K-频繁项集挖掘并行化算法

1. 设计思路

本题目的要求是在规定的标准数据集上进行频繁项集的挖掘，因而自然而然地想到了两种著名的算法：Apriori算法和FP-Growth算法。由于测试数据集约含1800万条事务记录，考虑到Apriori算法在计算过程中需要重复扫描数据集（如题要求1-频繁项集到8-频繁项集的挖掘，则需要扫描8遍数据集）以及产生大量的候选集，而FP-Growth算法只需进行2遍数据集的扫描，以将事务记录压缩成树来推导频繁项集的方式来代替候选集的生成，效率更高。因而最终决定采用并行化的FP-Growth算法来解决这一问题。

2. 实现方案

本算法的实现主要分为以下主要5个步骤：

2.1 挖掘1-频繁项集并编码

利用简单的类单词计数算法求出不同项的出现频数，并根据支持度进行过滤，然后将过滤后的项存储到一个哈希表oneItemMap中，该哈希表的键为项，值为根据项出现次数从大到小排列的编码（即出现次数最多的项编码为0，以此类推）。

2.2 根据编码后的1-频繁项集，对原始数据进行过滤和分组

根据上一步得到的哈希表对原始数据中的每一条事务记录进行过滤，只保留住包含在哈希表中的项，并根据哈希表中每一项的编号从小到大排序。然后将每一条过滤后的事务记录进行分组（根据groupNum值的设置），分组方案为：假设1-频繁项集按从大到小排列为{a, b, c, d, e, f, g}，将每条事务分为两组，组编号分别为0和1。分组0中包含的项目为{a, b, c, d, e, f, g}，分组1中包含的项目为{e, f, g}。针对每一条事务，若其中包含了组0和组1的项目，则输出两条事务并分别归为组0和组1，组0为该条事务，组1为截取相应部分得到的事务；若其中只含有组1的项，则该条事务只归为组1；若其中只含有组0的项，则只归为组0。该步骤为算法的Mapper。

注：根据运行时环境的配置，groupNum定为1-频繁项集的项数

2.3 对分组后的数据分别压缩建树

针对每个分组中的数据采用传统FP-Growth算法中建树的过程来建立FP树。

2.4 对每一棵FP树进行频繁项集的挖掘

同样采用FP-Growth算法中的对FP树的挖掘方式得到每一棵FP树的频繁项集。该步骤与上一步骤共同组成算法的Reducer。

2.5 整合每棵树的频繁项集，得到最终结果

通过groupByKey将所有分组的频繁项集进行整合，并进行去重、按要求格式化之后得到最终的结果。

3. 测试结果

3.1 jar包使用说明

spark-submit

—class tongji.dataspark.p3.Problem3

—master <master-url>

DataSpark_problem3.jar

input-path

output-path

3.2 任务提交

```
[bookcold@namenode ~]$ cd mathpanda/
[bookcold@namenode mathpanda]$ nohup ../spark-1.0.1/bin/spark-submit --class ton
gji.dataspark.p3.Problem3 --master spark://192.168.1.18:7077 ./DataSpark_problem
3.jar  /user/mathpanda/apriori_data.bat `pwd`/problem3_result &
[1] 13181
[bookcold@namenode mathpanda]$ nohup: 忽略输入并把输出追加到"nohup.out"

[bookcold@namenode mathpanda]$
```

3.3 运行情况

Completed Applications

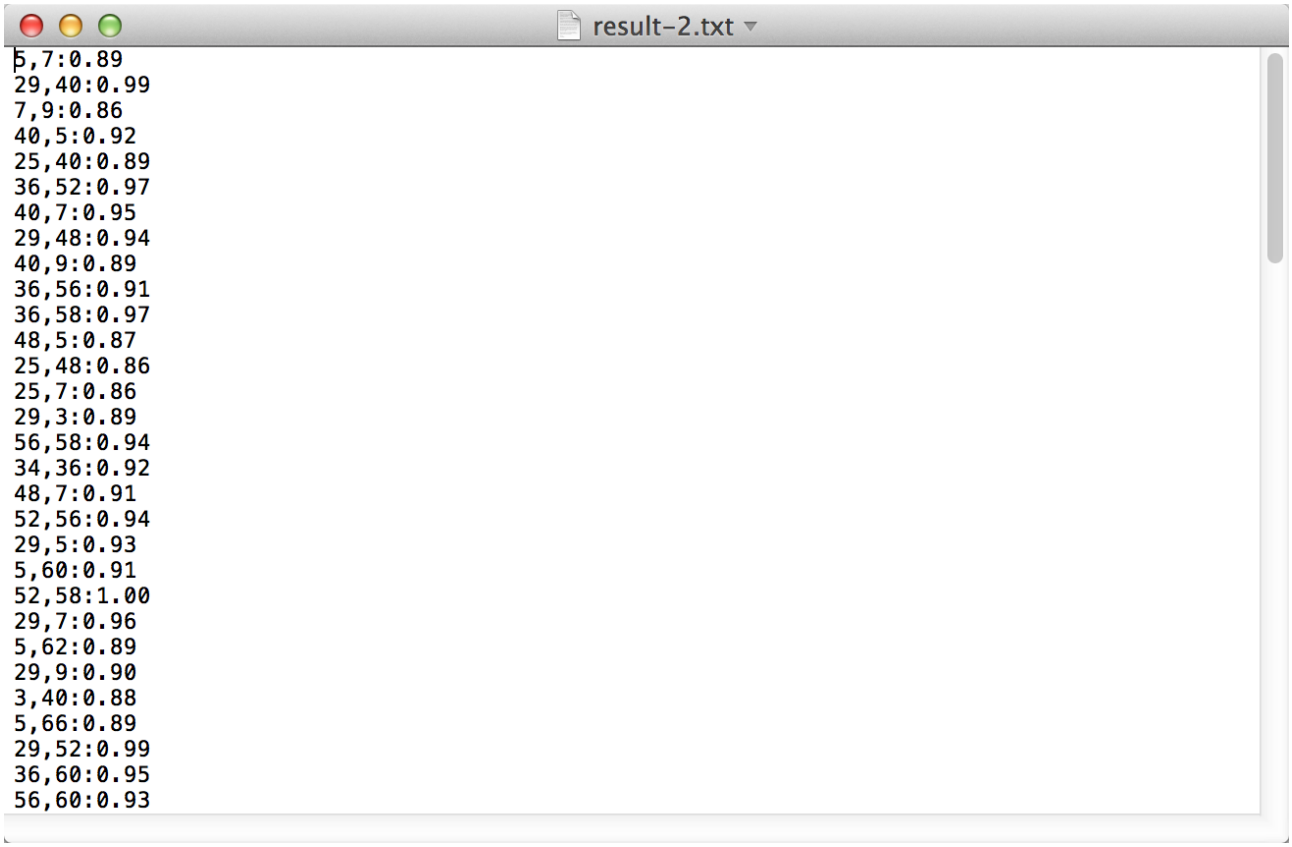
ID	Name	Cores	Memory per Node	Submitted Time	User	State	Duration
app-20141211203427-0015	Problem3	192	40.0 GB	2014/12/11 20:34:27	bookcold	FINISHED	2.2 h
app-20141211173102-0014	Problem3	192	40.0 GB	2014/12/11 17:31:02	bookcold	FINISHED	2.5 h
app-20141208144919-0008	Problem3	192	40.0 GB	2014/12/08 14:49:19	bookcold	FINISHED	5.6 h

minPatitions=16、64、128时，Duration=5.6h、2.5h、2.2h

3.4 执行结果

下图为2-频繁项集的文件截图：

（具体结果请查看PPT或附件problem3_result)


result-2.txt
5,7:0.89
29,40:0.99
7,9:0.86
40,5:0.92
25,40:0.89
36,52:0.97
40,7:0.95
29,48:0.94
40,9:0.89
36,56:0.91
36,58:0.97
48,5:0.87
25,48:0.86
25,7:0.86
29,3:0.89
56,58:0.94
34,36:0.92
48,7:0.91
52,56:0.94
29,5:0.93
5,60:0.91
52,58:1.00
29,7:0.96
5,62:0.89
29,9:0.90
3,40:0.88
5,66:0.89
29,52:0.99
36,60:0.95
56,60:0.93