A Comparative Study of Hadoop And Spark for Big Data Analytics

Prof. Krupa Trambadiya
Assistant Professor Dept of IT
Vadodara Institute of Engineering, Kotambi

Abstract- *The success of Big Data systems in recent years will continue its speedy development over the next decade. The challenges of big data systems drive users to find diversity and choices for various tools to be used for analytics purpose. Hadoop uses Map Reduce as its computational unit which has two phases, Map and Reduce whereas Spark uses Resilient Distributed Datasets (RDD) and Directed Acyclic Graph (DAG) for processing of large datasets. This paper works on comparative study of Hadoop and Spark and then discussing the type of applications where Hadoop is appropriate and some in which spark does better.*

Keywords- Big Data, Hadoop, Spark, Analytics, Mapreduce, RDD.

I. INTRODUCTION

BIG data systems are absolute success in recent years and will continue its hasty development over the next decade. The services like Search engine, Social networks, Multimedia and Ecommerce and number of scientific research are covered by Big data systems. The complication, miscellany and speedy evolution of big data systems give rise to various new challenges about how we design generators to produce data with the 5V properties (i.e. volume, velocity, variety, value and veracity).

Definition of Big Data

Big Data means the data which is large in volume and the one which cannot be stored and processed by traditional database management systems like RDBMS.

Types of Data under umbrella of Big Data [6]

Structured Data

All data which can be stored in table format with rows and columns. They have certain constraints and relational key and can easily be mapped into pre-designed fields. These data is mostly processed in development and is easy to manage. But only five percent of Big data is structured data.

Semi-structured Data

Semi-structured data is information that doesn't reside in a relational database but that does have some organizational properties that make it easier to analyze. CSV, XML and JSON documents are semi structured documents, NoSQL databases are considered as semi structured data.

Unstructured data

Unstructured data contributes 85% of Big data. It includes text and multimedia content. Example of such data are e-mail messages, word processing documents, videos, images, audio files, presentations, web pages and many other kinds of business documents. Unstructured data is either machine generated or human generated. Some examples of machine generated unstructured data are Radar data, satellite images, social media data, Mobile data, weather forecasting data etc.

5 V's of Big data: Volume, Velocity, Variety, Value and Veracity [7] [1]

In 2001, research report, META group (Gartner)defined data growth challenges and opportunities as being three- dimensional that is increasing amount of data, increasing speed of data for in and out and types of data from different sources. Big data carries information characterized by high volume, velocity and variety. Data is being produced at enormous rates. In fact, 90% of the data in the world today was created in the last two years! In order to make sense out of this overwhelming amount of data it is often broken down using five V's: Volume, Velocity, Variety, Value and Veracity. Gartner's 3 V's model has been added with more V's with nowadays.

Volume: This represents large amount of data.

Velocity: Velocity refers to the speed at which data is being processed.

Variety: This refers to the various types of data produced from different sources, for example semi-structured data like email

contains date with structured form and email text, which is unstructured data.

Value: One more V added to this is for Value of data to distinguish business value of big data.

Veracity: Originality of data matters a lot. So, fifth V (Veracity) refers the trustworthiness and authenticity of data.

Hadoop

Hadoop is an open source software framework that enables distributed storage and processing of large data sets across clusters of commodity hardware and is a possible solution to big data storage and computation problems. Hadoop is an open source software which provides scalability and fault tolerance in efficient manner.

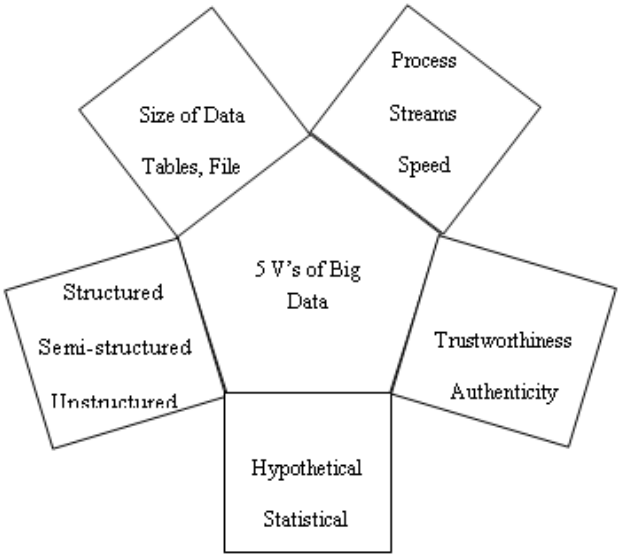

Figure 1.1: 5 V's of Big Data

II. SOFTWARE FRAMEWORKS TO PROCESS BIG DATA

Hadoop is designed to scale up from a single server to thousands of servers and so it can process large amount of data concurrently to afford results faster. The Hadoop framework is able to develop applications running on clusters of computers that support a master - slave architecture where slave nodes are controlled and managed by the master node.

Components of Hadoop

A. Data Processing

Mapreduce: A computational unit of hadoop used for processing large datasets concurrently. It consists of two parts, a map phase, which takes raw data and organizes it into key/value pairs, and a reduce phase which processes data.

YARN (Yet Another Resource Negotiator) architecture, which is responsible for providing computational resources needed for applications execution. The Resource Manager is the ultimate authority that sorts out resources among all the applications in the system. The Node Manager monitors resource usage and reports it to the resource manager.

Data Storage

HDFS (Hadoop Distributed File System): It is a distributed file system that stores data on cluster hardware that provides massive comprehensive bandwidth. It is a file system designed to store large amounts of data across multiple nodes of commodity hardware.

HBase: Apache HBase is the Hadoop database, a distributed, scalable, big data store. It can be used when random, real time read/write access is needed to Big Data. This project's goal is the hosting of very large tables like billions of rows with millions of columns with clusters of commodity hardware.

Data Management

It includes tools for user interaction with scheduling, monitoring, coordination and user interface. Oozie is a workflow scheduler and manages jobs for many of the tools in the processing layer. It also facilitates scheduling of jobs which need to run on regular intervals. Zookeeper manages coordination and synchronization of distributed systems. It provides tools to handle coordination of data and protocols and is able to handle partial network failures which are commonplace in distributed systems. Flume handles collection, aggregation and movement of log data into HDFS.

Data Access

Hive is data warehousing tool which is built on top of Hadoop. It is a query processing tool which queries data stored in hdfs and uses query language like HiveQL which is similar to SQL for RDBMS databases. Pig uses pig latin which is a scripting language to simplify programming process. Apache Mahout is an open source project used for creating scalable machine learning algorithms. It implements popular machine learning techniques such as recommendation, classification and clustering. Apache Avro provides data serialization and exchange services for apache hadoop. Avro facilitates the exchange of big data between programs written

in any language. With the serialization service, programs can efficiently serialize data into files or into messages. The data storage is compact and efficient. Avro stores both the data definition and the data together in one message or file. Apache Sqoop transfers data in bulk between Apache Hadoop and structured data stores such as relational databases. Sqoop can also be used to extract data from Hadoop and export it into external structured data stores. Sqoop works with relational databases such as Teradata, Netezza, Oracle, MySQL, Postgres, and HSQLDB.

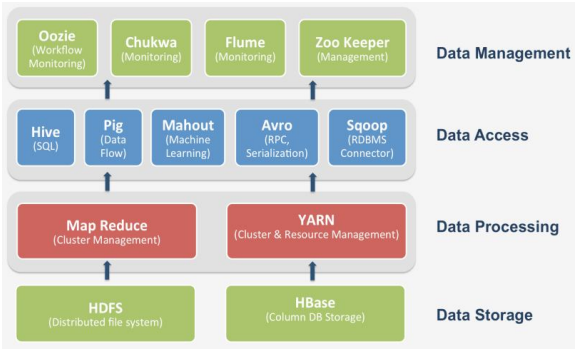

Figure 1.2: Hadoop Ecosystem

Apache Spark

Apache Spark is an open source big data processing framework which was originally developed in 2009 in UC Berkeley's AMPLab, and open sourced in 2010 as an Apache project. Apache Spark has become one of the key frameworks in big data distributed processing in the world. Spark can be deployed in a variety of ways, provides native bindings for the Java, Scala, Python and R programming languages and supports SQL, streaming data, machine learning, and graph processing. It is used by banking personnels, communication companies, gaming companies, governments, trading organizations and all of the major tech giants such as Facebook, Yahoo, Apple, IBM, Microsoft, Amazon etc.

Apache Spark is an open-source cluster computing framework for big data processing [2]. It has emerged as the next generation big data processing engine, surpasses Hadoop MapReduce which helped ignite the big data revolution. Spark maintains MapReduce's linear scalability and fault tolerance, but extends it in a few important ways: it is faster as well as much easier to programbecause of its rich APIs in Python, Java, Scala and R and its core data abstraction, the distributed data frame, and it goes far beyond batch applications to support a variety of compute-intensive tasks, including interactive queries, streaming, machine learning, and graph processing.

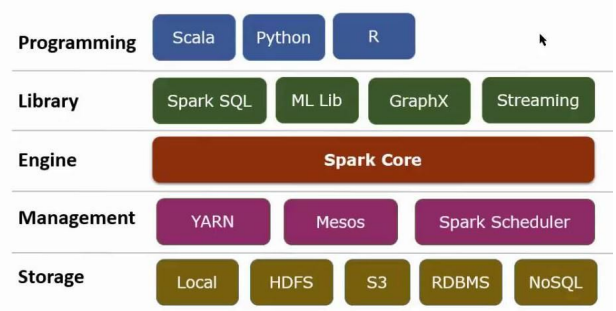

Figure 1.3: Spark Ecosystem [7]

High programming languages are supported by Apache Spark like R, Python, Scala/Java. Based on the nature of application of data, any of these languages can be used like R is good for research, plotting, and data analysis. Whereas Python is Good for small- or medium-scale projects to build models and analyze data, especially for fast startups or small teams. Scala/Java are good for robust programming with many developers and teams. it has fewer machine learning utilities than Python and R, but it makes up for it with increased code maintenance.

Spark SQL is an abstraction used on top of spark called SchemaRDD which is suitable for structured and semistructured data.

ML Lib is machine learning library for spark. It is a distributed machine learning framework on top of Spark. It is, according to benchmarks, done by the MLlib developers against the Alternating Least Squares (ALS) implementations. Spark MLlib is nine times faster than Hadoop disk-based version of Apache Mahout.

GraphX is a distributed graph-processing framework on top of Spark. It provides an API for graph working out that can model the user-defined graphs. It also provides an optimized runtime.

Streaming analytics are performed by **Spark Streaming** which leverages Spark Core's fast scheduling capability. It consumes data in mini-batches and performs RDD (Resilient Distributed Datasets) transformations on those mini-batches of data [8].

Apache Spark Core is heart of Spark ecosystem. It delivers speed by providing in-memory computation capability. Thus Spark Core is the foundation of parallel and distributed processing of large dataset [4].

Some key features of Apache Spark Core are:

- Essential I/O functionalities.

- Significant in programming and observing the role of the Spark cluster.
- Task dispatching.
- Fault recovery.
- Overcomes the obstacles of Map Reduce by using in-memory computation.

Spark Core is embedded with a special collection called RDD (resilient distributed dataset). RDD is among the abstractions of Spark. *Spark RDD* handles partitioning data across all the nodes in a cluster. It holds them in the memory pool of the cluster as a single unit. There are two operations performed on RDDs: *Transformation* and *Action*-

- Transformation: It is a function that produces new RDD from the existing RDDs.
- Action: In Transformation, RDDs are created from each other. But when we want to work with the actual dataset, then, at that point we use Action.

RDD (Resilient Distributed Dataset)

It is the fundamental data structure of Apache Spark which are an absolute collection of objects which computes on the different node of the cluster. Each and every dataset in Spark RDD is logically partitioned across many servers so that they can be computed on different nodes of the cluster.

RDD (Resilient Distributed Dataset) - decomposing the name:

- **Resilient** means here fault-tolerant, with the help of RDD lineage graph (like Directed acyclic graph (DAG)), it is able to re compute missing or damaged partitions because of node failures.
- **Distributed** with data residing on multiple nodes in a cluster.
- **Dataset** is a collection of partitioned data with primitive values or values of values like tuples or other objects that represent records of the data we work with. The datasets can be loaded externally from any of the files like JSON, CSV, text or database via JDBC without specific data structure.

There are three ways to create RDDs in Spark such as using *data in stable storage, from other RDDs, and parallelizing already existing collection in driver program*. Spark RDDs can also be operated in parallel with a low-level API that offers *transformations* and *actions*. Spark RDD can also be **cached** and **manually partitioned**. When we need to use RDD several times, Caching is advantageous. Manual partitioning is imperative to balance partitions appropriately. Usually, smaller partitions allow distributing RDD data more equally when executors are more. Hence, smaller amount partitions make the work trouble free.

There are certain reasons to use RDD in Spark. The key motivations behind the concept of RDD are Iterative algorithms, Interactive data mining tools and **DSM** (Distributed Shared Memory).

III. HADOOP VS SPARK

Both are most prominent Big Data Frameworks. Spark processes the similar applications that of hadoop motive but main feature of spark is its in memory processing that increases the speed of processing.

Parameters	Hadoop	Spark
Storage	Persistent Storage	Resilient Distributed Datasets (RDD) which resides in memory
Processing	Map Reduce can process data in batch mode	batch, interactive, iterative and streaming
Speed	Map Reduce reads and writes from disk, as a result, it slows down the processing speed	runs applications up to 100x faster in memory and 10x faster on disk than Hadoop
Complexity	Difficult to code compared to Spark	Easy to program with RDD
Requirements	Other components are required for different processing	Installing Spark on a cluster will be enough to handle all the requirements for data analysis
Types of applications	Fails to process real time data. Appropriate for batch processing on large volume of data	Able to process live streams efficiently. So appropriate for real time data analysis
Scheduler	Needs external job scheduler like Oozie	Equipped with its own scheduler for in memory processing
Cost	Cheaper compared to spark	Requires large number of RAM and hence costly
Machine Learning	External Machine learning tool is required like Mahout	Has its own library named MLlib
SQL Supports	Hive is a tool to which runs HiveQL	Equipped with Spark SQL

Table 3.1: Comparative Analysis of Hadoop and Spark

IV. CONCLUSION

With this study, it may be extracted that Hadoop is a very good startup or introductive tool for big data processing which

can be used with number of supported tool for different analytics activities for business data in economical way. On the other hand, Spark is fully equipped with number of libraries for same analytical processing which supports Resilient Distributed Dataset (RDD) with in memory processing and can capture real time data.

REFERENCES

[1] Muthiah, Karthika Ms., "Performance Evaluation of Hadoop based Big Data Applications with HiBench Benchmarking tool on IaaS Cloud Platforms" (2017). UNF Theses and Dissertations. 771. https://digitalcommons.unf.edu/etd/771
[2] V Srinivas Jonnalagadda, P Srikanth, Krishnamachari Thumati, Sri Hari Nallamala "A Review Study of Apache Spark in Big Data Processing", International Journal of Computer Science Trends and Technology (I JCST) – Volume 4 Issue 3, May - Jun 2016
[3] W.-K.Chen,LinearNetworksandSystems.Belmont, CA:Wadsworth, 1993, pp. 123–135
[4] https://data-flair.training/blogs/apache-spark-ecosystem-components.
[5] https://www.youtube.com/watch?v=ZTFGwQaXJm8
[6] https://jeremyronk.wordpress.com/2014/09/01/structured-semi-structured-and-unstructured-data/
[7] https://spark.apache.org/
[8] Priya Dahiya 1, Chaitra.B 2, Usha Kumari 3 "Survey on Big Data using Apache Hadoop and Spark", International Journal of Computer Engineering In Research Trends, Volume 4, Issue 6, June-2017, pp. 195-201