Map Reduce Programming for Electronic Medical Records Data Analysis on Cloud using Apache Hadoop, Hive and Sqoop

Sreekanth Rallapalli[#] , Gondkar RR[*]

[#]*Research Scholar, R&D Center, Bharathiyar University, Coimbatore, Tamilnadu*
**Professor, Department of IT, AIT, Bangalore*

Abstract—**Health care organizations now a day's made a strategic decision to turn huge medical data coming from various sources into competitive advantage. This will help the health care organizations to monitor any abnormal measurements which require immediate reaction. Apache Hadoop has emerged as a software framework for distributed processing of large datasets across large clusters of computers. Hadoop is based on simple programming model called MapReduce. Hive is a data warehousing framework built on top of hadoop. Hive is designed to enable easy data summarization, ad-hoc querying and analysis of large volume of data. As health care and Electronic Medical Records (EMR) are generating huge data, it is necessary to store, extract and load such big data using a framework which support distributed processing. Cloud computing model provides efficient resources to store and process the data. In this paper we propose a MapReduce programming for Hadoop which can analyze the EMR on cloud. Hive is used to analyze large data of healthcare and medical records. Sqoop is used for easy data import and export of data from structured data stores such as relational databases, enterprise datawarehouses and NoSQL systems.**

Keywords—Hadoop;MapReduce;EMR;Hive;Healthcare

I. APACHE HADOOP FOR BIG DATA

User Interactions online generated a huge data and in order to extend the services to scale the collection of data companies like Google, Facebook, Yahoo and many other companies have scaled up the capabilities of traditional information Technology architectures. In order to store, extract, transform and load these Big data these companies build their own core infrastructure components rapidly and various papers were published for many of components. All these components were open source. Apache Hadoop has been standardized for managing a large volumes of unstructured data [1].

Hadoop is an open source distributed software platform for storing and processing data. We can store petabytes of data reliably on tens of thousands of servers while scaling performance cost-effectively by adding inexpensive nodes to the cluster. MapReduce programming will help the programmers to solve parallel-data problems for which the data set can be sub-divided into smaller parts and processed. The system splits data into multiple chunks which is assigned a map that process data in parallel. The map task reads input data as a set of (key,value) pairs and produce a transformed set of (key,value) pair as output.

A master node has the job of distributing the work to worker nodes. The worker node just does one thing and returns the work back to the master node (i.e data processing). Once the master gets the work from the worker nodes, the reduce step takes over and combines all the work. By combining the work you can form some answer and ultimately output.(i.e data collection and digesting).

Cloud computing is a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction [4]. Most of the health care organizations today are moving to cloud architecture to store and process the huge medical data.

II. HADOOP ARCHITECTURE

The Hadoop architecture is shown in the Fig1. The Hadoop distributed File system (HDFS) is a distributed file system providing fault tolerance and designed to run on commodity hardware. HDFS provides high throughput access to application data and is suitable for applications that have a large data sets. Hadoop provides a distributed file system called HDFS that can store data across thousands of servers, and a means of running work (Map/Reduce jobs) across those machines, which move code to data. HDFS have master/slave architecture. Hadoop runs on large clusters of commodity machines or on cloud computing services. Hadoop scales linearly to handle larger data by adding more nodes to the cluster.

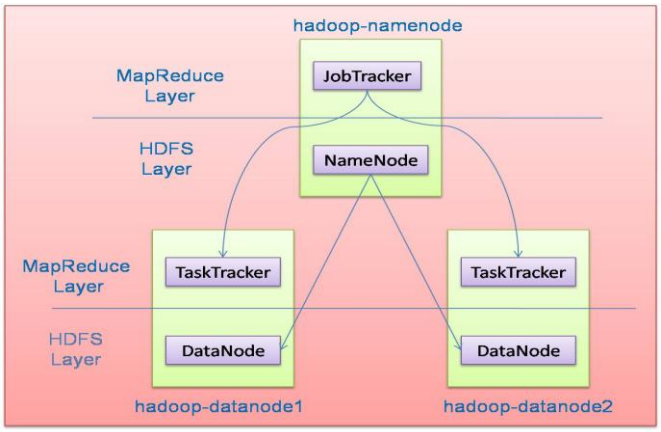

Fig 1: Hadoop Architecture with one master node and two slave nodes

Hadoop has 5 daemons or services running during the process. The 5 daemons are namenode, datanode, secondary namenode, jobtracker, tasktraker.

III. HIVE

Hive is a component of the hadoop. It was initially introduced by facebook in the year 2007 in order to full fill the requirements with respect to ETL (Extract Transformation Load) jobs. Later it becomes hadoop sub project. Hive is a data warehousing frame work built on top of the hadoop [2]. It is a hive default table. These tables will be completely managed by hive ware house. All manage tables are stored in following default directory /user/hive/warehouse. Fig2. Shows the hive architecture.

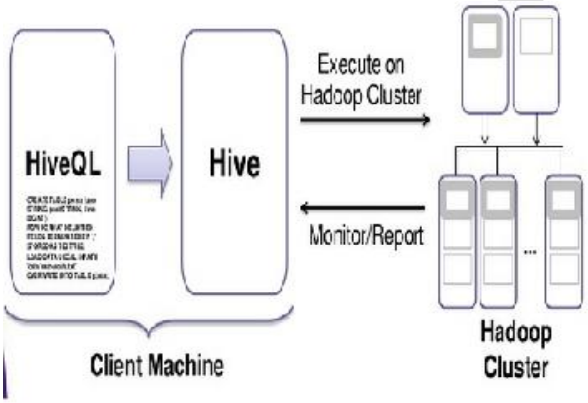

Fig2: Hive Architecture

IV. MAPREDUCE PROGRAMMING FOR EMR

MapReduce uses parallel programming which in turn improves performance and efficiency. Processing is broken up into parts and done concurrently. Instruction of each part runs on a separate CPU while many processors are connected [3]. Identification of set of tasks which can run concurrently is important. Input files are split into M pieces on distributed file systems. Intermediate files are created from map tasks are written to local disks. Output files are written to distributed file systems. Table 1 shows the main goal of MapReduce programming

Map Reduce Goal
Scalability to large data volumes:
Scan 100 TB on 1 node @ 50 MB/s = 24 days.
Scan on 1000 node cluster = 35 minutes.
Cost-efficiency:
Commodity nodes (cheap, but unreliable).
Commodity network.
Automatic fault-tolerance (fewer admins).
Easy to use (fewer programmers).

Table 1: Map Reduce Goal

In this section we take Electronic medical records of large number of patients and then write the MapReduce program/algorithm that find the occurrence of patient medical history in a large files [6].

```
package org.myorg;
import java.io.IOException;
import java.util.*;

import org.apache.hadoop.fs.Path;
import org.apache.hadoop.conf.*;
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapreduce.*;
import
org.apache.hadoop.mapreduce.lib.input.FileInputFo
rmat;
import
org.apache.hadoop.mapreduce.lib.input.TextInputFo
rmat;
import
org.apache.hadoop.mapreduce.lib.output.FileOutput
Format;
import
org.apache.hadoop.mapreduce.lib.output.TextOutput
Format;

public class medicalrecord {
 public static class Map extends
Mapper<LongWritable, Text, Text, IntWritable> {
    private final static IntWritable one = new
IntWritable(1);
    private Text word = new Text();

    public void map(LongWritable key, Text value,
Context context) throws IOException,
InterruptedException {
        String line = value.toString();
        StringTokenizer tokenizer = new
StringTokenizer(line);
        while (tokenizer.hasMoreTokens()) {
            word.set(tokenizer.nextToken());
            context.write(word, one);
        }
    }
 }

 public static class Reduce extends Reducer<Text,
IntWritable, Text, IntWritable> {

    public void reduce(Text key,
Iterator<IntWritable> values, Context context)
      throws IOException, InterruptedException {
        int sum = 0;
        while (values.hasNext()) {
            sum += values.next().get();
        }
        context.write(key, new IntWritable(sum));
    }
 }
```

```
 public static void main(String[] args) throws
Exception {
    Configuration conf = new Configuration();

    Job job = new Job(conf, "medicalrecord");
    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(IntWritable.class);
    job.setMapperClass(Map.class);
    job.setReducerClass(Reduce.class);

job.setInputFormatClass(TextInputFormat.class);

job.setOutputFormatClass(TextOutputFormat.class);
    FileInputFormat.addInputPath(job, new
Path(args[0]));
    FileOutputFormat.setOutputPath(job, new
Path(args[1]));

    job.waitForCompletion(true);
 }

}
```

Fig 3: Java source code for MapReduce to find the occurance of patients medical data

The MapReduce algorithm is used to find out the medical record occurrences in the file. In the above example the map input key is the provided data chunk with a value of 1. The map output key is the word itself and the value is 1 everytime the word exists in the processed data chunk. The reducers perform the aggregation of the key-values pair output from the maps word. MapReduce programs are usually written in java and can also be coded in languages such as C++, perl, python, Ruby, R.

V. LOADING ELECTRONIC MEDICAL RECORDS INTO HIVE

In this section we will see how the EMR data which is stored as a text file in unstructured format can be loaded into Hive and then processed.

```
hive> CREATE EXTERNAL TABLE IF NOT EXISTS
emr (
patientname STRING,
patientid STRING,
test1 STRING,
test2 STRING,test3 STRING,test4 STRING) ROW
FORMAT DELIMITED FIELDS TERMINATED BY '/t'
LOCATION '/user/botho/emr';
>load data local inpath '/home/botho/MEDICAL' into table
emr;
```

This command will load the text file containing all medical records and format it into hive table.

```
Hive>create database EMR;
Hive>use database EMR;
Hive>select * from emr;
```

If the text file contains a millions of records it is sent to hive table and using sqoop the dataset being transferred is sliced up into different partitions and a map-only job is launched with individual mappers responsible for transferring a slice of this dataset.

VI. RESULTS

We have first started with Hadoop 5 Services such as Namenode, datanode, Job tracker, task tracker and Secondary name node. All 5 deamons should be running in order to work further with hive, and sqoop.

Fig4 shows the Hadoop deamons are running

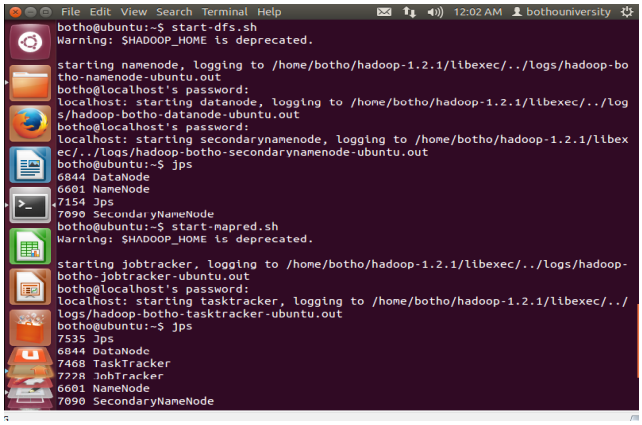

Fig 4 Hadoop Daemons

Fig 5 shows the database created for Electronic medical records using hive and created a table using hive. The data is loaded into hive by using a text file which has records. Fig 6 shows the display of EMR records using a query.

Sqoop is used to export the bulk data of electronic medical records to Hadoop system and using map-reduce programming which is responsible to for transferring the slice of this dataset.

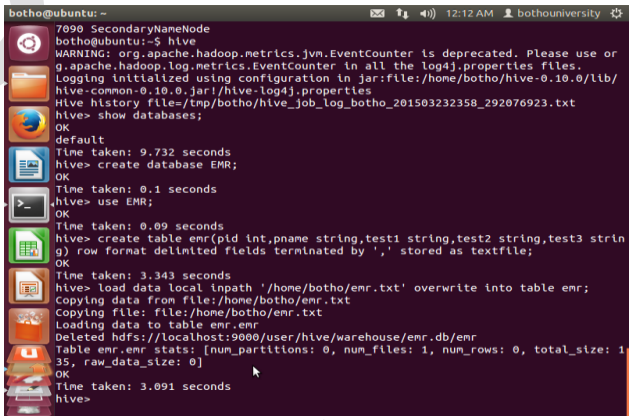

Fig 5 Loading of EMR Records

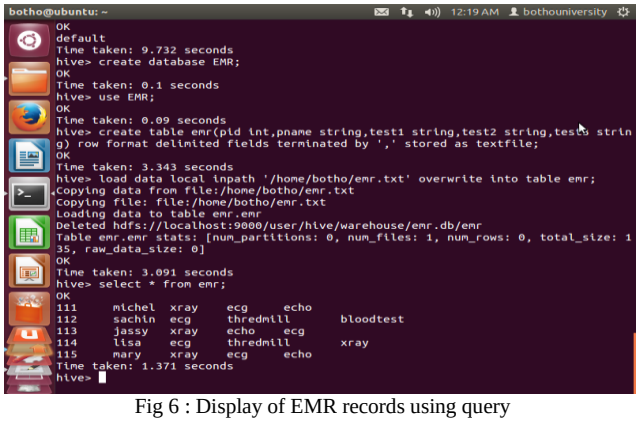

Fig 6 : Display of EMR records using query

VII. CONCLUSION

Sqoop is a bulk data transfer tool that allows easy import/export of data from structured datastores such as relational databases, enterprise data warehouses, and NoSQL systems. Using Sqoop, you can provision the data from an external system into HDFS, as well as populate tables in Hive and HBase. Similarly, Sqoop integrates with the workflow coordinator Apache Oozie (incubating), allowing you to schedule and automate import/export tasks. Sqoop uses a connector-based architecture which supports plugins that provide connectivity to additional external systems. In this paper we have taken bulk Elecronic medical records of unstructured data and then uses hive, to import the data into hadoop and then process it and further this is exported to external databases on cloud for further analysis.

REFERENCES

[1]. Dean J, Ghemawat S: MapReduce: simplified data processing on large clusters. Commun ACM 2008, 51(1):107–113.

[2]. Thusoo A, Sarma JS, Jain N, Shao Z, Chakka P, Anthony S, Liu H, Wyckoff P, Murthy R: Hive: a warehousing solution over a map-reduce framework. Proc VLDB Endowment 2009, 2(2):1626–1629.

[3]. Nguyen AV, Wynden R, Sun Y: HBase, MapReduce, and Integrated Data Visualization for Processing Clinical Signal Data. In AAAI Spring Symposium: Computational Physiology: 2011; 2011.

[4]. AWS | Amazon Elastic Compute Cloud (EC2) - Scalable Cloud Hosting. [http://aws.amazon.com/ec2/]

[5]. Sadasivam GS, Baktavatchalam G: A novel approach to multiple sequence alignment using hadoop data grids. In Proceedings of the 2010 Workshop on Massive Data Analytics on the Cloud: 2010, ACM; 2010:2.

[6]. Wang F, Lee R, Liu Q, Aji A, Zhang X, Saltz J: Hadoop-gis: A high performance query system for analytical medical imaging with mapreduce. In Atlanta – USA: Technical report, Emory University; 2011:1–13.

[7]. Musen MA, Middleton B, Greenes RA: Clinical decision-support systems. In Biomedical Informatics. New York – USA: Springer; 2014:643–674.

[8]. Mazurek M: Applying NoSQL Databases for Operationalizing Clinical Data Mining Models. In Beyond Databases, Architectures, and Structures. New York – USA: Springer; 2014:527–536.

[9]. Youssef AE: A framework for secure healthcare systems based on Big data analytics in mobile cloud computing environments. Int J Ambient Syst Appl 2014, 2(2):1–11.

[10]. Jeffrey Dean and Sanjay Ghemawat, " MapReduce: a flexible data processing tool," Communications of the ACM, Volume 53 Issue 1, January 2010, Pages 72-77

[11]. MapReduce: Simplified Data Processing on Large Clusters. Available at http://labs.google.com/papers/mapreduceosdi04.pdf

Appendix-I

Map Reduce Programming Flow for EMR