

《MapReduce海量数据处理》

实验4 图的三角形计数

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **学院名称** | **：** | 计算机科学与技术系 | | | | | |
| **小组成员** | **：** | 林喜鹏 李朝阳 饶璐 唐玉婷 | | | | | |
| **学号** | **：** | MG1833046 MF1833035 MF1833059 MF1833070 | | | | | |
| **小组组长** | **：** | 林喜鹏 | | | | | |
| **时间** | **：** | 2018 | 年 | 11 | 月 | 11 | 日 |

1. 实验要求

**实验背景**

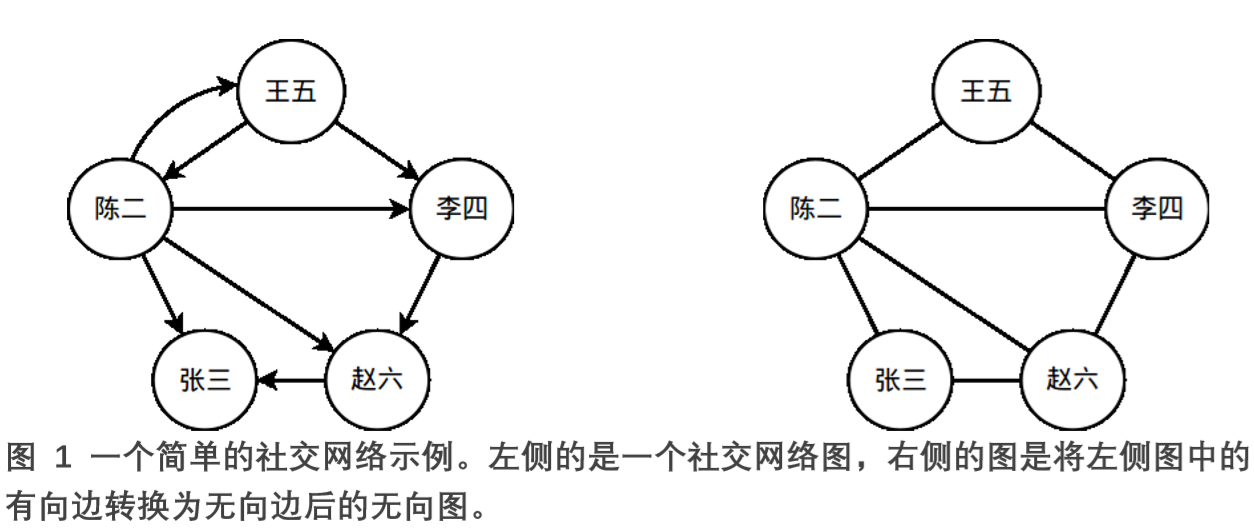
图的三角形计数问题是一个基本的图计算问题，是很多复杂网络分析（比如社交网络 分析）的基础。目前图的三角形计数问题已经成为了 Spark 系统中 GraphX 图计算库所提 供的一个算法级 API。本次实验任务就是要在 Hadoop 系统上实现图的三角形计数任务。

**实验任务**

一个社交网络可以看做是一张图（离散数学中的图）。社交网络中的人对应于图的顶 点；社交网络中的人际关系对应于图中的边。在本次实验任务中，我们只考虑一种关系— —用户之间的关注关系。假设“王五”在 Twitter/微博中关注了“李四”，则在社交网络 图中，有一条对应的从“王五”指向“李四”的有向边。图 1 中展示了一个简单的社交网 络图，人之间的关注关系通过图中的有向边标识了出来。。 本次的实验任务就是在给定的社交网络图中，统计图中所有三角形的数量。在统计前，需 要先进行有向边到无向边的转换，依据如下逻辑转换：

IF ( A→B) OR (B→A) THEN A-B

“A→B”表示从顶点 A 到顶点 B 有一条有向边。A-B 表示顶点 A 和顶点 B 之间有一条 无向边。一个示例见图 1，图 1 右侧的图就是左侧的图去除边方向后对应的无向图。 请在无向图上统计三角形的个数。在图 1 的例子中，一共有 3 个三角形。 本次实验将提供一个 Twitter 局部关系图[1]作为输入数据（给出的图是有向图），请统 计该图对应的无向图中的三角形个数。

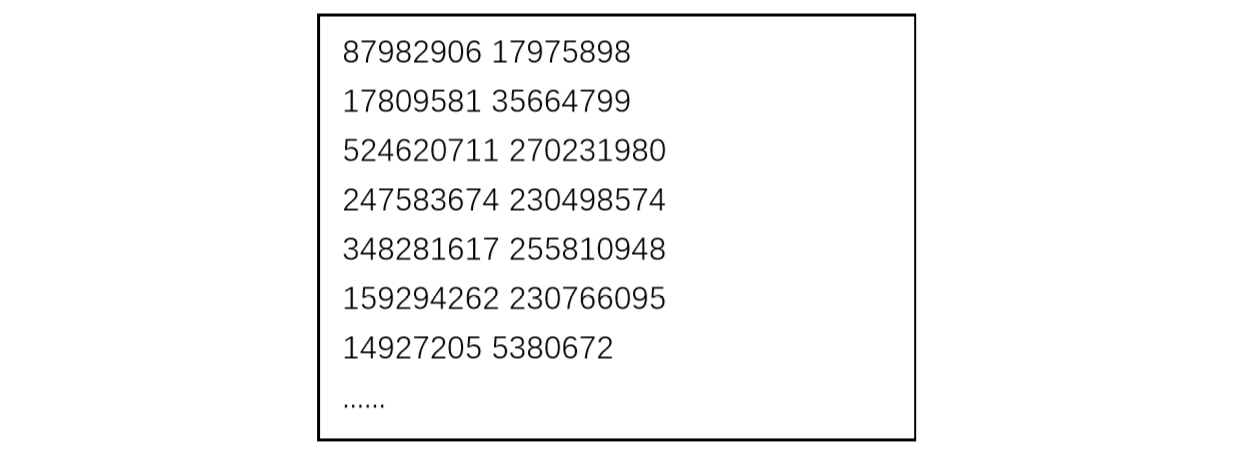


**输入要求**

输入数据仅一个文件。该文件由若干行组成，每一行由两个以空格分隔的整数组成：

A B

A，B 分别是两个顶点的 ID。这一行记录表示图中具有一条由 A 到 B 的有向边。整个 图的结构由该文件唯一确定。 下面的框中是文件部分内容的示例：



**输出要求**

请将统计出的三角形个数输出到一个 HDFS 的文件中。输出文件的路径可以自定义， 但请在实验报告中说明你所采用的输出文件路径。

**其他要求**

本次实验需要多个 MapReduce Job 才能完成。请再编写一个 Driver 程序，将多个 MapReduce Job 组织在一个程序内自动执行。

**选做内容**

1. 选做 1：本次实验中，我们在做有向边到无向边的转换时，依赖如下的逻辑：

IF ( A→B) OR (B→A) THEN A-B

现在请将该逻辑替换为：

IF ( A→B) AND (B→A) THEN A-B

再次进行统计，看看统计出的三角形的个数是多少。

1. 选做2：挑战更大的数据集！使用 Google+的社交关系网数据集作为输入数据集。 请报告计算出的三角形个数和总用时。输入文件的 HDFS 路径为：

/data/graphTriangleCount/gplus\_combined.unique.txt

注意：该文件中可能有类似“x x”这样的自己指向自己的边！

**提交格式**

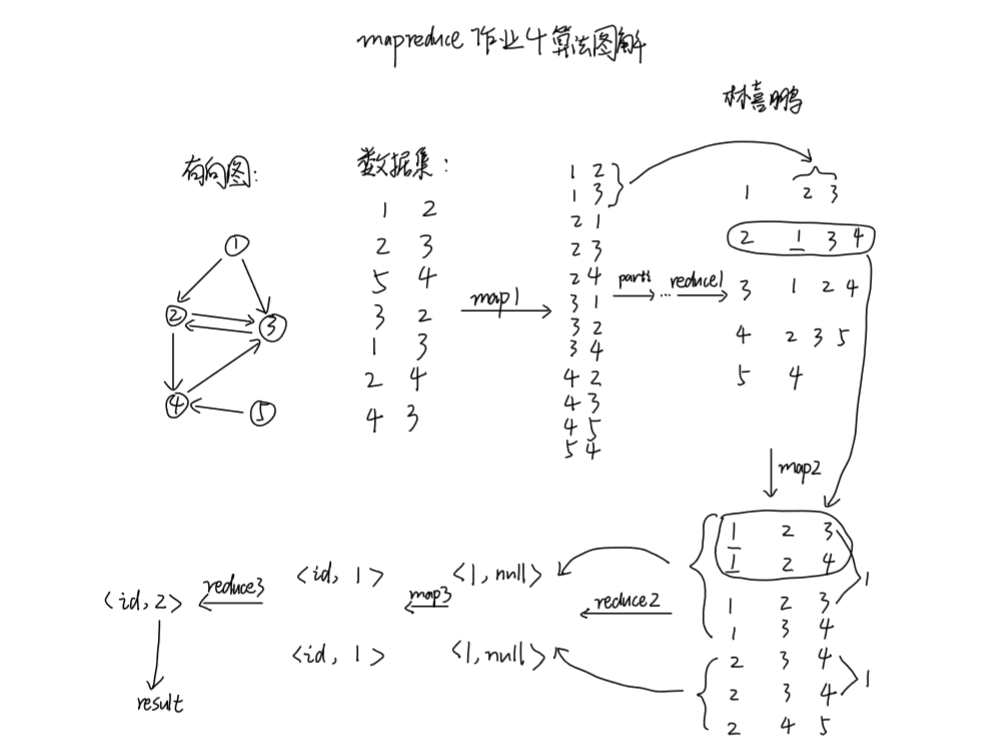
压缩包,包括:源代码、JAR 包、JAR 包执行说明及实验报告。

1. Map和Reduce的设计思路

统计三角形个数共由3个MapReduce job完成。Job1将有向图转换为无向图，并得到点的邻接关系；job2得到满足上述要求的所有三点关系，并由此得出三角形个数；job3将job2得到的三角形个数汇总，得到三角形总个数。

具体地，首先是将有向图转换为无向图，得到每个顶点的所有邻接点，如A->(B, C, D)。对得到的邻接点，考虑如B->A->C，B与A相邻，A与C相邻，则BAC可能构成三角形。考虑另一邻接情况C->(A, B)，得到B->C->A，因B与C相邻，C与A相邻；由于B->A->C及B->C->A，则确定BAC构成三角形；在生成三点关系时，如果将后两个点按照由小到大顺序排序，则如果得到2个后两个点值和顺序完全一样的情况，则可以确定其构成三角形。考虑A、B、C在无向图中构成三角形情况，ABC、ACB、BAC、BCA、CAB、CBA，上一步已经确定后两个点的大小情况，此时如果再加限制：第一个点一定比后两个点值小，则可以唯一确定三个点构成的三角形。照以上思路即可找出所有三角形。

为了更加便于理解，我画了算法在简单测试数据下的图解：



具体的Map和Reduce的设计如下：

Job1:

1. Mapper

输入:

key: Object, 行偏移

value: Text, 行内容，有向边A->B，由空格分开

输出:

key: Text, 两个点，由空格分开；对每个A->B，分别输出A->B和B->A。

value: NullWritable, null。

将有向图转换为无向图，即对任一有向边，输出两条边。

后面为了利于编程实现，需要其邻接点有序，故将两个点一起作为key输出。

2）Partitioner

将同一点的“A->B”发送到同一reduce节点，方便reduce时收集此顶点的所有邻接点。

3）Reducer:

输入:

key: Text, 邻接关系，A-B（注意这里是无向边）

value: NullWritable, null

输出:

key : Text, 顶点

value: Text, 此顶点的所有邻接点，有序，以space隔开

将得到的邻接关系收集起来，得到某一顶点的所有邻接点。

Job2:

1. Mapper

输入：

key: Object, 行偏移

value: Text, 顶点和其所有邻接点，顶点后以tab分开，邻接点间以space分开

输出：

key: Text, 第一个顶点

value: Text, “第二个顶点和第三个顶点“的集合，第二个和第三个顶点间以space分开，不同”第二个顶点和第三个顶点“以tab分开。

得到三点关系，即某点、其邻接点、邻接点的邻接点。三点间满足的条件之前已经说明。实现上即，要求第一个点一定比后两个点值小，后两个点按照值由小到大排序，即一定是三点值由小到大。为了减少map输出，对于一次map，将同一第一个点的所有满足条件的“第二个和第三个点“组合在一起输出。

1. Reducer

输入：

key: Text, 第一个顶点

value: Text, 多个“第二个顶点和第三个顶点”集合

输出：

key: IntWritable, 此reducer得到的总的三角形个数

value: NullWritable, null

Reduce得到的是所有相同第一个顶点的，“第二个顶点和第三个顶点“的所有map输出情况。这时如果存在相同的”第二个顶点和第三个顶点“，就确定构成三角形。由于题目要求不需要知道具体哪三个点构成三角形，因此可以统计出所有map输出个数，以及去除重复后的个数，其差值即为重复个数，也就是三角形个数。

Job3：

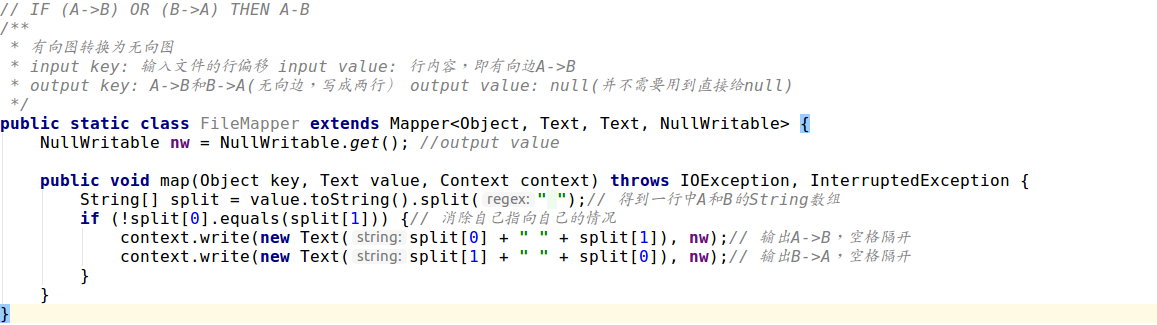
Job3将job2得到的三角形个数相加汇总，得到三角形总个数。

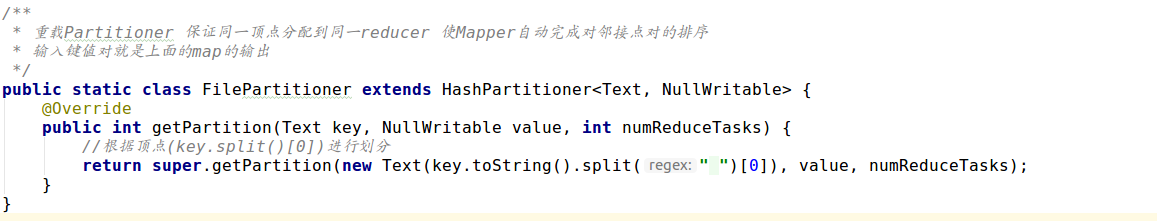
1. Mapper

读取上一步生成的文件，得到三角形个数。为了简化编程，mapper的key设为同一个值，这样reducer会一次性得到全部的个数。

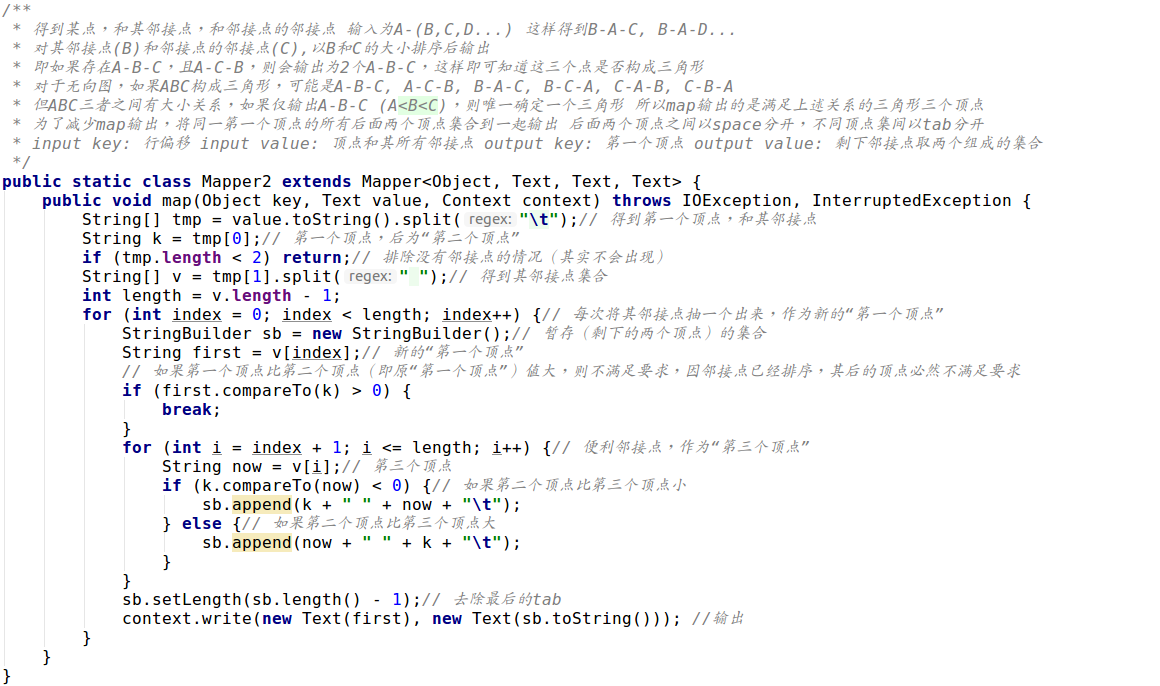
1. 将所有三角形个数汇总，得到总的三角形个数。
2. Map和Reduce代码

Job1:





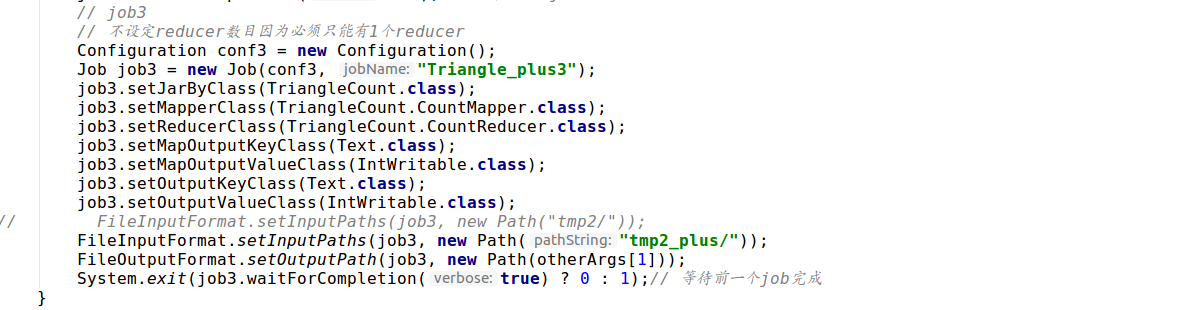
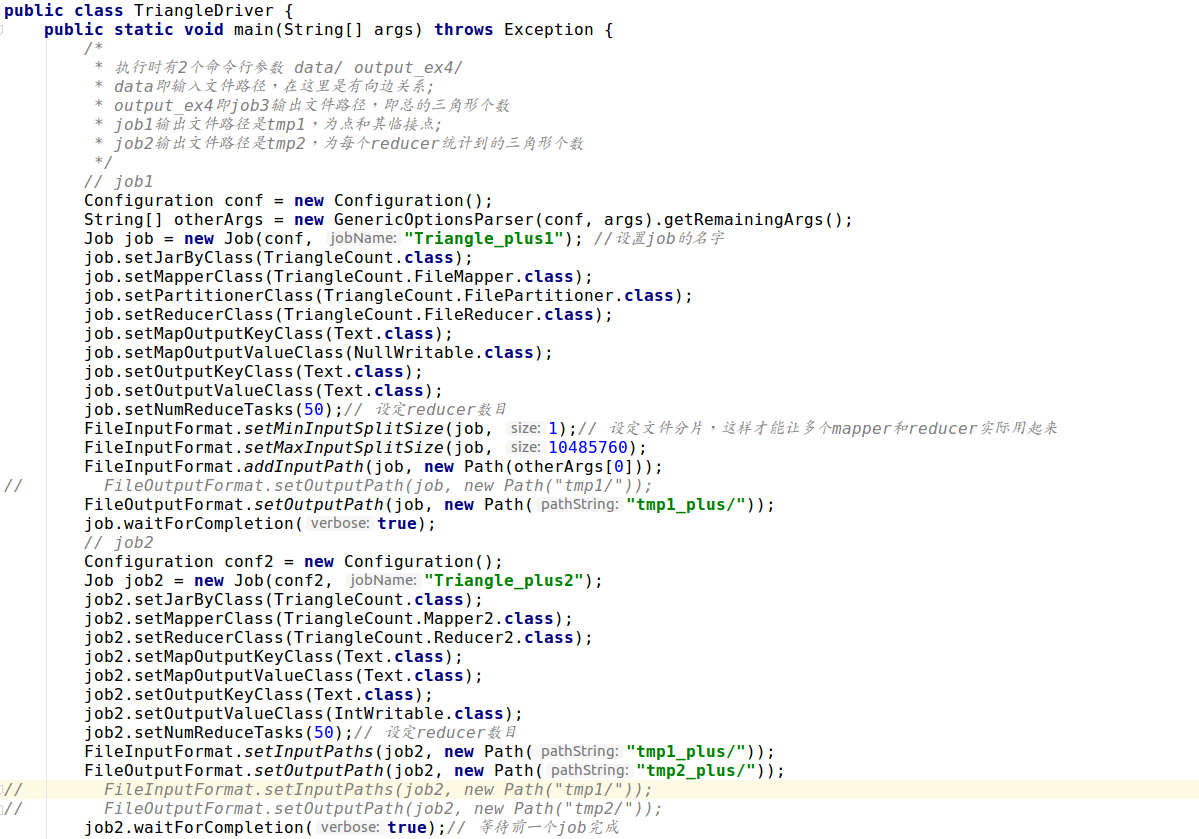
Job2：



Job3：



Driver：



1. 实验结果

|  |  |  |
| --- | --- | --- |
| 数据集 | 三角形个数 | 集群上运行时间 |
| Twitter | 13082506 | 97s |
| Google+（选做） | 1073677742 | 1h 2m 43s |

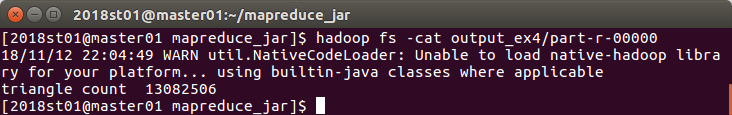
表1 实验结果-单向一条有向边即认为构成无向边

|  |  |  |
| --- | --- | --- |
| 数据集 | 三角形个数 | 集群上运行时间 |
| Twitter | 3653762 | 72s |
| Google+（选做） | 124552628 | 47m 47s |

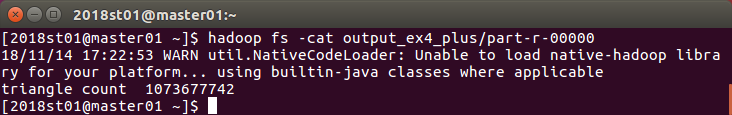
表2 选做题实验结果-双向有向边才认为构成无向边

1. 输出结果文件

Twitter数据集实验结果文件（路径见截图）

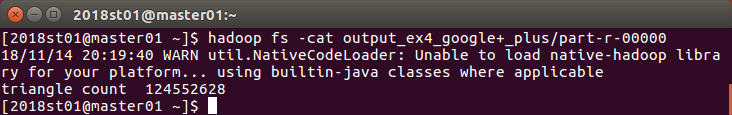


Google+数据集实验结果文件

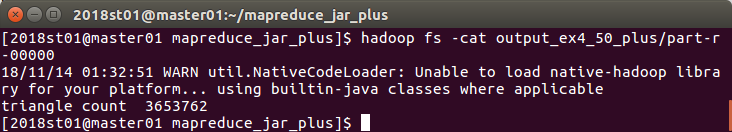


选做题

Twitter数据集实验结果文件



Google+数据集实验结果文件



1. 程序运行性能的分析

程序在Twitter数据集上的表现还是不错的，考虑到运行的实际时间还跟集群的使用情况以及设置的reduce数量有比较大的关系，我们做了三次对比实验，得到程序的执行时间与reduce数量的关系如下表：

|  |  |  |  |
| --- | --- | --- | --- |
| reduce数量 | 5×5 | 50×50 | 500×500 |
| 运行时间 | 207s | **97s** | 518s |

表3 程序在Twitter数据集上运行时间

而在Google+数据集上的表现为：

|  |  |  |  |
| --- | --- | --- | --- |
| reduce数量 | 5×5 | 50×50（Or） | 500×500(And) |
| 运行时间 | - | **1h 2m 43s** | 47m 47s |

分析：

首先，程序性能的瓶颈主要集中在Job2（各个Job详细的执行时间见第8节的WebUI执行报告截图），这是可以理解的，因为Job2实际上承担了计算三角形个数的主要工作（计算相同的三角关系的个数）。

其次，在Twitter这个比较小的数据集上，reduce的数量过大反而会降低性能，可能的原因是，多个reduce的时间收益抵不上对中间生成的文件的IO读取时间（因为有500个reduce，所以会有500个中间文件的生成），这反而成为了制约运行性能的瓶颈。而像Google+这个大的数据集上，使用更多的reduce数量还是对性能有一定的提升的（当然也可能是集群之前比较拥挤）。

1. 性能不足与可能改进之处
2. 程序的Job2性能还是比较一般，是整个程序性能的瓶颈；

可能改进的方案：

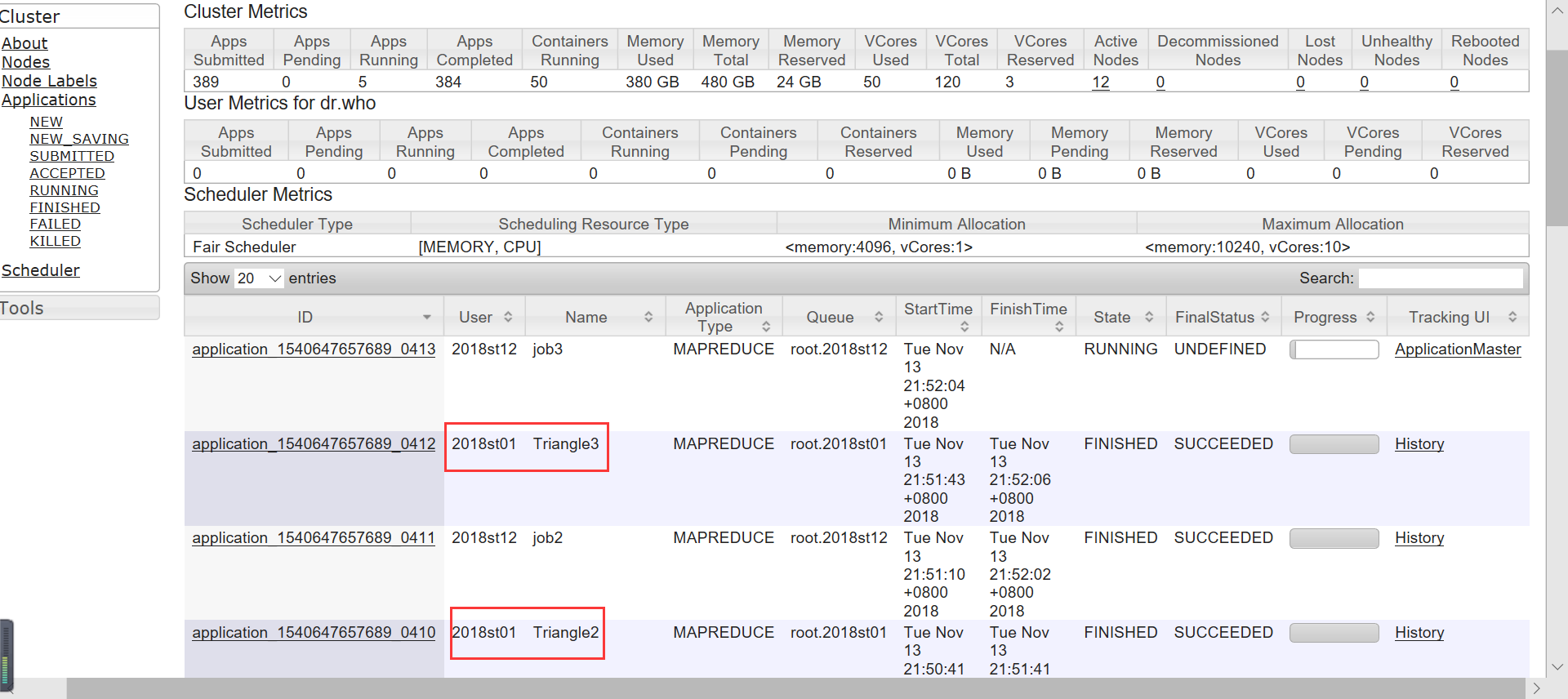
想办法减少Mapper2的输入（例如对Reducer1的输出进一步优化）或者改进Reducer2判断重复的三角关系的方法（暂时没想到更好的）。

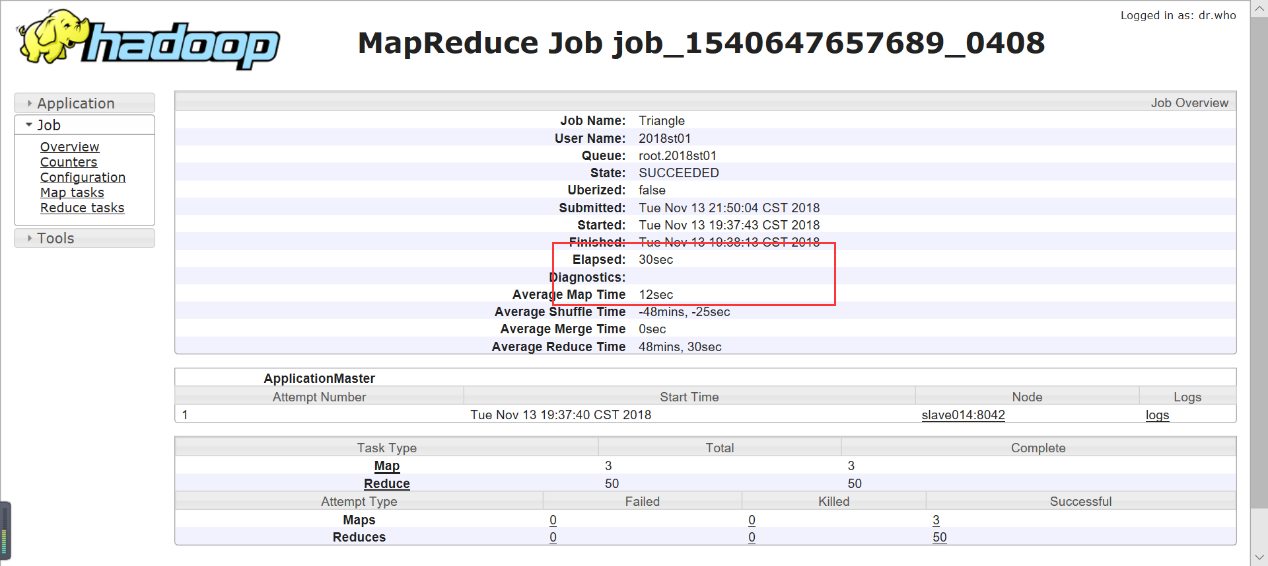
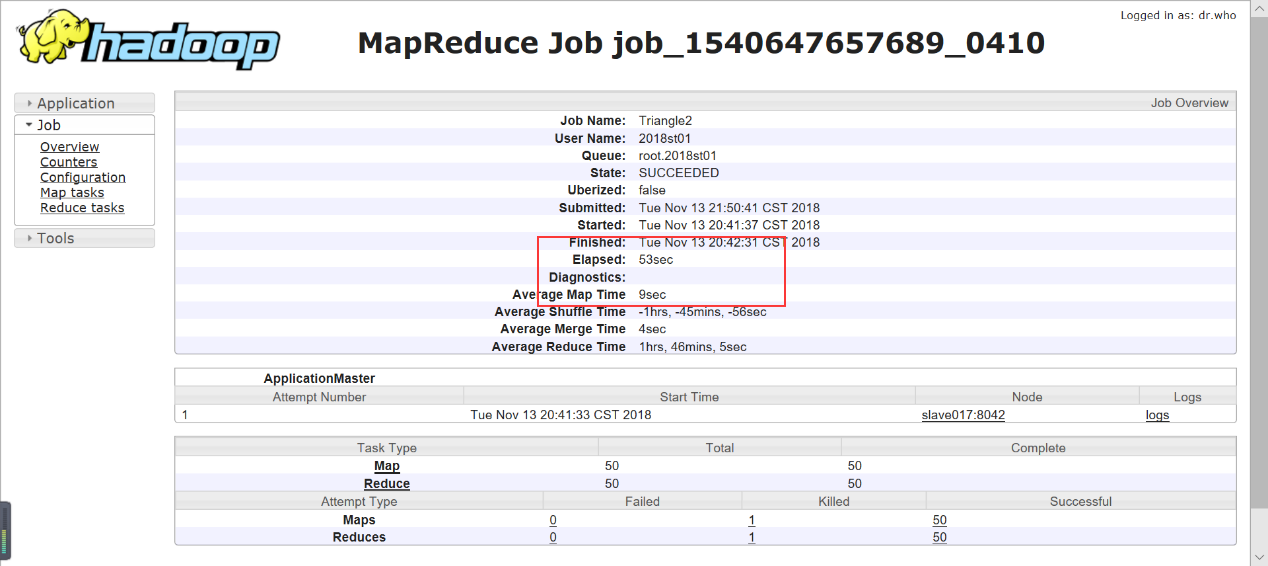
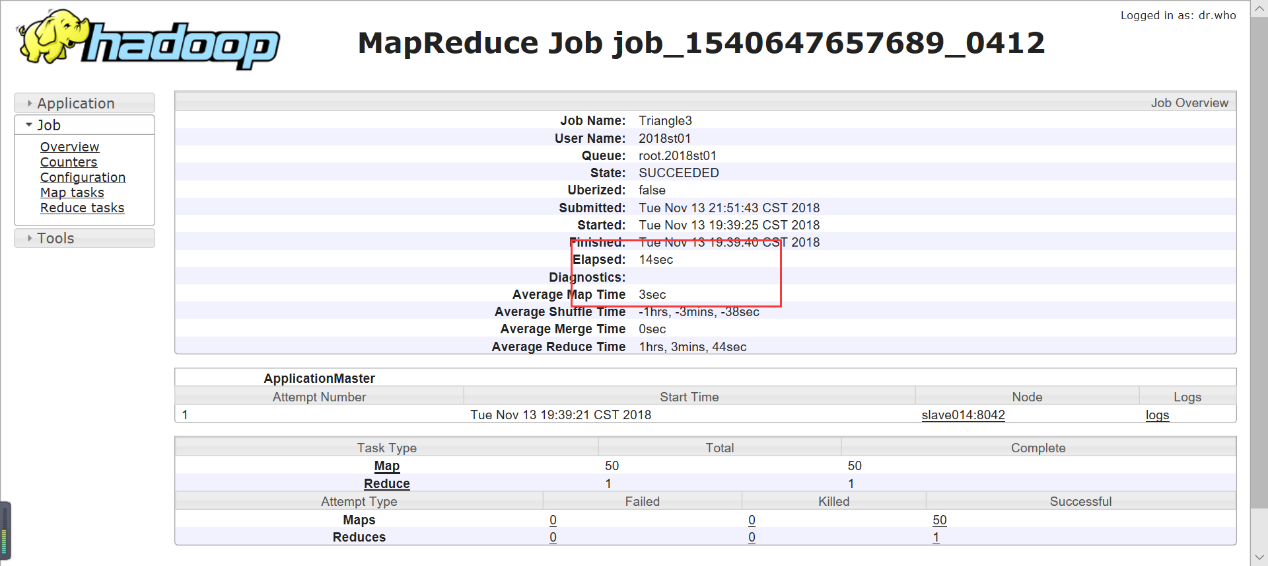
1. Job3仅仅是将不同reduce的结果相加，却相比一般的脚本更加耗时；

可能的改进方案：

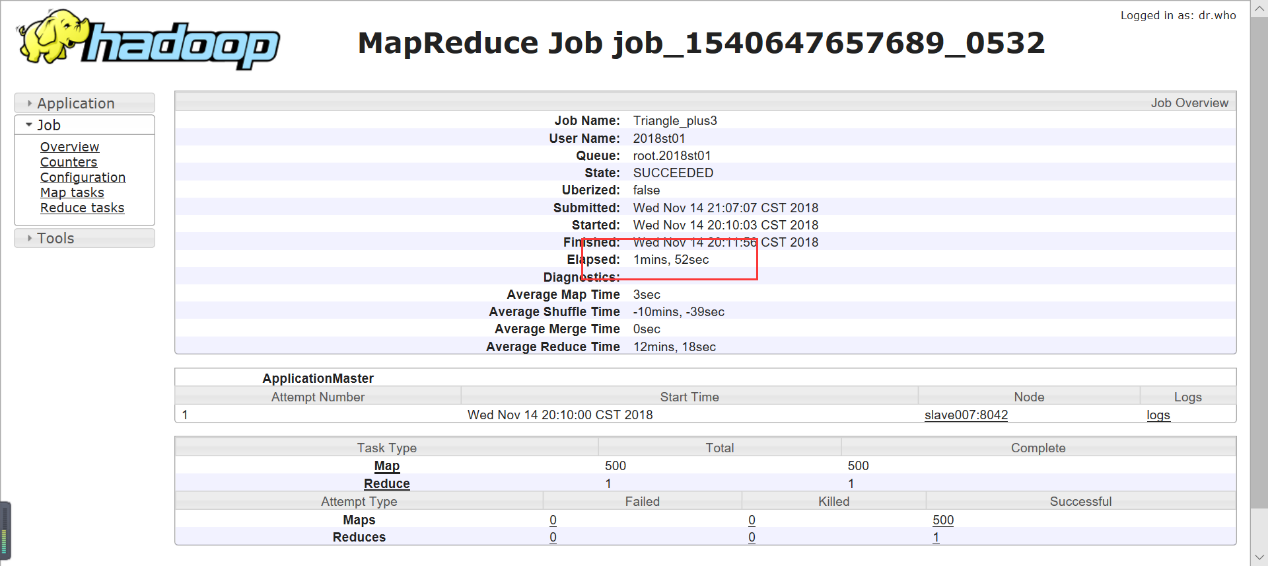
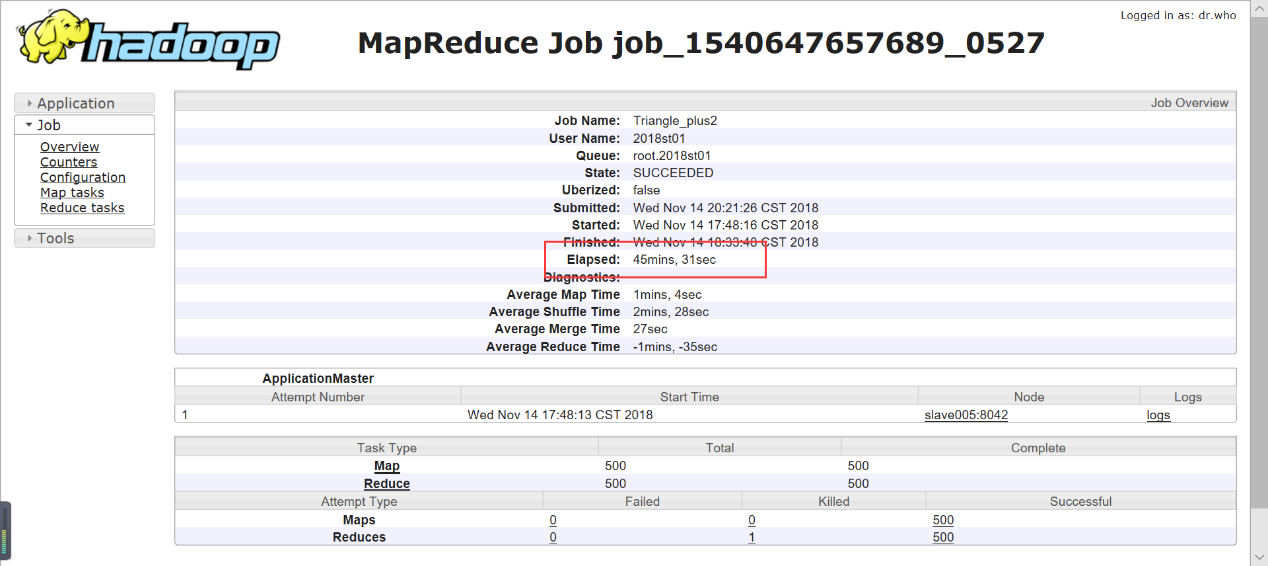
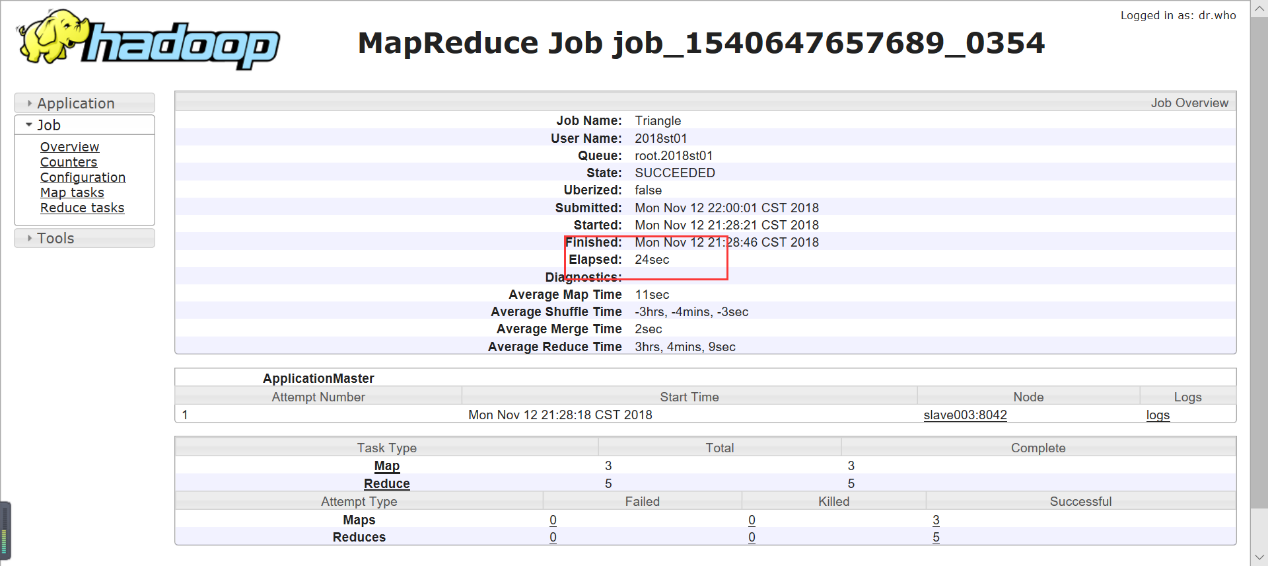
去掉Job3，自己写个脚本进行求和（这与Job3的执行时间相比可以忽略）。

1. WebUI执行报告截图





选做题Google+数据集



1. 实验总结