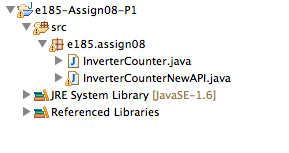
**HU Extension Assignment 08 E185 Big Data Analytics**

**Handed out: 04/05/2013 Due by 5:30PM on Friday, 04/12/2013**

**Problem 1) Rewrite attached InverterCounter.java class in new MapReduce API. Demonstrate that new class generates the same values as the class written in old API.**

**While testing new class demonstrate that you can output intermediate results only and not run any reducers.**

For this problem we created a new project in Eclipse where we created a package that has both the original InverterCounter.java and a new InverterCounterNewAPI.java with the implementation in the new API. All the necessary JAR libraries were added as in a previous exercise.



Our final code of the InverterCounterNewAPI.java is found below and in an attached file for Problem 1. The main changes are highlighted in bold.

package e185.assign08;

import java.io.IOException;

**import java.lang.InterruptedException;**

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.conf.Configured;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.io.IntWritable;

**import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;**

**import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;**

**import org.apache.hadoop.mapreduce.Job;**

**import org.apache.hadoop.mapreduce.lib.input.KeyValueTextInputFormat;**

**import org.apache.hadoop.mapreduce.Mapper;**

**import org.apache.hadoop.mapreduce.Reducer;**

**import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;**

import org.apache.hadoop.util.Tool;

import org.apache.hadoop.util.ToolRunner;

public class **InverterCounterNewAPI** extends Configured implements Tool {

**public static class MapClass extends Mapper<Text, Text, Text, Text> {**

**public void map(Text key, Text value,**

**Context context) throws IOException, InterruptedException {**

**context.write(value, key);**

**}**

**}**

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

**public void reduce(Text key, Iterable<Text> values,**

**Context context) throws IOException, InterruptedException {**

**int count = 0;**

**for (Text value : values) {**

**count++;**

**}**

**context.write(key, new IntWritable(count));**

**}**

**}**

public int run(String[] args) throws Exception {

Configuration conf = getConf();

**conf.set("mapreduce.input.keyvaluelinerecordreader.key.value.separator", ",");**

**Job job = new Job(conf);**

**job.setJarByClass(InverterCounterNewAPI.class);**

**job.setJobName("Inverter Counter");**

Path in = new Path(args[0]);

Path out = new Path(args[1]);

FileInputFormat.setInputPaths(job, in);

FileOutputFormat.setOutputPath(job, out);

job.setMapperClass(MapClass.class);

job.setReducerClass(Reduce.class);

job.setInputFormatClass(KeyValueTextInputFormat.class);

job.setOutputFormatClass(TextOutputFormat.class);

job.setOutputKeyClass(Text.class);

job.setOutputValueClass(Text.class);

**System.exit(job.waitForCompletion(true)?0:1);**

return 0;

}

public static void main(String[] args) throws Exception {

int res = ToolRunner.run(new Configuration(), new **InverterCounterNewAPI**(), args);

System.exit(res);

}

}

After the classes are compiled, we generate the aggregated JAR file and upload it to our virtual machine, accompanied by the input file.

**aperture-2:e185-Assign08-P1 alexcp$ ls**

Assign08-P1.jar bin cite75\_99.txt src

**aperture-2:e185-Assign08-P1 alexcp$ scp Assign08-P1.jar cite75\_99.txt cloudera@192.168.2.138:**

cloudera@192.168.2.138's password:

Assign08-P1.jar 100% 7312 7.1KB/s 00:00

cite75\_99.txt 100% 252MB 21.0MB/s 00:12

We ssh to it and setup the input file on our HDFS path. Before uploading it we edited the file to remove the header("CITING","CITED"):

**aperture-2:e185-Assign08-P1 alexcp$ ssh cloudera@192.168.2.138**

cloudera@192.168.2.138's password:

Last login: Tue Apr 9 19:37:21 2013

**[cloudera@centos-e185 ~]$ ls**

Assign08-P1.jar cite75\_99.txt hadoop-jar.tar.gz hadoop-mapreduce-jar.tar.gz

**[cloudera@centos-e185 ~]$ hadoop fs -mkdir Assign08**

**[cloudera@centos-e185 ~]$ hadoop fs -mkdir Assign08/input**

**[cloudera@centos-e185 ~]$ hadoop fs -copyFromLocal cite75\_99.txt Assign08/input**

**[cloudera@centos-e185 ~]$ hadoop fs -ls Assign08/input**

Found 1 items

-rw-r--r-- 1 cloudera supergroup 264075431 2013-04-09 23:43 Assign08/input/cite75\_99.txt

Before we begin, we need to patch out MapReduce libraries, because of a know bug that was detected by a fellow student. We replace the hadoop-mapreduce-client-core-2.0.0-cdh4.2.0.jar as directed on <https://issues.cloudera.org/browse/DISTRO-461>

**aperture-2:e185-Assign08-P1 alexcp$ scp hadoop-mapreduce-client-core-2.0.0-cdh4.2.0.jar cloudera@192.168.2.138:**

cloudera@192.168.2.138's password:

hadoop-mapreduce-client-core-2.0.0-cdh4.2.0.jar 100% 1406KB 1.4MB/s 00:00

**aperture-2:e185-Assign08-P1 alexcp$ ssh cloudera@192.168.2.138**

cloudera@192.168.2.138's password:

Last login: Tue Apr 9 23:40:00 2013 from 192.168.2.1

[cloudera@centos-e185 ~]$ **sudo service hadoop-mapreduce-historyserver stop**

[sudo] password for cloudera:

Stopping Hadoop historyserver: [ OK ]

stopping historyserver

[cloudera@centos-e185 ~]$ **sudo service hadoop-yarn-nodemanager stop**

Stopping Hadoop nodemanager: [ OK ]

stopping nodemanager

[cloudera@centos-e185 ~]$ **sudo service hadoop-yarn-resourcemanager stop**

Stopping Hadoop resourcemanager: [ OK ]

stopping resourcemanager

**[cloudera@centos-e185 ~]$ sudo mv hadoop-mapreduce-client-core-2.0.0-cdh4.2.0.jar /usr/lib/hadoop-mapreduce/**

**[cloudera@centos-e185 ~]$ sudo service hadoop-yarn-resourcemanager start**

Starting Hadoop resourcemanager: [ OK ]

starting resourcemanager, logging to /var/log/hadoop-yarn/yarn-yarn-resourcemanager-centos-e185.aperture.out

**[cloudera@centos-e185 ~]$ sudo service hadoop-yarn-nodemanager start**

Starting Hadoop nodemanager: [ OK ]

starting nodemanager, logging to /var/log/hadoop-yarn/yarn-yarn-nodemanager-centos-e185.aperture.out

**[cloudera@centos-e185 ~]$ sudo service hadoop-mapreduce-historyserver start**

Starting Hadoop historyserver: [ OK ]

starting historyserver, logging to /var/log/hadoop-mapreduce/yarn-mapred-historyserver-centos-e185.aperture.out

Now, we start by running the original InverterCounter class to check its result:

**[cloudera@centos-e185 ~]$ hadoop jar Assign08-P1.jar e185.assign08.InverterCounter Assign08/input Assign08/outputP1-old**

(...)

File Input Format Counters

Bytes Read=264087719

File Output Format Counters

Bytes Written=32858613

**[cloudera@centos-e185 ~]$ hadoop fs -cat Assign08/outputP1-old/part-00000 | head -15**

1 2

10000 1

100000 1

1000006 1

1000007 1

1000011 1

1000017 1

1000026 1

1000033 2

1000043 1

1000044 2

1000045 1

1000046 2

1000049 1

1000051 1

cat: Unable to write to output stream.

**[cloudera@centos-e185 ~]$ hadoop fs -cat Assign08/outputP1-old/part-00000 | tail -15**

999941 1

999945 2

999949 1

999951 2

999957 1

999961 9

999965 1

999968 1

999971 1

999972 1

999973 2

999974 3

999977 1

999978 1

999983 3

Now, we run the new one and check if the results are the same.

[cloudera@centos-e185 ~]$ hadoop jar Assign08-P1.jar e185.assign08.InverterCounterNewAPI Assign08/input Assign08/outputP1-new

(...)

File Input Format Counters

Bytes Read=264087702

File Output Format Counters

Bytes Written=32858603

**[cloudera@centos-e185 ~]$ hadoop fs -cat Assign08/outputP1-new/part-r-00000 | head -15**

1 2

10000 1

100000 1

1000006 1

1000007 1

1000011 1

1000017 1

1000026 1

1000033 2

1000043 1

1000044 2

1000045 1

1000046 2

1000049 1

1000051 1

cat: Unable to write to output stream.

**[cloudera@centos-e185 ~]$ hadoop fs -cat Assign08/outputP1-new/part-r-00000 | tail -15**

999941 1

999945 2

999949 1

999951 2

999957 1

999961 9

999965 1

999968 1

999971 1

999972 1

999973 2

999974 3

999977 1

999978 1

999983 3

And they are the same, as expected.

For some additional testing, let's run the Job without any reducers so we can have a peek at the intermediate results from the Map function:

**[cloudera@centos-e185 ~]$ hadoop jar Assign08-P1.jar e185.assign08.InverterCounterNewAPI -D mapred.reduce.tasks=0 Assign08/input Assign08/outputP1-new-map**

File Input Format Counters

Bytes Read=264087702

File Output Format Counters

Bytes Written=264075414

**[cloudera@centos-e185 ~]$ hadoop fs -ls Assign08/outputP1-new-map/**

Found 5 items

-rw-r--r-- 1 cloudera supergroup 0 2013-04-10 00:25 Assign08/outputP1-new-map/\_SUCCESS

-rw-r--r-- 1 cloudera supergroup 67108871 2013-04-10 00:25 Assign08/outputP1-new-map/part-m-00000

-rw-r--r-- 1 cloudera supergroup 67108873 2013-04-10 00:25 Assign08/outputP1-new-map/part-m-00001

-rw-r--r-- 1 cloudera supergroup 67108849 2013-04-10 00:25 Assign08/outputP1-new-map/part-m-00002

-rw-r--r-- 1 cloudera supergroup 62748821 2013-04-10 00:25 Assign08/outputP1-new-map/part-m-00003

**[cloudera@centos-e185 ~]$ hadoop fs -cat Assign08/outputP1-new-map/part-m-00000 | head -15**

956203 3858241

1324234 3858241

3398406 3858241

3557384 3858241

3634889 3858241

1515701 3858242

3319261 3858242

3668705 3858242

3707004 3858242

2949611 3858243

3146465 3858243

3156927 3858243

3221341 3858243

3574238 3858243

3681785 3858243

cat: Unable to write to output stream.

**[cloudera@centos-e185 ~]$ hadoop fs -cat Assign08/outputP1-new-map/part-m-00000 | tail -15**

4306925 4604160

4354895 4604160

4465538 4604160

3247576 4604161

3936329 4604161

4019248 4604161

4269653 4604161

4325182 4604161

4025411 4604162

4307180 4604162

4333965 4604162

4361600 4604162

4374011 4604162

4377438 4604162

4407851 4604162

As expected we get the inverted Citations list, that then would be used by the Reducer to count how many citations a patent has.

**Problem 2) Imagine that you are a Linux person and you have not a single machine with an Eclipse. Your boss is adamnt and wants a graphical histogram of citation counts you are calculating with a class like the attached CitationHistogram.java class. Write a new Hadoop program that will print a true (graphical) histogram of log10(citation\_count) values using a string of asterixes (\*) to indicate the value. You histogram will look approximately like this:**

**1-20 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**21-40 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**41-60 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**. . . .**

**760-780 \*\*\*\*\*\***

**Use buckets of size 20.**

As a Linux person with no Eclipse in sight, I need access my Linux box to do some Java coding! First of all, I upload the CitationHistogram.java and compile it in order to be able to generate the input file for this Problem.

**aperture-2:Assignmnet08\_materials alexcp$ ls**

CitationHistogram.java Inverter.java e185\_Assign08\_MapReduce APIs.docx

CitationHistogramNewApi.java InverterCounter.java

**aperture-2:Assignmnet08\_materials alexcp$ scp CitationHistogramNewApi.java cloudera@192.168.2.138:**

cloudera@192.168.2.138's password:

CitationHistogramNewApi.java 100% 3764 3.7KB/s 00:00

Compiling the class:

**[cloudera@centos-e185 ~]$ ls**

Assign08 CitationHistogramNewApi.java cite75\_99.txt hadoop-jar.tar.gz hadoop-mapreduce-jar.tar.gz

**[cloudera@centos-e185 ~]$ mkdir Assign08/P2**

**[cloudera@centos-e185 ~]$ mv CitationHistogramNewApi.java Assign08/P2/**

**[cloudera@centos-e185 ~]$ cd Assign08/P2/**

**[cloudera@centos-e185 P2]$ mkdir class**

**[cloudera@centos-e185 P2]$ javac -classpath `hadoop classpath` -Xlint:deprecation -d class CitationHistogramNewApi.java**

CitationHistogramNewApi.java:56: warning: [deprecation] Job(org.apache.hadoop.conf.Configuration,java.lang.String) in org.apache.hadoop.mapreduce.Job has been deprecated

Job job = new Job(conf, "CitationHistogram");

^

1 warning

**[cloudera@centos-e185 P2]$ ls -l class/edu/hu/bgd/**

total 12

-rw-rw-r--. 1 cloudera cloudera 2241 Apr 10 00:40 CitationHistogramNewApi.class

-rw-rw-r--. 1 cloudera cloudera 1703 Apr 10 00:40 CitationHistogramNewApi$MapClass.class

-rw-rw-r--. 1 cloudera cloudera 1698 Apr 10 00:40 CitationHistogramNewApi$Reduce.class

Creating the JAR file:

**[cloudera@centos-e185 P2]$ cd class/**

**[cloudera@centos-e185 class]$ jar -cvf CitationHistogramNewApi.jar edu/\***

added manifest

adding: edu/hu/(in = 0) (out= 0)(stored 0%)

adding: edu/hu/bgd/(in = 0) (out= 0)(stored 0%)

adding: edu/hu/bgd/CitationHistogramNewApi$MapClass.class(in = 1703) (out= 688)(deflated 59%)

adding: edu/hu/bgd/CitationHistogramNewApi.class(in = 2241) (out= 1041)(deflated 53%)

adding: edu/hu/bgd/CitationHistogramNewApi$Reduce.class(in = 1698) (out= 686)(deflated 59%)

**[cloudera@centos-e185 class]$ ls**

CitationHistogramNewApi.jar edu

Finally, we run it against our Citation Count output from the last problem

**[cloudera@centos-e185 class]$ hadoop jar CitationHistogramNewApi.jar edu.hu.bgd.CitationHistogramNewApi Assign08/outputP1-new Assign08/input-P2**

(...)

File Input Format Counters

Bytes Read=32858603

File Output Format Counters

Bytes Written=1721

**[cloudera@centos-e185 class]$ hadoop fs -cat Assign08/input-P2/part-r-00000 | head -10**

1 921127

2 552246

3 380319

4 278438

5 210814

6 163149

7 127941

8 102155

9 82126

10 66634

**[cloudera@centos-e185 class]$ hadoop fs -cat Assign08/input-P2/part-r-00000 | tail -10**

411 1

605 1

613 1

631 1

633 1

654 1

658 1

678 1

716 1

779 1

The output is the same as shown in the lecture slides. Now we can start working in Problem 2.

We create the a CitationGraph.java file to perform the calculation. The file is added to the directory of this problem, but we would like to highlight the Mapper and Reducer functions:

For the Mapper class, we calculate what bucket the citation group falls into by performing an integer division and storing the output as the first Integer of the corresponding bucket. By using IntWritable instead of Text on the Mapper output, we are able to have the rows in the proper order on the Reducer function without any additional computation.

25 public static class MapClass extends Mapper<Text, Text, IntWritable, IntWritable> {

26 public static final int bucketSize = 20;

27 private IntWritable citationBucket = new IntWritable();

28 private IntWritable citationCount = new IntWritable();

29 public void map(Text key, Text value,

30 Context context) throws IOException, InterruptedException {

31 int bucket = Integer.parseInt(key.toString()) / bucketSize; // Integer division

32 int startNum = bucket\* (bucketSize) + 1;

33

34 citationBucket.set(startNum);

35 citationCount.set(Integer.parseInt(value.toString()));

36

37 context.write(citationBucket, citationCount);

38 }

39 }

Finally, for the Reducer, we create the text for the bucket range based on the first integer we get and the bar is formatted based on the log10 of the total number of rows that are a part of the bucket.

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

41 public static final int bucketSize = 20;

42 private Text bar = new Text();

43 private Text range = new Text();

44 public void reduce(IntWritable key, Iterable<IntWritable> values,

45 Context context) throws IOException, InterruptedException {

46

47 // Creating the range string

48 int startBucket = key.get();

49 range.set(startBucket + "-" + (startBucket + bucketSize - 1));

50

51 // Calculating the number of stars

52 int count = 0;

53 for (IntWritable value : values) {

54 count+= value.get();

55 }

56 double logCount = Math.log10( (double) count);

57 int stars = (int) Math.ceil(logCount);

58 String histBars = new String(new char[stars]).replace("\0", "\*");

59 bar.set(histBars);

60

61 context.write(range, bar);

62 }

63 }

We compile it and run against out data that was prepared on the beginning of this problem:

**[cloudera@centos-e185 P2]$ ls**

CitationGraph.java CitationHistogramNewApi.java class

**[cloudera@centos-e185 P2]$ javac -classpath `hadoop classpath` -Xlint:deprecation -d class CitationGraph.java**

CitationGraph.java:66: warning: [deprecation] Job(org.apache.hadoop.conf.Configuration) in org.apache.hadoop.mapreduce.Job has been deprecated

Job job = new Job(conf);

^

1 warning

**[cloudera@centos-e185 P2]$ cd class/**

**[cloudera@centos-e185 class]$ jar -cvf CitationGraph.jar e185/\***

added manifest

adding: e185/assign08/(in = 0) (out= 0)(stored 0%)

adding: e185/assign08/CitationGraph$MapClass.class(in = 1731) (out= 725)(deflated 58%)

adding: e185/assign08/CitationGraph.class(in = 2279) (out= 1072)(deflated 52%)

adding: e185/assign08/CitationGraph$Reduce.class(in = 2325) (out= 1039)(deflated 55%)

**[cloudera@centos-e185 class]$ hadoop jar CitationGraph.jar e185.assign08.CitationGraph Assign08/input-P2 Assign08/output-P2**

(...)

File Input Format Counters

Bytes Read=1721

File Output Format Counters

Bytes Written=275

**[cloudera@centos-e185 class]$ hadoop fs -cat Assign08/output-P2/part-r-00000**

1-20 \*\*\*\*\*\*\*

21-40 \*\*\*\*\*

41-60 \*\*\*\*\*

61-80 \*\*\*\*

81-100 \*\*\*\*

101-120 \*\*\*

121-140 \*\*\*

141-160 \*\*\*

161-180 \*\*

181-200 \*\*

201-220 \*\*

221-240 \*\*

241-260 \*\*

261-280 \*

281-300 \*

301-320 \*

321-340

361-380

381-400

401-420 \*

601-620 \*

621-640 \*

641-660 \*

661-680

701-720

761-780

The rows with no stars means that there is only one member of that citation group, with only one occurrence. So, since the total value of that bucket is 1, we have log10(1) = 0, thus, no stars.

**Problem 3) Attached set of slides under the title “MapReduce calculation of Pi.ppt” It shows to you how you could calculate the value of number Pi (π) by randomly generating points in a two dimensional square. The ratio of points that fall into the circle and the number of points that fall into the square appears to be equal (close) to . Try to formulate this approach as a MapReduce job. Experiment also with the number of mappers and reducers your job will use. Class JobConf has methods setNumMapTasks() and setNumReduceTasks(). Use them. Report on your findings. You are not expected to calculate to the precision of 1000 decimal places. You just want to do a better job than you would do with a single random number generator.**

In order to perform this calculation, we implement the pseudo-code described in the slide deck. we have the following implementation on the class CalculatePi.java:

First of all, we need to be able to specify the number of mappers to execute on the job. As the new API does not have the setNumMapTasks() function, we have to change our InputFormatClass to be the NLineInputFormat, such as each input line is assigned to a new Mapper instance.

job.setMapperClass(MapClass.class);

job.setReducerClass(Reduce.class);

job.setInputFormatClass(NLineInputFormat.class);

job.setOutputFormatClass(TextOutputFormat.class);

job.setMapOutputKeyClass(IntWritable.class);

job.setMapOutputValueClass(IntWritable.class);

We set out number of points and number of mappers in the source code, but we also make sure we have an input file with 1000 lines:

private static int NUMPOINTS = 1000000;

private static int nMappers = 1000;

The Mapper function is set to generate the random numbers inside the square and calculate how many of them can be found inside the circle. We then send to the Reducer the number of points we found inside the circle out of the total points the Mapper generated:

public static class MapClass extends Mapper<LongWritable, Text, IntWritable, IntWritable> {

public void map(Text key, Text value,

Context context) throws IOException, InterruptedException {

int countCircle = 0;

int numPoints = NUMPOINTS/nMappers;

Random gen = new Random();

for (int i =0; i < numPoints; i++) {

// Generating random numbers from -1 to 1

double xcoord = (-1) + gen.nextDouble() \* 2;

double ycoord = (-1) + gen.nextDouble() \* 2;

// Calculating if it falls in the circle

if (Math.sqrt((xcoord \* xcoord) + (ycoord \* ycoord)) < 1) {

countCircle++;

}

}

context.write(new IntWritable(numPoints), new IntWritable(countCircle));

}

}

The Reducer aggregates these results and calculates the value of PI given the formula that was derived from the slide deck:

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

public void reduce(IntWritable key, Iterable<IntWritable> values,

Context context) throws IOException, InterruptedException {

// Calculating the number of points in all circles

int countCircle = 0;

for (IntWritable value : values) {

countCircle+= value.get();

}

double pi = 4.0 \* (double) countCircle / (double) NUMPOINTS;

context.write(new Text("PI:"), new DoubleWritable(pi));

}

}

We compile our class, setup the input file with the correspondent number of rows (a 1000 for this run) and we run the job:

[cloudera@centos-e185 ~]$ mkdir Assign08/P3

[cloudera@centos-e185 ~]$ mv CalculatePi.java Assign08/P3/

[cloudera@centos-e185 ~]$ cd Assign08/P3/

[cloudera@centos-e185 P3]$ ls

CalculatePi.java

[cloudera@centos-e185 P3]$ mkdir class

[cloudera@centos-e185 P3]$ javac -classpath `hadoop classpath` -Xlint:deprecation -d class CalculatePi.java

CalculatePi.java:72: warning: [deprecation] Job(org.apache.hadoop.conf.Configuration) in org.apache.hadoop.mapreduce.Job has been deprecated

Job job = new Job(conf);

^

1 warning

[cloudera@centos-e185 P3]$ cd class

[cloudera@centos-e185 class]$ jar -cvf CalculatePi.jar e185/\*

added manifest

adding: e185/assign08/(in = 0) (out= 0)(stored 0%)

adding: e185/assign08/CalculatePi$Reduce.class(in = 1833) (out= 784)(deflated 57%)

adding: e185/assign08/CalculatePi$MapClass.class(in = 1722) (out= 776)(deflated 54%)

adding: e185/assign08/CalculatePi.class(in = 2591) (out= 1238)(deflated 52%)

[cloudera@centos-e185 class]$ cd ..

[cloudera@centos-e185 P3]$ hadoop fs -mkdir Assign08/input-P3

[cloudera@centos-e185 P3]$ hadoop fs -copyFromLocal inputP3.txt Assign08/input-P3/

[cloudera@centos-e185 P3]$ cd class/

[cloudera@centos-e185 class]$ hadoop jar CalculatePi.jar e185.assign08.CalculatePi Assign08/input-P3 Assign08/output-P3

(...)

13/04/10 04:10:05 INFO mapreduce.Job: map 0% reduce 0%

(...)

13/04/10 04:54:20 INFO mapreduce.Job: map 100% reduce 100%

(...)

Job Counters

Launched map tasks=1000

Launched reduce tasks=1

(...)

File Input Format Counters

Bytes Read=2700628

File Output Format Counters

Bytes Written=12

[cloudera@centos-e185 class]$ hadoop fs -cat Assign08/output-P3/part-r-00000

PI: 3.14112

Running it with a 1000 Mappers turns out to be quite ambicious, because it takes 40 minutes to complete in our poor virtual machine, and we get a 3 decimal point precision on Pi. Just for the fun of it, we run it again with 100 Mappers (modifying the code and the input file), also reducing the total number of points by 10.

[cloudera@centos-e185 class]$ hadoop jar CalculatePi.jar e185.assign08.CalculatePi Assign08/input-P3 Assign08/output-P3

(...)

Job Counters

Launched map tasks=100

Launched reduce tasks=1

Other local map tasks=100

(...)

File Input Format Counters

Bytes Read=30399

File Output Format Counters

Bytes Written=12

[cloudera@centos-e185 class]$ hadoop fs -cat Assign08/output-P3/part-r-00000

PI: 3.14124

And we get a similar result as well.

**Problem 4) We can execute the two MapReduce jobs manually one after the other, the way we did it in the class with InverterCounter job, the outputof which we used as the input to CitationHistogram job. It would be more convenient to automate the execution sequence. You can chain two or more MapReduce jobs to run sequentially, with the output of one MapReduce job being the input to the next. Chaining MapReduce jobs is analogous to Unix pipes .**

**mapreduce-1 | mapreduce-2 | mapreduce-3 | ...**

**Chaining MapReduce jobs sequentially is quite straightforward. Recall that a driver sets up a JobConf object with the configuration parameters for a MapReduce job and passes the JobConf object to JobClient.runJob() to start the job. As JobClient.runJob() blocks until the end of a job, chaining MapReduce jobs involves calling the driver of one MapReduce job after another. The driver at each job will have to create a new JobConf object and set its input path to be the output path of the previous job. You can delete the intermediate data generated at each step of the chain or at the time when they are not needed any more, at the end. You can perform the file deletion with command like this**

**FileSystem.delete(Path f, boolean recursive);**

**You can control input and output file locations (paths) programmatically using Path class.**

**Path in = new Path("HDFSDirectory/filename");**

**Path first\_out = new Path("hdfs\_directory”);**

**Demonstrate that your chained MapReduce job for calculating Citation Histogram starting with patents citation file cite75\_99.tx will produce the same result as the sequence of two jobs used in class.**

For this problem, we create a new class called P4FullCitation.java, where we compose the two Mappers and the two reducers from the InverterCounterNewAPI.java and CitationHistogramNewAPI.java. The classes themselves are unchanged, but we added the prefix Counter and Histogram to them in order to differentiate them:

public static class CounterMapClass extends Mapper<Text, Text, Text, Text> {

public void map(Text key, Text value,

Context context) throws IOException, InterruptedException {

context.write(value, key);

}

}

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

public void reduce(Text key, Iterable<Text> values,

Context context) throws IOException, InterruptedException {

int count = 0;

for (Text value : values) {

count++;

}

context.write(key, new IntWritable(count));

}

}

public static class HistogramMapClass extends Mapper<Text, Text, IntWritable, IntWritable> {

private final static IntWritable uno = new IntWritable(1);

private IntWritable citationCount = new IntWritable();

public void map(Text key, Text value, Context context)

throws IOException, InterruptedException {

citationCount.set(Integer.parseInt(value.toString()));

context.write(citationCount, uno);

}

}

public static class HistogramReduce extends Reducer <IntWritable,IntWritable,IntWritable,IntWritable> { //public void reduce(IntWritable key, Iterator<IntWritable>values,

public void reduce(IntWritable key, Iterable<IntWritable> values, Context context)

throws IOException, InterruptedException {

int count = 0;

for (IntWritable val:values){ // Iterable allows

count += val.get(); // for looping

}

context.write(key, new IntWritable(count));

}

}

But the real changes are in the run() method, were we create two separate jobs and we execute them in sequence, using an intermediate directory for the output of the first MapReduce job, and deleting it afterwards.

public int run(String[] args) throws Exception {

Configuration conf = getConf();

conf.set("mapreduce.input.keyvaluelinerecordreader.key.value.separator", ",");

FileSystem fs = FileSystem.get(conf);

Path in = new Path(args[0]);

Path out = new Path(args[1]);

Path intermediate = new Path("Assign08/inter-P4");

Job jobCounter = new Job(conf, "P4-InverterCounter");

jobCounter.setJarByClass(P4FullCitation.class);

FileInputFormat.setInputPaths(jobCounter, in);

FileOutputFormat.setOutputPath(jobCounter, intermediate);

jobCounter.setJobName("P4-InverterCounter");

jobCounter.setMapperClass(CounterMapClass.class);

jobCounter.setReducerClass(CounterReduce.class);

jobCounter.setInputFormatClass(KeyValueTextInputFormat.class);

jobCounter.setOutputFormatClass(TextOutputFormat.class);

jobCounter.setOutputKeyClass(Text.class);

jobCounter.setOutputValueClass(Text.class);

jobCounter.waitForCompletion(true);

conf.unset("mapreduce.input.keyvaluelinerecordreader.key.value.separator");

Job jobHistogram = new Job(conf, "P4-CitationHistogram");

jobHistogram.setJarByClass(P4FullCitation.class);

FileInputFormat.setInputPaths(jobHistogram, intermediate);

FileOutputFormat.setOutputPath(jobHistogram, out);

jobHistogram.setJobName("P4-CitationHistogram");

jobHistogram.setMapperClass(HistogramMapClass.class);

jobHistogram.setReducerClass(HistogramReduce.class);

jobHistogram.setInputFormatClass(KeyValueTextInputFormat.class);

jobHistogram.setOutputFormatClass(TextOutputFormat.class);

jobHistogram.setOutputKeyClass(IntWritable.class);

jobHistogram.setOutputValueClass(IntWritable.class);

jobHistogram.waitForCompletion(true);

fs.delete(intermediate, true);

System.exit(0);

return 0;

}

We then compile and run this aggregated class with the 2 MapReduce jobs and check the output.

**[cloudera@centos-e185 P4]$ javac -classpath `hadoop classpath` -Xlint:deprecation -d class P4FullCitation.java**

P4FullCitation.java:78: warning: [deprecation] Job(org.apache.hadoop.conf.Configuration,java.lang.String) in org.apache.hadoop.mapreduce.Job has been deprecated

Job jobCounter = new Job(conf, "P4-InverterCounter");

^

P4FullCitation.java:93: warning: [deprecation] Job(org.apache.hadoop.conf.Configuration,java.lang.String) in org.apache.hadoop.mapreduce.Job has been deprecated

Job jobHistogram = new Job(conf, "P4-CitationHistogram");

^

2 warnings

**[cloudera@centos-e185 P4]$ cd class**

**[cloudera@centos-e185 class]$ jar -cvf P4FullCitation.jar e185/\***

added manifest

adding: e185/assign08/(in = 0) (out= 0)(stored 0%)

adding: e185/assign08/P4FullCitation$CounterMapClass.class(in = 1286) (out= 491)(deflated 61%)

adding: e185/assign08/P4FullCitation$CounterReduce.class(in = 1630) (out= 677)(deflated 58%)

adding: e185/assign08/P4FullCitation$HistogramReduce.class(in = 1695) (out= 689)(deflated 59%)

adding: e185/assign08/P4FullCitation.class(in = 2912) (out= 1341)(deflated 53%)

adding: e185/assign08/P4FullCitation$HistogramMapClass.class(in = 1700) (out= 690)(deflated 59%)

**[cloudera@centos-e185 class]$ hadoop jar P4FullCitation.jar e185.assign08.P4FullCitation Assign08/input Assign08/output-P4**

(...)

File Input Format Counters

Bytes Read=264087702

File Output Format Counters

Bytes Written=32858603

(...)

File Input Format Counters

Bytes Read=32858603

File Output Format Counters

Bytes Written=1721

**[cloudera@centos-e185 class]$ hadoop fs -cat Assign08/output-P4/part-r-00000 | head**

1 921127

2 552246

3 380319

4 278438

5 210814

6 163149

7 127941

8 102155

9 82126

10 66634

**[cloudera@centos-e185 class]$ hadoop fs -cat Assign08/output-P4/part-r-00000 | tail**

411 1

605 1

613 1

631 1

633 1

654 1

658 1

678 1

716 1

779 1

And we get the same result as we would with the jobs running separately.