**Documentation of Aggregation - Map reduce**

-Prepared by Vignesh.R

If the Mapper output key and value were the same the Reducer output key and value, then it would be enough to just invoke the setOutputKeyClass and setOutputValueClass. The framework would assume the Mapper output key Class and Mapper output value class by invoking the getOutputKeyClass and getOutputValueClass methods. This example, clearly illustrates when it is required to set both sets of functions. Revisit this section when you have read the sections on the Aggregation Mapper and Aggregation Reducer classes.

We set the Mapper and Reducer class and indicate that we will use one Reducer. The last line is optional because the MapReduce framework uses one Reducer for a MapReduce job by default. Chapter 6 explores situations in which this default should be changed.

**Creating .jar file for Aggregation and exporting it:-**

**Step 1:**

Open Eclipse and Click on File > New > Java Project.

**Step 2:**

Give the name ‘Aggregation’ as project name and click ‘Finish’.

**Step 3:**

Right click on ‘Aggregation’ project and select ‘Properties’. Click ‘Java Build Path’ and switch to Libraries tab and click on ‘Add external JARs’.

**Step 4:**

Select all the JAR files in usr >> lib >> hadoop directory to add them.

**Step 5:**

Again add all jar files in usr >> lib >> hadoop >> client directory and press OK.

**Step 6:**

**Right click on src, New >>Class.**

**Step 7:**

Enter the project name as ‘AggregationMRJob’ and click ‘Finish’.

**Step 8:**

Open browser and copy and paste the Java Source code of Aggregationprogram from the link given. The packages are automatically generated by the Eclipse.

**Website link:**<https://github.com/Apress/pro-apache-hadoop/blob/master/prohadoop/src/main/java/org/apress/prohadoop/c5/AggregationMRJob.java>

**Website link:**<https://github.com/Apress/pro-apache-hadoop/blob/master/prohadoop/src/main/java/org/apress/prohadoop/utils/AirlineDataUtils.java>

**Website link:**<https://github.com/Apress/pro-apache-hadoop/blob/master/prohadoop/src/main/java/org/apress/prohadoop/c6/MonthDoWOnlyWritable.java>

**Website link:**<https://github.com/Apress/pro-apache-hadoop/blob/master/prohadoop/src/main/java/org/apress/prohadoop/c6/MonthDoWWritable.java>

**Website link:**<https://github.com/Apress/pro-apache-hadoop/blob/master/prohadoop/src/main/java/org/apress/prohadoop/c6/DelaysWritable.java>

**Step 9:**

Right click on the AggregationMRJob Java project and select Export >> Java >> JAR file. Then click on ‘Next’.

**Step 10:**

Name the JAR file and click ‘Finish’.

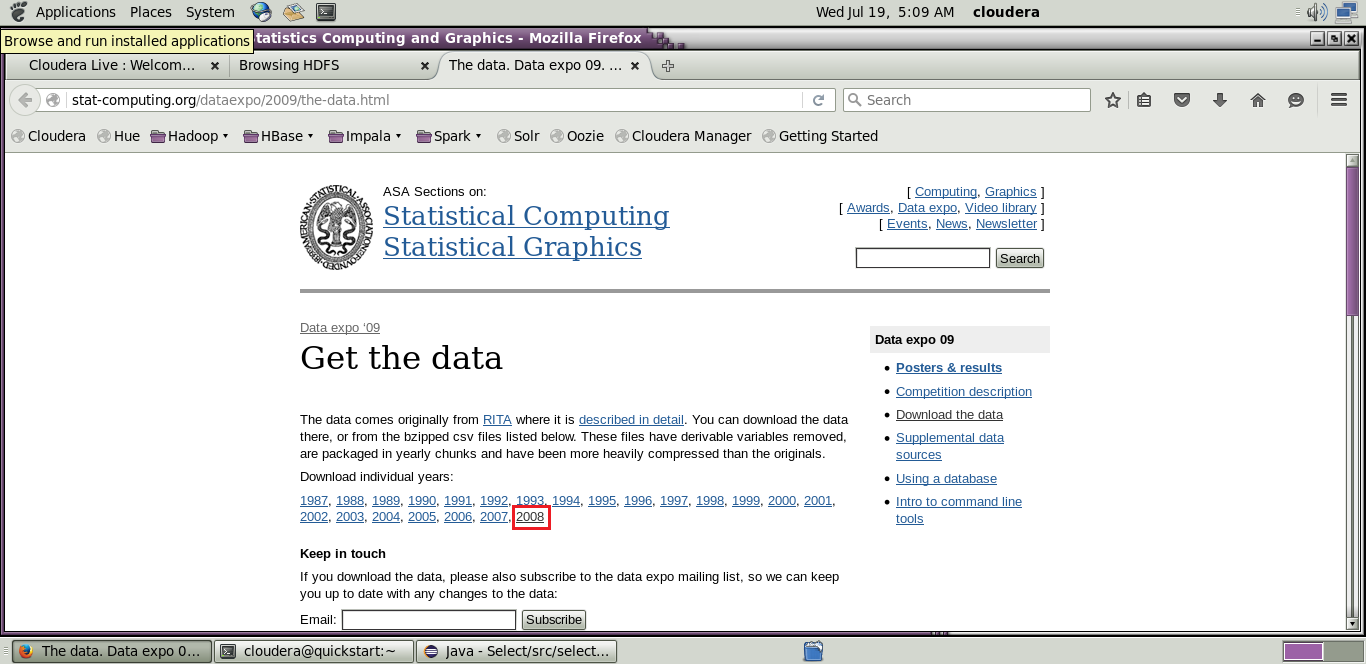
**Creating a file for Mapreduce job to work on:**

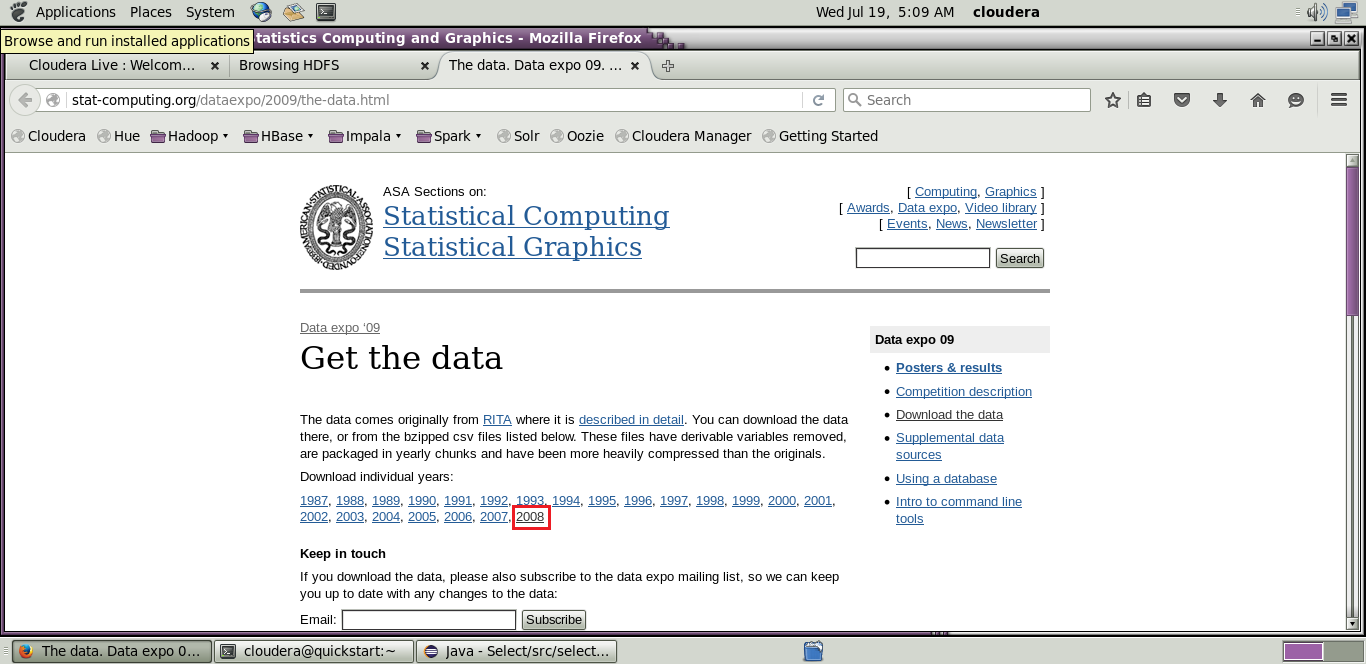
**Step 11:**Make a new Directory using the following command.

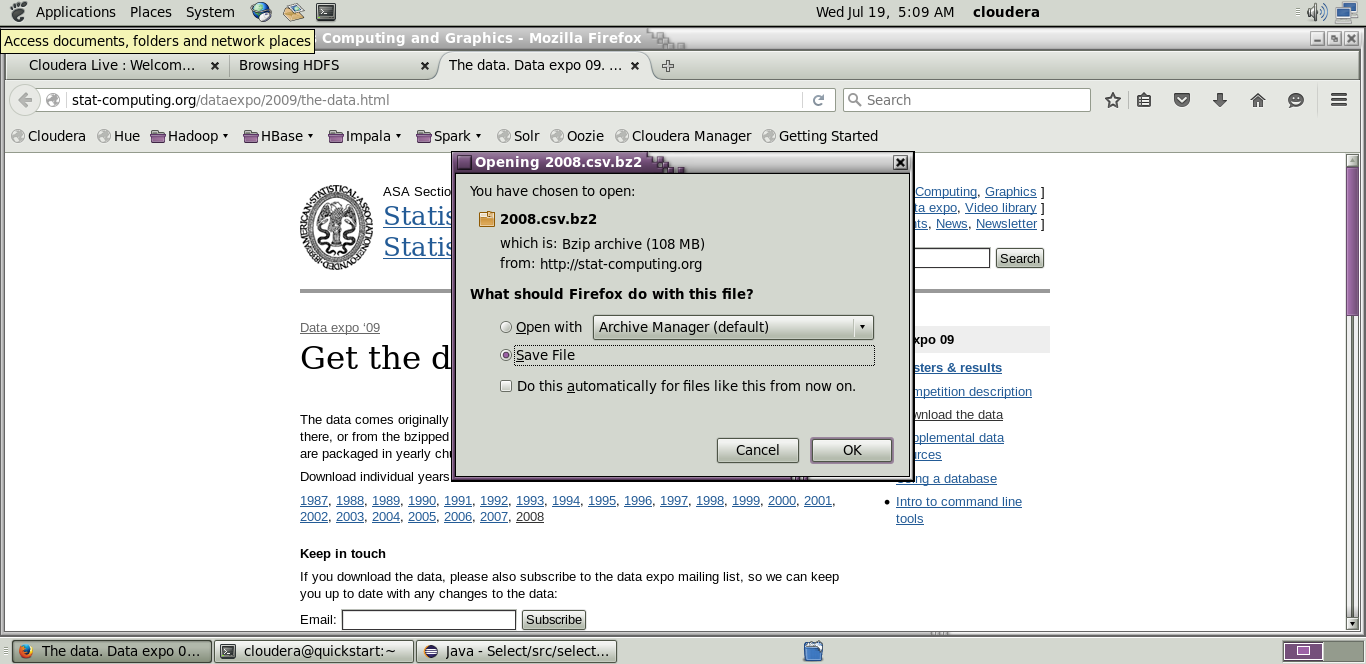
**Code:hadoop fs -mkdir /airline**

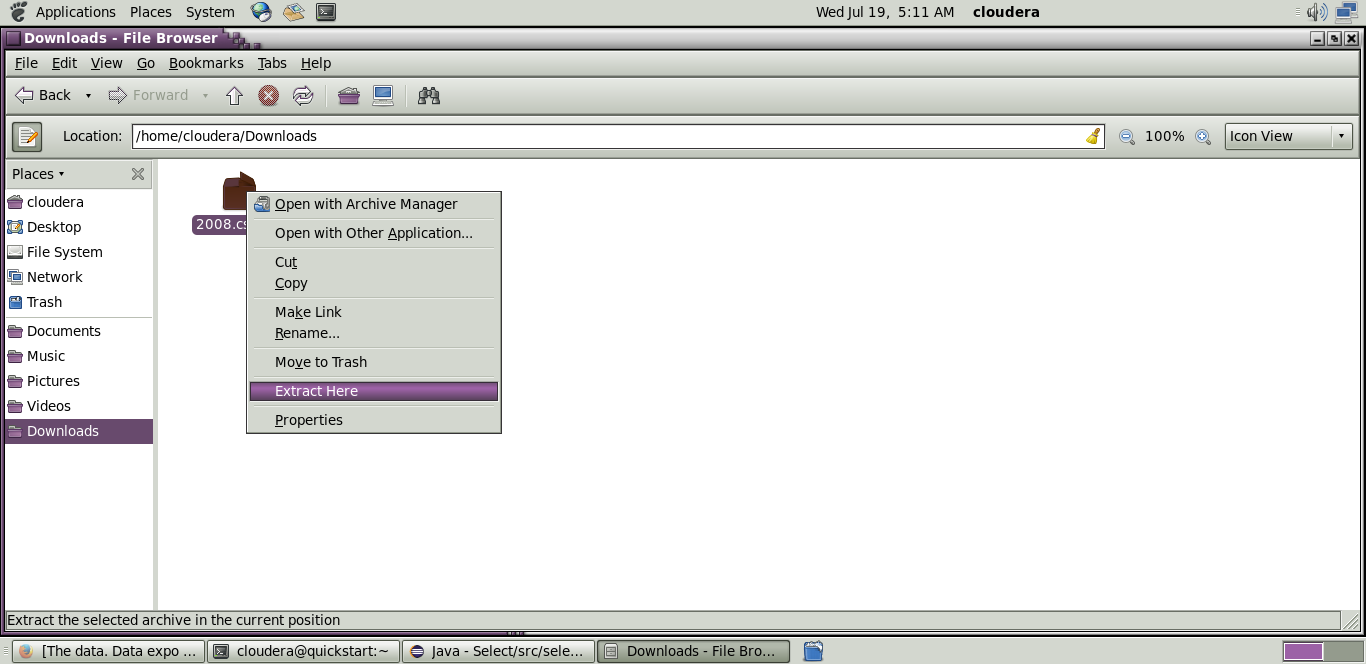
**Step 12:**Use the following link to download the csv file.

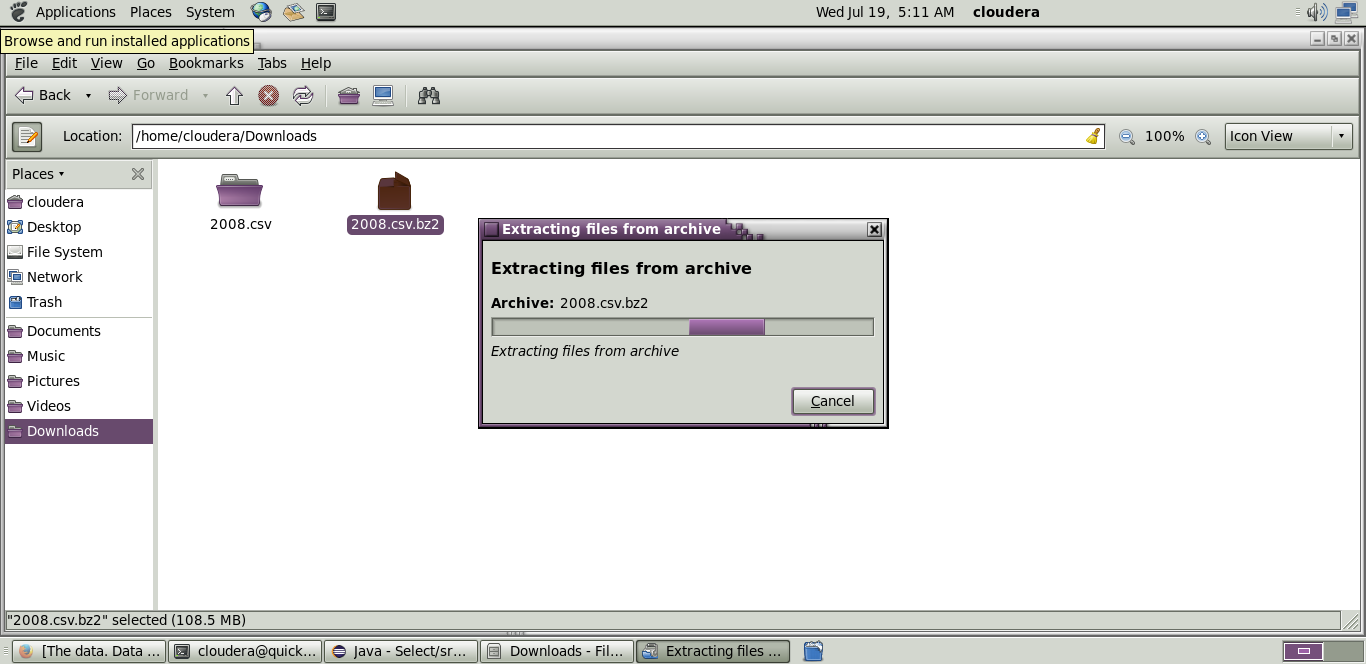
**Website link:**

[http://stat-computing.org/dataexpo/2009/the-data.html](http://stat-computing.org/dataexpo/2009/the-data.html )

[](http://stat-computing.org/dataexpo/2009/the-data.html )







**Step 13:** Copy the downloaded file to the new directory created in HDFS.

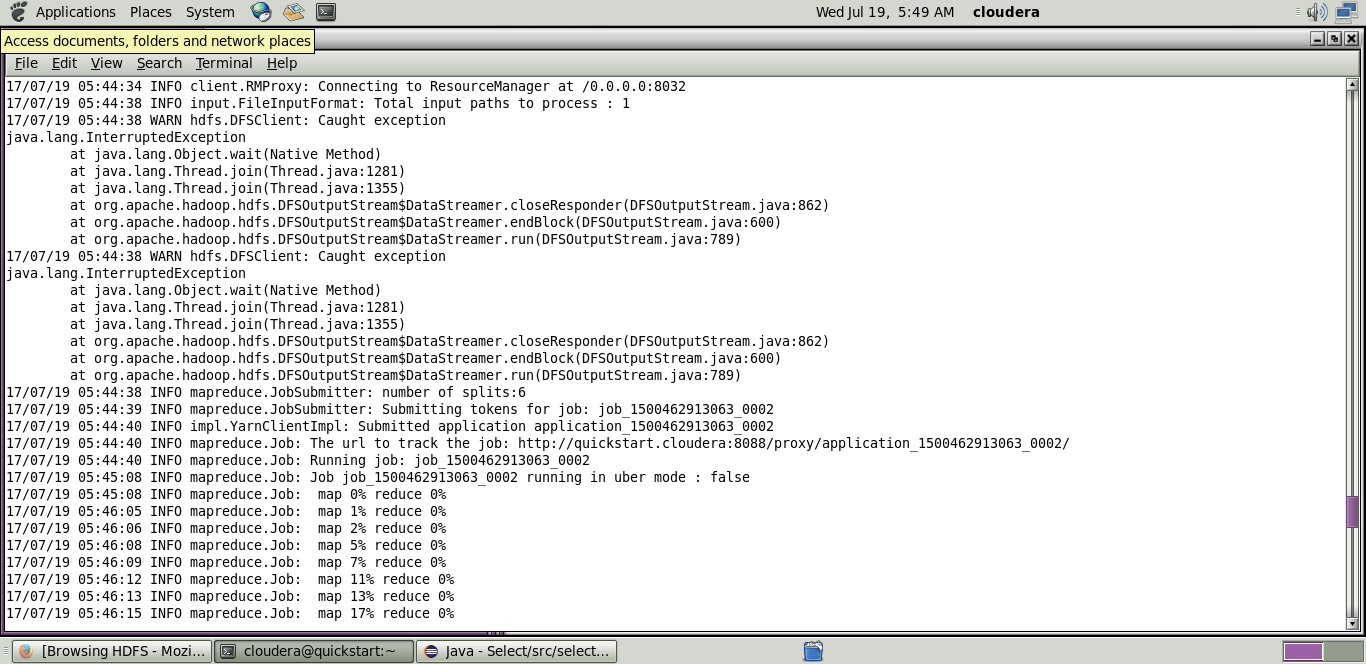
**Code:hadoop fs -put /home/cloudera/Downloads/2008.csv/airline**

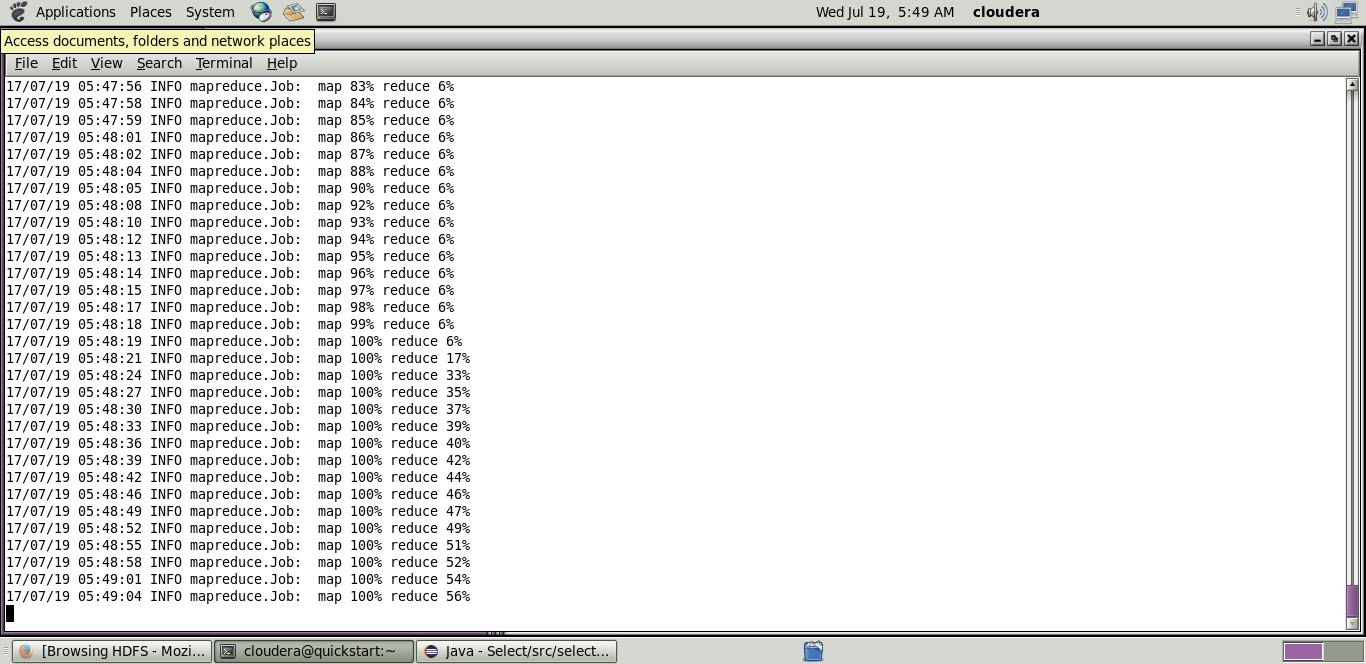
**Initializing mapreduce job:**

**Step 14:**

Initialize the mapreduce job by giving the following command and wait for sometime.

**Code:** hadoop jar Desktop/select.jar select.AggregationMRJob /airline/2008.csv /airlineout/aggrout





Now wait for about 50-70 seconds while the mapreduce job is being performed for the data created earlier.

**Output mapreduce job:**

**Step 15:**

The output directory of the mapreduce program is listed using the following command.

**Code:** hadoop fs -ls/airlineout/aggrout/\*

**Step 16:**

The final output of the mapreduce program is found using the following command.

**Code:**hadoop fs -cat /airlineout/aggrout/\*