

# 20377199 赵芮箐 第11周作业

**作业内容：** MapReduce是利用多进程并行处理文件数据的典型场景。作为一种编程模型，其甚至被称为Google的“三驾马车”之一(尽管目前由于内存计算等的普及已经被逐渐淘汰)。在编程模型中，Map进行任务处理，Reduce进行结果归约。本周作业要求利用Python多进程实现MapReduce模型下的文档库（搜狐新闻数据(SogouCS)（下载地址：<https://www.sogou.com/labs/resource/cs.php>），注意仅使用页面内容，即新闻正文）词频统计功能。具体地：

1. Map进程读取文档并进行词频统计，返回该文本的词频统计结果。
2. Reduce进程收集所有Map进程提供的文档词频统计，更新总的文档库词频，并在所有map完成后保存总的词频到文件。
3. 主进程可提前读入所有的文档的路径列表，供多个Map进程竞争获取文档路径；或由主进程根据Map进程的数目进行分发；或者单独实现一个分发进程，与多个Map进程通信。
4. 记录程序运行时间，比较不同Map进程数量对运行时间的影响，可以做出运行时间-进程数目的曲线并进行简要分析。进程数量并非越多越好。

## 总体思路：

- 主进程中实现一个队列，多个Map进程，一个Reduce进程
- 主进程中根据Map进程的数目将文档分发给Map进程
- 每个Map进程同步进行任务处理，将词频统计结果put进队列
- Reduce进程get到队列中的词频统计结果，将词频结果进行合并
- 直到所有Map进程结束（即get到None），就将结果写入文件
- 记录不同Map进程数目下程序的运行时间，做出曲线并进行分析

## Part1: Map进程读取文档并进行词频统计

```
def Map(q, content):  
    '''  
    Map进程读取文档路径并进行词频统计，返回该文本的词频统计结果。  
    '''  
  
    with open('week2/stopwords_list.txt', 'r', encoding='utf-8') as s:  
        stopwords = s.read()  
        stopwords_list = stopwords.split('\n')  
  
    word_list = []  
    word_count = {}  
    for con in content:  
        seg_list = jieba.cut(con, cut_all=False)           # 分词  
        word_list.extend(seg_list)  
    for word in word_list:                                   # 过滤停用词并且统计词频  
        if word not in stopwords_list:  
            count = word_count.get(word, 0)  
            word_count[word] = count + 1  
    q.put(word_count)                                       # 将词频统计结果放进队列
```

## Part2: Reduce进程更新总的文档库词频

```
def Reduce(q, save_path = 'week11/word_count.txt'):  
    '''  
    Reduce进程收集所有Map提供的文档词频统计，更新总文档库词频，并在所有map完成后保存总词频到文件。  
    '''  
  
    total_count = {}  
    while True:  
        word_count = q.get()  
        if word_count is None:                               # 所有Map结束后保存总词频  
            with open('week11/word_count.txt', 'w', encoding='utf-8') as f:
```

```

        json_str = json.dumps(total_count, indent=0, ensure_ascii=False)
        f.write(json_str)
        f.write('\n')
    break
else:
    for word in word_count:                # 合并词频
        if word not in total_count:
            total_count[word] = word_count[word]
        else:
            total_count[word] += word_count[word]

```

## Part3: 主进程根据Map进程数目将文档分发给Map进程

```

if __name__=='__main__':
    content_list = json_to_list('week11/sohu_data.json')
    q = Queue()
    r = Process(target=Reduce, args=(q,))

    N = 4
    maps = []
    num = int(len(content_list)/N)
    for i in range(0,N):
        content = content_list[i*num:(i+1)*num]                # 根据Map进程数目分发文档
        m = Process(target=Map, args=(q, content))
        maps.append(m)

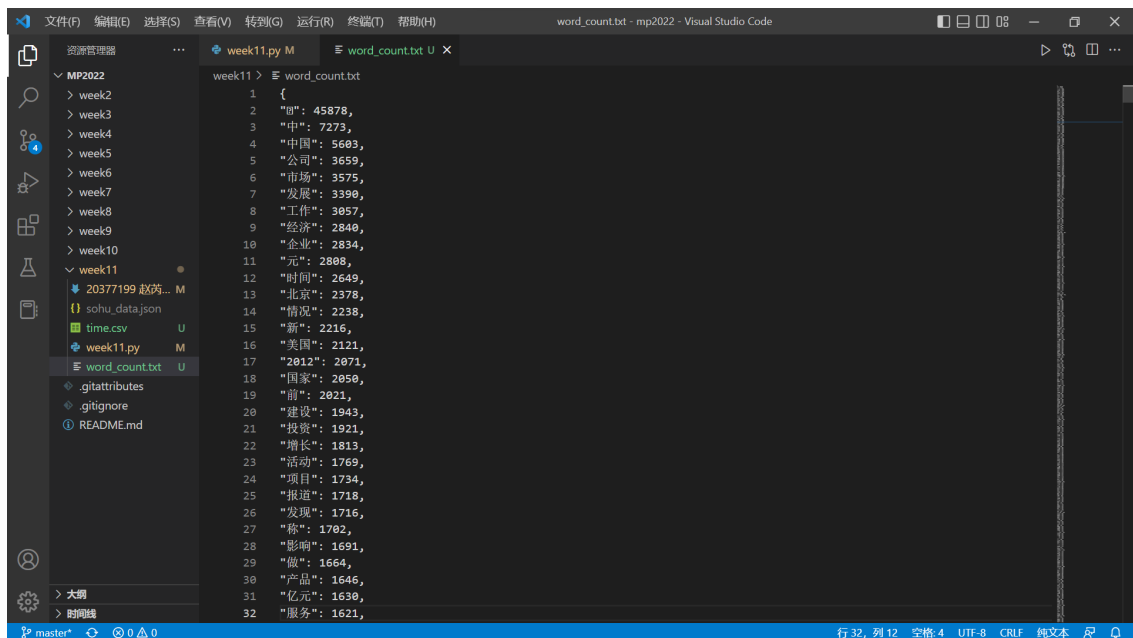
    start_time = time.time()
    for m in maps:                # 启动Map进程
        m.start()
    r.start()                    # 启动Reduce进程
    for m in maps:
        m.join()
    q.put(None)                # 主进程发信号结束
    end_time = time.time()
    print('-----所有任务结束-----')

```

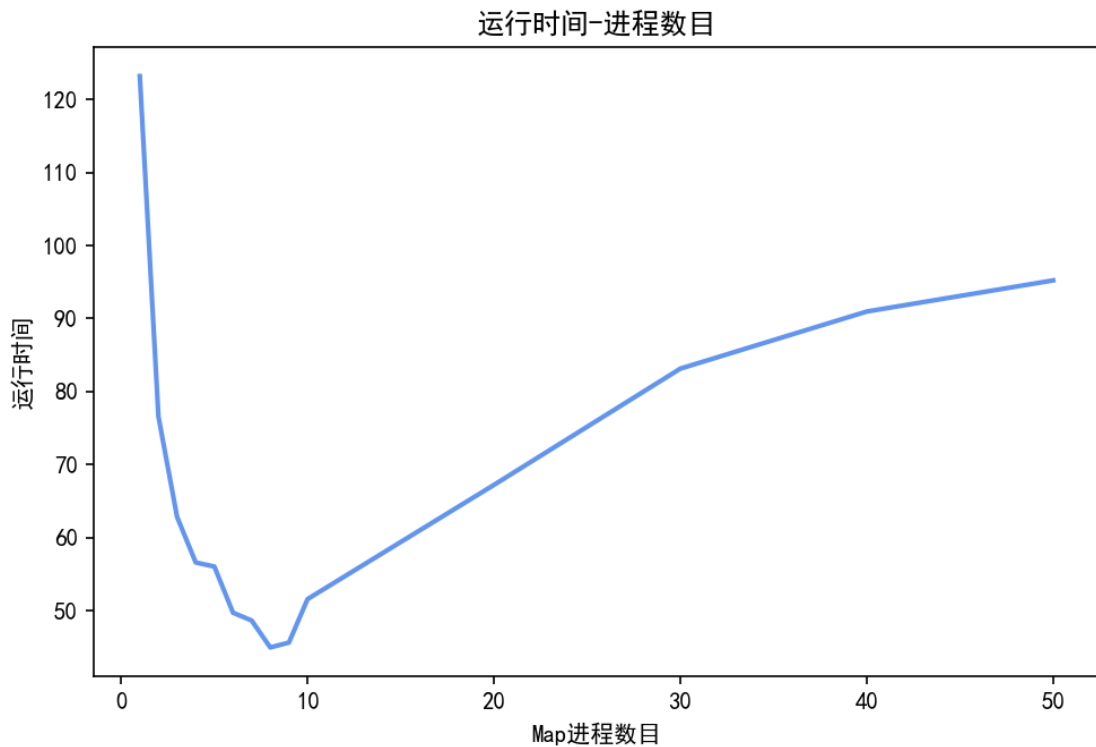
## 结果分析：

共有 1245835 条正文，但只取其中的 10000 条进行测试

- 词频统计结果



- 不同Map进程数量下的运行时间



可以发现进程并非越多越好。由于每个进程同步进行词频统计，故总统计时间会减少，但随着进程数的增多，**进程转换所需要的时间也在增多，可能大于同步处理节约的时间**，总时间反而增大。

对于10000条的数据而言，进程数为8左右运行时间最短。

代码：

[https://github.com/rachhning/mp2022\\_python/blob/master/week11/week11.py](https://github.com/rachhning/mp2022_python/blob/master/week11/week11.py)

Ref：

- MapReduce: <https://zhuanlan.zhihu.com/p/82399103>