# 0. 工作总结（Review）

## Week 01 2016/10/17

* Flink初探（基本概念，编程模型，环境搭建）(重点花在这里)；
* Spark的GraphX和Flink的Gelly的图算法总结;
* Flink的Gelly相关的基础知识学习(重点花在这里);
* Flink的Triangle Count算法学习;

## Week 02 2016/10/31

* 深入理解Flink的架构设计
* 设计benchmark的测试方案

## Week 03 2015/11/07

* 学习Flink图算法的API.
* 深入探讨Flink其中一个图算法.

# 1. 实验设计（Benchmark）

## 1.1 如何衡量一个算法的好坏？

（1）准确性：计算的结果是否准确？

（2）扩展性：在不同规模的数量级上算法的性能趋势？

（3）可重复性：在相同数据集上多次重复试验的结果是否相同？

应用参数的影响。

## 1.2 如何设计图的benchmark？

（1）纵向比较

a. 图的性质：有向/无向图？随机图？全连通图？空图？超立方体图？

b. 图的规模：顶点和边的数量？

（2）横向比较

a. Flink和Spark做对比？

b. 自己实现的做对比？

# 2. Flink图算法

表1. Flink图算法表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 编号 | 名称 | 描述 | 应用 | 难度 |
|  | Community Detection |  |  |  |
|  | Label Propagation |  |  |  |
|  | Connected Components |  |  |  |
|  | GSA Connected Components |  |  |  |
|  | PageRank |  |  |  |
|  | GSA PageRank |  |  |  |
|  | Single Source Shortest Paths |  |  |  |
|  | GSA Single Source Shortest Paths |  |  |  |
|  | Triangle Count |  |  |  |
|  | Triangle Listing |  |  |  |
|  | Triangle Enumerator |  |  |  |
|  | Hyperlink-Induced Topic Search |  |  |  |
|  | Summarization |  |  |  |
|  | Adamic-Adar |  |  |  |
|  | Jaccard Index |  |  |  |
|  | Local Clustering Coefficient |  |  |  |
|  | Global Clustering Cofficient |  |  |  |

## 2.1 Flink 图模型



图1. Flink图模型

## 2.2 Gelly Library

### AF07 Single Source Shortest Paths

|  |  |
| --- | --- |
| 名称 |  |
| 描述 |  |
| 应用 |  |
| 逻辑 |  |
| 程序 |  |
| 实验 |  |
| 结果 |  |
| 参考 |  |

### AF09 Triangle Count

|  |  |
| --- | --- |
| 名称 | Triangle Count |
| 描述 | 统计图中不同三角形的数目 |
| 应用 | 一般运用在社交网络分析中。社交网络中的三角形越多，说明关系网越强。（聚集系数） |
| 逻辑 | 1. **def** triangleCount(graph): 2. count = 0 3. tringle = [] 4. **for** srcId **in** graph:  //1. 遍历图中的每个节点 5. srcSet = graph.get(srcId)  //2. 针对节点srcId，找出它的所有邻接点 6. **for** destId **in** srcSet: 7. **if** (destId > srcId): 8. destSet = graph.get(destId)  //3. 针对邻接点desctId,找出他的邻接点 9. //4. 如果srcId和descId的邻接点相同，则构成三角形 10. **for** vertexId **in** destSet: 11. **if**(vertexId **in** srcSet) **and** (vertexId > destId): 12. count += 1 13. tringle.append((srcId, destId, vertexId)) |
| 程序 | **class**:org.apache.flink.graph.library.clustering.(un)directed.TriangleCount  **input**: a (un)directed graph  **output**: The analytic takes an **undirected** **graph** as input and returns as a result a **Long** corresponding to the number of triangles in the graph. The graph ID type must be Comparable and Copyable.  **analysis:**  **如下代码调用triangleCount算法**  graph.run(triangleCount);  实际上是反过来调用：  analytic.run(this);  而triangleCount方法体为：  public TriangleCount<K, VV, EV> run(Graph<K, VV, EV> input)  throws Exception {  super.run(input);   DataSet<Tuple3<K, K, K>> triangles = input  .run(new TriangleListing<K, VV, EV>()  .setSortTriangleVertices(false)  .setLittleParallelism(littleParallelism));   triangles  .output(new CountHelper<Tuple3<K, K, K>>(id))  .name("Count triangles");   return this; }  即实际上调用了TriangleListing算法的runInternal(graph)方法：  public DataSet<Tuple3<K, K, K>> runInternal(Graph<K, VV, EV> input)  throws Exception {  // u, v where u < v  DataSet<Tuple2<K, K>> filteredByID = input  .getEdges()  .flatMap(new FilterByID<K, EV>())  .setParallelism(littleParallelism)  .name("Filter by ID");   // u, v, (edge value, deg(u), deg(v))  DataSet<Edge<K, Tuple3<EV, LongValue, LongValue>>> pairDegree = input  .run(new EdgeDegreePair<K, VV, EV>()  .setParallelism(littleParallelism));   // u, v where deg(u) < deg(v) or (deg(u) == deg(v) and u < v)  DataSet<Tuple2<K, K>> filteredByDegree = pairDegree  .flatMap(new FilterByDegree<K, EV>())  .setParallelism(littleParallelism)  .name("Filter by degree");   // u, v, w where (u, v) and (u, w) are edges in graph, v < w  DataSet<Tuple3<K, K, K>> triplets = filteredByDegree  .groupBy(0)  .sortGroup(1, Order.*ASCENDING*)  .reduceGroup(new GenerateTriplets<K>())  .setParallelism(littleParallelism)  .name("Generate triplets");   // u, v, w where (u, v), (u, w), and (v, w) are edges in graph, v < w  DataSet<Tuple3<K, K, K>> triangles = triplets  .join(filteredByID, JoinOperatorBase.JoinHint.*REPARTITION\_HASH\_SECOND*)  .where(1, 2)  .equalTo(0, 1)  .with(new ProjectTriangles<K>())  .setParallelism(littleParallelism)  .name("Triangle listing");   if (sortTriangleVertices.get()) {  triangles = triangles  .map(new SortTriangleVertices<K>())  .name("Sort triangle vertices");  }   return triangles; } |
| 参考 | <http://blog.csdn.net/u010376788/article/details/50223157>  <http://book.51cto.com/art/201409/451628.htm> |

# 3. Spark图算法

表2. Spark 图算法表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 编号 | 名称 | 描述 | 应用 | 难度 |
|  | Label Propagation |  |  |  |
|  | Connected Components |  |  |  |
|  | PageRank |  |  |  |
|  | Shortest Paths |  |  |  |
|  | Strongly Connected Components |  |  |  |
|  | SVDPPlusPlus |  |  |  |
|  | Triangle Count |  |  |  |

# #. 问题

1. Flink和Spark的编程模型趋同，而且在调试的时候发现Flink也属于延迟计算类型，只记录数据源和算子，并没有真正的去执行计算（需要深入理解）=> 总结Flink和Spark的编程模型，并且给出系统全面的比较。（10）interesting.

(1) <https://www.zhihu.com/question/30151872>

(2)<http://blog.madhukaraphatak.com/introduction-to-flink-for-spark-developers-flink-vs-spark/>

2. Flink和Spark的图算法的布局都趋于相同。（6）

数据分析和运算是Python/R/Matlab的强项，现在在Flink/Spark上构建数据分析运算模块是否合理？ give up.

3. 可以扩展的地方：（8）

AF9算法中Flink的实现版本是比较直观的，网上还给出了一种基于矩阵运算的算法，是否能够加速？=>Flink是否支持矩阵运算？=>图论中有很多利用矩阵来进行运算的例子，而Flink的算法是建立在顶点和边的集合上的运算，如果能够将这种运算转换为矩阵运算，构建矩阵运算模型来代替图运算模型，是否能够加速呢？maybe difficult, give up.

4. Flink的Graph模型是建立在DataSet上的，即批处理上，如何在流处理上建立Graph模型，或者跑Graph相关算法？（10）key problem

5. 之前上过一门课程是社交媒体大数据挖掘，里面提到非常多的算法，是否可以在Flink/Spark上实现其中的某些有价值的算法？（10）maybe interesting.

6. 如果提出针对图算法的测试Benchmark，算不算成果？（10）

7. Benchmark

测试方法：

（1）顶点和边的数量

（2）边的稀疏度

（3）机器的扩展性

数据来源：

（1）真实的DataSet，twitter等；

（2）人为生成的。

性能评价：

扩展性：数据量增大时的曲线图。

可靠性：出错？

可重复性：多次跑同样的数据集，结果是否一致或相差不大？

8. 优化思路

算法角度：

系统角度：

结合。

9. 当提交的任务占用内存资源过多时，会报OOM错误，整个集群宕机。这种问题如何侦测，解决？（云计算环境下，如何检测用户提交的任务不是非法的，不会影响到整个集群）

10. 现在有很多更加专一和强大的计算框架，为什么不直接用例如Giraph / Pregel等这样的图计算框架而要研究Flink?

# #. 记录

## 2016.10.17

**郑莹莹**

1. Iterative Graph Processing

barrier是如何实现的？barrier加在哪里？

2. scatter-gather

发送时，发送的顺序？

接收时，接收有顺序？接收了所有的之后（缓存）再更新，还是接收了之后就立刻更新。

3. 单源点最短路径 => 任意两两之间的最短路径的实现

4. 三种算法的区别？

scatter：以顶点为视角，发送边；

gather：以边为视角，收集边；

两段式，三段式的设计区别和举例。

**深入理解框架模型的设计理念。**

5. Scatter-Gather 和任何节点通信？如何理解？

任意的，满足交换律，结合律。

**重点：深入理解三种模型。**

6. Graph能否画出来？（D3?）

7. Spark VS Flink算法实现的对比=>迭代模型的对比。

8. 强连通图的实现细节，边排序？

9. 算法使用什么模式来实现的？读取数据的过程和迭代计算的过程？

## 2016.11.07

1. 基本算子的使用：map, flatMap, join等等。

2. 算法在执行时考虑并行度问题。

3. 幂律分布。

4. 和师兄讨论毕设问题。

## 2016.11.11

1. 分片的策略如何。

2. 数据分布图。数据流图。数据之间的依赖。

3. 如何进行pipeline划分的。