Assignment #4 - Page Rank in Hadoop

Due on August 9th, 2020
Distributed Systems and Cloud Computing
CS549WS−Summer 2020
Professor Dominic Duggan

Daniel Kadyrov
10455680

1 Introduction

This assignment focuses on implementing a Page Ranking algorithm to sort the most popular Wikipedia pages from a provided previously downloaded set. Changes were made to the files shown in the following section. A test using a graph was performed using Hadoop on a local machine was performed. The Page Ranking script was then used on AWS EMR on 5, 10, and 20 clusters.

2 Changes within Provided Files

2.1 DiffMap1

Listing 1: DiffMap1.java

```
1  package edu.stevens.cs549.hadoop.pagerank;
2
3  import java.io.IOException;
4
5  import org.apache.hadoop.mapreduce.*;
6  import org.apache.hadoop.io.*;
7
8  public class DiffMap1 extends Mapper<LongWritable, Text, Text, Text> {
9
10         public void map(LongWritable key, Text value, Context context) throws
               IOException, InterruptedException,
11                       IllegalArgumentException {
12                 String line = value.toString(); // Converts Line to a String
13                 String[] sections = line.split("\t"); // Splits each line
14                 if (sections.length > 2) // checks for incorrect data format
15                 {
16                         throw new IOException("Incorrect data format");
17                 }
18                 /**
19                 *  TODO: read node-rank pair and emit: key:node, value:rank
20                 */
21
22                 // Split into Node and Rank
23                 String[] node_rank = sections[0].split("\\+");
24
25                 // Emit Node and Rank
26                 context.write(new Text(node_rank[0]),  new Text(node_rank[1]))
                       ;
27         }
28
29 }
```

2.2 DiffMap2

Listing 2: DiffMap2.java

```
1  package edu.stevens.cs549.hadoop.pagerank;
2
3  import java.io.IOException;
4
5  import org.apache.hadoop.mapreduce.*;
6  import org.apache.hadoop.io.*;
7
8  public class DiffMap2 extends Mapper<LongWritable, Text, Text, Text> {
9
10         public void map(LongWritable key, Text value, Context context) throws
               IOException, InterruptedException,
11                     IllegalArgumentException {
12                 String s = value.toString(); // Converts Line to a String
13
14                 /*
15                  * TODO: emit: key:"Difference" value:difference calculated in
                        DiffRed1
16                  */
17
18                 String[] node_rank = s.split("\t+");
19
20                 // Emit Difference
21                 context.write(new Text("Difference"), new Text(node_rank[1]));
22         }
23
24 }
```

2.3 DiffRed1

Listing 3: DiffRed1.java

```
1  package edu.stevens.cs549.hadoop.pagerank;
2
3  import java.io.*;
4
5  import org.apache.hadoop.mapreduce.*;
6  import org.apache.hadoop.io.*;
7  import java.util.Iterator;
8
9  public class DiffRed1 extends Reducer<Text, Text, Text, Text> {
10
11         public void reduce(Text key, Iterable<Text> values, Context context)
               throws IOException, InterruptedException {
12                 double[] ranks = new double[2];
13                 /*
14                  * TODO: The list of values should contain two ranks.  Compute
                        and output their difference.
15                  */
16
17                 Iterator<Text> iterator = values.iterator();
18
19                  // Default Diff set at 0
20                 double diff = 0;
21
22                 // Rank 1 Calculated
23                 if(iterator.hasNext()) {
24                         ranks[0] = Double.valueOf(iterator.next().toString());
25                 }
26
27                 // Rank 2 Calculated
28                 if(iterator.hasNext()) {
29                         ranks[1] = Double.valueOf(iterator.next().toString());
30                 }
31
32                 // Calculate Difference Diff
33                 diff = Math.abs(ranks[0] - ranks[1]);
34                 System.out.println(key.toString() + "  " + diff);
35                 context.write(key, new Text(String.valueOf(diff)));
36         }
37 }
```

2.4 DiffRed2

Listing 4: DiffRed2.java

```
1  package edu.stevens.cs549.hadoop.pagerank;
2
3  import java.io.*;
4  import java.util.Iterator;
5  import org.apache.hadoop.mapreduce.*;
6  import org.apache.hadoop.io.*;
7
8  public class DiffRed2 extends Reducer<Text, Text, Text, Text> {
9
10         public void reduce(Text key, Iterable<Text> values, Context context)
               throws IOException, InterruptedException {
11                 double diff_max = 0.0; // sets diff_max to a default value
12                 /*
13                  * TODO: Compute and emit the maximum of the differences
14                  */
15
16                  Iterator<Text> iterator = values.iterator();
17
18                 // Calculate Maximum Difference and Emit
19                 while(iterator.hasNext()) {
20                         diff_max = Math.max(diff_max, Double.valueOf(iterator.
                               next().toString()));
21                 }
22                 context.write(new Text(""), new Text(String.valueOf(diff_max))
                       );
23         }
24 }
```

2.5 FinMapper

Listing 5: FinMapper.java

```
1  package edu.stevens.cs549.hadoop.pagerank;
2
3  import java.io.IOException;
4  import org.apache.hadoop.mapreduce.*;
5  import org.apache.hadoop.io.*;
6
7  public class FinMapper extends Mapper < LongWritable, Text, DoubleWritable,
       Text > {
8          public void map(LongWritable key, Text value, Context context) throws
               IOException, InterruptedException,IllegalArgumentException {
9                  String line = value.toString(); // Converts Line to a String
10                 /*
11                  * TODO output key:-rank, value: node
12                  * See IterMapper for hints on parsing the output of
                        IterReducer.
13                  */
14                 String[] sections = line.split("\t"); // nodeId+nodeName |
                       rank
15
16                 // Check Format of Data
17                 if (sections.length > 2) {
18                         throw new IOException("Incorrect data format");
19                 }
20                 if (sections.length != 2) {
21                         return;
22                 }
23
24                 // 0-Rank will Reverse Shuffle the Reducer
25                 context.write(new DoubleWritable(0 - Double.valueOf(sections
                       [1])), new Text(sections[0]));
26         }
27 }
```

2.6 FinReducer

Listing 6: FinReducer.java

```
1  package edu.stevens.cs549.hadoop.pagerank;
2
3  import java.io.IOException;
4  import java.util.Iterator;
5  import org.apache.hadoop.io.DoubleWritable;
6  import org.apache.hadoop.io.Text;
7  import org.apache.hadoop.mapreduce.Reducer;
8
9  public class FinReducer extends Reducer < DoubleWritable, Text, Text, Text > {
10
11         public void reduce(DoubleWritable key, Iterable < Text > values,
               Context context) throws IOException,
12         InterruptedException {
13                 /*
14                  * TODO: For each value, emit: key:value, value:-rank
15                  */
16
17                 Iterator <Text> iterator = values.iterator();
18                 String node;
19
20                 while (iterator.hasNext()) {
21                         node = iterator.next().toString();
22
23                         // Convert Negative Rank to Rank
24                         context.write(new Text(node), new Text(String.valueOf
                               (0 - key.get())));
25                 }
26         }
27 }
```

2.7 InitMapper

Listing 7: InitMapper.java

```
1  package edu.stevens.cs549.hadoop.pagerank;
2
3  import java.io.IOException;
4
5  import org.apache.hadoop.mapreduce.*;
6  import org.apache.hadoop.io.*;
7
8  public class InitMapper extends Mapper < LongWritable, Text, Text, Text > {
9
10         public void map(LongWritable key, Text value, Context context) throws
               IOException,
11         InterruptedException,
12         IllegalArgumentException {
13                 String line = value.toString(); // Converts Line to a String
14                 /*
15                  * TODO: Just echo the input, since it is already in adjacency
                        list format.
16                  */
17
18                 // Split the Line using ":" and Emit the Key and Adjacent List
19                 String[] pair = line.split(":");
20
21                 if (pair != null && pair.length == 2) {
22                         context.write(new Text(pair[0].trim()), new Text(pair
                               [1]));
23                 }
24         }
25 }
```

2.8 InitReducer

Listing 8: InitReducer.java

```
1  package edu.stevens.cs549.hadoop.pagerank;
2
3  import java.io.*;
4  import java.util.Iterator;
5  import org.apache.hadoop.mapreduce.*;
6  import org.apache.hadoop.io.*;
7
8  public class InitReducer extends Reducer<Text, Text, Text, Text> {
9
10         public void reduce(Text key, Iterable<Text> values, Context context)
               throws IOException, InterruptedException {
11             /*
12              * TODO: Output key: node+rank, value: adjacency list
13              */
14             // Default Rank is 1
15             int default_rank = 1;
16             Iterator<Text> iterator = values.iterator();
17
18             while(iterator.hasNext()) {
19                     // Emit Node and Rank, Value
20                     context.write(new Text(key + "+" + default_rank),
                           iterator.next());
21             }
22     }
23 }
```

2.9 JoinMapper

Listing 9: JoinMapper.java

```
1  package edu.stevens.cs549.hadoop.pagerank;
2
3  import java.io.File;
4  import java.io.IOException;
5  import java.net.URI;
6
7  import org.apache.commons.io.FileUtils;
8  import org.apache.hadoop.fs.Path;
9  import org.apache.hadoop.io.LongWritable;
10 import org.apache.hadoop.io.Text;
11 import org.apache.hadoop.mapreduce.Mapper;
12
13 public class JoinMapper extends Mapper < LongWritable, Text, Text, Text > {
14         @Override
15         protected void setup(Context context) throws IOException,
               InterruptedException {
16                 if (context.getCacheFiles() != null && context.getCacheFiles()
                       .length > 0) {
17                         URI mappingFileUri = context.getCacheFiles()[0];
18
19                         if (mappingFileUri != null) {
20                                 System.out.println("Mapping File: " +
                                       FileUtils.readFileToString(new File("./
                                       cache")));
21                         } else {
22                                 System.out.println("no mapping file");
23                         }
24                 } else {
25                         System.out.println("no cache file");
26                 }
27         }
28
29         public void map(LongWritable key, Text value, Context context) throws
               IOException, InterruptedException, IllegalArgumentException {
30                 String line = value.toString();
31                 String[] sections;
32
33                 if (line.contains(":")) {
34                         int index = line.indexOf(":");
35                         sections = new String[2];
36                         sections[0] = line.substring(0, index);
37                         sections[1] = line.substring(index + 1, line.length())
                               ;
38                 } else {
39                         sections = line.split("\t");
40                 }
41
42                 if (sections.length > 2) {
43                         throw new IOException("Incorrect data format");
44                 }
45
46                 String[] node_rank = sections[0].split("\\+");
47                 if (node_rank.length == 1) {
48                         context.write(new Text(node_rank[0]), new Text("name:"
                                + sections[1].trim()));
49                 }
50                 if (node_rank.length == 2) {
51                         context.write(new Text(node_rank[0]), new Text("rank:"
                                + node_rank[1]));
52                 }
53         }
54 }
```

2.10 JoinReducer

Listing 10: JoinReducer.java

```
1  package edu.stevens.cs549.hadoop.pagerank;
2
3  import java.io.IOException;
4  import java.util.Iterator;
5
6  import org.apache.hadoop.io.Text;
7  import org.apache.hadoop.mapreduce.Reducer;
8
9  public class JoinReducer extends Reducer < Text, Text, Text, Text > {
10         public void reduce(Text key, Iterable < Text > values, Context context
               ) throws IOException, InterruptedException,
               IllegalArgumentException {
11                 Iterator <Text> iterator = values.iterator();
12                 String node_name = "";
13                 String rank = "";
14
15                 while (iterator.hasNext()) {
16                         String tmp = iterator.next().toString();
17                         if (tmp.startsWith("name:")) {
18                                 node_name = tmp.replaceAll("name:", "");
19                         }
20                         if (tmp.startsWith("rank:")) {
21                                 rank = tmp.replaceAll("rank:", "");
22                         }
23                 }
24
25                 context.write(new Text(key + "+" + node_name), new Text(rank))
                       ;
26         }
27 }
```

3 Running on Local Machine

3.1 Graph Files

Listing 11: sample.txt

```
1: 2 3
2: 4
3: 1 4 5
4:
5: 1 4
```

Listing 12: names.txt

```
1: v1
2: v2
3: v3
4: v4
5: v5
```

Listing 13: output.txt

```
4:0     1.708333333333333
1:0     0.8583333333333333
3:0     0.575
2:0     0.575
5:0     0.43333333333333335
```

3.2 Run Tests

The following command was used to test the local hadoop graph ranking algorithm.

Listing 14: Local Command

```
edu.stevens.cs549.hadoop.pagerank.PageRankDriver composite s3://cs549/input s3
    ://cs549/output inter1 inter2 diff 10
```

Table 1: Spread of COVID-19 by Continent

Number of Reducers	Processing Time
5	13s
10	18s
20	40s

Increasing the number of reducers increases the processing time of the algorithm.

4 Running on EMR

Listing 15: Custom JAR Command

```
edu.stevens.cs549.hadoop.pagerank.PageRankDriver composite s3://cs549/input s3
    ://cs549/output inter1 inter2 diff 10
```

The algorithm took 23 minutes to run on a 5 machine cluster with 10 reducers. The following ranking was outputted:

Listing 16: output.txt

```
      5302153:0   12810.69631667126
      84707:0     8159.113728956683
      88822:0     7886.093686178045
      1921890:0   7275.127835822817
      5300058:0   5826.635099830321
      81615:0     5014.404767519376
      1804986:0   4197.8888148045
      5535280:0   4164.226675136804
      896161:0    3769.910788409694
      5535664:0   3714.2571317056295
      79583:0     3583.7680798360816
      1601519:0   3488.3225964390367
      687324:0    3455.841896902911
      1948883:0   3364.344625704658
      5308545:0   3190.9223377001085
      505135:0    3064.802824618247
      1603276:0   2941.357165895597
      5596267:0   2914.369814878134
      2497500:0   2825.4558209203
      2995510:0   2730.416154609054
      1650573:0   2659.3656421983587
      2370447:0   2565.2363930365195
      77935:0     2528.8948762668074
      4141787:0   2407.816159087741
      3492254:0   2356.5909651517313
      2437900:0   2304.3565862362984
      2401294:0   2295.6349478790767
      4189168:0   2293.5300406199854
      687618:0    2224.083020041053
      3072654:0   2102.6401724186544
      434174:0    2102.335180035135
      3988566:0   2096.664388916008
      4015997:0   2018.9134163444628
      686242:0    2005.6101902332678
      4696900:0   1961.1958496038294
      76573:0     1960.9973826414164
      4351989:0   1959.317983917727
      478879:0    1904.298745200261
      3603437:0   1898.3236554574517
      5115901:0   1879.5875947888485
      2876077:0   1846.0542169601665
      1386743:0   1841.197756971188
      5492723:0   1780.7479241121116
      3997849:0   1712.6427307459228
      75323:0     1630.2723881913055
      3587465:0   1575.4630614468424
      4089591:0   1571.4228562480328
      181909:0    1553.487944478633
      4015913:0   1537.6157012671888
      3013310:0   1487.1735404348779
```

When trying to run more than 5 reducers or more than 5 clusters on the Amazon EMR service resulted in the following error:

> Terminated with errorsThe request to create the EMR cluster or add EC2 instances to it failed. The number of vCPUs for instance type m5.xlarge exceeds the EC2 service quota for that type. Request a service quota increase for your AWS account or choose a different instance type and retry the request.
>
> For more information, see https://docs.aws.amazon.com/console/elasticmapreduce/vcpu-limit

I cannot find any information on Canvas about setting up EMR or addressing this issue. I asked a question on Piazza and awaiting for the reply at the time of submitting this assignment.