Assignment 5

Intro

For this assignment, the most important thing is the process of how to decompose the complicated program into several simple and sequential tasks which can be executed parallel by multiple computers. The basic workflow is:

initial mapreduce -> iterate mapreducr -> evaluate difference -> join name -> finish

I should emphasize on the finish phase. Actually, in my implementation, I tear down the FINISH phase into two parts: FINISH_JOIN, FINISH_COMBINE. Meanwhile, I also change the threshold to adapt to different tasks. Let's move on the next process, DIFF, of which major responsibility is to calculate difference between the intermediate values produced by ITER. If the difference is small enough, we then consider mapreduce is stable and output the result. ITER is a process where we compute the rank. INIT is the first phase we initialize all the value.

Implement

1.Initial

This is the beginning phase we start to prepare our program. In this phase, we initialize all the values. Initialize default rank = 1 And out put the final result with the format (node+rank, adjacency list)

InitMapper.java

```
String[] pair = line.split(":");
if(pair != null && pair.length == 2) {
    //System.out.println(pair[1]+"  test");
    context.write(new Text(pair[0].trim()), new Text(pair[1]));
```

we simply split the record and emit them to the reducer without any changes.

InitReducer.java

```
int defualtrank = 1;
Iterator<Text> v = values.iterator();
while(v.hasNext()) {
    // emit node+rank, value
    context.write(new Text(key + "+" + defualtrank), v.next());
```

we initialize the default rank.

2.Iterate

Iterating is the core phase where we calculate the rank of links. The major algorithm implement here. It consists of 2 steps in mapper and 1 step in reducer.

IterMapper.java

```
String[] noderank = sections[0].split("\\+"); // split node+rank
String node = String.valueOf(noderank[0]);
double rank = Double.valueOf(noderank[1]);
String ajacentlist = sections[1].toString().trim(); // keep ajacent list

String[] ajacentnodes = ajacentlist.split(" ");
int N = ajacentnodes.length; // outgoing links number
// 1/n * rank
double weightOfPage = (double)1/N * rank; // calculate current page weight if outgoing
for(String ajacentnode : ajacentnodes) {
    context.write(new Text(ajacentnode), new Text(String.valueOf(weightOfPage)));
}
// at the same time, emit current node's ajacent list with marker "ADJ:"
context.write(new Text(node), new Text(PageRankDriver.MARKER + sections[1]));
```

Calculate current node's weight and notify adjacent node and notify reducer current node's adjacent

IterReducer.java

```
Iterator<Text> iterator = values.iterator();
double currentRank = 0; // default rank is 1 - d
String ajacentlist = "";
while(iterator.hasNext()) {
    String line = iterator.next().toString();
    if(!line.startsWith(PageRankDriver.MARKER)) {
        currentRank += Double.valueOf(line);
    } else {
        ajacentlist = line.replaceAll(PageRankDriver.MARKER, "");
    }
}
// (1-d) + d * sum(bac
currentRank = 1 - d + currentRank * d;
context.write(new Text(key + "+" + currentRank), new Text(ajacentlist));
```

Calculate current node's weight and combine with its adjacent list. According to the formula, rank = (1-d) + sum(Nb) * d, we can present code (1-d) + d * sum(Nb) * d currentRank = 1 - d + currentRank * d. Emit the final out put context.write(new Text(key + "+" + currentRank), new Text(ajacentlist));

3.Diff

Diff is a special phase which mainly focuses on the difference between 2 successive Iteration operations. That's if the difference is less than the threshold we set, we can consider the Iteration operation has already done enough and we should output the final result. Diff contain 2 jobs, the first one is to get the list of difference of each node. Then the second job could compute the maximum values among them.

DiffMap1.java

Simply split the key(node+rank), emit the node, rank to reducer

```
String[] noderank = sections[0].split("\\+");// split node+rank;
context.write(new Text(noderank[0]), new Text(noderank[1])); // emit node, rank
```

DiffReducer1.java

Calculate the difference of each node

```
// caculate diff
diff = Math.abs(ranks[0] - ranks[1]); context.write(key, new Text(String.valueOf(diff)));
```

DiffMap2.java

Emit each node's difference to reducer

```
String[] noderank = s.split("\t+");
context.write(new Text("Difference"), new Text(noderank[1]));
```

DiffReducer2.java

Find the maximum difference among all the node

```
while(iterator.hasNext()) {
    double diff = Double.valueOf(iterator.next().toString());

    diff_max = diff_max > diff ? diff_max : diff;

}
context.write(new Text(""), new Text(String.valueOf(diff_max)));
```

4.Finish

Finish phase is not quite hard but can only output nodeId with the rank. To get the result with the format I personally decompose the finish phase into 2 phases. The first phase is Finish_join, which join the output produce by iterate and nodes' names relation data. After that, we reformat the data and descend order them by rank in finish phase.

FinJoinMapper:

Emit (node, rank or node name) relation to reducer with different maker

```
if(noderank.length == 1) {
        // it's nodeID with its name
        context.write(new Text(noderank[0]), new
Text(PageRankDriver.MARKER_NAME + sections[1].trim()));
}

if(noderank.length == 2) {
        // it's nodeID with its rank
        context.write(new Text(noderank[0]), new
        Text(PageRankDriver.MARKER_RANK + noderank[1]));
}
```

FinJoinReducer:

Capture node's ID and node's rank and recombine them.

```
/*
 * output: key: nodeId+names, text: rank
 */
context.write(new Text(key + "+" + nodeName) , new
Text(rank));
```

FinMapper:

To shuffle the intermediate value in descend order by rank, we need emit (-rank, node) to reducer.

```
// to reverse shuffle the reducer, we need minus rank with 0
        context.write(new DoubleWritable(0 -
        Double.valueOf(sections[1])), new Text(sections[0]));
```

FinReducer:

Restore back the value and output the final result.

```
// convert -rank back to rank
context.write(new Text(node), new Text(String.valueOf(0 -
key.get())));
```

Test Case

1.Test graph locally (video:// 1.Local_Graph)

I have two test file for this graph, they respectively are sample.txt and names.txt

Use *hadoop namenode -format* to formate hfs directory.

```
hadoop — java -Dproc_namenode -Xmx1000m -Djava.net.preferIPv4S
YangTenfeideMBP:hadoop tengfei$ hadoop namenode -format
DEPRECATED: Use of this script to execute hdfs command is deprecated.
Instead use the hdfs command for it.
```

Use *hstart* to start hadoop

```
YangTenfeideMBP:hadoop tengfei$ hstart
15/11/29 03:43:32 WARN util.NativeCodeLoader: Unable to load native-hadoop libra
ry for your platform... using builtin-java classes where applicable
Starting namenodes on [localhost]
localhost: starting namenode, logging to /usr/local/Cellar/hadoop/2.7.1/libexec/
logs/hadoop-tengfei-namenode-YangTenfeideMBP.home.out
```

Use the command below to make directory and put input data to them.

hdfs dfs -mkdir /input
hdfs dfs -put sample.txt /input

hdfs dfs -mkdir /cache
hdfs dfs -put names.txt /cache

```
YangTenfeideMBP:hadoop tengfei$ hdfs dfs -mkdir /input
15/11/29 03:43:59 WARN util.NativeCodeLoader: Unable to load native-hadoop libra
ry for your platform... using builtin-java classes where applicable
YangTenfeideMBP:hadoop tengfei$ hdfs dfs -mkdir /cache
15/11/29 03:44:05 WARN util.NativeCodeLoader: Unable to load native-hadoop libra
ry for your platform... using builtin-java classes where applicable
YangTenfeideMBP:hadoop tengfei$ cd ~/Desktop/
YangTenfeideMBP:Desktop tengfei$ hdfs dfs -put sample.txt /input
15/11/29 03:44:19 WARN util.NativeCodeLoader: Unable to load native-hadoop libra
ry for your platform... using builtin-java classes where applicable
YangTenfeideMBP:Desktop tengfei$ hdfs dfs -put names.txt /cache
15/11/29 03:44:27 WARN util.NativeCodeLoader: Unable to load native-hadoop libra
ry for your platform... using builtin-java classes where applicable
YangTenfeideMBP:Desktop tengfei$
```

Use the command below to run page rank
hadoop jar PageRank-1.0.0.jar edu.stevens.cs549.hadoop.pagerank.PageRankDriver composite /input /output inter1 inter2 diffdir 10

```
YangTenfeideMBP:Desktop tengfei$ hadoop jar PageRank-1.0.0.jar edu.stevens.cs549
.hadoop.pagerank.PageRankDriver composite /input /output inter1 inter2 diffdir 1
0
Tengfei Yang (10395116)
Init Job Started
15/11/29 03:44:56 WARN util.NativeCodeLoader: Unable to load native-hadoop libra
ry for your platform... using builtin-java classes where applicable
15/11/29 03:44:57 INFO Configuration.deprecation: session.id is deprecated. Inst
ead, use dfs.metrics.session-id
15/11/29 03:44:57 INFO jvm.JvmMetrics: Initializing JVM Metrics with processName
=JobTracker, sessionId=
```

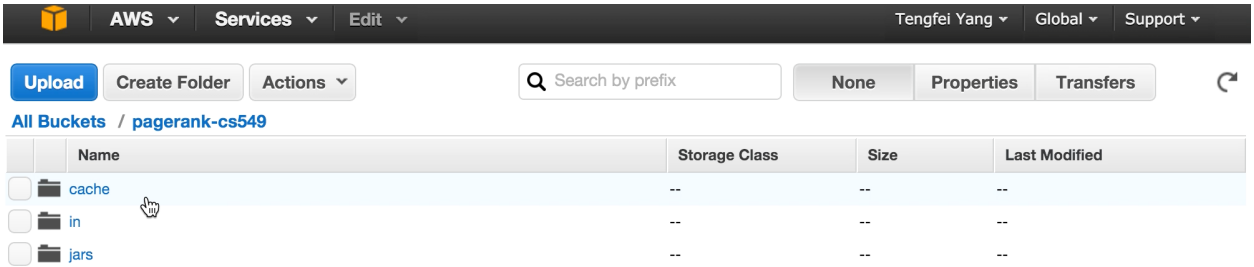
Use *hdfs dfs -cat /output/output.txt* to check the out put result.

```
YangTenfeideMBP:Desktop tengfei$ hdfs dfs -cat /output/output.txt
15/11/29 03:46:02 WARN util.NativeCodeLoader: Unable to load native-hadoop libra
ry for your platform... using builtin-java classes where applicable
v4      0.5904162570273318
v1      0.3359758507212698
v3      0.29526023321487993
v2      0.29526023321487993
v5      0.23481346876868733
```

For more details you can see in video ***1.Local_Graph***

2.Test wikipedia in EMR (video:// 2.EMR_Setup, 3.Wiki_Result)

Creating directory



Uploading the test data, jar to S3.

AWS Services Edit Tengfei Yang Global Support

Upload Create Folder Actions Search by prefix None Properties Transfers

All Buckets / pagerank-cs549 / jars

Name	Storage Class	Size	Last Modified
PageRank-1.0.0.jar	Standard	22.2 KB	Sun Nov 29 03:57:18 GMT-500 2015

AWS Services Edit Tengfei Yang Global Support

Upload Create Folder Actions Search by prefix None Properties Transfers

All Buckets / pagerank-cs549 / cache

Name	Storage Class	Size	Last Modified
wikipedia-pages-0001.txt	Standard	2.9 MB	Sun Nov 29 04:31:23 GMT-500 2015
wikipedia-pages-0002.txt	Standard	2.4 MB	Sun Nov 29 04:31:24 GMT-500 2015
wikipedia-pages-0003.txt	Standard	2.5 MB	Sun Nov 29 04:31:26 GMT-500 2015
wikipedia-pages-0004.txt	Standard	2.6 MB	Sun Nov 29 04:31:27 GMT-500 2015
wikipedia-pages-0005.txt	Standard	2.6 MB	Sun Nov 29 04:31:28 GMT-500 2015
wikipedia-pages-0006.txt	Standard	2.5 MB	Sun Nov 29 04:31:29 GMT-500 2015
wikipedia-pages-0007.txt	Standard	2.5 MB	Sun Nov 29 04:31:30 GMT-500 2015
wikipedia-pages-0008.txt	Standard	2.5 MB	Sun Nov 29 04:31:31 GMT-500 2015
wikipedia-pages-0009.txt	Standard	2.4 MB	Sun Nov 29 04:31:32 GMT-500 2015
wikipedia-pages-0010.txt	Standard	2.6 MB	Sun Nov 29 04:31:35 GMT-500 2015
wikipedia-pages-0011.txt	Standard	2.7 MB	Sun Nov 29 04:31:35 GMT-500 2015
wikipedia-pages-0012.txt	Standard	2.9 MB	Sun Nov 29 04:31:36 GMT-500 2015
wikipedia-pages-0013.txt	Standard	2.5 MB	Sun Nov 29 04:31:37 GMT-500 2015
wikipedia-pages-0014.txt	Standard	2.7 MB	Sun Nov 29 04:31:40 GMT-500 2015
wikipedia-pages-0015.txt	Standard	2.5 MB	Sun Nov 29 04:31:41 GMT-500 2015

Feedback English Privacy Policy Terms of Use

AWS Services Edit Tengfei Yang Global Support

Upload Create Folder Actions Search by prefix None Properties Transfers

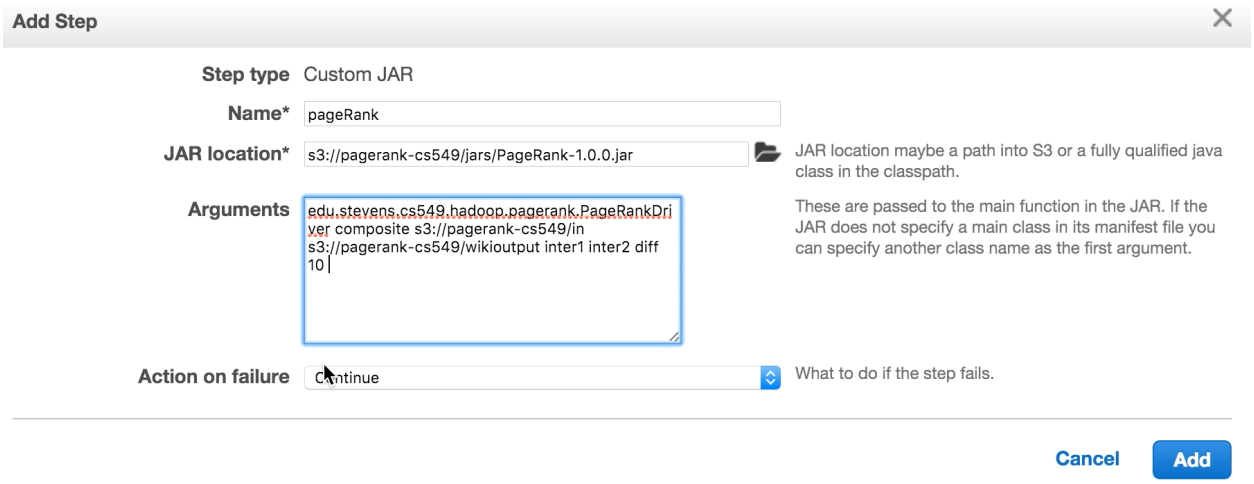
All Buckets / pagerank-cs549 / in

Name	Storage Class	Size	Last Modified
wikipedia-links-0001.txt	Standard	37.9 MB	Sun Nov 29 04:22:12 GMT-500 2015
wikipedia-links-0002.txt	Standard	16.3 MB	Sun Nov 29 04:22:23 GMT-500 2015
wikipedia-links-0003.txt	Standard	15.3 MB	Sun Nov 29 04:22:27 GMT-500 2015
wikipedia-links-0004.txt	Standard	16.6 MB	Sun Nov 29 04:22:31 GMT-500 2015
wikipedia-links-0005.txt	Standard	17.8 MB	Sun Nov 29 04:22:35 GMT-500 2015
wikipedia-links-0006.txt	Standard	16.9 MB	Sun Nov 29 04:22:40 GMT-500 2015
wikipedia-links-0007.txt	Standard	17.8 MB	Sun Nov 29 04:22:44 GMT-500 2015
wikipedia-links-0008.txt	Standard	19.2 MB	Sun Nov 29 04:22:49 GMT-500 2015
wikipedia-links-0009.txt	Standard	17.4 MB	Sun Nov 29 04:22:53 GMT-500 2015
wikipedia-links-0010.txt	Standard	16.8 MB	Sun Nov 29 04:22:57 GMT-500 2015
wikipedia-links-0011.txt	Standard	17.5 MB	Sun Nov 29 04:23:01 GMT-500 2015
wikipedia-links-0012.txt	Standard	16.4 MB	Sun Nov 29 04:23:05 GMT-500 2015
wikipedia-links-0013.txt	Standard	16.8 MB	Sun Nov 29 04:23:10 GMT-500 2015
wikipedia-links-0014.txt	Standard	16 MB	Sun Nov 29 04:23:13 GMT-500 2015
wikipedia-links-0015.txt	Standard	14.7 MB	Sun Nov 29 04:23:17 GMT-500 2015

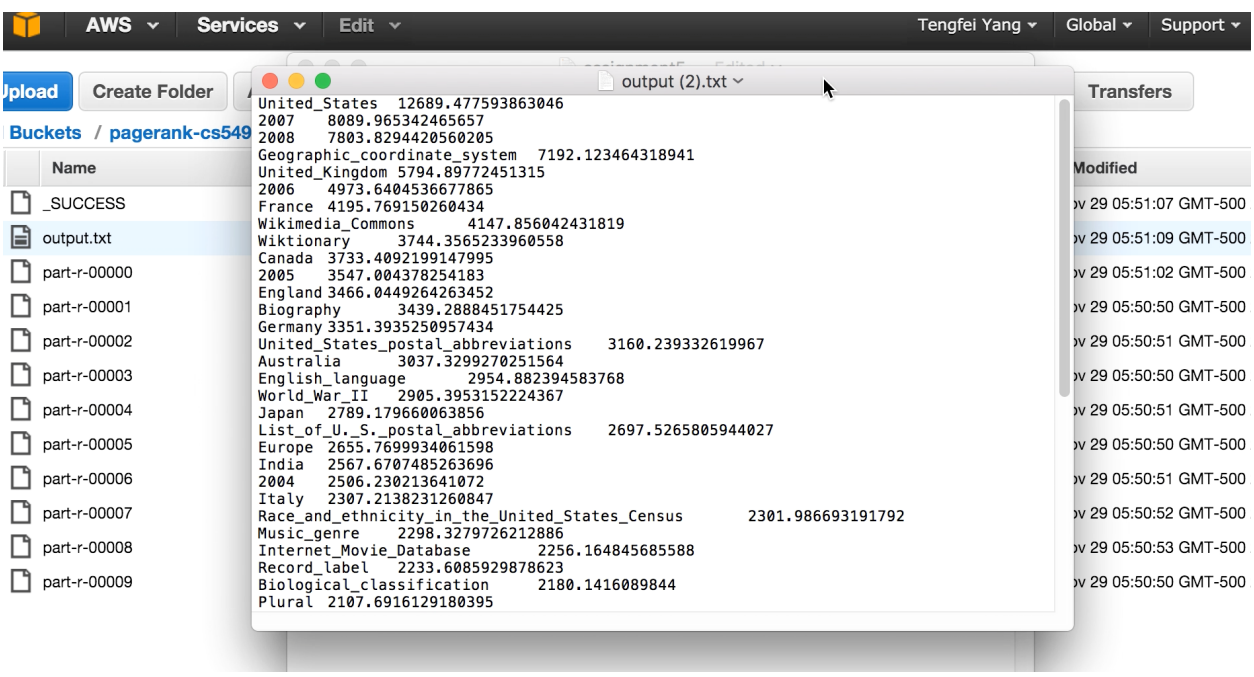
Feedback English Privacy Policy Terms of Use

Stepping EMR and using custom jar selection and give the parameters.

edu.stevens.cs549.hadoop.pagerank.PageRankDriver composite s3://pagerank-cs549/in s3://pagerank-cs549/wikioutput inter1 inter2 diff 10



Running EMR and Waiting for completement and check the output result in S3.



To see more details, please check video *2.EMR_Setup* and *3.Wiki_Result*.