HamaWhite 原创，转载请注明出处！欢迎大家加入**Giraph 技术交流群**： 228591158

目的：科研中，需要分析在每次迭代过程中参与计算的顶点数目，来进一步优化系统。比如，在SSSP的compute()方法最后一行，都会把当前顶点voteToHalt，即变为InActive状态。所以每次迭代完成后，所有顶点都是InActive状态。在大同步后，收到消息的顶点会被激活，变为Active状态，然后调用顶点的compute()方法。本文的目的就是统计每次迭代过程中，参与计算的顶点数目。下面附上SSSP的compute()方法：

**[java]** [view plaincopyprint?](http://blog.csdn.net/xin_jmail/article/details/25078723)[在CODE上查看代码片](https://code.csdn.net/snippets/328979)

1. @Override
2. **public** **void** compute(Iterable<DoubleWritable> messages) {
3. **if** (getSuperstep() == 0) {
4. setValue(**new** DoubleWritable(Double.MAX\_VALUE));
5. }
6. **double** minDist = isSource() ? 0d : Double.MAX\_VALUE;
7. **for** (DoubleWritable message : messages) {
8. minDist = Math.min(minDist, message.get());
9. }
10. **if** (minDist < getValue().get()) {
11. setValue(**new** DoubleWritable(minDist));
12. **for** (Edge<LongWritable, FloatWritable> edge : getEdges()) {
13. **double** distance = minDist + edge.getValue().get();
14. sendMessage(edge.getTargetVertexId(), **new** DoubleWritable(distance));
15. }
16. }
17. //把顶点置为InActive状态
18. voteToHalt();
19. }

附：giraph中算法的终止条件是：没有活跃顶点且worker间没有消息传递。

hama-0.6.0中算法的终止条件只是：判断是否有活跃顶点。不是真正的pregel思想，半成品。

修改过程如下：

1. org.apache.giraph.partition. PartitionStats 类

添加变量和方法，用来统计每个Partition在每个超步中参与计算的顶点数目。添加的变量和方法如下：

**[java]** [view plaincopyprint?](http://blog.csdn.net/xin_jmail/article/details/25078723)[在CODE上查看代码片](https://code.csdn.net/snippets/328979)

1. /\*\* computed vertices in this partition \*/
2. **private** **long** computedVertexCount=0;
4. /\*\*
5. \* Increment the computed vertex count by one.
6. \*/
7. **public** **void** incrComputedVertexCount() {
8. ++ computedVertexCount;
9. }
11. /\*\*
12. \* @return the computedVertexCount
13. \*/
14. **public** **long** getComputedVertexCount() {
15. **return** computedVertexCount;
16. }

修改readFields()和write()方法，每个方法追加最后一句。当每个Partition计算完成后，会把自己的computedVertexCount发送给Master，Mater再读取汇总。

**[java]** [view plaincopyprint?](http://blog.csdn.net/xin_jmail/article/details/25078723)[在CODE上查看代码片](https://code.csdn.net/snippets/328979)

1. @Override
2. **public** **void** readFields(DataInput input) **throws** IOException {
3. partitionId = input.readInt();
4. vertexCount = input.readLong();
5. finishedVertexCount = input.readLong();
6. edgeCount = input.readLong();
7. messagesSentCount = input.readLong();
8. //添加下条语句
9. computedVertexCount=input.readLong();
10. }
12. @Override
13. **public** **void** write(DataOutput output) **throws** IOException {
14. output.writeInt(partitionId);
15. output.writeLong(vertexCount);
16. output.writeLong(finishedVertexCount);
17. output.writeLong(edgeCount);
18. output.writeLong(messagesSentCount);
19. //添加下条语句
20. output.writeLong(computedVertexCount);
21. }

2. org.apache.giraph.graph. GlobalStats 类

  添加变量和方法，用来统计每个超步中参与计算的顶点总数目，包含每个Worker上的所有Partitions。

**[java]** [view plaincopyprint?](http://blog.csdn.net/xin_jmail/article/details/25078723)[在CODE上查看代码片](https://code.csdn.net/snippets/328979)

1. /\*\* computed vertices in this partition
2. \*  Add by BaiSong
3. \*/
4. **private** **long** computedVertexCount=0;
5. /\*\*
6. \* @return the computedVertexCount
7. \*/
8. **public** **long** getComputedVertexCount() {
9. **return** computedVertexCount;
10. }

修改addPartitionStats(PartitionStats partitionStats)方法，增加统计computedVertexCount功能。

**[java]** [view plaincopyprint?](http://blog.csdn.net/xin_jmail/article/details/25078723)[在CODE上查看代码片](https://code.csdn.net/snippets/328979)

1. /\*\*
2. \* Add the stats of a partition to the global stats.
3. \*
4. \* @param partitionStats Partition stats to be added.
5. \*/
6. **public** **void** addPartitionStats(PartitionStats partitionStats) {
7. **this**.vertexCount += partitionStats.getVertexCount();
8. **this**.finishedVertexCount += partitionStats.getFinishedVertexCount();
9. **this**.edgeCount += partitionStats.getEdgeCount();
10. //Add by BaiSong，添加下条语句
11. **this**.computedVertexCount+=partitionStats.getComputedVertexCount();
12. }

当然为了Debug方便，也可以修改该类的toString()方法（可选），修改后的如下：

**[java]** [view plaincopyprint?](http://blog.csdn.net/xin_jmail/article/details/25078723)[在CODE上查看代码片](https://code.csdn.net/snippets/328979)

1. **public** String toString() {
2. **return** "(vtx=" + vertexCount + ", computedVertexCount="
3. + computedVertexCount + ",finVtx=" + finishedVertexCount
4. + ",edges=" + edgeCount + ",msgCount=" + messageCount
5. + ",haltComputation=" + haltComputation + ")";
6. }

3. org.apache.giraph.graph. ComputeCallable<I,V,E,M>

添加统计功能。在computePartition()方法中，添加下面一句。

**[java]** [view plaincopyprint?](http://blog.csdn.net/xin_jmail/article/details/25078723)[在CODE上查看代码片](https://code.csdn.net/snippets/328979)

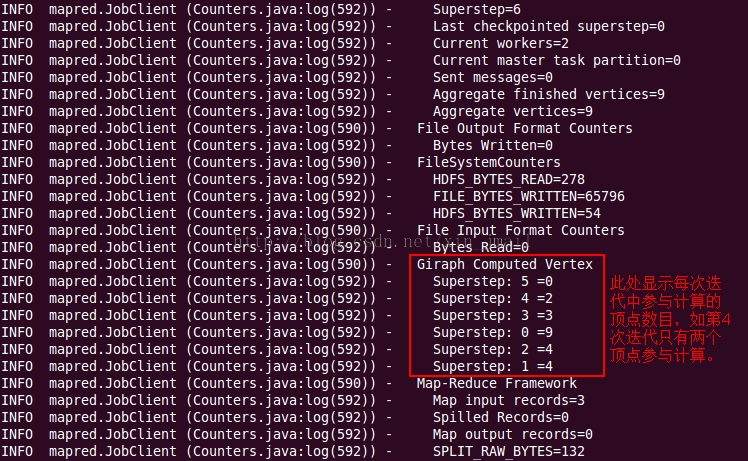
1. **if** (!vertex.isHalted()) {
2. context.progress();
3. TimerContext computeOneTimerContext = computeOneTimer.time();
4. **try** {
5. vertex.compute(messages);
6. //添加下面一句，当顶点调用完compute()方法后，就把该Partition的computedVertexCount加1
7. partitionStats.incrComputedVertexCount();
8. } **finally** {
9. computeOneTimerContext.stop();
10. }
11. ……

4. 添加Counters统计，和我的博客[Giraph源码分析（七）—— 添加消息统计功能](http://blog.csdn.net/xin_jmail/article/details/24411891) 类似，此处不再详述。添加的类为：org.apache.giraph.counters.GiraphComputedVertex，下面附上该类的源码：

**[java]** [view plaincopyprint?](http://blog.csdn.net/xin_jmail/article/details/25078723)[在CODE上查看代码片](https://code.csdn.net/snippets/328979)

1. **package** org.apache.giraph.counters;
3. **import** java.util.Iterator;
4. **import** java.util.Map;
6. **import** org.apache.hadoop.mapreduce.Mapper.Context;
7. **import** com.google.common.collect.Maps;
9. /\*\*
10. \* Hadoop Counters in group "Giraph Messages" for counting every superstep
11. \* message count.
12. \*/
14. **public** **class** GiraphComputedVertex **extends** HadoopCountersBase {
15. /\*\* Counter group name for the giraph Messages \*/
16. **public** **static** **final** String GROUP\_NAME = "Giraph Computed Vertex";
18. /\*\* Singleton instance for everyone to use \*/
19. **private** **static** GiraphComputedVertex INSTANCE;
21. /\*\* superstep time in msec \*/
22. **private** **final** Map<Long, GiraphHadoopCounter> superstepVertexCount;
24. **private** GiraphComputedVertex(Context context) {
25. **super**(context, GROUP\_NAME);
26. superstepVertexCount = Maps.newHashMap();
27. }
29. /\*\*
30. \* Instantiate with Hadoop Context.
31. \*
32. \* @param context
33. \*            Hadoop Context to use.
34. \*/
35. **public** **static** **void** init(Context context) {
36. INSTANCE = **new** GiraphComputedVertex(context);
37. }
39. /\*\*
40. \* Get singleton instance.
41. \*
42. \* @return singleton GiraphTimers instance.
43. \*/
44. **public** **static** GiraphComputedVertex getInstance() {
45. **return** INSTANCE;
46. }
48. /\*\*
49. \* Get counter for superstep messages
50. \*
51. \* @param superstep
52. \* @return
53. \*/
54. **public** GiraphHadoopCounter getSuperstepVertexCount(**long** superstep) {
55. GiraphHadoopCounter counter = superstepVertexCount.get(superstep);
56. **if** (counter == **null**) {
57. String counterPrefix = "Superstep: " + superstep+" ";
58. counter = getCounter(counterPrefix);
59. superstepVertexCount.put(superstep, counter);
60. }
61. **return** counter;
62. }
64. @Override
65. **public** Iterator<GiraphHadoopCounter> iterator() {
66. **return** superstepVertexCount.values().iterator();
67. }
68. }

5. 实验结果，运行程序后。会在终端输出每次迭代参与计算的顶点总数目。 测试SSSP（SimpleShortestPathsVertex类），输入图中共有9个顶点和12条边。输出结果如下：



上图测试中，共有6次迭代。红色框中，显示出了每次迭代过冲参与计算的顶点数目，依次是：9,4,4,3,4,0

解释：在第0个超步，每个顶点都是活跃的，所有共有9个顶点参与计算。在第5个超步，共有0个顶点参与计算，那么就不会向外发送消息，加上每个顶点都是不活跃的，所以算法迭代终止。

完！