugh, and R. H. Katz. "Interactive Analytical Processing in Big Data Systems: A Cross-Industry Study of MapReduce Workloads". PVLDB 5.12 (2012), pp. 1802–1813.
[8] Ashish Thusoo et. al. Hive: a petabyte scale data warehouse using Hadoop. In ICDE, 2010.

## BIOGRAPHIES

D. Prudhvi is a final year student of Computer Science & Engineering branch in Dadi Institute of Engineering & Technology.

D. Jaswitha is a final year student of Computer Science & Engineering branch in Dadi Institute of Engineering & Technology.

B. Monika is a final year student of Computer Science & Engineering branch in Dadi Institute of Engineering & Technology.

B. Mounika is a final year student of Computer Science & Engineering branch in Dadi Institute of Engineering & Technology.