

《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**

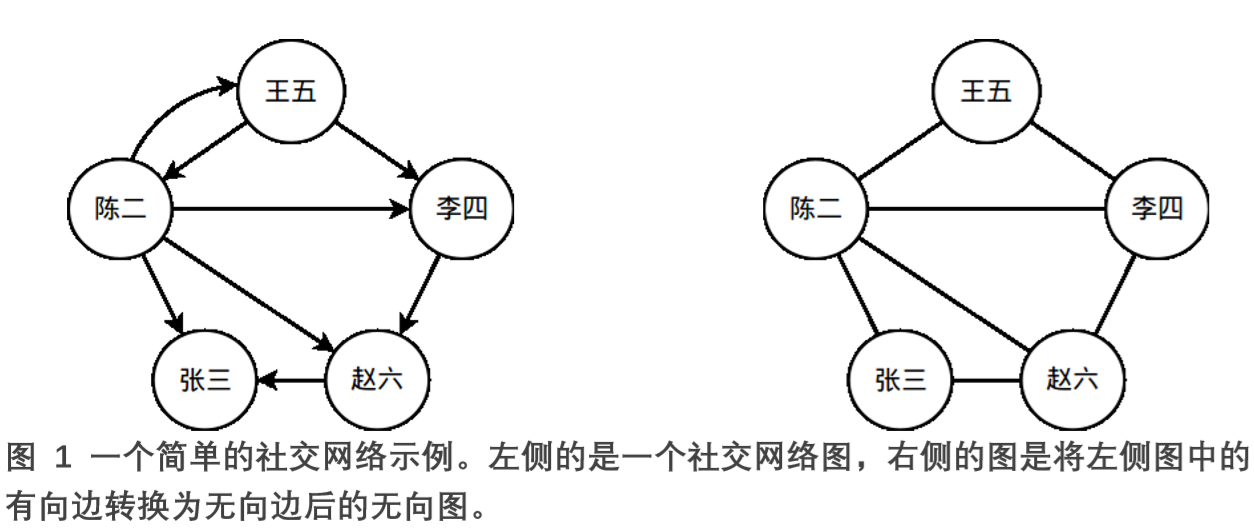
图的三角形计数问题是一个基本的图计算问题，是很多复杂网络分析（比如社交网络 分析）的基础。目前图的三角形计数问题已经成为了 Spark 系统中 GraphX 图计算库所提 供的一个算法级 API。本次实验任务就是要在 Hadoop 系统上实现图的三角形计数任务。

**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。

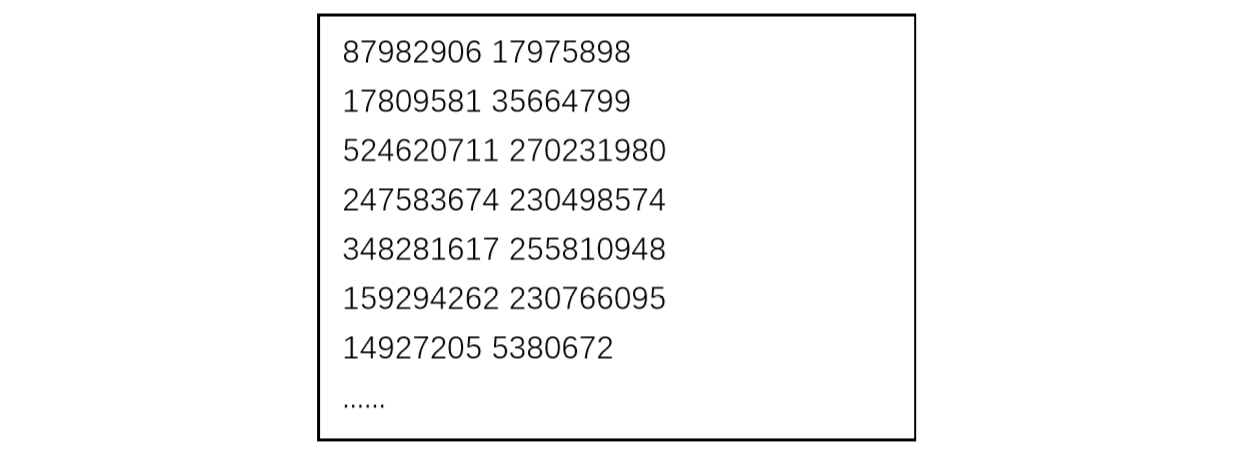


**输入要求**

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

要使得两个计算任务请在同一个 MapReduce Job 中完成，则除了基本的Mapper和Reducer之外，还必须重载Combiner和Partitioner,具体如下：

Map:

1. Mapper

输入:key是文件当前的偏移量，value是文件当前行内容

输出:key是word#filename，value是1

获取当前处理文件名filename,然后对value值进行切分，得到多个word值，此时将每个word和filename拼接到一起作为输出key，其计数值为1，也即value为1。

1. Combiner

输入:key是word#filename，value为对应的[1, 1, 1, …]

输出:key是word#filename，value是同一key下的累加和

作用是将Mapper输出的中间结果相同key部分的value累加，减少向Reduce节点传输的数据量。

Reduce:

1. Partitioner

输入:key是word#filename，value则是累加和

输出:key是word#filename，value则为累加和

这里是为了保证同一word都在同一Reduce节点进行处理。

1. Reducer

输入:key是word#filename，vaule为[累加和1, 累加和2, …]

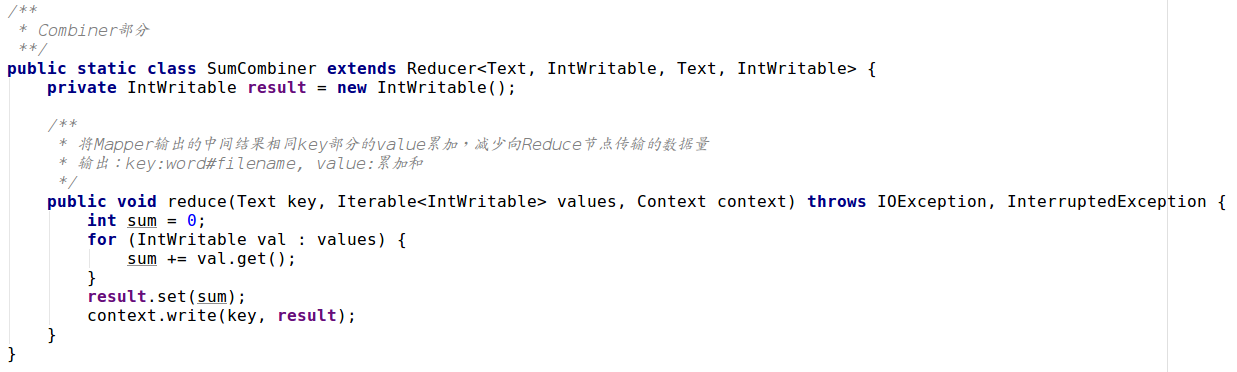
输出:key是word，value为[平均出现次数，filename:词频;filename:词频;…]

Reducer稍微麻烦一点，首先需要利用Reduce节点输入的key值都是有序的，将key拆分，对于同一word，每次都保存其filename和词频，并统计其总出现次数和总出现文档数；当同一word处理完后，计算平均出现次数，将其与filename及其词频作为value输出。

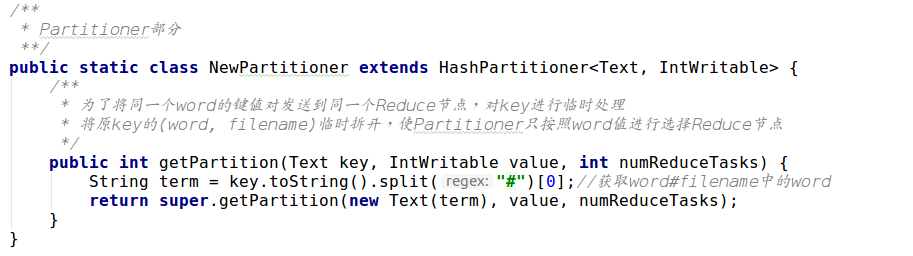
1. Map和Reduce代码

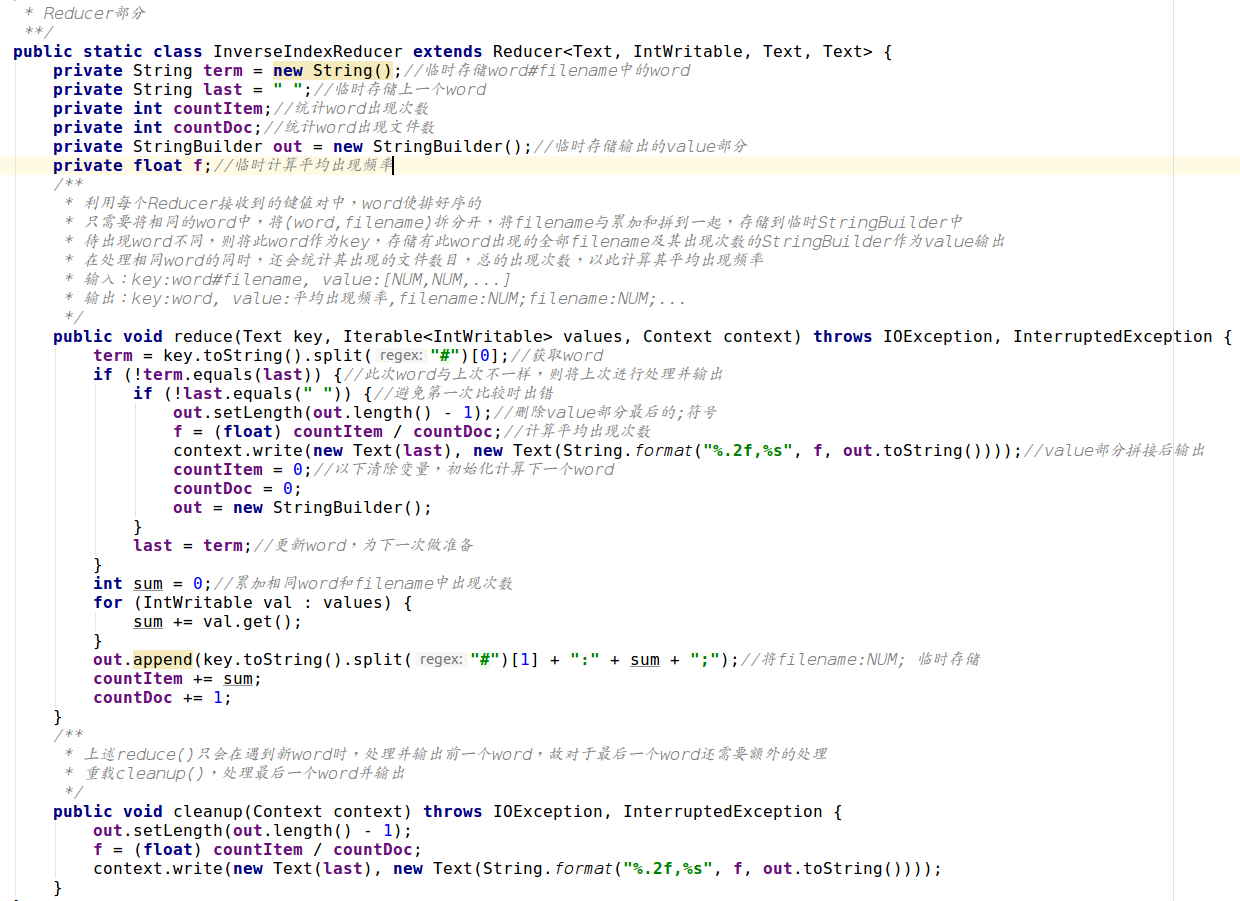
Map:





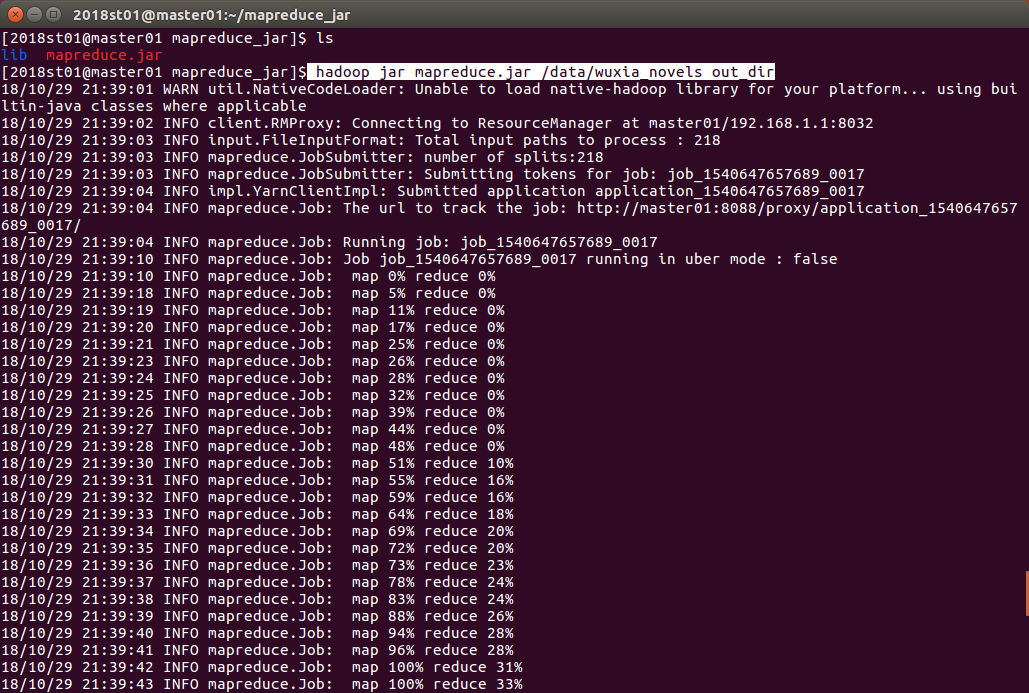
Reduce:

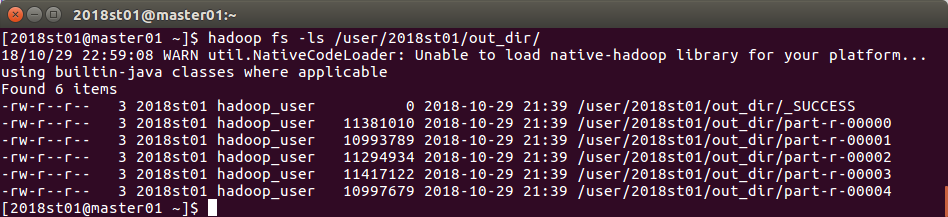




1. 本机实现后将jar包传到集群进行测试并检查输出文件

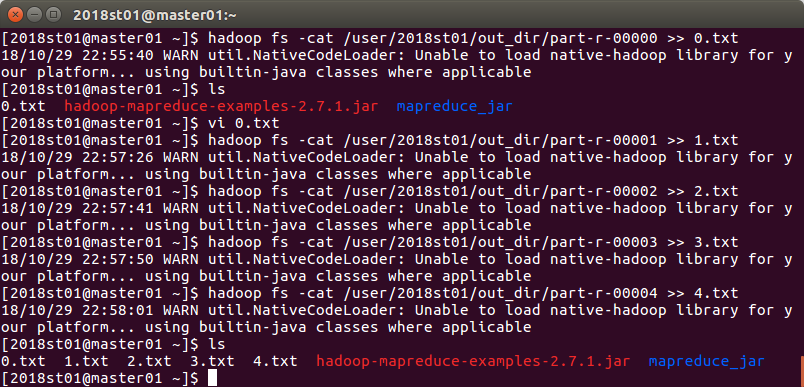
使用5个reduce节点，执行情况如下：



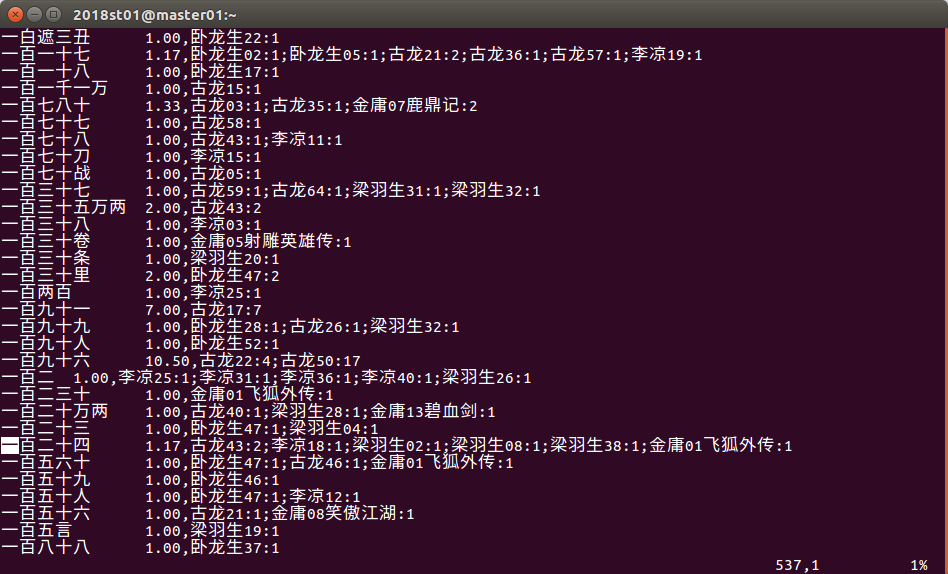


1. 输出结果文件的部分截图

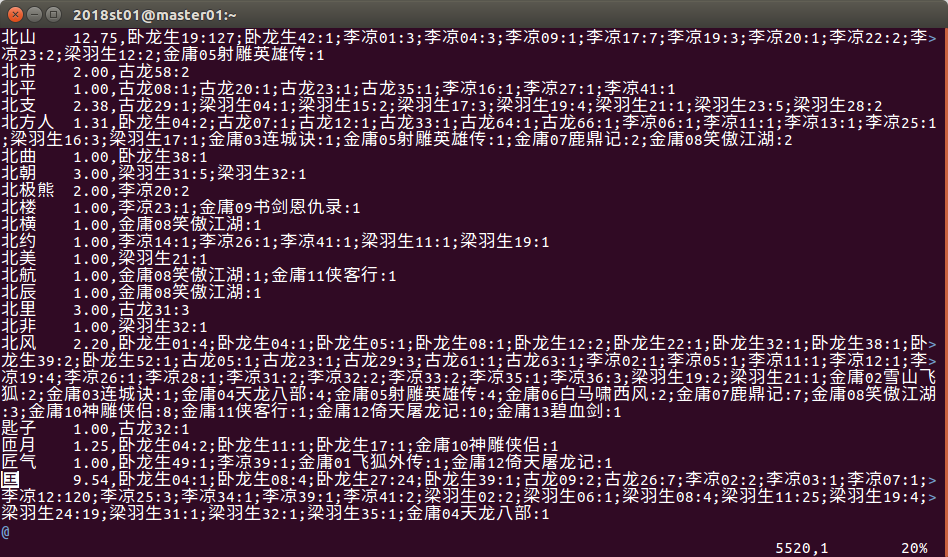
输出文件内容过多，为了便于查看，我把它们重定向到txt文件，这样也方便用vim进行搜索



查看part-00000部分截图:

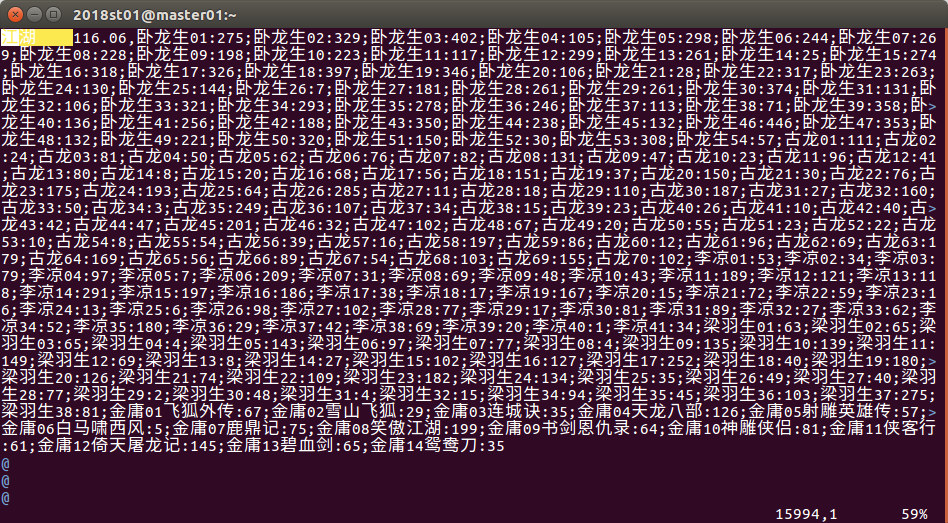


查看part-00002部分截图:

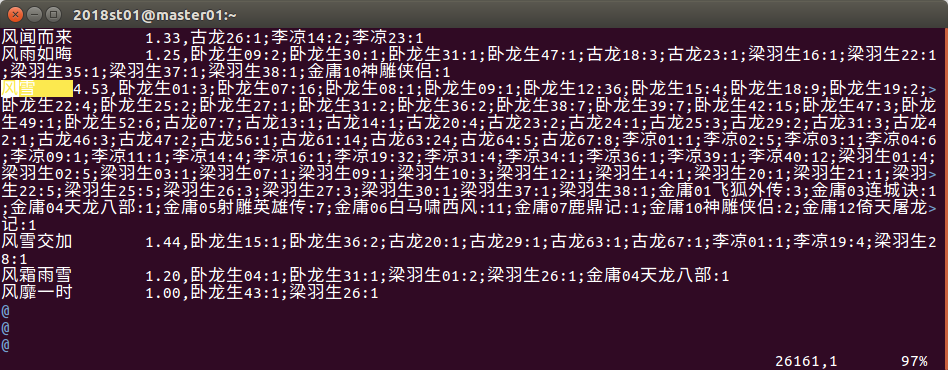


1. “江湖”、“风雪”两个单词的输出结果

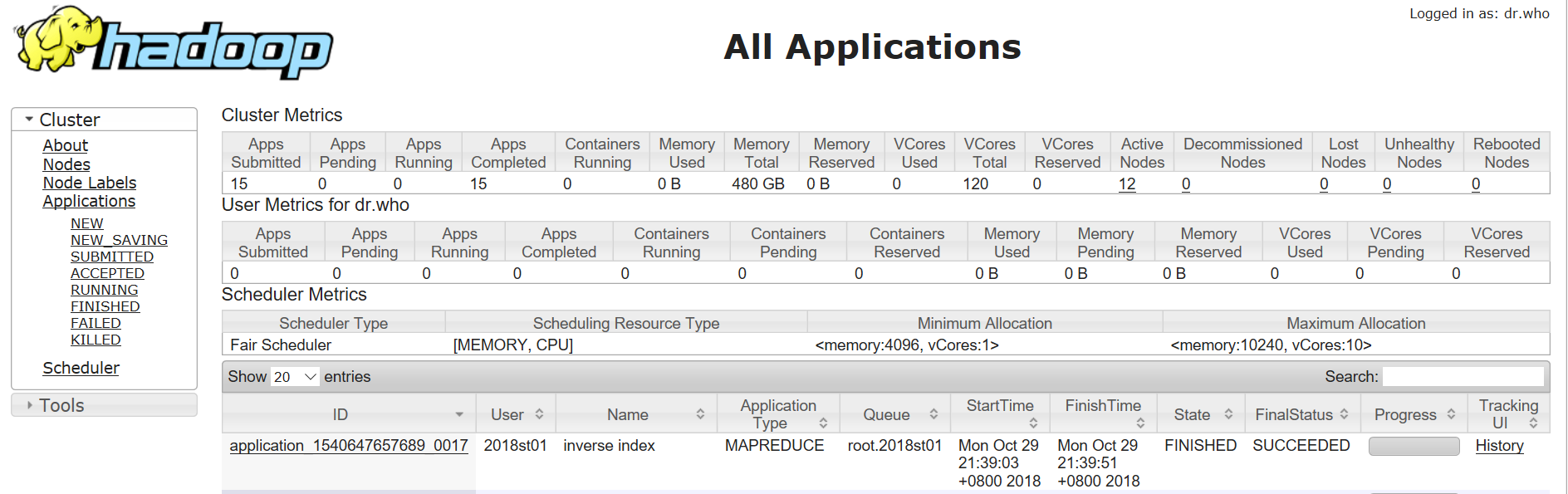
“江湖”:

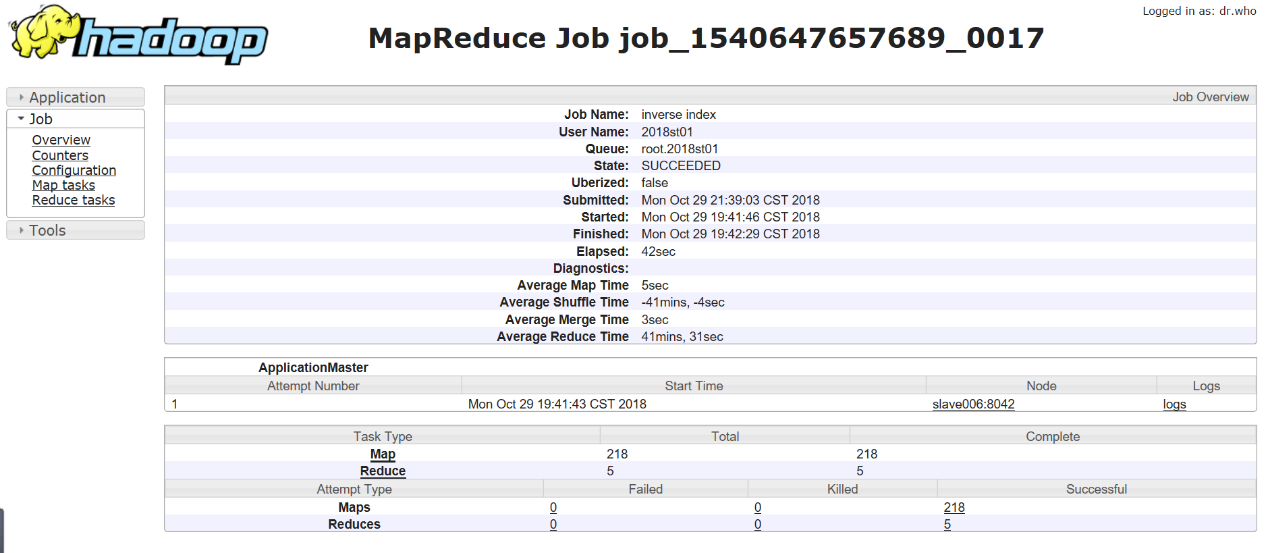


“风雪”:



1. WebUI执行报告截图





1. 实验总结

本次实验主要包括统计词语的倒排索引并输出带词频属性的倒排索引和计算每个词语的平均出现次数两个任务。实验的主要难点在于如何设计和实现Map和Reduce，这部分在课件上其实已经有非常详细的介绍了。实验结果达到了预期实验要求。通过本次实验我们提高了对MapReduce的算法设计与实现有了更加深刻认识，同时对Combiner和Partitioner的作用也有了进一步的理解。