基于六度理论的演员关系搜索

吴行行
朱雨

2017-07-19

1 数据集

数据集是上映电影和演员数据，共有 70 万部电影，96 万位演员。首先进行预处理和统计，将原数据集处理成 “演员 + 出演电影列表” 和 “电影 + 演员列表” 两个数据文件。统计得到平均每个演员出演 3.99 部电影，每个电影平均有 8.5 个演员。

2 算法

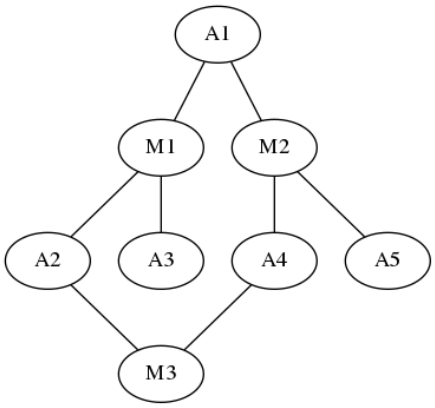
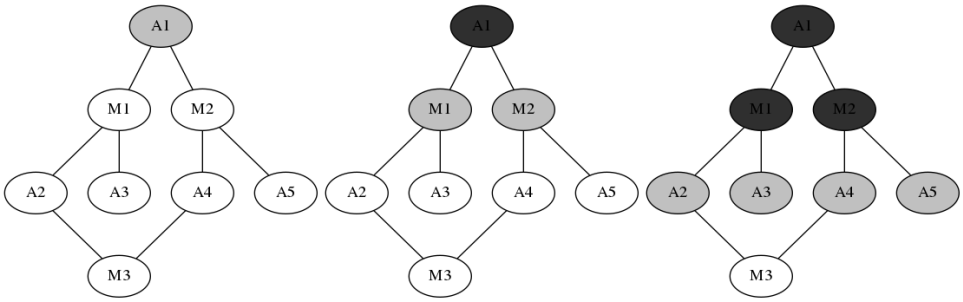


Figure 1: model

图1是算法采用的模型，其中 A 表示演员，M 表示电影，演员出演电影则两个节点存在一条边。所有数据构成一个无向有环图，问题转化成在图中查找两个 A 节点之间的路径。采用 Breadth-First-Search 算法。



(a) bfs:1 (b) bfs:2 (c) bfs:3

Figure 2: BFS

灰色点表示 open 点，黑色点表示 close 点，白色点表示 unknown 点。循环进行下去，直到发现目标节点，或者遍历深度达到上限值 (6*2+1=13)。

3 Hadoop 实现

3.1 输入数据

所有节点及其子节点 <name, children>;

已经处理过的节点 <name,children,distance,status,parent>。

3.2 Map 阶段

所有节点发送 <name,children>

已经处理过的节点发送 <name,distance,status,parent>

open 节点发送 <child,distance,parent>

3.3 Reduce 阶段

接收到 children 数据，用于组成新的 open 节点

接受到 distance,status,parent 数据，则已经是 close 节点

接收到 parent's distance 数据，则成为新的 open 节点

写入所有的 close 节点数据和新的 open 节点数据：<name,children,distance,status,parent>

4 优化

4.1 优化 close 节点数据

对于 close 节点，不再需要 children,distance,status,parent 数据，只需要保留 name 即可。能够减小写入文件大小和 M-R 之间传输的数据。

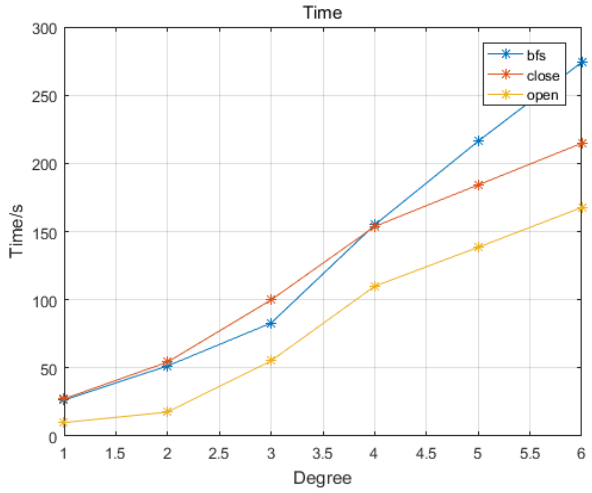
4.2 优化 open 数据

Map 阶段所有节点都需要发送 <name,children> 数据，但是 Reduce 中只有新的 open 节点才需要这个数据。

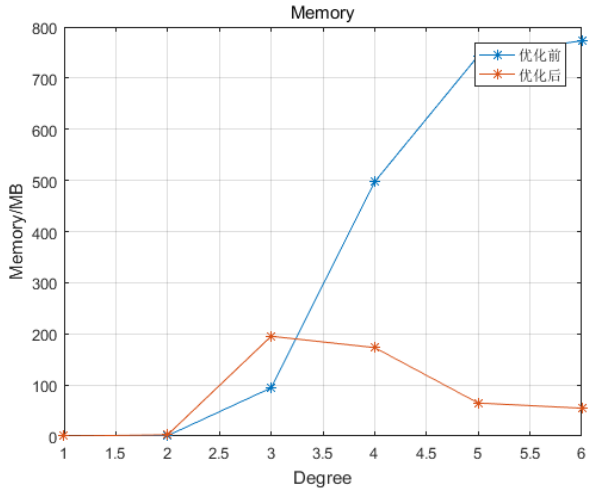
上一轮中保存下一轮会 open 的节点 name，Map 的时候只发送新 open 节点的数据。通过 CacheFile 传送这个数据。

当然 CacheFile 文件不能太大，否则会 heap 溢出并且查询时间过长，这里取深度为 6 以上的时候就不再使用 CacheFile，而是采用原来的办法发送所有数据。

5 结果



(a) time



(b) memory

Figure 3: 时间开销和空间开销