MapReduce Job Creation

Step1: Write the Java code for MapReduce program called **OuterClass.java**

Public class OuterClass

{

Mapper Class

Reducer Class

Main method (MapReduce Driver)

}

Step2: Add all the Hadoop jars from Hadoop Home and Hadoop Lib to CLASSPATH variable.

- **export CLASSAPTH=$CLASSPATH:$HADOOP_HOME/*.jars:$HADOOP_HOME/lib/*.jars**

Step3: Compile the java class which contains the Main Method.

- **Javac OuterClass.java**

Step4: Create mapreduce program as a java archive.

- **Jar –cvf jarfilename.jar –C ./ OuterClass*.class**

Step5: Run the program against hadoop script

- **Hadoop jar jarfilename.jar OuterClass <input path> <output path>**

Hadoop Streaming:

Hadoop Streaming is a **Generic API** to run the MapReduce Programs which are written in other programming languages like **shell, python, Perl, R** etc.

There is a **contrib directory in hadoop home** directory. In **contrib**, we have streaming directory in which there is a jar file name called as **hadoop-streaming-x.x.x.jar**

Cmd> hadoop jar contrib/streaming/hadoop-streaming-1.0.3.jar –help

Cmd> Usage: $HADOOP_HOME/bin/hadoop jar $HADOOP_HOME/contrib/streaming/hadoop-streaming-1.03.jar [options]

Options:

-input	<path> DFS input file(s) for the Map step
-output	<path> DFS output directory for the Reduce step
-mapper	<cmd|JavaClassName> the streaming command to run

-combiner <cmd|JavaClassName> the streaming command to run

-reducer <cmd|JavaClassName> the streaming command to run

-file <file> File/dir to be shipped in the Job jar file

-inputformat TextInputFormat (default)|SequenceFileAsTextInputFormat|JavaClassName Optional.

-outputformat TextOutputFormat(default)|JavaClassName Optional.

-partitioner JavaClassName Optional.

-numReduceTasks <num> Optional.

-inputreader <spec> Optional.

-cmdenv <n>=<v> Optional. Pass env.var to streaming commands

-mapdebug <path> Optional. To run this script when a map task fails

-reducedebug <path> Optional. To run this script when a reduce task fails

-io <identifier> Optional.

-verbose

Example: hadoop jar contrib/streaming/hadoop-streaming-1.0.3.jar -input /data/daily -output /streaming -mapper "cat" -reducer "wc -l"

Pig Installation:

Download pig binary distribution from apache pig website.

Extract tar file into a specified directory.

Export HADOOP_HOME=Hadoop Home Directory

Export HADOOP_CONF_DIR= Hadoop Home Directory/conf

Export PIG_HOME= Pig Home Directory

Export PIG_CLASSPATH=$HADOOP_CONF_DIR

Run Pig in local Mode

- **Pig –x local**

Run Pig in Hadoop Mode

- **Pig**

HIVE Installation:

Download the hive binary distribution from the apache hive website.

Extract tar file into a specified directory

Export HIVE_HOME= hive home directory

Export HADOOP_HOME=hadoop home directory

Hive uses Any RDBMS to store its Meta information (databases, tables, indexes etc. info)

Derby: Hive has an embedded database called Apache Derby (java DB). It is useful for development activities. It has some limitations like it allows only one connection; there is no global database management. It uses current directory has the database location.

MySQL: We can store Meta information in Full featured RDBMS like MySQL or Oracle for production deployments where many developers trying to connect Hadoop cluster concurrently.

We have to options to run MySQL server.

- Run MySQL on local machine
- Run MySQL on dedicated machine (remote machine)
    - Here MySQL is running on one machine all other hive clients are trying to connect shared Mata Information which is in MySQL. This communication is happening with protocol called **thrift**. We have to run **hiveserver** for this communication.

Using Apache Derby:

There is no installation for Apache Derby:

Create the data warehouse location in HDFS by default /user/hive/warehouse it is in hive-default.xml.template.

- **Hadoop fs –mkdir /user/hive/warehouse**

Change the warehouse directory permissions as 755

- **Hadoop fs –chmod –R 755 /user/hive/warehouse**

Hive uses hive-default.xml.template for default configurations.

If we are overriding any of the configuration parameters, we need copy the hive-default.xml.template to hive-site.xml and then modify the customized configuration parameters.

We can use our **own directory** by modifying the value of **hive.metastore.warehouse.location** in hive-site.xml.

- **Hadoop fs –mkdir "/my ware house"**
- **Hadoop fs –chmod –R 755 "/my ware house"**

Run hive by using script called *hive* which is in *bin* directory of Hive Home. It works for single connection.

Using Apache MySQL:

Installing and configuring MySQL:

- **run mysql in Linux shell**

If MySQL is not installed, then installing it by using apt-get command

- **sudo apt-get install mysql-server.x.x (provide username as root and password as xxxxxxx)**

Enter into MySQL command line shell to create a user name called **hive** and its **password.**

- **Mysql –u root –p password**
- **Create user "*hive*"@"*hostname*" identified by "*password*"**

Create database name called hivemetastore.

- **Create database *hivemetastore***

Give all the permissions to user hive on database ***hivemetastore***.

- **Grant all on *hivemetastore*.* to '*hive*'@'*hostname*'**

Finally flush the permissions.

> *flush privileges* ;

Here we are modifying all the default database configurations of hive-default.xml.template. So we have to copy hive-default.xml.template to hive-site.xml.

Property	**Default**	**Overriding value**
hive.metastore.local	**true**	**false**
Hive.metastore.uris	**empty**	**thrift://hostname:10000**
javax.jdo.option.ConnectionURL	**jdbc:derby:;databaseName=metastore_db;create=true**	**jdbc:mysql://hostname:3306/hivemetastore**
javax.jdo.option.ConnectionDriverName	**org.apache.derby.jdbc.EmbeddedDriver**	**com.mysql.jdbc.Driver**
javax.jdo.option.ConnectionUserName	**APP**	**hive**

javax.jdo.option.ConnectionPassword	**Mine**	**hive**

Create the data warehouse location in HDFS by default /user/hive/warehouse it is in hive-site.xml.

- **Hadoop fs –mkdir /user/hive/warehouse**

Change the warehouse directory permissions as 755

- **Hadoop fs –chmod –R 755 /user/hive/warehouse**

We can use our **own directory** by modifying the value **hive.metastore.warehouse.location** in hive-site.xml.

- **Hadoop fs –mkdir "/my ware house"**
- **Hadoop fs –chmod –R 755 "/my ware house"**

Change the "**hive.metastore.local**" from **true to false**. The default value is true means metastore is running on local machine. But here we decided to run metastore on remote machine so change it to false.

Change the "**hive.metastore.uris**" value to "**thrift://hostname:10000**". It says hive server is running on particular hostname with port number 10000. It uses thrift as the protocol.

- **Hadoop fs –mkdir "my ware house"**
- **Hadoop fs –chmod –R 755 "my ware house"**

Run the hiveserver

- **Hive –service hiveserver &**

Run the metastore

- **Hive –service metastore &**

Run multiple hive clients from different machines.