Programming Assignment 2

Big Data System Course

Objective

- Install and configure HAMA
- Implement Weighted PageRank in HAMA

PageRank HAMA example

```
public abstract class Vertex<V extends Writable, E extends
Writable, M extends Writable>
      implements VertexInterface<V, E, M> {

    public void compute(Iterator<M> messages) throws
IOException;
    ..

}
```

- V: Vertex Key type
- E: Edge type
- M: **Message** type also **Vertex Value** type

PageRank HAMA example (cont’d)

```
public static class PagerankSeqReader
      extends
      VertexInputReader<Text, TextArrayWritable, Text, NullWritable,
DoubleWritable> {
    @Override
    public boolean parseVertex(Text key, TextArrayWritable value,
        Vertex<Text, NullWritable, DoubleWritable> vertex)  throws
Exception {
      vertex.setVertexID(key);

      for (Writable v : value.get()) {
        vertex.addEdge(new Edge<Text, NullWritable>((Text) v,  null));
      }

      return true;
    }
  }
```

set vertex key

Identify neighbor by neighbor vertex ID

- Implement a VertexInputReader to Read data from sequence file

PageRank HAMA example (cont’d)

```
public static class PageRankVertex extends
      Vertex<Text, NullWritable, DoubleWritable> {

    @Override
    public void compute(Iterator<DoubleWritable> messages) throws IOException {
      if (this.getSuperstepCount() == 0) {
        this.setValue(new DoubleWritable(1.0 / (double) this.getNumVertices()));
      }

      if (this.getSuperstepCount() >= 1) {
        double sum = 0;
        while (messages.hasNext()) {
          DoubleWritable msg = messages.next();
          sum += msg.get();
        }

        double ALPHA = (1 - 0.85) / (double) this.getNumVertices();
        this.setValue(new DoubleWritable(ALPHA + (0.85 * sum)));
      }

      if (this.getSuperstepCount() < this.getMaxIteration()) {
        int numEdges = this.getOutEdges().size();
        sendMessageToNeighbors(new DoubleWritable(this.getValue().get()
            / numEdges));
      }
    }
  }
```

Weighted PageRank

- Core Idea: assigns larger rank values to **more important (popular) pages** instead of dividing the rank value of a page evenly among its outlink pages.

- $$PR(u) = (1-d) + d \sum_{v \in B(u)} PR(v) W^{in}_{(v,u)} W^{out}_{(v,u)}$$

  $$W^{in}_{(v,u)} = \frac{I_u}{\sum_{p \in R(v)} I_p}$$

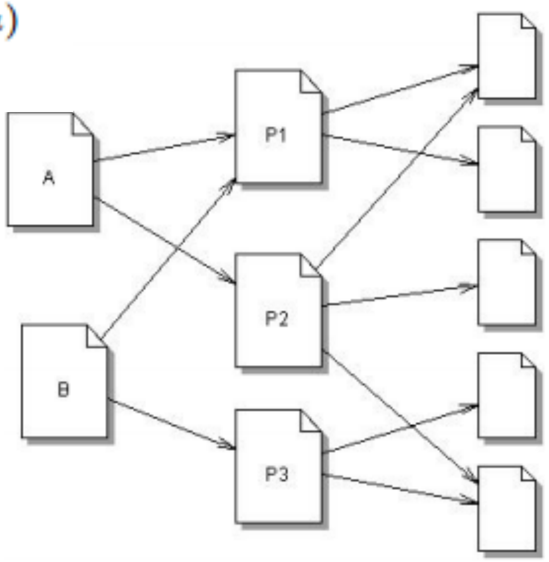
  $$W^{out}_{(v,u)} = \frac{O_u}{\sum_{p \in R(v)} O_p}$$

Example

$$PR(u) = (1-d) + d \sum_{v \in B(u)} PR(v) W^{in}_{(v,u)} W^{out}_{(v,u)}$$

- $$W^{in}_{(A,p1)} = I_{p1}/(I_{p1} + I_{p2}) = \frac{2}{3}$$

  $$W^{out}_{(A,p1)} = O_{p1}/(O_{p1} + O_{p2}) = \frac{2}{5}$$



- $$PR(P1) = (1-d) + d\{\frac{2}{3} \cdot \frac{2}{5} PR(A) + \frac{2}{3} \cdot \frac{2}{4} PR(B)\}$$

Input Format

- A text file with vertex ID and edges separated by tab, edges are separated by space.

```
1	7 4 2 8 10
2	3 6 5 1
3	2 10 8 5 7
...
```

- You should implement your own VertexInputReader, and also set the correct vertex r

```
wpJob.setVertexInputReaderClass(MyVertexReader.class);
wpJob.setInputFormat(TextInputFormat.class);
wpJob.setInputKeyClass(LongWritable.class);
wpJob.setInputValueClass(Text.class);
```

Output Format

- The output format should be a **text file** with vertex ID as key and weighted pagerank value as value separated by tab in each line.

```
1<tab>1.23455332
2<tab>2.54543522
3<tab>1.23243323
4<tab>0.72342344
```

- Note that the sequence of key does not matter as long as all the keys are listed.

Workflow

- When sending contribution to output edges, you need to know the number of input and number of output edges of the output edges.
- So the workflow should be like follow
   a. Calculate the #input and #output edges of all output edges. Hints:
      - Send message to out edges, out edges receive messages and thus know what are it’s in edges.
      - Send message to all input edges telling it how many input and output edges it has.
   b. Calculate weighted pagerank based on the result.

Hints

- Since you need to store information of #input and #output edges, the built-in classes are not enough. You need to implement your own Writable class, e.g.
  - `class` **`NodeWritable`** `implements Writable`
- You may also need to implement your own ArrayWritable type, e.g. **NodeArrayWritable**
  - See the TextArrayWritable for example.
- If you were to implement the homework based on the pagerank example, remember to **remove** this line:
  - `pageJob.set("hama.graph.self.ref", "true");`

Test Data

- We provide two test data for you to verify your implementation.
- **small.graph**
  - A small graph containing only 10 nodes
  - small.graph.13.out is the result of running 13 iterations
- **big.graph**
  - A big graph containing 200 nodes
  - big.graph.6.out is the result of running 6 iterations
- We only check for the **6** digits of the floating point precision.

Debugging

- Sometimes errors don’t show up in console!!
  - Check the logs under $HAMA_HOME/logs/tasklogs
- You can use `org.apache.commons.logging`

Compile & Execute

- Like usual, this homework requires you to provide two script, namely compile.sh and run.sh.
- **./compile.sh**
  - We will provide the **CLASSPATH** environment variable containing all the jar required.
- **./run.sh <iter> <input> <output>**
  - iter: number of pagerank iteration
  - input: input file path
  - output: output file path

Submission

Upload a zip file named in “Your School ID"

Use “r00922000” as an example :

```
r00922000.zip
 └── r00922000
      ├── Report.pdf
      ├── compile.sh
      ├── run.sh
      └── Other source files
```

Submission

- Due: 2013/11/21 19:00:00
- Wrong submission format (including compile.sh and run.sh) will recieve 10% ponishment.
- Late submissions lose 10% per day
- Plagiarism will not be tolerated