Assignment

You are given a data log file of say aircel having subscribers ID,tower id and and Data Downloaded.

Calculate Total Data Downloaded for each customer id using

a) Java Program run in your local system.
b) Map reduce program in hadoop.
c) Sort the result from b) according to Data Downloaded.

Solutions :)

a)JAVA program

This Program Calculates the Downloaded bytes for individual subscriber using only java code developed in eclipse.

```
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.File;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.util.HashMap;
import java.util.Iterator;
import java.util.Map.Entry;

public class File_read {
	public static void main(String args[]) throws IOException{
		//reading a file
		File file = new File("E:/Data_File.txt");
		BufferedReader br = new BufferedReader(new FileReader(file));
		String str,id,bytes_str;
		double bytes,V;
		HashMap<String, Double> hashmap = new HashMap<String, Double>();
		while ((str=br.readLine())!=null && str.length()!=0){
			id=str.substring(15, 26);
			bytes_str=str.substring(87,97);
			bytes= Double.parseDouble(bytes_str);
			V=0;
		if (hashmap.containsKey(id)){
			V=bytes+ Double.parseDouble(hashmap.get(id).toString());
			hashmap.remove(id);
			hashmap.put(id, V);}
		else{
			hashmap.put(id, bytes);}
			}
		br.close();
		File filo = new File("E:/file_out.txt");
		BufferedWriter bw=new BufferedWriter(new FileWriter(filo));

		Iterator<Entry<String, Double>> itr = hashmap.entrySet().iterator();
		while (itr.hasNext()){
				Entry<String, Double> mp = itr.next();
				str = (mp.getKey().toString());

				V =Double.parseDouble(mp.getValue().toString());

				bw.write(str+"___________"+String.format("%f", V));
				bw.newLine();
		}
		bw.close();}}
```

"Data_File" required log file for program

Using a hashmap for storing id and their corresponding data

If ID already present,Data downloaded is added to the previous data. If ID is absent ,new entry is created in the HashMap

Readings are taken out of HashMap using an Iterator in the form pf Map-Entry Set.

b) Map Reduce Program

This Program Adds Total bytes downloaded for each individual customer-id. Data is extracted from a text file "Data_File.txt"
Make sure you have Data file in your HDFS system.

Mapper Class

```
import java.io.IOException;
import org.apache.hadoop.io.DoubleWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.MapReduceBase;
import org.apache.hadoop.mapred.Mapper;
import org.apache.hadoop.mapred.OutputCollector;
import org.apache.hadoop.mapred.Reporter;

public class aircel_mapper extends MapReduceBase implements Mapper<LongWritable,Text,Text,DoubleWritable> {
        public static final double MISSING=0;
        public void map(LongWritable key, Text value,OutputCollector<Text,DoubleWritable> output, Reporter reporter) throws
IOException {
                String line = value.toString();
                String subID="Dummy",subId;
                if(line.isEmpty()){
                        output.collect(new Text (subID), new DoubleWritable(1));
                }
                else{
                        subId = line.substring(15,26);
                        Double bytes = Double.parseDouble(line.substring(87,97));
                        if(bytes==null){
                                bytes=MISSING;
                        }
                        output.collect(new Text(subId), new DoubleWritable(bytes));
                }}}
```

Hadoop map-reduce has its own classes for data i/o ,namely LongWritable,IntWritable,Text,SequenceFileInputFormat,etc

Signature of mapper class
Mapper<K1,V1,K2,V2>
K1-Input Key
V1-Input Value
K2-Output Key
V2-Output Value

1) Variable "value" of Text type is converted to string.Required values of id and data_download are taken out from value using substring().

2) Here each "value" is complete line in Data_File.txt.

3) Finally output of map function is fed to Output Collector "output" with final key(Text) and value(DoubleWritable)

Reducer Class

```
import java.io.IOException;
import java.util.Iterator;
import org.apache.hadoop.io.DoubleWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.MapReduceBase;
import org.apache.hadoop.mapred.OutputCollector;
import org.apache.hadoop.mapred.Reducer;
import org.apache.hadoop.mapred.Reporter;

public class aircel_reducer extends MapReduceBase implements Reducer<Text,DoubleWritable,Text,DoubleWritable>{
	public void reduce(Text key, Iterator<DoubleWritable> value,OutputCollector<Text,DoubleWritable> output, Reporter
reporter) throws IOException{
		while (value.hasNext()) {
			total_bytes += value.next().get();}
		output.collect(key,new DoubleWritable(total_bytes));}}
```

Reducer(K1,V1,K2,V2)
K1-Key input(Text)
V1-Value input(Double Writable)
K2-Key Output(Text)
V2-Value Output(DoubleWritable)

Runner Class

```
import java.io.IOException;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.DoubleWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.FileInputFormat;
import org.apache.hadoop.mapred.FileOutputFormat;
import org.apache.hadoop.mapred.JobClient;
import org.apache.hadoop.mapred.JobConf;
import org.apache.hadoop.mapred.TextInputFormat;
import org.apache.hadoop.mapred.TextOutputFormat;

public class aircel_runner {
	public static void main(String[] args) throws IOException {

		JobConf conf=new JobConf(aircel_runner.class);
		conf.setJobName("Subscriber_ID");

		conf.setOutputKeyClass(Text.class);
		conf.setOutputValueClass(DoubleWritable.class);

		conf.setMapperClass(aircel_mapper.class);
		conf.setCombinerClass(aircel_reducer.class);
		conf.setReducerClass(aircel_reducer.class);

		conf.setInputFormat(TextInputFormat.class);
		conf.setOutputFormat(TextOutputFormat.class);

		FileInputFormat.setInputPaths(conf, new Path(args[0]));
		FileOutputFormat.setOutputPath(conf, new Path(args[1]));

		JobClient.runJob(conf);}}
```

Runner Class

1) Initialise a JobConf class mentioning runner class in the constructor.
2) Set Output Key and Output Value Class. In our case we have output key as Text(Subscriber's id) and output value as DoubleWritable(Downloaded Bytes).
3) Mapper,combiner and reducer class are set.
4) Input and Output Format are set. In our case our input data and output result both are in Text Format.
5) Paths are mentioned through command line arguments.

Running the JAR Program in VMware

1> **Make sure you have the data file in your HDFS system.**

2> **Output Directory should NOT exist for map reduce to run.**

```
cloudera@cloudera-vm:~/Desktop$
cloudera@cloudera-vm:~/Desktop$
cloudera@cloudera-vm:~/Desktop$
cloudera@cloudera-vm:~/Desktop$
cloudera@cloudera-vm:~/Desktop$
cloudera@cloudera-vm:~/Desktop$
cloudera@cloudera-vm:~/Desktop$
cloudera@cloudera-vm:~/Desktop$ hadoop jar aircel.jar aircel_runner /user/cloudera/data_file.txt /user/cloudera/Subscriber
```

JAR file

HDFS path to data file

Output Directory

Runner Class

3> **After the program run is completed, open the completed file by**

Hadoop fs –cat /user/cloudera/Subscriber/part-00000

4> **Compare the results with the one obtained by running only java program.**

C) Sort Program on b) output

Objective of this exercise is to take the input from previous Subcribers data download program and sort the results according to downloaded data.

Mapreduce program always gives the sorted result according to "Key". In previous program the key was ID hence the results that we will get will be already sorted according to "ID". In this program we only need to change the key-value pairs and hence mapper will give sorted list as per new key "Downloaded Data".

No reduce program is required.

We cannot use Input Format as Text because for this input format, Map takes Key as offset from start and Value as the entire line. Hence we will be using **SequenceFileInputFormat** This kind of file has binary key/value pairs.

Changes to be done in previous program code :

```
import org.apache.hadoop.mapred.TextInputFormat;
import org.apache.hadoop.mapred.SequenceFileOutputFormat;


public class aircel_runner {

    public static void main(String[] args) throws IOException {
        // TODO Auto-generated method stub
        JobConf conf=new JobConf(aircel_runner.class);
        conf.setJobName("Subscriber_ID");

        conf.setOutputKeyClass(Text.class);
        conf.setOutputValueClass(DoubleWritable.class);

        conf.setMapperClass(aircel_mapper.class);
        conf.setCombinerClass(aircel_reducer.class);
        conf.setReducerClass(aircel_reducer.class);

        conf.setInputFormat(TextInputFormat.class);
        conf.setOutputFormat(SequenceFileOutputFormat.class);

        FileInputFormat.setInputPaths(conf, new Path(args[0]));
        FileOutputFormat.setOutputPath(conf, new Path(args[1]));

        JobClient.runJob(conf);
    }
```

Mapper Class

```
import java.io.IOException;
import org.apache.hadoop.io.DoubleWritable;

import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.MapReduceBase;
import org.apache.hadoop.mapred.Mapper;
import org.apache.hadoop.mapred.OutputCollector;
import org.apache.hadoop.mapred.Reporter;

public class sortmap extends MapReduceBase implements Mapper<Text,DoubleWritable,DoubleWritable,Text> {
        public void map(Text key, DoubleWritable value,OutputCollector<DoubleWritable,Text> output, Reporter reporter) throws
IOException {
                output.collect(value,key);
}}
```

Key-Value pairs interchanged.

Reducer Class

```
import java.io.IOException;
import java.util.Iterator;
import org.apache.hadoop.io.DoubleWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.MapReduceBase;
import org.apache.hadoop.mapred.OutputCollector;
import org.apache.hadoop.mapred.Reducer;
import org.apache.hadoop.mapred.Reporter;

public class sortreducer extends MapReduceBase implements Reducer<DoubleWritable,Text,DoubleWritable,Text> {
	public void reduce(DoubleWritable key, Iterator<Text> value,OutputCollector<DoubleWritable,Text> output, Reporter
reporter) throws IOException {
		output.collect(key,value.next());}}
```

No need of a reducer class in this case. Just for simplicity this reducer class emits out exactly same things it got as an input.

Runner Class

```
import java.io.IOException;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.DoubleWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.FileInputFormat;
import org.apache.hadoop.mapred.FileOutputFormat;
import org.apache.hadoop.mapred.JobClient;
import org.apache.hadoop.mapred.JobConf;
import org.apache.hadoop.mapred.SequenceFileInputFormat;
import org.apache.hadoop.mapred.TextOutputFormat;

public class sortrunner {
	public static void main(String[] args) throws IOException{
		JobConf conf = new JobConf(sortrunner.class);
		conf.setJobName("Sort");

		conf.setOutputValueClass(Text.class);
		conf.setOutputKeyClass(DoubleWritable.class);

		conf.setMapperClass(sortmap.class);
		conf.setReducerClass(sortreducer.class);

		conf.setInputFormat(SequenceFileInputFormat.class);
		conf.setOutputFormat(TextOutputFormat.class);

		FileInputFormat.setInputPaths(conf,new Path(args[0]));
		FileOutputFormat.setOutputPath(conf,new Path(args[1]));

		JobClient.runJob(conf);
	}}
```

Input Format Different than he previous code.

Running the File