14CS IT 703

Hall Ticket Number:

IV/IV B.Tech (Regular/Supplementary) DEGREE EXAMINATION

October, 2018 — **Common to CSE & IT**

Seventh Semester — **Advanced Data Analytics**

Time: Three Hours — **Maximum:** 60 Marks

Answer Question No.1 compulsorily. (1X12 = 12 Marks)

Answer ONE question from each unit. (4X12=48 Marks)

1. Answer all questions (1X12=12 Marks)

a) What is RAID? Explain

RAID or Redundant Array of Independent Disks, is a technology to connect multiple secondary storage devices and use them as a single storage media.**RAID** consists of an array of disks in which multiple disks are connected together to achieve different goals.

b) Define HDFS and explain

The Hadoop Distributed File System (HDFS) is a distributed file system designed to run on commodity hardware. It has many similarities with existing distributed file systems.

c) What is YARN?

The fundamental idea of YARN is to split up the functionalities of resource management and job scheduling/monitoring into separate daemons. The idea is to have a global ResourceManager (*RM*) and per-application ApplicationMaster (*AM*). An application is either a single job or a DAG of jobs.

d) Define cluster.

A **Hadoop cluster** is a special type of computational **cluster** designed specifically for storing and analyzing huge amounts of unstructured data in a distributed computing environment.

e) What is Pig Latin?

Pig is a high level scripting language that is used with Apache **Hadoop**. **Pig** enables data workers to write complex data transformations without knowing Java. **Pig's**simple SQL-like scripting language is called **Pig Latin**, and appeals to developers already familiar with scripting languages and SQL.

f) What is Metastore? Explain

The **Hive metastore** is a service that stores the metadata for**Hive** tables and partitions in a relational database. IT provides clients (including **Hive**) access to this information using the**metastore** service API. Default **metastore** is Derby(in-memory). RDBMS is used at enterprise level.

g) What is Task Scheduling?

he **task scheduling** algorithm has a direct impact on the **task** execution. **Task scheduling** for**Hadoop** platform means that allocating the appropriate **tasks** of appropriate job to appropriate **task**server.

h) List the aggregation functions in Hive

MIN,MAX,SUM,AVG,COUNT

i) Define Schema

A database **schema** is the skeleton structure that represents the logical view of the entire database. It defines how the data is organized and how the relations among them are associated.

j) Write data types in Hive

The primitive data types supported by Hive are listed below:

- **Numeric Types**. TINYINT (1-byte signed **integer**, from -128 to 127) SMALLINT (2-byte signed **integer**, from -32,768 to 32,767) ...
- Date/Time Types. TIMESTAMP. DATE.
- **String Types**. **STRING**. **VARCHAR**. ...
- Misc Types. **BOOLEAN**. ...
- **Complex** Types. arrays: **ARRAY**<data_type>

k) Define DAG and explain

(Directed Acyclic Graph) DAG in Apache Spark is a set of **Vertices** and **Edges**, where *vertices* represent the **RDDs** and the *edges* represent the **Operation to be applied on RDD**. In Spark DAG, every edge is directed from earlier to later in the sequence. On calling of *Action*, the created DAG is submitted to **DAG Scheduler**which further splits the graph into the **stages** of the **task**.

l) What is Export in Sqoop?

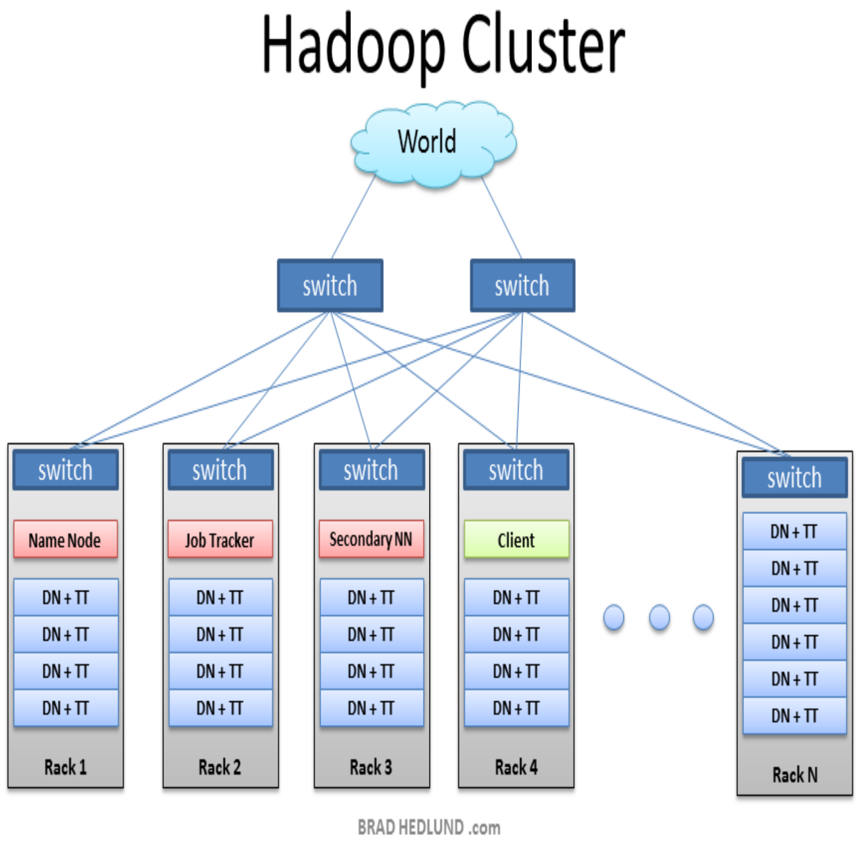
qoop - Export. This chapter describes how to **export** data back from the HDFS to the RDBMS database. The target table must exist in the target database. The files which are given as input to the**Sqoop** contain records, which are called rows in table.

UNIT I

2. a) Describe typical two-level network architecture for a Hadoop cluster 6M

The three major categories of machine roles in a Hadoop deployment are Client machines, Masters nodes, and Slave nodes. The Master nodes oversee the two key functional pieces that make up Hadoop: storing lots of data (HDFS), and running parallel computations on all that data (Map Reduce). The Name Node oversees and coordinates the data storage function (HDFS), while the Job Tracker oversees and coordinates the parallel processing of data using Map Reduce. Slave Nodes make up the vast majority of machines and do all the dirty work of storing the data and running the computations. Each slave runs both a Data Node and Task Tracker daemon that communicate with and receive instructions from their master nodes. The Task Tracker daemon is a slave to the Job Tracker, the Data Node daemon a slave to the Name Node.

Client machines have Hadoop installed with all the cluster settings, but are neither a Master or a Slave. Instead, the role of the Client machine is to load data into the cluster, submit Map Reduce jobs describing how that data should be processed, and then retrieve or view the results of the job when its finished. In smaller clusters (~40 nodes) you may have a single physical server playing multiple roles, such as both Job Tracker and Name Node. With medium to large clusters you will often have each role operating on a single server machine.



b) Describe important Hadoop properties. 6M

o run HDFS, you need to desinate one machine as a namenode. In this case the property fs.defaultFS is an HDFS filesystem URI whose host is the namenode's hostname or IP address and whose port is the port that the namenode will listen on for RPCs. If no port is specified, the default of 8020 is used.

There are a few other configuration properties you should set for HDFS: those that set the storage directories for the namenode and for datanodes. The property dfs.namenode.name.dir specifies a list of directories where the namenode stores persistent filesystem metadata. A copy of each metadata file is stored in each directory for redundancy.

as for dfs.datanode.data.diir property, which specifies a list of directories for a datanode to store its blocks in. Unlike the namenode, a datanode round-robins writes between its storage directories, so

for performance you should specify a storage directory for each local disk. Read performance also benefits from having multiple disks for storage.
The dfs.namenode.checkpoint.dir property specifies a list of directories where the checkpoints are kept. Like the storage directories for the namenode, whick keep redundant copies of the namenode metadata. the checkpointed filesystem image is stored in each checkpoint directory for redundancy.

(OR)

3. a) Write the procedure for installing Hadoop and SSH Configuration. 6M

We need to copy this Commands to bashrc file:-(on bascrh file)

```
export HADOOP_INSTALL=/usr/local/hadoop

export PATH=$PATH:$HADOOP_INSTALL/bin

export PATH=$PATH:$HADOOP_INSTALL/sbin

export HADOOP_MAPRED_HOME=$HADOOP_INSTALL

export HADOOP_HDFS_HOME=$HADOOP_INSTALL

export HADOOP_COMMON_HOME=$HADOOP_INSTALL

export YARN_HOME=$HADOOP_INSTALL

export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_INSTALL/lib/native

export HADOOP_OPTS="-Djava.library.path=$HADOOP_INSTALL/lib/native"
```

copy this Commands on terminal

```
mkdir -p /usr/local/hadoopdata/hdfs/namenode

mkdir -p /usr/local/hadoopdata/hdfs/datanode

sudo mkdir -p /app/hadoop/tmp

sudo chown hadoop1:hadoop1 /app/hadoop/tmp(Instead of hadoop1:hadoop1 give your user name   ex:  lavanya: lavanya)

sudo chmod 750 /app/hadoop/tmp
```

copy this Commands on terminal

```
mkdir -p /usr/local/hadoopdata/hdfs/namenode

mkdir -p /usr/local/hadoopdata/hdfs/datanode

sudo mkdir -p /app/hadoop/tmp

sudo chown hadoop1:hadoop1 /app/hadoop/tmp(Instead of hadoop1:hadoop1 give your user name   ex:  lavanya: lavanya)

sudo chmod 750 /app/hadoop/tmp
```

b) Discuss about the Hadoop file systems. 6M

- HDFS is distributed file system designed for storing very large files with streaming data access patterns, running on cluster of commodity hardware.

- HDFS is a logical file system across all the nodes local system; it provides special capabilities to handle the storage of big data efficiently.
- We are giving the files to HDFS, and it will divide the file into no of blocks of manageable size
- (either 64MB or 128 MB based on configuration)
- These blocks will be replicated three times and stored in local file systems as a separate file.
- Blocks are replicated across a multiple machines, known as **DataNodes**.
- **DataNode** is slave machine in Hadoop cluster running the data node daemon.

Daemon-A process continuously running

- A Master Node(high end configurations like dual power supply, dual n/w cards. Etc) called the Name Node(a node which is running name node daemon) keeps track of which blocks make up a file, and where those blocks are located , known as meta data.
- The Name Node keep track of the file metadata –which files are running in the system and how each file is broken down into .
- Name Node to keep the metadata current.

UNIT II

4. a) How Hadoop runs a MapReduce job using the classic framework? Explain. 6M

WordCountMapper Class

```
package wc.hadoop.training.it.bec;

import java.io.IOException;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;


public class WordCountMapper extends Mapper<LongWritable, Text, Text, IntWritable>

{
         private String tokens = "[_|$#<>\\^=\\[\\]\\*/\\\\,;,.\\-:()?!\""]";
         public void map(LongWritable recadd, Text rec, Context con) throws IOException,
     InterruptedException

      {
               String cleanline = rec.toString().toLowerCase().replaceAll(tokens, " ");
               String[] words = cleanline.split(" ");
               for(String kw : words)

                {

                        con.write(new Text(kw.trim()), new IntWritable(1));

               }


     }

}
```

WordCountReducer.Class

```
package wc.hadoop.training.it.bec;

import java.io.IOException;

import java.util.HashMap;

import java.util.Map;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;


public class WordCountReducer extends Reducer<Text, IntWritable, Text, IntWritable>

 {

        private Map<String, Integer> countMap = new HashMap<String, Integer>();
        public void reduce(Text key, Iterable<IntWritable> values, Context con)
        throws IOException, InterruptedException

         {

                int sum = 0;
                for (IntWritable el : values)

                 {

                        sum = sum + el.get();

                }
                countMap.put(key.toString(), new Integer(sum));

        }




        @Override
        protected void cleanup(Context con) throws IOException,
        InterruptedException

        {

                super.cleanup(con);

                Map<String, Integer> sortedMap = MiscUtils.sortByValues(countMap);

                // int counter = 0;

                for (String key : sortedMap.keySet())

                {
                    //if (counter++ == 5) {

                    //break;

                    //}

                    con.write(new Text(key), new IntWritable(sortedMap.get(key)));

                }

        }

}
```

WordCountJob.Class

```
package wc.hadoop.training.cse;
import org.apache.hadoop.conf.Configured;

import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.util.Tool;
import org.apache.hadoop.util.ToolRunner;
public class WordCountJob extends Configured implements Tool

{

        public static void main(String[] cla) throws Exception

        {
                int exitstatus = ToolRunner.run(new WordCountJob(), cla);

                System.exit(exitstatus);

        }

@Override
public int run(String[] args) throws Exception

{

        Job jb = Job.getInstance(getConf());

        jb.setJobName("Word Count");
        jb.setMapperClass(WordCountMapper.class);

        //jb.setCombinerClass(WordCountCOM.class);
        //jb.setSortComparatorClass(WordCountSC.class);
        jb.setReducerClass(WordCountReducer.class);

        //jb.setNumReduceTasks(0);
        jb.setMapOutputKeyClass(Text.class);
        jb.setMapOutputValueClass(IntWritable.class);
        jb.setOutputKeyClass(Text.class);

        jb.setOutputValueClass(IntWritable.class);
        jb.setJarByClass(WordCountJob.class);
        FileInputFormat.setInputPaths(jb, new Path(args[0]));

        FileOutputFormat.setOutputPath(jb, new Path(args[1]));
        return jb.waitForCompletion(true) ? 0 : 1;


    }

    }
```

b) How YARN is different from MapReduce-1? Explain. 6M

- The problem is solved by splitting the responsibility of JobTracker (in Classic MapReduce) to different components. Because of which, there are more entities involved in YARN (compared to Classic MR). The entities in YARN are as follows;
  - <u>Client</u>: which submits the MapReduce job
  - <u>Resource Manager</u>: which manages the use of resources across the cluster. It creates new containers for Map and Reduce processes.
  - <u>Node Manager</u>: In every new container created by Resource Manager, a Node Manager process will be run which oversees the containers running on the cluster nodes. It doesn't matter if the container is created for Map or Reduce or any other process. Node Manager ensures that the application does not use more resources than what it is allocated with.
  - <u>Application Master</u>: which negotiates with the Resource Manager for resources and runs the application-specific process (Map or Reduce tasks) in those clusters. The Application Master & the MapReduce tasks run in containers that are scheduled by the resource manager and managed by the node manager.
  - <u>HDFS</u>

(OR)

5. a) Discuss the various types of failures in classic MapReduce 6M

- In the real world, user code is buggy, processes crash, and machines fail.
- One of the major benefits of using Hadoop is its ability to handle such failures and allow your job to complete successfully.

We need to consider the failure of any of the following entities:

- ➢ The task
- ➢ The application master
- ➢ The node manager

The resource manager.

b) Write MapReduce program to count the number words in a given file. 6M

WordCountMapper Class

```
package wc.hadoop.training.it.bec;

import java.io.IOException;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;


public class WordCountMapper extends Mapper<LongWritable, Text, Text, IntWritable>

{
          private String tokens = "[_|$#<>\\^=\\[\\]\\*/\\\\,;,.\\-:()?!\""]";
          public void map(LongWritable recadd, Text rec, Context con) throws IOException,
        InterruptedException

         {
                String cleanline = rec.toString().toLowerCase().replaceAll(tokens, " ");
                String[] words = cleanline.split(" ");
                for(String kw : words)

                 {

                        con.write(new Text(kw.trim()), new IntWritable(1));

                }


        }

}
```

WordCountReducer.Class

```
package wc.hadoop.training.it.bec;

import java.io.IOException;

import java.util.HashMap;

import java.util.Map;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;


public class WordCountReducer extends Reducer<Text, IntWritable, Text, IntWritable>

 {

        private Map<String, Integer> countMap = new HashMap<String, Integer>();
        public void reduce(Text key, Iterable<IntWritable> values, Context con)
        throws IOException, InterruptedException

         {

                int sum = 0;
                for (IntWritable el : values)
                 {
                        sum = sum + el.get();
                }
                countMap.put(key.toString(), new Integer(sum));
        }



        @Override
        protected void cleanup(Context con) throws IOException,
        InterruptedException
        {
                super.cleanup(con);
                Map<String, Integer> sortedMap = MiscUtils.sortByValues(countMap);
                // int counter = 0;
                for (String key : sortedMap.keySet())
                {
                        //if (counter++ == 5) {
                        //break;
                        //}
                        con.write(new Text(key), new IntWritable(sortedMap.get(key)));
                }
        }
}
```

WordCountJob.Class

```
package wc.hadoop.training.cse;
import org.apache.hadoop.conf.Configured;

import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.util.Tool;
import org.apache.hadoop.util.ToolRunner;
public class WordCountJob extends Configured implements Tool

{

        public static void main(String[] cla) throws Exception

        {
                int exitstatus = ToolRunner.run(new WordCountJob(), cla);

                System.exit(exitstatus);

        }

@Override
public int run(String[] args) throws Exception

{

        Job jb = Job.getInstance(getConf());

        jb.setJobName("Word Count");
        jb.setMapperClass(WordCountMapper.class);

        //jb.setCombinerClass(WordCountCOM.class);

        //jb.setSortComparatorClass(WordCountSC.class);
        jb.setReducerClass(WordCountReducer.class);

        //jb.setNumReduceTasks(0);
        jb.setMapOutputKeyClass(Text.class);
        jb.setMapOutputValueClass(IntWritable.class);
        jb.setOutputKeyClass(Text.class);

        jb.setOutputValueClass(IntWritable.class);
        jb.setJarByClass(WordCountJob.class);
        FileInputFormat.setInputPaths(jb, new Path(args[0]));

        FileOutputFormat.setOutputPath(jb, new Path(args[1]));
        return jb.waitForCompletion(true) ? 0 : 1;


}

}
```

UNIT III

6. a) How to execute and run the Pig programs? Explain. 6M

Start Apache Pig in local mode and register the jar file sample_udf.jar as shown below.

```
$cd PIG_HOME/bin
```

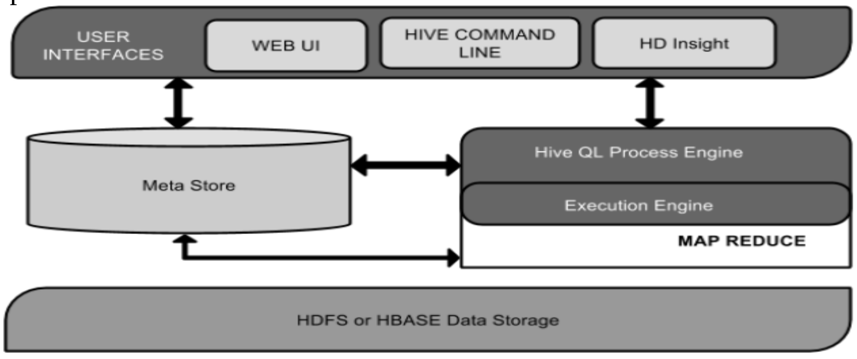
```
$./pig –x local
REGISTER '/$PIG_HOME/sample_udf.jar
```

b) What is UDF? Write UDF to trim leading and trailing whitespace from chararray values. 6M

- Pig having some Built-in functions,we can use that Built-in functions for our Pig Script with out adding any extra code but some times user requirement is not available in that built-in functions at that time user can write some own custom user defined functions called UDF (user defined function).
- For writing UDF's, complete support is provided in Java and limited support is provided in all the remaining languages.
- Using Java, you can write UDF's involving all parts of the processing like data load/store, column transformation, and aggregation.
- Since Apache Pig has been written in Java, the UDF's written using Java language work efficiently compared to other languages.
- In Apache Pig, we also have a Java repository for UDF's named **Piggybank**.
- Using Piggybank, we can access Java UDF's written by other users, and contribute our own UDF's.

(OR)

7. a) Explain the Architecture of Hive. 6M



This component diagram contains different units :

- **User Interface :**

Hive is a data warehouse infrastructure software that can create interaction between user and HDFS. The user interfaces that Hive supports are Hive Web UI, Hive command line, and Hive HD Insight (In Windows server).

- **Meta Store :**

Hive chooses respective database servers to store the schema or Metadata of tables, databases, columns in a table, their data types, and HDFS mapping.

- **HiveQL Process Engine :**

HiveQL is similar to SQL for querying on schema info on the Metastore. It is one of the replacements of traditional approach for MapReduce program. Instead of writing MapReduce program in Java, we can write a query for MapReduce job and process it.

- **Execution Engine :**

The conjunction part of HiveQL process Engine and MapReduce is Hive Execution Engine. Execution engine processes the query and generates results as same as MapReduce results. It uses the flavor of MapReduce.

- **HDFS or HBASE :**

Hadoop distributed file system or HBASE are the data storage techniques to store data into file system.

b) Describe the high-level comparison of SQL and HiveQL. 6M

1. Database Engine Model : Hive is built on Hadoop and it's widely used as a Dataware-house model for distributed large set of data managing and querying. While Oracle is used as RDBMS Model for dealing with small datasets not **Big -data**
2. Verification Design : **In RDBMS**, a table's schema is enforced at data load time, If the data being loaded doesn't conform to the schema, then it is rejected. This design is called **schema on write.** But **Hive** doesn't verify the data when it is loaded, but rather when it is retrieved. This is called **schema on read**
3. Storage Capability : In RDBMS, maximum data size allowed will be in 10's

of **Terabytes** but whereas Hive can 100's **Petabytes** very easily.

4. Working Mechanism : Hive is based on the notion of **Write once, Read many times** but RDBMS is designed for **Read and Write many times**
5. Support : As Hadoop is a batch-oriented system, Hive **doesn't support OLTP** (Online Transaction Processing) but it is **closer to OLAP** (Online Analytical Processing). While Oracle supports both kind of workloads.

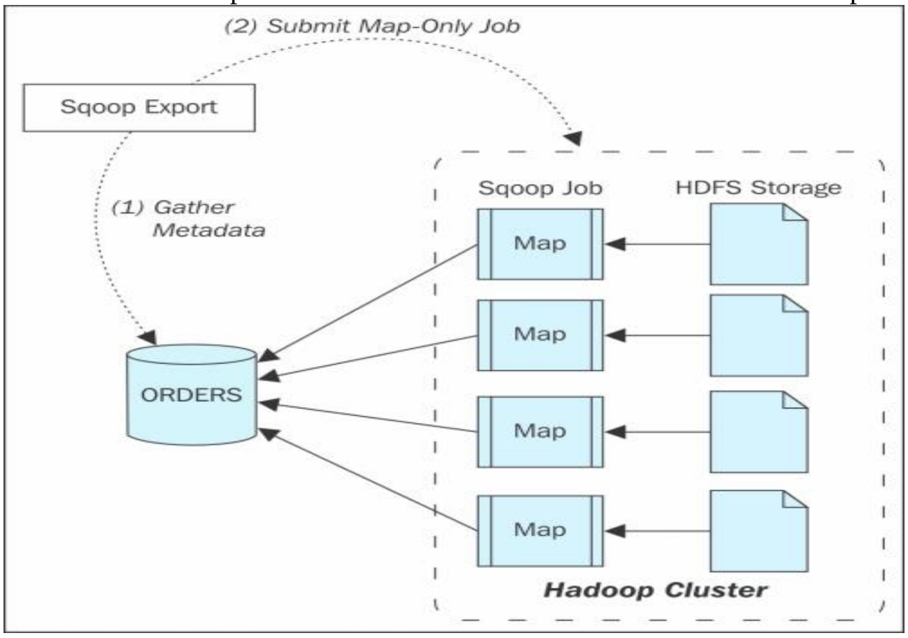
UNIT IV

8. a) Write the procedure for anatomy of spark job run. 12M

(OR)

9. a) Explain the export process of Sqoop. 6M

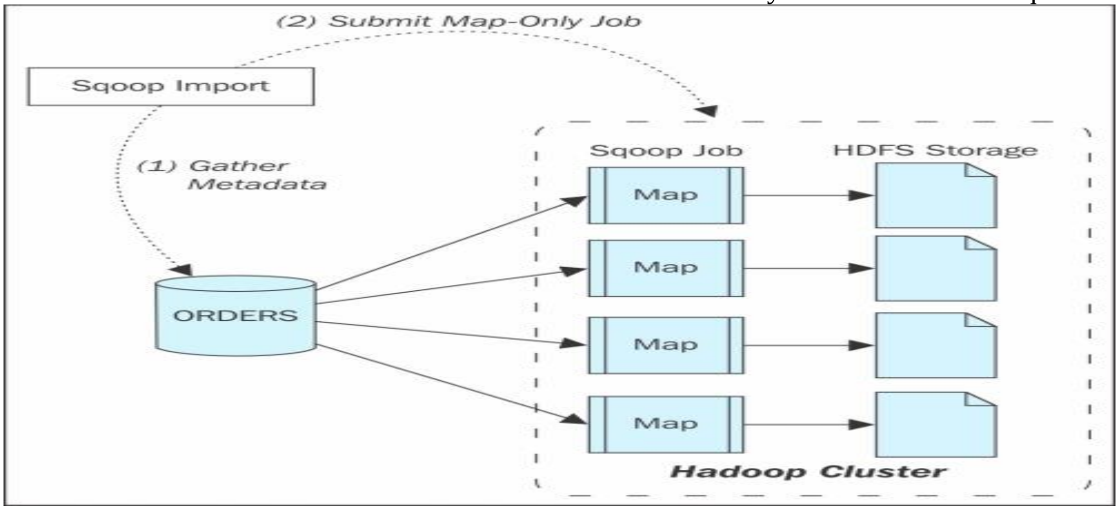
Sqoop Export

- The export tool exports a set of files from HDFS back to an RDBMS.
- The files given as input to Sqoop contain records, which are called as rows in table.
- Those are read and parsed into a set of records and delimited with user-specified delimiter.
- 

b) Explain the import process of Sqoop. 6M

Sqoop Import

- The import tool imports individual tables from RDBMS to HDFS.
- Each row in a table is treated as a record in HDFS.
- All records are stored as text data in text files or as binary data in Avro and Sequence files.


(2) Submit Map-Only Job
Sqoop Import
(1) Gather Metadata
Sqoop Job
HDFS Storage
Map
Map
Map
Map
ORDERS
Hadoop Cluster