程设第十次作业

20377383 樊思涵

前提准备：

Step1.查看本次数据文件的格式

```
with open(filename,'r',encoding = 'utf-8') as fp:
    print(type(fp))
    data = json.load(fp) #使用json模块读取文件
print(type(data))    #data的类型是列表
print(type(data[0])) #data是一个字典列表，data[0]的类型是dict
print(data[0]) #查看具体某个字典内容
print(len(data)) #查看文件总长度
print(len(data[0])) #单个字典长度
```

结果：

```
PS E:\code\py_code> python -u "e:\code\py_code\week11\week11.py"
<class '_io.TextIOWrapper'>
<class 'list'>
<class 'dict'>
{'title': '深圳地铁将设立VIP头等车厢 买双倍票可享坐票', 'content': '南都讯 记者刘凡 周昌和 任笑一 继推出日票后,深圳今后将设地铁VIP头等车厢,设坐票制。昨日,《南都METRO》创刊仪式暨2012年深港地铁圈高峰论坛上透露,在未来的11号线上将增加特色服务,满足不同消费层次的乘客的不同需求,如特设行李架的车厢和买双倍票可有座位坐的VIP车厢等。\ue40c论坛上,深圳市政府副秘书长、轨道交通建设办公室主任赵鹏林透露,地铁未来的方向将分等级,满足不同层次的人的需求,提供不同层次的有针对的服务。其中包括一些档次稍微高一些的服务。“我们要让公共交通也能满足档次稍高一些的服务”。比如,尝试有座位的地铁票服务。尤其是一些远道而来的乘客,通过提供坐票服务,让乘坐地铁也能享受到非常舒适的体验。他说,这种坐票的服务有望在地铁3期上实行,将加挂2节车厢以实施花钱可买座位的服务。\ue40c“我们希望轨道交通和家里开的车一样,分很多种。”赵鹏林说,比如有些地铁是“观光线”,不仅沿途的风光非常好,还能凭一张票无数次上下,如同旅游时提供的“通票服务”。再比如,设立可以放大件行李的车厢,今后通过设专门可放大件行李的座位,避免像现在放行李不太方便的现象。“未来地铁初步不仅在干线上铺设,还会在支线、城际线上去建设。”\ue40c“觉得如果车费不太贵的话,还是愿意考虑的。”昨日市民曹小姐表示,尤其是从老街到机场这一段,老街站每次上下客都很多人,而如果赶上上下班高峰期,特别拥挤,要一路从老街站站到机场,40、50分钟还是挺吃力的,宁愿多花点钱也能稍微舒适一点。但是白领林先生则表示,自己每天上下班都要坐地铁,出双倍车资买坐票费用有点高。'}
1245835
2
```

Step2.创建测试文件，取原数据文件的前 1000 条数据作为测试文件

```
16      #下面截取原json文件的一部分创建新的文件
17      data_new=data[:1000]
18      with open(filename_test,'w',encoding = 'utf-8') as fp:
19          json.dump(data_new,fp,ensure_ascii=False)
20
```

错误尝试：

一开始选择使用 Queue 作为多进程的共享对象，可 python 的 multiprocessing 库中的 Queue 在队列长度过长时会错误地阻塞，导致程序会在 p.join()过程中永久阻塞，结合 week12 所学内容并查询相关资料后决定放弃使用 Queue 转而使用更为方便有效的 Manager

调用库：

```
1   import json
2   from multiprocessing import Process
3   from multiprocessing import Manager
4   import time
5   import jieba
```

文件地址：

```
7   filename = r'C:\Users\LF\Desktop\sohu_data\sohu_data.json'
8   filename_test = r'C:\Users\LF\Desktop\sohu_data\sohu_data_test.json'
9   filename_result = r'C:\Users\LF\Desktop\result.txt'
```

录入文件函数

```
11  def load(path):
12      with open(path,'r',encoding = 'utf-8') as fp:
13          data = json.load(fp) #使用json模块读取文件
14      content_lis=[]
15      for dic in data:
16          content_lis.append(dic['content'])
17      return content_lis
```

Map 函数

```
19  def Map(content_lis,Lis):
20      for i in content_lis:
21          text_lis = jieba.lcut(i)
22          for j in text_lis:
23              Lis.append(j)
```

Reduce 函数

```
25  def Reduce(Lis):
26      dic = {}
27      for i in Lis:
28          dic[i] = dic.get(i,0)+1
29      dic_order=sorted(dic.items(),key=lambda x:x[1],reverse=True)
30      with open(filename_result,'w',encoding='utf-8') as file:
31          for k,v in dic_order:
32              file.write(str(k)+':'+str(v)+'\n')
```

在主程序中读取数据并根据进程数分配任务

```
34  if __name__=='__main__':
35      content_lis = load(filename_test)
36      num_lis = len(content_lis)
37      m = Manager()
38      Lis = m.list([])
39      process = []
40      num_process = 2          #可给定任意的进程数
41      num_task = int(num_lis / num_process)
```

生成多进程 Process 完成 Map 任务

```
42      for i in range(num_process):
43          if i == num_process - 1:
44              content_i_lis = content_lis[i*num_task:]
45          else:
46              content_i_lis = content_lis[i*num_task:(i+1)*num_task]
47          p = Process(target = Map,args = (content_i_lis,Lis))
48          process.append(p)
```

启动进程并在进程全部完成前阻塞主程序，同时记录 Map 花费时间

```
49      start_time = time.time() #进程启动前记录启动时间
50      for p in process:
51          p.start()  #启动进程
52      for p in process:
53          p.join()  #阻滞主进程
54      Mapped_time = time.time()
55      Map_cost_time = Mapped_time - start_time
56      print('Map进程结束')
```

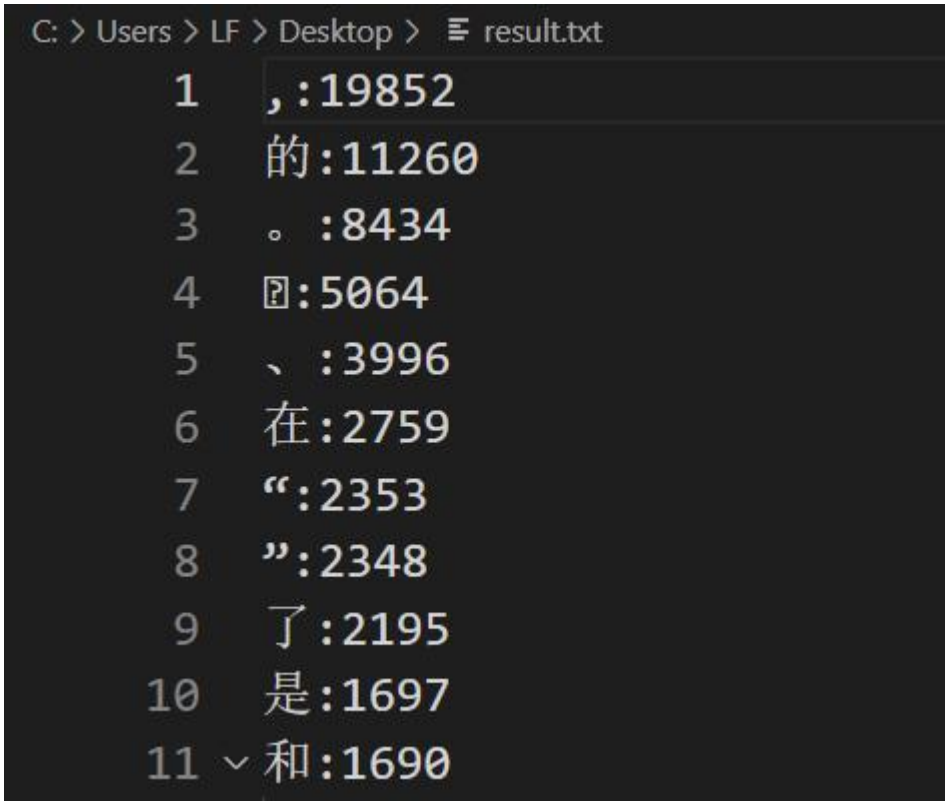
Map 进程结束后启动 Reduce 进程
（由于是单进程不需要额外使用 Process）

```
54      Mapped_time = time.time()
55      Map_cost_time = Mapped_time - start_time
56      print('Map进程结束')
57      Reduce(Lis)
58      Reduced_time = time.time()
59      Reduce_cost = Reduced_time - Mapped_time
60      print('Reduce进程结束')
61      print('-'*25)
62      print(f'{num_process}个进程时，Map进程耗时{Map_cost_time:.2f}秒')
63      print(f'Reduce进程耗时{Reduce_cost:.2f}秒')
```

先使用 test 数据检测程序可行性（仅 1000 条 content）

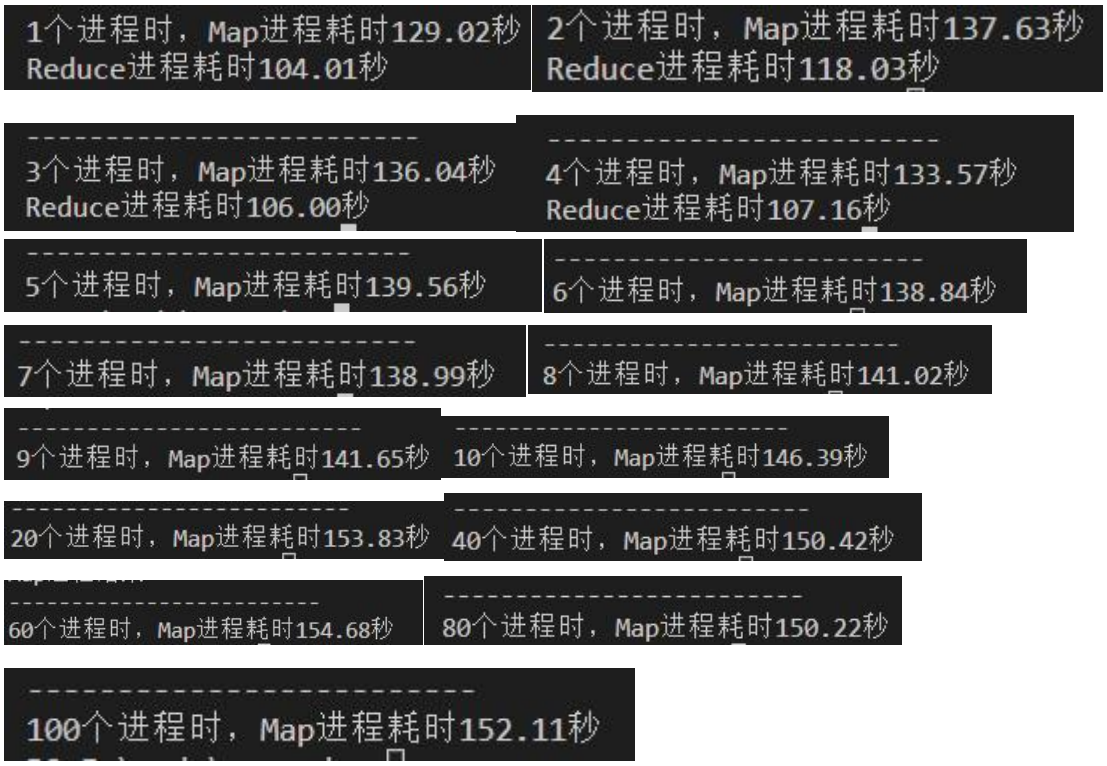
```
PS E:\code\py_code> python -u "e:\code\py_code\week11\week11.py"
Building prefix dict from the default dictionary ...
Loading model from cache C:\Users\LF\AppData\Local\Temp\jieba.cache
Building prefix dict from the default dictionary ...
Loading model from cache C:\Users\LF\AppData\Local\Temp\jieba.cache
Loading model cost 0.530 seconds.
Prefix dict has been built successfully.
Loading model cost 0.540 seconds.
Prefix dict has been built successfully.
Map进程结束
Reduce进程结束
-------------------------
2个进程时，Map进程耗时7.71秒
Reduce进程耗时7.63秒
PS E:\code\py_code>
```

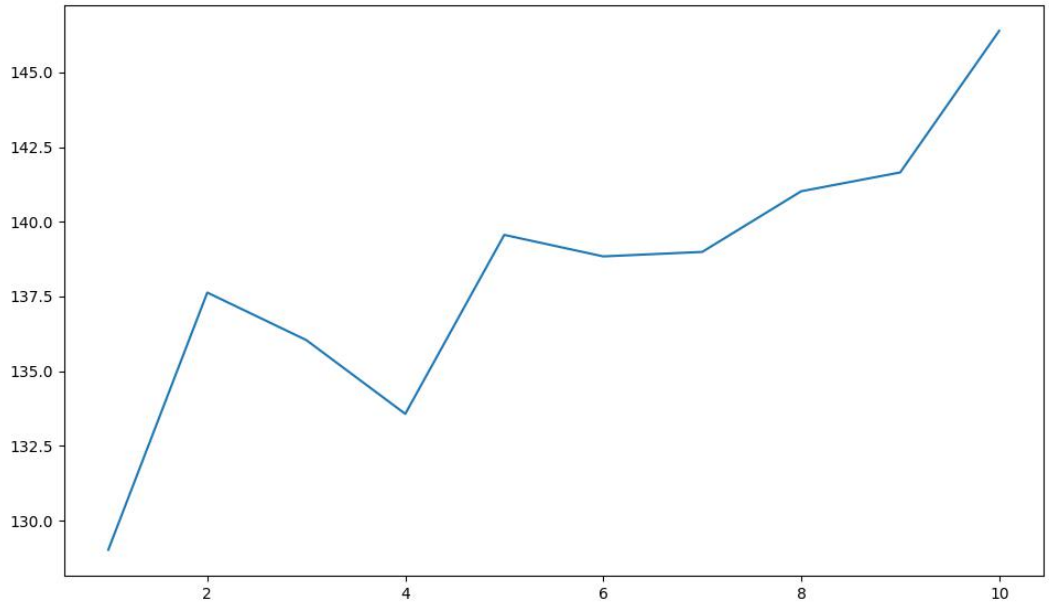
处理结果展示：

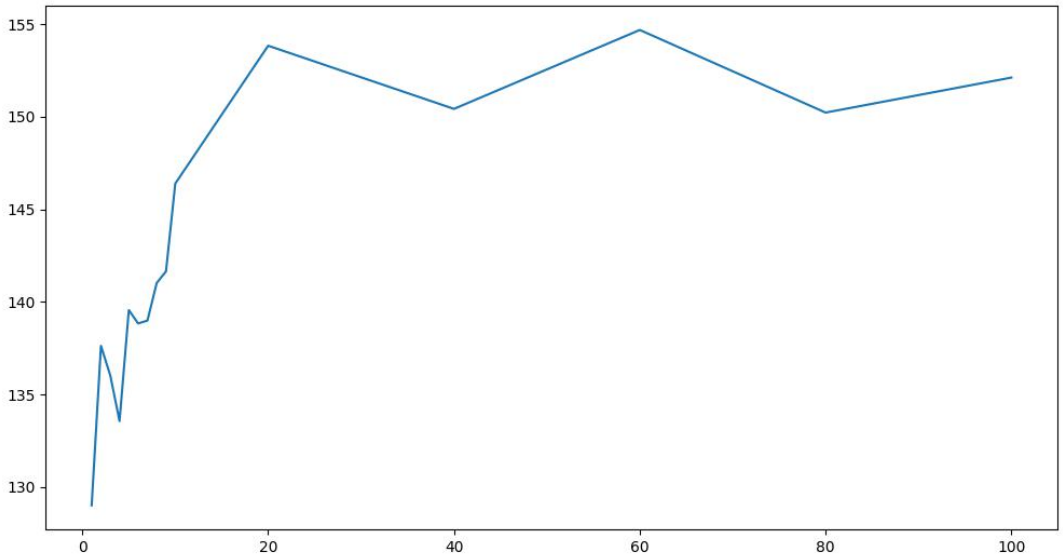


最后使用原始数据并记录不同进程数对应的 Map 进程耗时

…跑了一个小时还是没有跑出来结果，最后决定只采用前 1w 条数据展示结果分别在进程数为 1,2,3,4,5,6,7,8,9,10,20,40,60,80,100 的条件下计算 Map 进程的用时







分析：在本任务中，多个多进程共享 Lis 明显不能减小运行时间，相反，由于多个进程同时访问 Lis 导致在进程较多时 Map 进程耗时增多，应当设计调用共享空间更少的多进程程序来达到优化程序运行时间的效果。