

华中科技大学

课程实验报告

课程名称： 大数据分析

专业班级： 计科 2003 班
学 号： U202015374
姓 名： 张隽翊
指导教师： 崔金华
报告日期： 2022 年 12 月 28 日

计算机科学与技术学院

目录

实验一 wordCount 算法及其实现.....	1
1.1 实验目的	1
1.2 实验内容	1
1.3 实验过程	1
1.3.1 编程思路.....	1
1.3.2 遇到的问题及解决方式.....	3
1.3.3 实验测试与结果分析.....	4
1.4 实验总结	6

实验一 wordCount 算法及其实现

1.1 实验目的

- 1、理解 map-reduce 算法思想与流程；
- 2、应用 map-reduce 思想解决 wordCount 问题；
- 3、（可选）掌握并应用 combine 与 shuffle 过程。

1.2 实验内容

提供 9 个预处理过的源文件（source01-09）模拟 9 个分布式节点，每个源文件中包含一百万个由英文、数字和字符（不包括逗号）构成的单词，单词由逗号与换行符分割。

要求应用 map-reduce 思想，模拟 9 个 map 节点与 3 个 reduce 节点实现 wordCount 功能，输出对应的 map 文件和最终的 reduce 结果文件。由于源文件较大，要求使用多线程来模拟分布式节点。

学有余力的同学可以在 map-reduce 的基础上添加 combine 与 shuffle 过程，并可以计算线程运行时间来考察这些过程对算法整体的影响。

提示：实现 shuffle 过程时应保证每个 reduce 节点的工作量尽量相当，来减少整体运行时间。

1.3 实验过程

1.3.1 编程思路

本实验模拟 9 个分布式节点使用的 map-reduce 方法包括 4 个环节：map→combine→shuffle→reduce。最后将各个统计结果组合为一个文件，即 result.csv 文件做为最终结果。各个环节的逻辑关系示意图如图 1.1 所示。

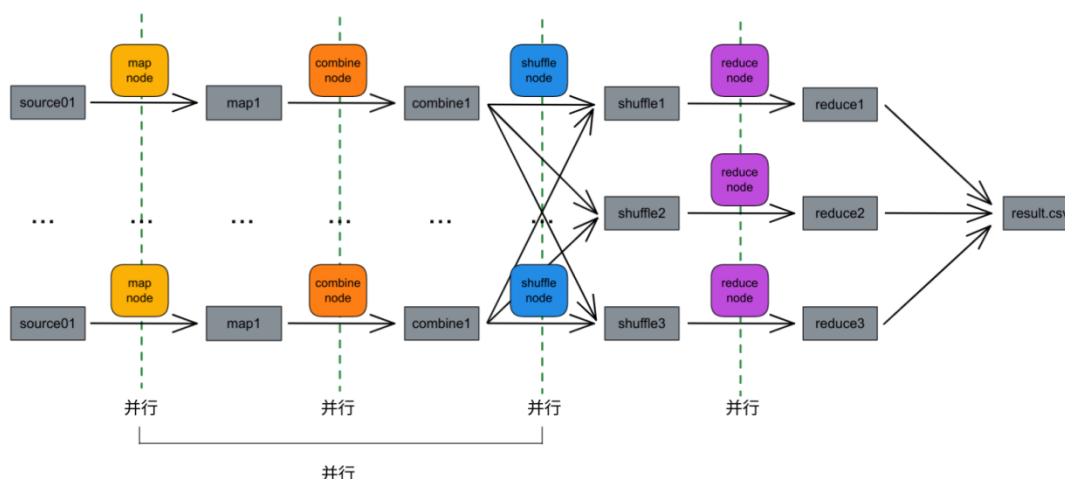


图 1.1 map-reduce 各环节之间关系示意图

（1）map 环节

map 过程将提供的 source 文件中的信息提取出来，形成<word, count>键值对。对于仅提取关键字的 map 操作，count 值应为 1。

```

1 nonritualistic, freedoot, coleslaws, argalas, Essam, Vinnie, immaculateness, Listerised, moisturizer, dicrotic
2 dingey's, Ajanta, Bajaj, audiovisuals, jargonised, recoaled, strictured, klunk, Gangamopteris, teenage
3 responsor, wingspreads, microtonally, uninsidious, Lundale, amphigory, filterableness, pimplyest, aflat, vibrancy
4 heathberries, penitentiaries, enspheres, Locrus, wire-drawn, dustrag, loquitur, Olm, jojoba, crurogenital

```

图 1.2 source01 文件内容

观察 source 文件，每行的单词以英文逗号“,”为分隔符，可以利用 Python 中的 yield 生成器读取 source 文件，以<word, 1>的形式存放在对应的 map 文件中，一共 9 个。实验中设立了 9 个 map 结点进行 map 操作，将 source 文件转换为 map 文件。

（2）combine 环节

combine 过程对 map 文件中预处理好的<word, 1>键值对进行统计，累加相同的关键词。实现上，首先建立一个字典，读取 map 文件中的记录录入字典，若 word 不存在于字典的键中则新建条目，否则将键对应的值加一。最后将字典中的键值对按顺序输出到 combine 文件内。实验中设立了 9 个 combine 结点进行 combine 操作，将 map 文件转换为 combine 文件。

（3）shuffle 环节

shuffle 过程对 combine 文件中的 word 进行分类，按照首字母不同划分至 3 个文件中。以 word 的首字母作为分类依据，首字母为 a~i 和 A~I 的键值对放入 shuffle1 文件中，首字母为 j~r 和 J~R 的键值对放入 shuffle2 文件中，其他的键值对放入 shuffle3 文件中。实验中设立了 9 个 shuffle 结点进行 shuffle 操作，将 combine 文件转换为 shuffle 文件。

（4）reduce 环节

reduce 过程对 shuffle 内的键值对进行统计，操作与 combine 过程一致。最后将字典中的键值对按顺序输出到 reduce 文件内。由于前面的 shuffle 环节已经对键值对进行了初步分类，这一步仅设立了 3 个 reduce 结点进行 reduce 操作，

将 shuffle 文件转换为 reduce 文件。

(5) 生成 result

将 reduce 环节产生的 3 个 reduce 文件合并为一个 result.csv 文件，操作与 combine 类似。

(6) 并行分析

每个环节各个结点之间可以并行，map、combine、shuffle 环节中前一个过程的结点完成工作后下一个过程的结点可以开始工作。Python 实现中，在每个环节启用设定数量的线程用于模拟结点，当线程完成工作时向消息队列发送序号，用于下一个环节接受并启动其线程。

1.3.2 遇到的问题及解决方式

(1) 线程计数器累加出错

问题：在为每个环节启用多个线程时，一开始对计数器没有加锁保护，导致运行时计数器累加不是原子操作，最终启动的线程数可能少于设定的数量。

解决：在对计数器进行加一操作时用 threading.Lock() 进行保护。

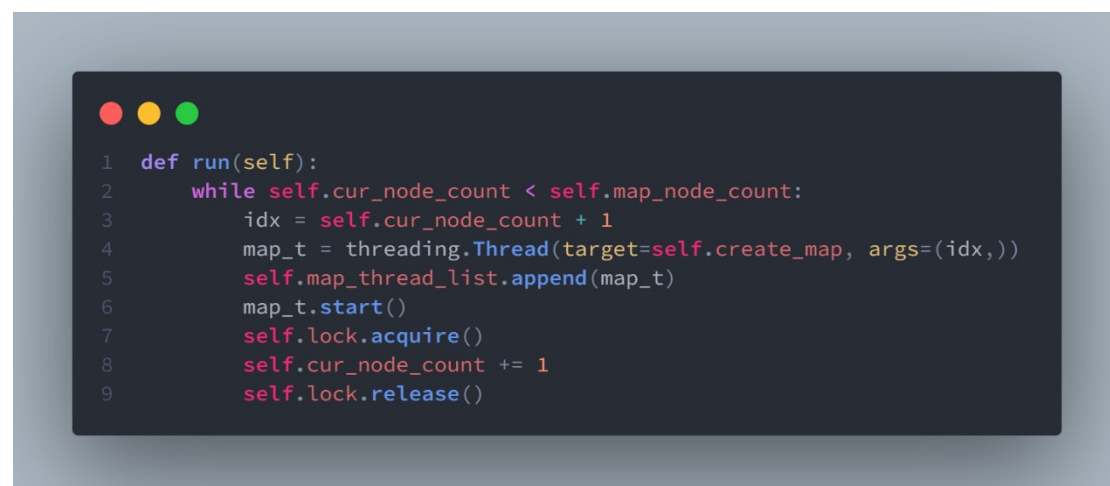


图 1.3 计数器加锁保护

(2) 线程间通信效率

问题：map、combine、shuffle 这三个环节中相邻两个环节之间的线程需要通信，前一个环节的线程完成工作后需要通知下一个环节的线程开始工作。采用了 Python 中的队列 Queue 后，每个线程向队列发送自己完成工作的信号，从队列中读取前一环节线程发送的信号，但整体上通信效率较低，对信号需要进行判断分析。

解决：为每相邻的两个环节单独设立通信队列，即 map、combine 之间通过 map_queue 进行通信，combine、shuffle 之间通过 combine_queue 进行通信。以 map 和 combine 之间的通信为例，当 map1 结点完成工作后，它会向 map_queue 队列发送“1”这个信号，则 combine 读取到“1”这一信号后会启动 combine1 结

点, 处理 map1 结点生成的 map1 文件。combine 和 shuffle 之间的通信与之相似。

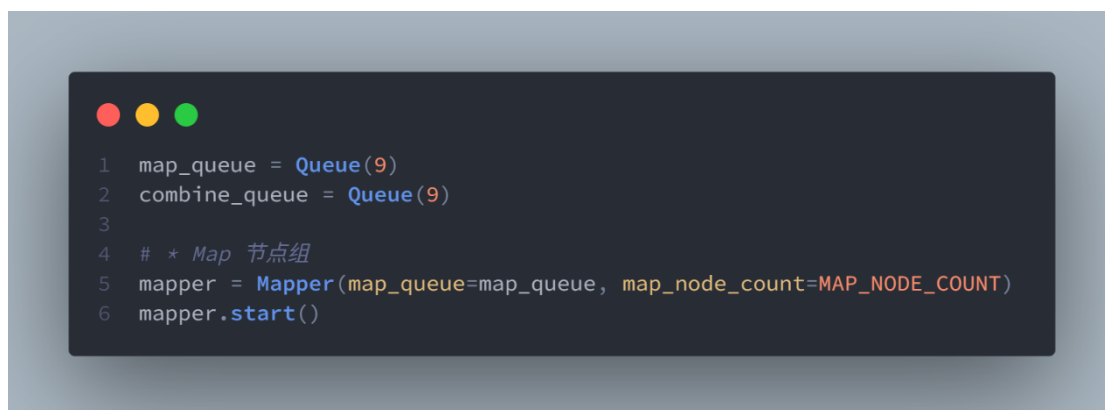


图 1.4 map 与 combine 间通信

1.3.3 实验测试与结果分析

(1) map 测试

map 环节结束后生成的 9 个 map 文件如图 1.5 所示。

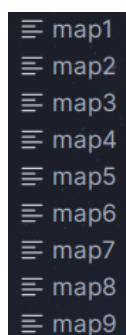


图 1.5 map 测试结果

其中 map1 文件的开始部分如图 1.6 所示。

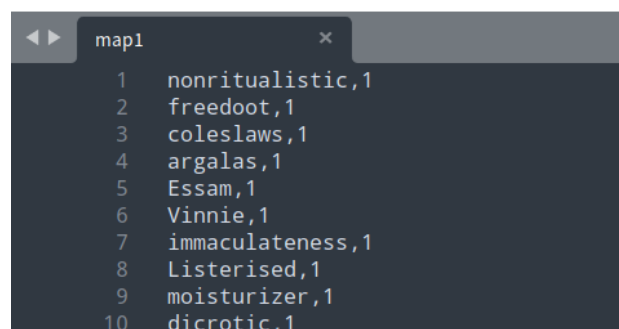


图 1.6 map1 文件开始部分

(2) combine 测试

combine 环节结束后生成的 9 个 combine 文件如图 1.7 所示。

```
≡ combine1
≡ combine2
≡ combine3
≡ combine4
≡ combine5
≡ combine6
≡ combine7
≡ combine8
≡ combine9
```

图 1.7 combine 测试结果

其中 combine1 文件的开始部分如图 1.8 所示。

```
combine1
1  &c,3
2  'll,3
3  'm,5
4  'midst,3
5  'mongst,1
6  'prentice,1
7  're,1
8  'sblood,1
9  'sbodikins,3
10 'sdeath,2
```

图 1.8 combine1 文件开始部分

(3) shuffle 测试

shuffle 环节结束后生成的 3 个 shuffle 文件如图 1.9 所示。

```
≡ shuffle1
≡ shuffle2
≡ shuffle3
```

图 1.9 shuffle 测试结果

其中 shuffle1 文件的开始部分如图 1.10 所示。

```
shuffle1
1  A&M,1
2  A&P,1
3  A'asia,3
4  A-1,3
5  A-OK,4
6  A-axes,3
7  A-axis,2
8  A-blast,2
9  A-day,3
10 A-flat,1
```

图 1.10 shuffle1 文件开始部分

(4) reduce 测试

reduce 环节结束后生成的 3 个 reduce 文件如图 1.11 所示。

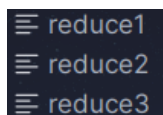
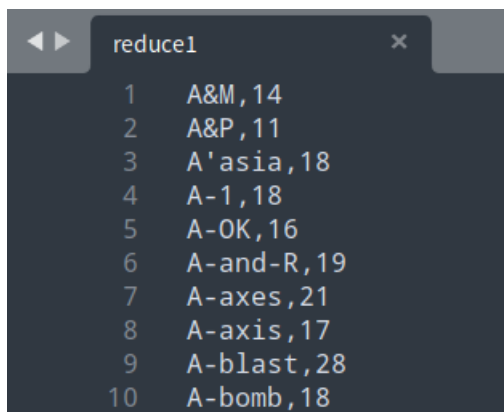


图 1.11 reduce 测试结果

其中 reduce1 文件的开始部分如图 1.12 所示。

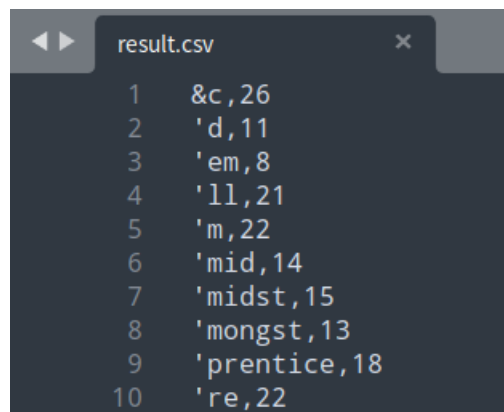


1	A&M,14
2	A&P,11
3	A'asia,18
4	A-1,18
5	A-OK,16
6	A-and-R,19
7	A-axes,21
8	A-axis,17
9	A-blast,28
10	A-bomb,18

图 1.12 reduce1 文件开始部分

(5) 结果测试

最终生成的 result.csv 文件如图 1.13 所示。



1	&c,26
2	'd,11
3	'em,8
4	'll,21
5	'm,22
6	'mid,14
7	'midst,15
8	'mongst,13
9	'prentice,18
10	're,22

图 1.13 result.csv 文件开始部分

综上所述，各个环节的结果均符合预期，最终生成的文件按照字典序排列，满足要求。

1.4 实验总结

本次实验实现了基于 map-reduce 算法的 wordCount 工作，了解了 map-reduce 算法中 map、combine、shuffle、reduce 等各个环节的功能和作用，同时感受到了 Python 多线程编程的优势。在 map-reduce 算法中，通过增加 combine 和 shuffle 操作，进一步分解任务，在多处理器环境下能够提高运行效率。

map-reduce 算法的核心思想是“分而治之”，即分解和并行，适用于大量复杂的任务处理场景也即大规模数据处理场景。map 负责“分”，将复杂的任务分

解为若干个相对简单的任务并行处理。可以拆分的前提是这些子任务可以并行计算，彼此之间几乎没有依赖关系。**reduce** 负责“合”，对 **map** 阶段的结果进行全局汇总。这种编程方式可以降低总的数据处理压力，各个环节协同合作可以降低数据出错带来的影响，单个任务的简单化减轻了单个结点的工作压力，还可以通过流水线技术进一步提高效率。