PageRank-Hadoop

- Ankai Liang
- 10411998

- CS 549 Assignment 5

Code Explanation

PageRankDriver.java

In this snippet, I print my name in the method `composite()`, write another `composite()` to deal with the order which contains an equipartitioned join. And I change the code in `summarizeResult()` in order to output the result with an equipartitioned join.

```
public class PageRankDriver {

    public static final double DECAY = 0.85;
    public static final String MARKER_DELIMITER = "+";
    public static final String MARKER_ADJ = "ADJACENCY";
    public static String nameInput = null;
    ...
```

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

        String job = "";
        ...
         else if (args.length == 7) // If number of input arguments is 7 and
                                    // first input is == "composite"; parses
                                    // data to composite
        {
            if (job.equals("composite")) {
                composite(args[1], args[2], args[3], args[4], args[5],
                        Integer.parseInt(args[6]));
            } else // In case the function name doesn't match up
            {
                System.err
                        .println("Please check the name of the function you wish to c
all and try again");
            }
```

```
        } else if (args.length == 8) // If number of input arguments is 8 and
            // first input is == "composite"; parses
            // data to composite
        {
            if (job.equals("composite")) {
                composite(args[1], args[2], args[3], args[4], args[5], args[6],
                        Integer.parseInt(args[7]));
        }

        ...

                if (i % 2 == 1) // As i increments at the last step, for odd i, inter
im2
                        // is the input directory
        {
            deleteDirectory(interim1); // deletes other directory
            counter++;
            if (nameInput != null) {
                translate(nameInput, interim2, interim1, reducers);
                finish(interim1, output, reducers);
            } else {
                finish(interim2, output, reducers);
            }
            summarizeResult(output);
        } else // for even i, interim1 is the input directory
        {
            deleteDirectory(interim2); // Deletes other directory
            counter++;
            if (nameInput != null) {
                translate(nameInput, interim1, interim2, reducers);
                finish(interim2, output, reducers);
            } else {
                finish(interim1, output, reducers);
            }
            summarizeResult(output);
        }
        ...
```

```
    public static void composite(String input, String output, String interim1,
            String interim2, String diff, int reducers) throws Exception {
        /*
         * TODO
         */
        System.out.println("Ankai Liang (10411998)");

        int counter = 0;
        ...
```

```
    public static void composite(String name, String input, String output, String int
erim1,
            String interim2, String diff, int reducers) throws Exception {
        nameInput = name;
        composite(input, output, interim1, interim2, diff, reducers);
        nameInput = null;
    }
```

```
    static void summarizeResult(String path) throws Exception {
        Path finpath = new Path(path); // Creates new Path
        Configuration conf = new Configuration();
        FileSystem fs = FileSystem.get(URI.create(path), conf);
        HashMap<String, Double> values = new HashMap<String, Double>();
        ...
```

InitMapper.java

- input: nodeIdentifier : adjacency list
- emit: key = nodeIdentifier, value = adjacency list

```
public class InitMapper extends Mapper<LongWritable, Text, Text, Text> {

    public void map(LongWritable key, Text value, Context context) throws IOException
, InterruptedException,
            IllegalArgumentException {
        String line = value.toString(); // Converts Line to a String
        /*
         * TODO: Just echo the input, since it is already in adjacency list format.
         */
        String[] sections = line.split(":");
        if (sections.length == 2){
            context.write(new Text(sections[0].trim()),new Text(sections[1].trim()));
        }else
            throw new IOException("Incorrect data format");
    }

}
```

InitReducer.java

- input: key = nodeIdentifier, values = vertexes the node links to
- emit: key = nodeIdentifier+rank_value, value = adjacency list

```
public class InitReducer extends Reducer<Text, Text, Text, Text> {

    public void reduce(Text key, Iterable<Text> values, Context context) throws IOExc
eption, InterruptedException {
        /*
         * TODO: Output key: node+rank, value: adjacency list
         */
        StringBuilder sb = new StringBuilder();
        for (Text val : values){
            sb.append(val.toString() + " ");
        }
        context.write(new Text(key.toString() + PageRankDriver.MARKER_DELIMITER + "1.
0"), new Text(sb.toString().trim()));
    }
}
```

IterMapper.java

- input nodeIdentifier+rank_value | adjacency list
- emit key = adj vertex, value = computed weight

- emit key = this vertex, value = adj List

```
public class IterMapper extends Mapper<LongWritable, Text, Text, Text> {

    public void map(LongWritable key, Text value, Context context) throws IOException
, InterruptedException,
            IllegalArgumentException {
        String line = value.toString(); // Converts Line to a String
        String[] sections = line.split("\t"); // Splits it into two parts. Part 1: no
de+rank | Part 2: adj list

        if (sections.length > 2) // Checks if the data is in the incorrect format
        {
            throw new IOException("Incorrect data format");
        }
        if (sections.length != 2) {
            return;
        }

        /*
         * TODO: emit key: adj vertex, value: computed weight.
         *
         * Remember to also emit the input adjacency list for this node!
         * Put a marker on the string value to indicate it is an adjacency list.
         */
        String[] pair = sections[0].split("\\"+PageRankDriver.MARKER_DELIMITER);
        double rank = Double.valueOf(pair[1]);
        String[] adjList = sections[1].split(" ");
        double weight = rank / adjList.length;

        for (int i = 0; i < adjList.length; i++){
            context.write(new Text(adjList[i]), new Text(String.valueOf(weight)));
        }
        context.write(new Text(pair[0]), new Text(PageRankDriver.MARKER_ADJ + ":" + s
ections[1]));
    }

}
```

IterReducer.java

- input key = node, values = rankValues form adj vertexes Or adjacency list
- emit key = node+rank, value = adjacency list

```
public class IterReducer extends Reducer<Text, Text, Text, Text> {
    public void reduce(Text key, Iterable<Text> values, Context context) throws IOExc
eption, InterruptedException {
        double d = PageRankDriver.DECAY; // Decay factor
        /*
         * TODO: emit key:node+rank, value: adjacency list
         * Use PageRank algorithm to compute rank from weights contributed by incomin
g edges.
         * Remember that one of the values will be marked as the adjacency list for t
he node.
         */
        double sum = 0;
        String adjList = "";
        for (Text val:values){
            String val_s = val.toString();
            if (val_s.startsWith(PageRankDriver.MARKER_ADJ))
                adjList = val_s.split(":")[1];
            else sum += Double.valueOf(val.toString());
        }
        double value = (1-d) + d * sum;
        context.write(new Text(key.toString() + PageRankDriver.MARKER_DELIMITER + Str
ing.valueOf(value)), new Text(adjList));
    }
}
```

DiffMap1.java

- input node+rank ('+' is the delimiter)
- emit key = node, value = rank

```
public class DiffMap1 extends Mapper<LongWritable, Text, Text, Text> {

    public void map(LongWritable key, Text value, Context context) throws IOException
, InterruptedException,
            IllegalArgumentException {
        String line = value.toString(); // Converts Line to a String
        String[] sections = line.split("\t"); // Splits each line
        if (sections.length > 2) // checks for incorrect data format
        {
            throw new IOException("Incorrect data format");
        }
        /**
         *  TODO: read node-rank pair and emit: key:node, value:rank
         */
        String[] pair = sections[0].split("//"+PageRankDriver.MARKER_DELIMITER);
        if (pair.length == 2)
            context.write(new Text(pair[0]), new Text(pair[1]));
    }

}
```

DiffRed1.java

- input key = node , values = 2 ranks
- emit key = difference, value = ""

```
public class DiffRed1 extends Reducer<Text, Text, Text, Text> {

    public void reduce(Text key, Iterable<Text> values, Context context) throws IOExc
eption, InterruptedException {
        double[] ranks = new double[2];
        /*
         * TODO: The list of values should contain two ranks.  Compute and output the
ir difference.
         */
        int i = 0;
        for (Text val: values){
            ranks[i++] = Double.valueOf(val.toString());
            if (i >= 2) break;
        }
        context.write(new Text(String.valueOf(Math.abs(ranks[0] - ranks[1]))), new Te
xt());
    }
}
```

DiffMap2.java

- input difference from DiffRed1
- emit: key = "Difference" value = difference

```
public class DiffMap2 extends Mapper<LongWritable, Text, Text, Text> {

    public void map(LongWritable key, Text value, Context context) throws IOException
, InterruptedException,
            IllegalArgumentException {
        String s = value.toString(); // Converts Line to a String

        /*
         * TODO: emit: key:"Difference" value:difference calculated in DiffRed1
         */
        context.write(new Text("Difference"), new Text(s.trim()));
    }

}
```

DiffRed2.java

- input difference
- emit max_difference

```
public class DiffRed2 extends Reducer<Text, Text, Text, Text> {

    public void reduce(Text key, Iterable<Text> values, Context context) throws IOExc
eption, InterruptedException {
        double diff_max = 0.0; // sets diff_max to a default value
        /*
         * TODO: Compute and emit the maximum of the differences
         */
        for (Text val: values){
            double diff = Double.valueOf(val.toString());
            diff_max = Math.max(diff, diff_max);
        }
        context.write(new Text(String.valueOf(diff_max)), new Text());
    }
}
```

FinMapper.java

- input node+rank | adjList
- emit key = -rank, value = node

```
public class FinMapper extends Mapper<LongWritable, Text, DoubleWritable, Text> {

    public void map(LongWritable key, Text value, Context context)
            throws IOException, InterruptedException, IllegalArgumentException {
        String line = value.toString(); // Converts Line to a String
        /*
         * TODO output key:-rank, value: node
         * See IterMapper for hints on parsing the output of IterReducer.
         */
        String[] sections = line.split("\t");
        String[] pair = sections[0].split("\\" + PageRankDriver.MARKER_DELIMITER);
        if (pair.length == 2 && pair[1] != null && !pair[1].equals("null"))
            context.write(new DoubleWritable(-Double.valueOf(pair[1])), new Text(pair
[0]));
    }
}
```

FinReducer.java

- input key = -rank, values = node
- emit key = node, value = rank ``` public class FinReducer extends Reducer {

public void reduce(DoubleWritable key, Iterable values, Context context) throws IOException, InterruptedException { /* * TODO: For each value, emit: key:value, value:-rank */ for (Text val:values){ context.write(new Text(val), new Text(String.valueOf(-Double.valueOf(key.toString())))); } } } ```

TranslateMap.java

- input node+rank | adjList Or node: name
- emit key = node, value = "NAME" + name
- emit key = node, value = "RANK" + rank
- emit key = node, value = "ADJADENCY" + adjList

```
public class TranslateMap extends Mapper<LongWritable, Text, Text, Text> {

    public void map(LongWritable key, Text value, Context context) throws IOException
, InterruptedException, IllegalArgumentException {
        String line = value.toString();
        if (line.contains(":")) {
            //value = node : NAME
            String[] sections = line.split(": ");
            if (sections.length >= 2) {
                context.write(new Text(sections[0]),
                        new Text("NAME" + PageRankDriver.MARKER_DELIMITER + line.subs
tring(line.indexOf(':') + 1).trim()));
            }
        } else {
            // value = node+rank | adjList
            String[] sections = line.split("\t");
            String[] pair = sections[0].trim().split("\\"+PageRankDriver.MARKER_DELIM
ITER);
            String rank = pair[1];
            String list = (sections.length > 1) ? sections[1].trim() : "";
            context.write(new Text(pair[0]), new Text("RANK" + PageRankDriver.MARKER_
DELIMITER + rank));
            context.write(new Text(pair[0]), new Text(PageRankDriver.MARKER_ADJ + Pag
eRankDriver.MARKER_DELIMITER + list));
        }
    }
}
```

TranslateRed.java

- input node l "NAME" + name
- input node l "RANK" + rank
- input node l "ADJADENCY" + adjList
- emit key = name Or node(if can't match any name), value = adjList

```
public class TranslateRed extends Reducer<Text, Text, Text, Text> {

    public void reduce(Text key, Iterable<Text> values, Context context) throws IOExc
eption, InterruptedException {
        String node = key.toString();
        String name = null;
        String rank = null;
        String list = null;
        for(Text val : values) {
            String value = val.toString();
            String[] pair = value.split("\\"+PageRankDriver.MARKER_DELIMITER);
            if (value.startsWith("NAME")) {
                name = (pair.length > 1) ? pair[1]:null;
            } else if (value.startsWith("RANK")) {
                rank = (pair.length > 1) ? pair[1]:null;
            } else if (value.startsWith(PageRankDriver.MARKER_ADJ)) {
                list = (pair.length > 1) ? pair[1]:null;
            }
        }
        String outKey = ((name == null)? node : name) + PageRankDriver.MARKER_DELIMIT
ER + rank;
        String outValue = (list == null)? "" : list;
        context.write(new Text(outKey), new Text(outValue));
    }

}
```

Test

Local

I test using the example graph in pdf. Graph:

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

Name:

```
1: amazon
2: pixer
3: java
4: google
5: Ac Fun
```

Output:

```
google  1.708333333333333
amazon  0.8583333333333333
pixer   0.575
java    0.575
Ac Fun  0.43333333333333335
```

EMR

Result(5 reducers):

```
United_States_postal_abbreviations  17400.21276502926
United_States   13078.758142261851
Geographic_coordinate_system    11721.388760989967
Biography   9046.94404539674
2008    7074.6425361942565
2007    6800.352714147496
United_Kingdom  4910.857511340169
Music_genre 4873.710868083095
France  4742.359910022161
Record_label    4693.041866616612
Biological_classification   4504.695513826754
England 4235.032041106078
Canada  3690.885542951598
Personal_name   3576.978399648568
2006    3575.6745346145035
Internet_Movie_Database 3463.2832928822127
India   3106.5889721148874
Binomial_nomenclature   2900.531729940027
Germany 2813.7080608753836
Australia   2801.5539246554167
2005    2753.2105256013992
Japan   2738.5863561278975
Studio_album    2653.126455458513
```

```
Village 2443.744194498339
Record_producer 2390.2859118177685
Football_(soccer)   2309.2199488266556
Politician  2297.477993147134
Romania 2223.1518281845247
English_language    2213.6348292720913
Time_zone   1995.897478274637
Departments_of_France   1989.2385407660036
Wiktionary  1986.601595665923
Geocode 1957.2310712914546
UN/LOCODE   1927.7685672468049
2004    1923.7717011191974
Television  1887.5449027795114
Italy   1876.5179226178395
Europe  1876.4133549775547
Album   1862.5000237009542
Conservation_status 1822.3255838594016
Website 1786.3698129091722
Animal  1755.4690932603446
London  1750.4268351247983
IUCN_Red_List   1693.081548359981
Wikimedia_Commons   1691.2920868385286
Poland  1660.9849679859221
Population_density  1594.3872097107585
Public_domain   1524.3655342422294
Actor   1430.0997921719966
Digital_object_identifier   1394.3430354664986
2001    1372.891851411744
Elevation   1359.6035580473917
Norway  1354.81422265507
China   1290.9928602406003
School  1176.2465753102497
```

5 reducers ,10 reducers, 20 reducers

- 1 master, 4 cores, spent 9 mins.
- 1 master, 9 cores, spent 6 mins 11 secs (first)
- 1 master, 9 cores, spent 5 mins 30 secs (second)
- 1 master, 19 cores, spent 4 mins 25secs（first）.
- 1 master, 19 cores, spent 4 mins 52secs (second)

According to the different number reducers' different time, we know that running twice with the exact same arguments still show a notable difference. And we use the double reducers from 10 to 20, but the time cost

didn't improve as much as that. Because if we have too more reducers, more time will spend on comunication and calling function, rather than the benifit of increasing reducer. That means adding reducers may slow overall time.