SURVEY PAPER ON BIG DATA ANALYTICS USING HADOOP TECHNOLOGIES

Vikas Goyal[1], Deepak Soni[2]
[1]Assistant Professor, [2]Student, CSE Department, MIMIT, Malout, India
Email: vikas.735@gmail.com[1], deepaksonibti@gmail.com[2]

Abstract

Big data is a term that describes the large volume of data. That contains data in the form of both structured and un-structured data. These data sets are very large and complex so that it becomes difficult to process using traditional data processing applications. Big data is difficult to work with using most relational database management systems and desktop statistics and visualization packages, requiring instead "massively parallel software running on tens, hundreds, or even thousands of servers". The technologies used in Hadoop by big data application to handle the massive data are Hdfs, Map Reduce, Pig, Apache Hive, Hbase and Spark. These technologies handle massive amount of data in KB, MB, GB, TB, PB, EB, ZB, YB and BB.

Keywords: Apache hive, Big Data, Hadoop, Hbase, Map Reduce, Pig, Spark.

I. INTRODUCTION

Big Data: The whole world has a tendency to produce 2.5 quintillion bytes of data — so much that 90th of the data within the world these days has been created within the last four years alone. This much amount of data comes from everywhere: sensors used to gather climate data, posts to social media sites, digital pictures and videos, purchase transaction records, and cell phone GPS signals to call some. This huge amount of the data is understood as "Big data". Big data is a nonsensicality, or catch-phrase, utilizes to describe a {huge an enormous a vast a colossal} volume of both structured and unstructured data that's so huge that it's sophisticated to process using traditional database and software system techniques. In most enterprise scenarios the data is too massive or it moves too quick or it exceeds current process capability. Big data has the potential to facilitate organizations to enhance operations and build faster, more intelligent choices. Big Data, now a day this term becomes common in IT industries. As there's a large amount {of data of knowledge of information} lies within the trade however there is nothing before massive data comes into image. The need of massive data generated from the big corporations like Facebook, Yahoo, Google, YouTube etc. for the purpose of research of enormous quantity of data that is in unstructured type or maybe in structured type. Google contains the large quantity of data. So; there is the requirement of massive Data Analytics to process the complicated and large data. So there are several type of information round the world like Structured, Semi-structured, Un-structured here I mention the different presence of data that are within the means like:

- Five V's (volume, velocity, veracity, variety, value):

1. VOLUME: Volume refers to the vast amounts of data generated each second. Just suppose of all the emails, twitter messages, photos, video clips, sensor data etc. we turn out and share each second. We are not talking Terabyte show ever Zettabytes or Brontobytes. On Facebook alone we send ten billion

messages per day, click the "like" button 4.5 billion times and transfer 350 million new photos each and every day. If we take all the data generated within the world between the start of your time and 2008, the same amount of data{of|information|of|knowledge} can presently be generated each minute. This more and more makes data sets large to store and analyze using traditional database technology. With big data technology we can currently store and use these data sets with the assistance of distributed systems, where elements of the data are keep in several locations and brought along by software system.

2. VELOCITY: Velocity refers to the speed at that new data is generated and the speed at that data moves around. Just suppose of social media messages going viral in seconds, the speed at which credit card transactions are checked for dishonorable activities, or the milliseconds it takes trading systems to analyze social media networks to choose up signals that trigger selections to shop for or sell shares. Big data technology permits for United States currently to analyze the data whereas it's being generated, without ever putting it into databases.

3. VERACITY: Veracity refers to the messiness or trustiness of the data. With many forms of huge data, quality and accuracy are less manageable (just suppose of Twitter posts with hash tags, abbreviations, typos and colloquial speech as well because the reliableness and accuracy of content) however huge data and analytics technology currently permits United States to figure with these style of data. The volumes often build up for the shortage of quality or accuracy.

4. VARIETY: Variety refers to the totally different sorts of data we are able to currently use. In the past we targeted on structured data that neatly fits into tables or relational databases, such as financial data (e.g. sales by product or region). In fact, 80% of the world's data is currently unstructured, and therefore can't simply be place into tables (think of photos, video sequences or social media updates). With big data technology we can currently harness differed varieties of data (structured and unstructured) together with messages, social media conversations, photos, sensor data, video or voice recordings and bring them in conjunction with more traditional, structured data.

5. VALUE: Value! It is all well and good having access to huge data however unless we are able to flip it into price it's useless. So you will safely argue that 'value' is that the most vital V of massive data. It is important so that companies build a business case for any conceive to collect and leverage huge data. It is very easy to be the thrill trap and commence huge data initiative while not a transparent understanding of prices and advantages.

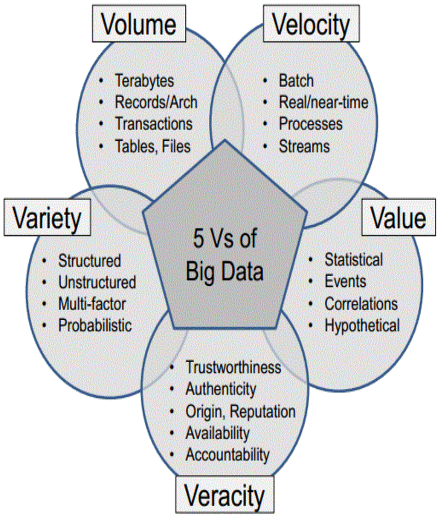


Fig. 1.1 Parameters of Big Data

II. HADOOP

Hadoop is a framework that enables for the distributed processing of huge data sets across clusters which has thousands of nodes computers using simple programming models. It is designed to rescale from single servers to thousands of machines, each providing native computation and storage. Rather than suppose hardware to deliver high-availability, itself is designed to detect and handle failures at the applying layer, so delivering a highly-available service on the top of a cluster (multiple nodes) of computers, each of that might be susceptible to failures. Hadoop is commonly used for distributed batch index building; it's fascinating to optimize the index capability in close to real time. Hadoop provides components for storage and analysis for massive scale process. The core of Apache Hadoop consists of a storage part, known as Hadoop Distributed file system (HDFS), and the processing part is called Map Reduce.

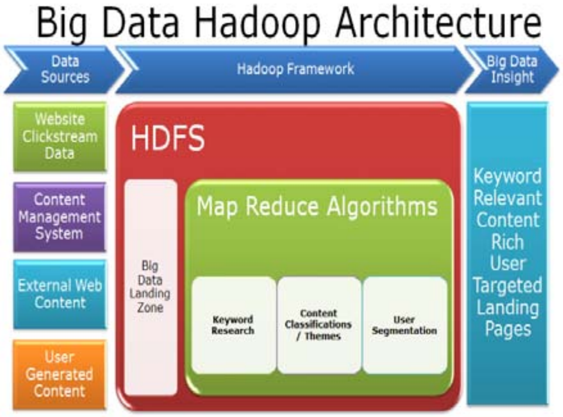


Fig. 2.1 Hadoop System

III. MAP REDUCE

Hadoop Map Reduce is a software system framework for simply writing applications that process large amounts of data (multi-terabyte data-sets) in-parallel on large clusters (thousands of nodes) of goods hardware in an exceedingly reliable, fault-tolerant manner. A Map Reduce job typically splits the input data-set into freelance chunks that are processed by the map tasks in a fully parallel manner. The framework sorts the outputs of the maps, which are then input to the reduce tasks. Typically, both the input and the output of the work are keep in an exceedingly file-system. The framework takes care of scheduling tasks, monitoring them and re-executes the failing tasks.

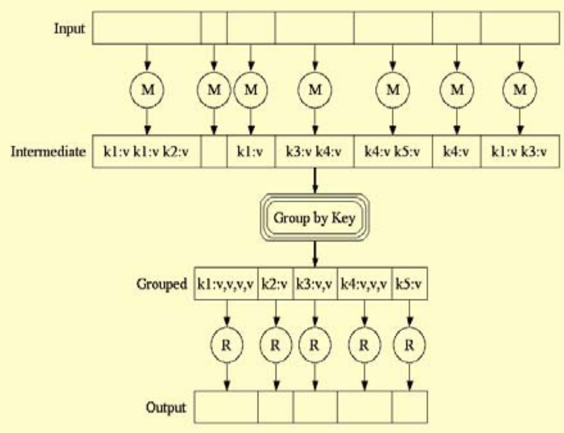


Fig 3.1 Map Reduce Operations

The Map Reduce framework consists of a single master Job tracker and one slave Task tracker per cluster-node. The master is responsible for scheduling the jobs' element tasks on the slaves, monitoring them and re-executing the unsuccessful tasks. The slaves execute the tasks as directed by the master.

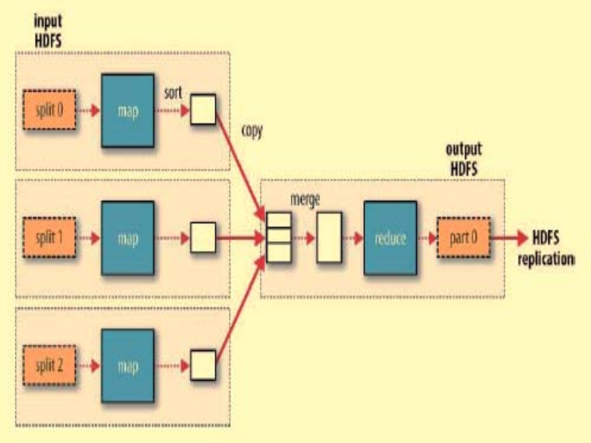


Fig 3.2 Map Reduce Data Flow with A Single Reduce Task

3.1 INPUTS AND OUTPUTS:

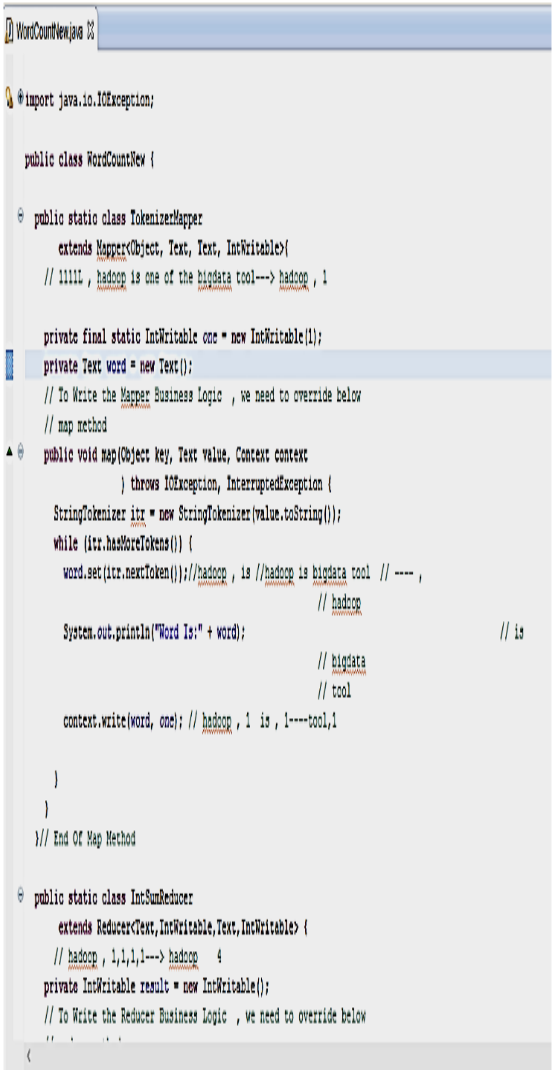
The Map Reduce framework operates completely on <key, value> pairs, that is, the framework views the input to the job as a group of <key, value> pairs and produces a set of <key, value> pairs as the output of the work, conceivably of different varieties. The key and value categories have to be serializable by the framework and hence got to implement the Writable interface. Additionally, the key classes have to implement the Writable Comparable interface to facilitate sorting by the framework. Input and Output types of a Map reduce job: (input) <k1, v1> -> map -> <k2, v2> -> combine -> <k2, v2> -> reduce -> <k3, v3> (output).

3.2 EXAMPLE: WORD COUNT V1.0

• Before we jump into the details, lets walk through an example Map reduce application to get a flavor for the way they work.

• Word Count is the simple application that counts the number of occurrences of each word in an exceedingly given input set.

• This works with a local-standalone, pseudo-distributed or fully-distributed Hadoop installation (Single Node Setup).



```
import java.io.IOException;

public class WordCountNew {

  public static class TokenizerMapper
       extends Mapper<Object, Text, Text, IntWritable>{
    // 1111L , hadoop is one of the bigdata tool---> hadoop , 1

    private final static IntWritable one = new IntWritable(1);
    private Text word = new Text();
    // To Write the Mapper Business Logic  , we need to override below
    // map method
    public void map(Object key, Text value, Context context
                    ) throws IOException, InterruptedException {
      StringTokenizer itr = new StringTokenizer(value.toString());
      while (itr.hasMoreTokens()) {
        word.set(itr.nextToken());//hadoop , is //hadoop is bigdata tool  // ---- ,
                                                    // hadoop
        System.out.println("Word Is:" + word);                              // is
                                                    // bigdata
                                                    // tool
        context.write(word, one); // hadoop , 1  is , 1----tool,1

      }
    }
  }// End Of Map Method

  public static class IntSumReducer
       extends Reducer<Text,IntWritable,Text,IntWritable> {
     // hadoop , 1,1,1,1---> hadoop    4
    private IntWritable result = new IntWritable();
    // To Write the Reducer Business Logic  , we need to override below
```

Fig 3.3 Word count java code

3.3 USAGE:

Assuming HADOOP_HOME is the root of the installation and HADOOP_VERSION.

- Hadoop version installed, compile WordCount.java and create a jar:

```
$ mkdir wordcount_classes
$ javac -classpath
${HADOOP_HOME}/hadoop-
${HADOOP_VERSION}-core.jar -d
wordcount_classes WordCount.java
$ jar -cvf /usr/joe/wordcount.jar -C
wordcount_classes/ .
```

- ASSUMING THAT:

• /usr/joe/wordcount/input - input directory in HDFS

• /usr/joe/wordcount/output - output directory in HDFS

- SAMPLE TEXT-FILES AS INPUT:

```
$ bin/hadoop dfs -ls /usr/joe/wordcount/input/
/usr/joe/wordcount/input/file01
/usr/joe/wordcount/input/file02
$ bin/hadoop dfs -cat
/usr/joe/wordcount/input/file01
Hello World Bye World (input file text)
$ bin/hadoop dfs -cat
/usr/joe/wordcount/input/file02
Hello Hadoop Goodbye Hadoop
```

- RUN THE APPLICATION:

```
$ bin/hadoop jar /usr/joe/wordcount.jar
org.myorg.WordCount
/usr/joe/wordcount/input
/usr/joe/wordcount/output
```

- OUTPUT:

```
$ bin/hadoop dfs -cat
/usr/joe/wordcount/output/part-00000
```

Bye 1
Goodbye 1
Hadoop 2
Hello 2
World 2

IV. PIG

Pig was initially developed at Yahoo! to allow individuals using Apache Hadoop to focus a lot of on analyzing massive data sets and pay less time having to put in writing mapper and reducer programs. Like actual pigs, who eat nearly something, the Pig programming language is meant to handle any reasonably data—hence the name!

Pig is made up of two components: the first component is that the language itself, which is known as Pig Latin (yes, people naming various Hadoop projects do tend to have a way of humor related to their naming conventions), and the second could be a runtime environment wherever Pig Latin programs are executed. Think of the link between a Java Virtual Machine (JVM) and a Java application. In this section, we'll just refer to the total entity as Pig.

- Let's first look at the programming language itself so you can see how it's significantly easier than having to write mapper and reducer programs.

1. The first step in an exceedingly Pig program is to LOAD the data you wish to control from HDFS.
2. Then you run the data through a group of transformations (which, under the covers, are translated into a set of mapper and reducer tasks).
3. Finally, you DUMP the data to the screen or you STORE the results in a file somewhere.

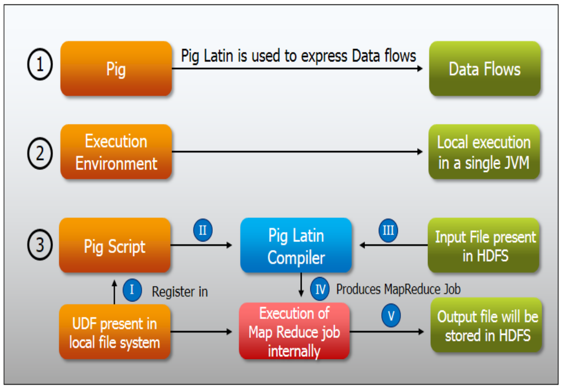

1
Pig
Pig Latin is used to express Data flows
Data Flows
2
Execution Environment
Local execution in a single JVM
3
Pig Script
II
Pig Latin Compiler
III
Input File present in HDFS
I
Register in
IV
Produces MapReduce Job
UDF present in local file system
Execution of Map Reduce job internally
V
Output file will be stored in HDFS

Fig 4.1 Apache Pig Operations

V. HIVE

Hive, allows SQL developers to write Hive query language (HQL) statements that are similar to standard SQL statements; now you must remember that HQL is proscribed in the commands it understands, but it is still pretty useful. HQL statements are broken down by the Hive service into Map reduce jobs and executed across a Hadoop cluster.

For anyone with a SQL or relational database background, this section will look very familiar to you. As with any database management system (DBMS), you can run your Hive queries in many ways. You can run them from a program line interface (known because the Hive shell), from a Java database connectivity (JDBC) or Open Database connectivity (ODBC) application leveraging the Hive JDBC/ODBC drivers, or from what is called a Hive Thrift consumer. The Hive Thrift Client is a lot of like every database consumer that gets installed on a user's consumer machine (or in an exceedingly middle tier of a three-tier architecture): it communicates with the Hive services running on the server.

Hive looks terribly abundant like traditional database code with SQL access. However, because Hive is primarily based on Hadoop and Map reduce operations, there are many key variations. The first is that Hadoop is meant for long successive scans, and because Hive is predicated on Hadoop, you can expect queries to possess a really high latency (many minutes). This means that Hive wouldn't be applicable for applications that require in no time response times, as you would expect with a database like DB2. Finally, Hive is read-based and therefore not acceptable for dealings processing that usually involves a high percentage of write operations

Currently, there are four file formats supported in Hive, which are TEXTFILE, SEQUENCEFILE, ORC and RCFILE.

Other features of Hive include:

- Indexing to provide acceleration, index type as well as compaction and image index as of 0.10, more index varieties are planned.
- Different storage types such as plain text, RCFile, HBase, ORC, and others.

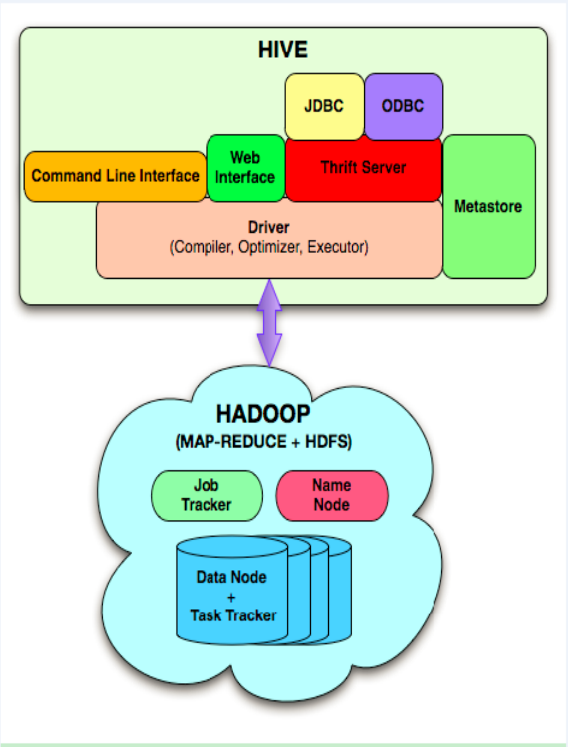


Fig 5.1 Hive Architecture

VI. HBASE

HBase is a column-oriented database management system that runs on top of HDFS. It is compatible for sparse data sets, which are common in several huge data use cases. Unlike relational database systems, HBase does not support a structured command language like SQL; actually, HBase isn't a relational data store at all. HBase applications are written in Java much like a typical Map reduce application.

For example, if the table is storing diagnostic logs from servers in your environment, where each row would possibly be a log record, a typical column in such a table would be the timestamp of when the log record was written, or perhaps the server name wherever the record originated. In fact, HBase allows for several attributes to be classified together into what are called column families, such that the weather of a column family is all keep together. This is different from a row-oriented on-line database, where all the columns of a given row are stored together. With HBase you must predefine the table schema and specify the column families. However, it's very versatile in that new columns are often additional to families at any time, making the schema flexible and thus ready to adapt to dynamic application needs. Just as HDFS includes a NameNode and slave nodes, and MapReduce has JobTracker and TaskTracker slaves, HBase is built on similar ideas. In HBase a master node manages the cluster and region servers store portions of the tables and perform the work on the data.

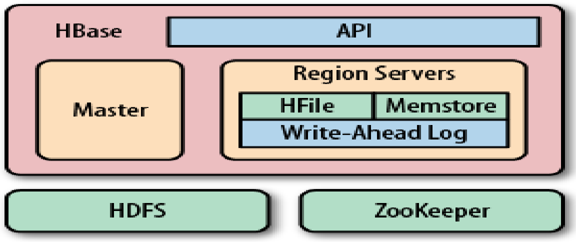


Fig. 6.1 Hbase Architecture

VII. SPARK

Apache Spark is a lightning-fast cluster computing technology, designed for fast computation. It is supported Hadoop MapReduce and it extends the MapReduce model to efficiently use it for more forms of computations, which includes interactive queries and stream process. The main feature of Spark is its in-memory cluster computing that increases the process speed of an application. Spark is designed to hide a large range of workloads like batch applications, iterative algorithms, interactive queries and streaming. Apart from supporting of these workloads in an exceedingly respective system, it reduces the management burden of maintaining separate tools.

7.1 FEATURES OF APACHE SPARK:

Apache Spark has following features.

- Speed − Spark helps to run an application in Hadoop cluster, up to 100 times quicker in memory, and 10 times quicker when running on disk. This is possible by reducing variety of read/write operations to disk. It stores the intermediate processing data in memory.

- Supports multiple languages − Spark provides built-in Apis in Java, Scala, or Python. Therefore, you can write applications in several languages. Spark comes up with 80 high-level operators for interactive querying.

- Advanced Analytics − Spark not only supports 'Map' and 'reduce'. It also supports SQL queries, Streaming data, Machine learning (ML), and Graph algorithms.

7.2 SPARK BUILT ON HADOOP:

The following diagram shows three ways of how Spark can be built with Hadoop components.

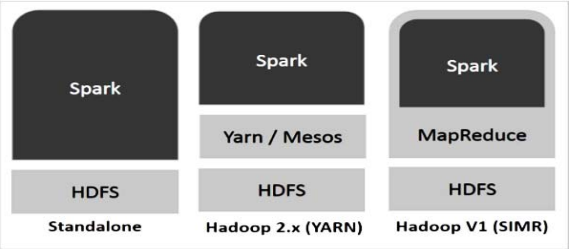


Fig.7.1 Spark Architecture

There are three ways of Spark deployment as explained below.

- Standalone− Spark is a Standalone deployment suggests that Spark occupies the place on high of HDFS (Hadoop Distributed File System) and house is allocated for HDFS, explicitly. Here, Spark and MapReduce will run side by side to cover all spark jobs on cluster.

- Hadoop Yarn− Hadoop Yarn deployment suggests that, simply, spark runs on Yarn without any pre-installation or root access needed. It helps to integrate Spark into Hadoop ecosystem or Hadoop stack. It allows alternative elements to run on top of stack.

- Spark In Mapreduce (SIMR)− Spark in MapReduce is used to launch spark job additionally to standalone deployment. With SIMR, user can begin Spark and uses its shell without any administrative access.7.3 Components of Spark.

7.3COMPONENETS OF SPARK:

- The following illustration depicts the different components of Spark:

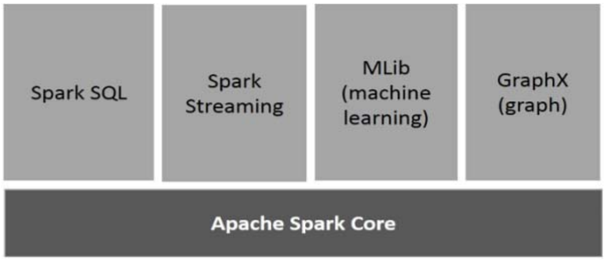


Fig.7.2 Spark Core Components

7.3.1 APACHE SPARK CORE

Spark Core is the underlying general execution engine for spark platform that all other functionality is built upon. It provides In-Memory computing and referencing datasets in external storage systems.

7.3.2 SPARK SQL

Spark SQL is a component on top of Spark Core that introduces a new data abstraction called Schema RDD, which provides support for structured and semi-structured data.

7.3.3 SPARK STREAMING

Spark Streaming leverages Spark Core's fast scheduling capability to perform streaming analytics. It ingests data in mini-batches and performs RDD (Resilient Distributed Datasets) transformations on those mini-batches of data.

7.3.4 MLLIB (MACHINE LEARNING LIBRARY)

MLlib is a distributed machine learning framework above Spark because of the distributed memory-based Spark architecture. It is, according to benchmarks, done by the MLlib developers against the Alternating Least Squares (ALS) implementations. Spark MLlib is nine times as fast as the Hadoop disk-based version of Apache Mahout (before Mahout gained a Spark interface).

7.3.5 GRAPHX

GraphX is a distributed graph-processing framework on top of Spark. It provides an API for expressing graph computation that can model the user-defined graphs by using Pregl abstraction API. It also provides an optimized runtime for this abstraction.

VIII. CONCLUSIONS

Hadoop and the MapReduce programming paradigm have already got a considerable base within the bioinformatics community, especially in the field of next-generation sequencing analysis, and such use is increasing. This is as a result of the cost-effectiveness of Hadoop-based analysis on commodity Linux clusters, and in the cloud via data transfer to cloud vendors who have implemented Hadoop/HBase; and due to the effectiveness and ease-of-use of the MapReduce technique in parallelization of the many data analysis algorithms.

IX. REFERENCES:

[1] Vibhavari Chavan etal, /(IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 5 (6)

[2] Yuri Demchenko "The Big Data Architecture Framework (BDAF)" Outcome of the Brainstorming Session at the University of Amsterdam 17 July 2013.

[3] Big Data, Data Mining, and Machine Learning: Value Creation for Business Leaders and Practitioners by Jared Dean

[4] D.Talia, "Clouds for scalable big data analytics," Computer, vol. 46, no. 5, Article ID 6515548, pp. 98–101, 2013.

[5] D. E. O'Leary, "Artificial intelligence and big data," IEEE Intelligent Systems, vol. 28, no. 2, pp. 96–99, 2013.

[6] The Hadoop Distributed File System, Konstantin Shvachko, Hairong Kuang, Sanjay Radia, Robert Chansler Yahoo! Sunnyvale, California USA 2010.

[7] Hadoop: The Definitive Guide by Tom White

[8] Dean, J. and Ghemawat, S., "MapReduce: a flexible data processing tool", ACM 2010

[9] Pig - Apache Software Foundation project home page. http://pig.apache.org/

[10] Hive - Apache Software Foundation project home page. http://hadoop.apache.org/hive/

[11] HBase - Apache Software Foundation project home page. http://hadoop.apache.org/hbase/

[12] Hadoop - Apache Software Foundation project home page. http://hadoop.apache.org/

[13] Cloudera recommendations on Hadoop/HBasehttp://www.cloudera.com/blog/2010/08/hadoophbase-capacity-planning/

[14]Hbasehttp://www01.ibm.com/software/data / infosphere/hadoop/hbase

[15] Sprark-http://spark.apache.org/