WriteHamaGraphFile

Overview

This article is about how to bring your graph into the graph module of Hama and successfully run algorithms on it.

Google Web dataset (local mode, pseudo distributed cluser)

For this example, the Google web graph from 2002 is used (http://snap.stanford.edu/data/web-Google.html), it contains 875,713 nodes and 5,105,039 edges.

The file itself looks like this:

```
# Directed graph (each unordered pair of nodes is saved once): web-Google.txt
# Webgraph from the Google programming contest, 2002
# Nodes: 875713 Edges: 5105039
# FromNodeId        ToNodeId
0        11342
0        824020
0        867923
0        891835
11342        0
11342        27469
11342        38716
```

To read this kind of file you need to provide a vertex reader like this:

```
  public static class PagerankTextReader extends
      VertexInputReader<LongWritable, Text, Text, NullWritable, DoubleWritable> {

    String lastVertexId = null;
    List<String> adjacents = new ArrayList<String>();

    @Override
    public boolean parseVertex(LongWritable key, Text value,
        Vertex<Text, NullWritable, DoubleWritable> vertex) {

      String line = value.toString();
      String[] lineSplit = line.split("\t");
      if (!line.startsWith("#")) {
        if (lastVertexId == null) {
          lastVertexId = lineSplit[0];
        }
        if (lastVertexId.equals(lineSplit[0])) {
          adjacents.add(lineSplit[1]);
        } else {
          vertex.setVertexID(new Text(lastVertexId));
          for (String adjacent : adjacents) {
            vertex.addEdge(new Edge<Text, NullWritable>(new Text(adjacent),
                null));
          }
          adjacents.clear();
          lastVertexId = lineSplit[0];
          adjacents.add(lineSplit[1]);
          return true;
        }
      }
      return false;
    }

  }
```

```
    HamaConfiguration conf = new HamaConfiguration(new Configuration());
    // if we are in local mode, prevent the file from beeing split
    conf.set("bsp.local.tasks.maximum", "1");
    GraphJob pageJob = new GraphJob(conf, PageRank.class);
    pageJob.setJobName("Pagerank");

    pageJob.setVertexClass(PageRankVertex.class);
    pageJob.setInputPath(new Path(
        "/tmp/google-in/"));
    pageJob.setOutputPath(new Path(
        "/tmp/google-out/"));

    // do a maximum of 60 iterations
    pageJob.setMaxIteration(60);
    pageJob.set("hama.pagerank.alpha", "0.85");
    // we need to include a vertex in its adjacency list,
    // otherwise the pagerank result has a constant loss
    pageJob.set("hama.graph.self.ref", "true");
    // run until the error is archived
    pageJob.set("hama.graph.max.convergence.error", "0.001");
    // hama takes care that the graph is complete
    pageJob.set("hama.graph.repair", "true");
    pageJob.setAggregatorClass(AverageAggregator.class);
    // don't get splitted
    pageJob.setNumBspTask(1);
    pageJob.setVertexIDClass(Text.class);
    pageJob.setVertexValueClass(DoubleWritable.class);
    pageJob.setEdgeValueClass(NullWritable.class);

    pageJob.setInputKeyClass(LongWritable.class);
    pageJob.setInputValueClass(Text.class);
    pageJob.setInputFormat(TextInputFormat.class);
    pageJob.setVertexInputReaderClass(PagerankTextReader.class);
    pageJob.setPartitioner(HashPartitioner.class);
    pageJob.setOutputFormat(TextOutputFormat.class);
    pageJob.setOutputKeyClass(Text.class);
    pageJob.setOutputValueClass(DoubleWritable.class);

    long startTime = System.currentTimeMillis();
    if (pageJob.waitForCompletion(true)) {
      System.out.println("Job Finished in "
          + (System.currentTimeMillis() - startTime) / 1000.0 + " seconds");
    }
```

You should see the algorithm converge quite fast after nine supersteps. The textfile is sadly not deterministically splitable, so it doesn't converge as fast as if it were executed in parallel.

If you read the results back from the text output, you will see the following top 10 ranked sites:

```
885605 = 0.00149900065779375
846221 = 0.0010280702392776039
557124 = 8.654234880507804E-4
537039 = 6.634317501245855E-4
163075 = 6.529762251084758E-4
597621 = 6.503367245789417E-4
41909 = 5.845160681337011E-4
551829 = 5.702205338951212E-4
504140 = 5.507901000809657E-4
765334 = 5.432108978490109E-4
486980 = 5.394792436341423E-4
```

Wikipedia dataset (for smallish clusters)

For this example, the Wikipedia link dataset is used (http://haselgrove.id.au/wikipedia.htm) / (http://users.on.net/~henry/pagerank/links-simple-sorted.zip).

The dataset contains 5,716,808 pages and 130,160,392 links and is unzipped ~1gb large. You should use a smallish cluster to crunch this dataset with Hama, based on the blocksize of HDFS a slot number of 16-32 is required.

The file is formatted like this

```
   from1: to11 to12 to13 ...
   from2: to21 to22 to23 ...
```

To read this kind of file you need to provide a vertex reader like this:

```
public static class WikipediaLinkDatasetReader extends
     VertexInputReader<LongWritable, Text, Text, NullWritable, DoubleWritable> {

   @Override
   public boolean parseVertex(LongWritable key, Text value,
       Vertex<Text, NullWritable, DoubleWritable> vertex) {
     String[] vertexAdjacents = value.toString().split(":");
     vertex.setVertexID(new Text(vertexAdjacents[0].trim()));
     String[] split = vertexAdjacents[1].split(" ");
     for (int i = 0; i < split.length; i++) {
       if (!split[i].isEmpty()) {
         vertex.addEdge(new Edge<Text, NullWritable>(
             new Text(split[i].trim()), null));
       }
     }
     return true;
   }
 }
```

Now we can setup the job with the following code (assuming that your input is in /tmp/wiki-in/:

```
    HamaConfiguration conf = new HamaConfiguration(new Configuration());
    GraphJob pageJob = new GraphJob(conf, PageRank.class);
    pageJob.setJobName("Pagerank");

    pageJob.setVertexClass(PageRankVertex.class);
    pageJob.setInputPath(new Path(
        "/tmp/wiki-in/"));
    pageJob.setOutputPath(new Path(
        "/tmp/wiki-out/"));

    // do a maximum of 60 iterations
    pageJob.setMaxIteration(60);
    pageJob.set("hama.pagerank.alpha", "0.85");
    // we need to include a vertex in its adjacency list,
    // otherwise the pagerank result has a constant loss
    pageJob.set("hama.graph.self.ref", "true");
    // run until the error is archived
    pageJob.set("hama.graph.max.convergence.error", "0.001");
    // hama takes care that the graph is complete
    pageJob.set("hama.graph.repair", "true");
    pageJob.setAggregatorClass(AverageAggregator.class);

    pageJob.setVertexIDClass(Text.class);
    pageJob.setVertexValueClass(DoubleWritable.class);
    pageJob.setEdgeValueClass(NullWritable.class);

    pageJob.setInputKeyClass(LongWritable.class);
    pageJob.setInputValueClass(Text.class);
    pageJob.setInputFormat(TextInputFormat.class);
    pageJob.setVertexInputReaderClass(WikipediaLinkDatasetReader.class);
    pageJob.setPartitioner(HashPartitioner.class);
    pageJob.setOutputFormat(TextOutputFormat.class);
    pageJob.setOutputKeyClass(Text.class);
    pageJob.setOutputValueClass(DoubleWritable.class);

    long startTime = System.currentTimeMillis();
    if (pageJob.waitForCompletion(true)) {
      System.out.println("Job Finished in "
          + (System.currentTimeMillis() - startTime) / 1000.0 + " seconds");
    }
```

Now you could export a jar with the job submission as main class and you can submit this via

```
hama/bin/hama jar YOUR_JAR.jar
```

to your Hama cluster.

If you sort the result descending by pagerank you can see the following top 10 sites:

```
  1    0.00222  United_States
  2    0.00141  2007
  3    0.00136  2008
  4    0.00126  Geographic_coordinate_system
  5    0.00101  United_Kingdom
  6    0.00087  2006
  7    0.00074  France
  8    0.00073  Wikimedia_Commons
  9    0.00066  Wiktionary
 10    0.00065  Canada
```

Note that you can decode the indices you may see with the page titels files.

Troubleshooting

If your job does not execute, your cluster may not have enough resources (task slots).
Symptoms may look like this in the bsp master log:

```
2012-05-27 20:00:51,228 WARN org.apache.hadoop.util.NativeCodeLoader: Unable to load native-hadoop library for
your platform... using builtin-java classes where applicable
2012-05-27 20:00:51,288 INFO org.apache.hama.bsp.JobInProgress: num BSPTasks: 16
2012-05-27 20:00:51,305 INFO org.apache.hama.bsp.JobInProgress: Job is initialized.
2012-05-27 20:00:51,313 ERROR org.apache.hama.bsp.SimpleTaskScheduler: Scheduling of job Pagerank could not be
done successfully. Killing it!
2012-05-27 20:01:08,334 INFO org.apache.hama.bsp.JobInProgress: num BSPTasks: 16
2012-05-27 20:01:08,339 INFO org.apache.hama.bsp.JobInProgress: Job is initialized.
2012-05-27 20:01:08,340 ERROR org.apache.hama.bsp.SimpleTaskScheduler: Scheduling of job Pagerank could not be
done successfully. Killing it!
```

This was run on a 8 slot cluster, but it required 16 slots because of 64m chunk size of HDFS. Either you can reupload the file with higher chunksize so the slots match the blocks or you can increase the slots in your Hama cluster.