

**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](#_Toc58252267)

**[1.1实验目的](#_Toc58252268)** [1](#_Toc58252268)

**[1.2 实验内容](#_Toc58252269)** [1](#_Toc58252269)

**[1.3 实验过程](#_Toc58252270)** [1](#_Toc58252270)

[1.3.1 编程思路 1](#_Toc58252271)

[1.3.2 遇到的问题及解决方式 1](#_Toc58252272)

[1.3.3 实验测试与结果分析 1](#_Toc58252273)

**[1.4 实验总结](#_Toc58252274)** [1](#_Toc58252274)

# 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。

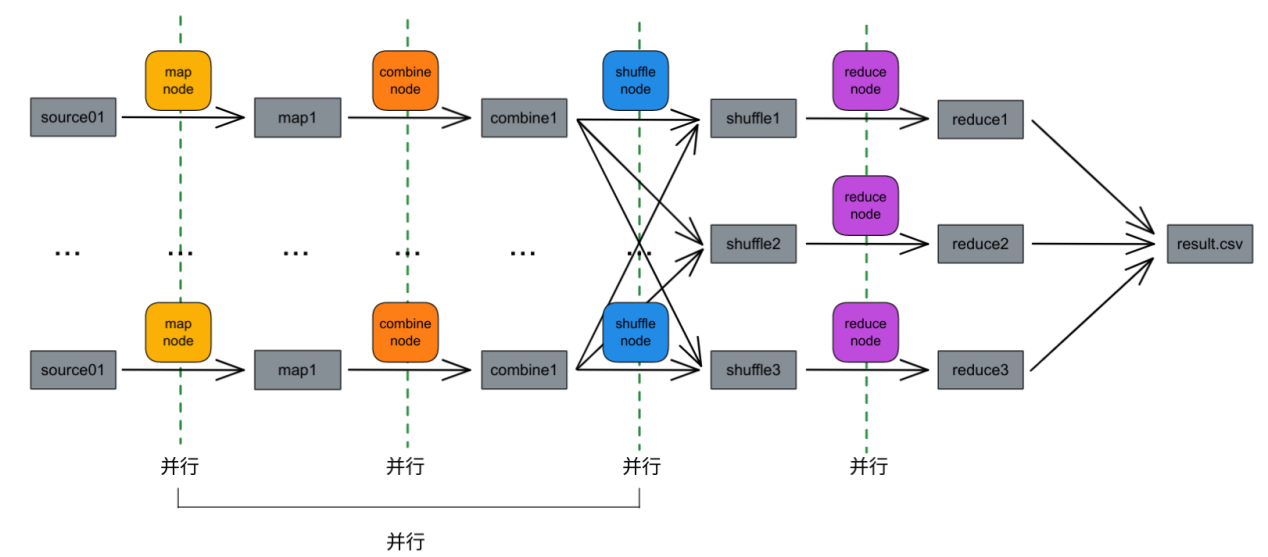
学有余力的同学可以在map-reduce的基础上添加combine与shuffle过程，并可以计算线程运行时间来考察这些过程对算法整体的影响。

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**

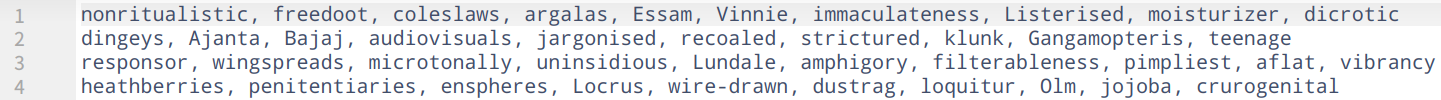
### 1.3.1 编程思路

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xxx所示。



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。



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。

1. 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，累加相同的关键字。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。

1. 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。

1. 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。

1. 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。

1. 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。

（代码）

1. 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。

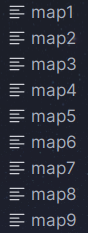
解决：为每相邻的两个环节单独设立通信队列，即map、combine之间通过map\_queue进行通信，combine、shuffle之间通过combine\_queue进行通信。以map和combine之间的通信为例，当map1结点完成工作后，它会向map\_queue队列发送“1”这个信号，则combine读取到“1”这一信号后会启动combine1结点，处理map1结点生成的map1文件。combine和shuffle之间的通信与之相似。

（代码）

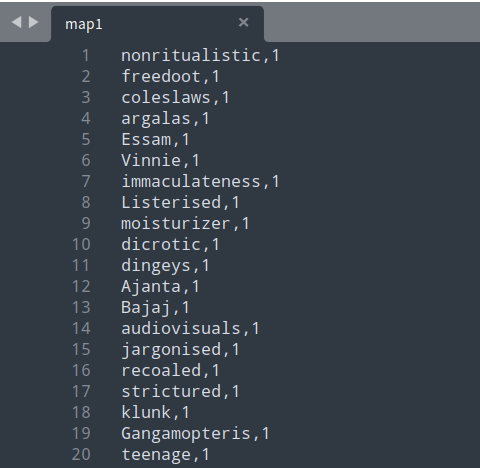
### 1.3.3 实验测试与结果分析

（1）map测试

map环节结束后生成的9个map文件如xxx所示。

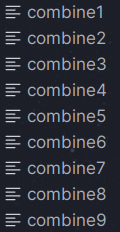


其中map1文件的开始部分如xxx所示。

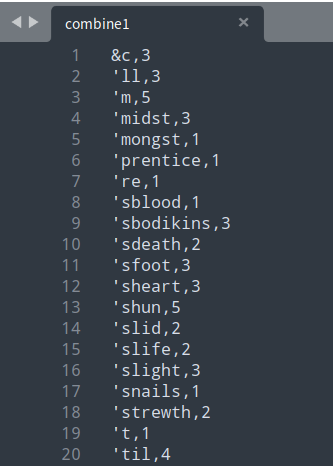


（2）combine测试

combine环节结束后生成的9个combine文件如xxx所示。



其中combine1文件的开始部分如xxx所示。

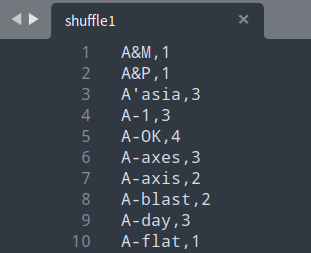


（3）shuffle测试

shuffle环节结束后生成的3个shuffle文件如xxx所示。

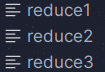


其中shuffle1文件的开始部分如xxx所示。

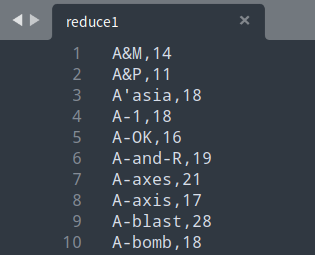


（4）reduce测试

reduce环节结束后生成的3个reduce文件如xxx所示。

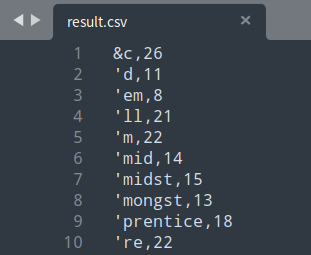


其中reduce1文件的开始部分如xxx所示。



（5）结果测试

最终生成的result.csv文件如xxx所示。



综上可知，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。