Параллельные вычисления

Материалы:

- Макрушин С.В. Лекция 10: Параллельные вычисления
- https://docs.python.org/3/library/multiprocessing.html (https://docs.python.org/3/library/multiprocessing.html)

Задачи для совместного разбора

1. Посчитайте, сколько раз встречается каждый из символов (заглавные и строчные символы не различаются) в файле `Dostoevskiy Fedor. Prestuplenie i nakazanie - BooksCafe.Net.txt` и в файле `Dostoevskiy Fedor. Igrok - BooksCafe.Net.txt` .

2. Решить задачу 1, распараллелив вычисления с помощью модуля `multiprocessing` . Для обработки каждого файла создать свой собственный процесс.

Лабораторная работа 10

1. Разбейте файл `recipes_full.csv` на несколько (например, 8) примерно одинаковых по объему файлов с названиями `id_tag_nsteps_*.csv` . Каждый файл содержит 3 столбца: `id` , `tag` и `n_steps` , разделенных символом `;` . Для разбора строк используйте `csv.reader` .

Важно: вы не можете загружать в память весь файл сразу. Посмотреть на первые несколько строк файла вы можете, написав код, который считывает эти строки.

Подсказка: примерное кол-во строк в файле - 2.3 млн.

```
id;tag;n_steps
137739;60-minutes-or-less;11
137739;time-to-make;11
137739;course;11
```

In [1]:
```
import multiprocessing as mp
import csv
import numpy as np
from ast import literal_eval
```

In [4]:
```
n_workers = 2*mp.cpu_count()
n_workers
```

Out[4]: 12

In [68]:
```
with open(r'C:\Users\sunya\Desktop\6семестр\интернетвещей\parallelcomputing\recipes_full.cs
    read = csv.reader(file, delimiter=',')

    rows = sum(1 for row in file)

    file.seek(0)

    next(read)

    header = ['id','tag','n_steps']

    lines = int(np.ceil(rows/8))

    for i in range(8):
        with open(f'id_tag_nsteps_{i+1}.csv', 'w', newline='',  encoding='utf-8') as new_f:
            write = csv.writer(new_f, delimiter=';')

            write.writerow(header)

#             row = next(read)
#             print(row)


            for j in range(lines):
                try:
                    row = next(read)
                    tags = literal_eval(row[5])
                    for tag in tags:
                        write.writerow([row[1], tag, row[6]])
                except StopIteration:
                    break
```

In [69]:
```
with open(r'C:\Users\sunya\Desktop\6семестр\интернетвещей\parallelcomputing\sem\id_tag_nstep
    read = csv.reader(file, delimiter=';')

    next(read)
    print(next(read))
```

```
['683970', 'mexican', '4']
```

2. Напишите функцию, которая принимает на вход название файла, созданного в результате решения задачи 1, считает среднее значение количества шагов для каждого тэга и возвращает результат в виде словаря.

In [23]:
```
import pandas as pd

df = pd.read_csv(r'C:\Users\sunya\Desktop\6семестр\интернетвещей\parallelcomputing\sem\id_ta

df[df.tag=='1-day-or-more']

# new = df.groupby(['tag']).mean('n_steps')

# new

# dic = new.T.to_dict('list')

# dic
```

Out[23]:

	id	tag	n_steps
484	1200113	1-day-or-more	2
529	1415513	1-day-or-more	4
888	400287	1-day-or-more	2
2885	2221917	1-day-or-more	6
3886	1217101	1-day-or-more	4
...	...	...	...
2043174	622766	1-day-or-more	2
2043802	2166082	1-day-or-more	3
2044579	660827	1-day-or-more	3
2044901	471987	1-day-or-more	2
2047410	1611790	1-day-or-more	2

2863 rows × 3 columns

In [20]:
```
df.groupby(['tag']).agg({'n_steps':['mean','count']}).T.to_dict('list')
```

Out[20]:
```
{'1-day-or-more': [4.589940621725463, 2863.0],
 '15-minutes-or-less': [4.971805773103602, 8938.0],
 '3-steps-or-less': [4.701848667239897, 9304.0],
 '30-minutes-or-less': [7.581541639282697, 10651.0],
 '4-hours-or-less': [10.105472738411272, 9794.0],
 '5-ingredients-or-less': [5.365748381363463, 7877.0],
 '60-minutes-or-less': [9.325785582255083, 12984.0],
 'Throw the ultimate fiesta with this sopaipillas recipe from Food.com.': [3.4874905231
235784,
  2638.0],
 'a1-sauce': [3.4948414214749715, 2617.0],
 'african': [4.3119266055045875, 3052.0],
 'american': [7.568643490115883, 7335.0],
 'amish-mennonite': [3.5787328640237126, 2699.0],
 'angolan': [3.4908205320344696, 2669.0],
 'appetizers': [6.240464947330185, 5506.0],
 'apples': [4.7726051924798565, 3351.0],
 'april-fools-day': [3.5518144407033296, 2673.0],
 'argentine': [3.577431906614786, 2570.0],
```

In [24]:
```
def to_dictionary(filename):
    df = pd.read_csv(filename, sep=';')

    dic = df.groupby(['tag']).agg({'n_steps':['mean','count']}).T.to_dict('list')

    return dic
```

In [6]:
```
to_dictionary('id_tag_nsteps_1.csv')
```

```
 3027: [10.0, 11.0],
 3034: [2.0, 19.0],
 3045: [32.0, 17.0],
 3047: [8.0, 8.0],
 3058: [10.0, 13.0],
 3110: [1.0, 11.0],
 3116: [12.0, 19.0],
 3119: [14.0, 15.0],
 3154: [11.0, 20.0],
 3158: [3.0, 14.0],
 3173: [16.0, 16.0],
 3179: [7.0, 17.0],
 3187: [7.0, 8.0],
 3195: [7.0, 15.0],
 3198: [12.0, 21.0],
 3199: [18.0, 9.0],
 3201: [10.0, 23.0],
 3219: [17.0, 16.0],
 3222: [21.0, 16.0],
 3252: [6.0, 10.0],
```

3. Напишите функцию, которая считает среднее значение количества шагов для каждого тэга по всем файлам, полученным в задаче 1, и возвращает результат в виде словаря. Не используйте параллельных вычислений. При реализации выделите функцию, которая объединяет результаты обработки отдельных файлов. Модифицируйте код из задачи 2 таким образом, чтобы иметь возможность получить результат, имея результаты обработки отдельных файлов. Определите, за какое время задача решается для всех файлов.

In [3]:
```
import os


all_tags = {}


def file_tags(filename):
    tags_steps = {}

    with open(filename, 'r', encoding='utf-8') as f:
        read = csv.reader(f, delimiter=';')

        next(read)

        for row in read:

            tag = row[1]
            n_steps = row[2]

            if tag in tags_steps:
                tags_steps[tag][0] += int(n_steps)
                tags_steps[tag][1] += 1
            else:
                tags_steps[tag]=[int(n_steps), 1]

    return tags_steps

def join_res(result):
    tags_av ={}
    ret = {}

    for res in result:
        for tag, arr in res.items():
            if tag in tags_av:
                tags_av[tag][0] += arr[0]
                tags_av[tag][1] += arr[1]
            else:
                tags_av[tag] = [arr[0], arr[1]]
        for tag, arr in tags_av.items():
            ret[tag] = arr[0]/arr[1]

    return ret



def all_files_tag_av(directory):
    results = []

    for filename in os.listdir(directory):
        if filename.endswith('.csv'):
            filepath = os.path.join(directory, filename)
            results.append(file_tags(filepath))

    return join_res(results)
```

In [41]:
```
%%time

all_files_tag_av(r'C:\Users\sunya\Desktop\6семестр\интернетвещей\parallelcomputing\sem')
```

```
 'spaghetti': 4.0825152293208475,
 'passover': 3.658676110051757,
 'quick-breads': 5.058895036887995,
 'californian': 3.74143203627544,
 'namibian': 3.5042895887529752,
 'candy': 4.229612689762553,
 'independence-day': 4.10637159533074,
 'baking': 3.6306821245618766,
 'pennsylvania-dutch': 3.5471966710468683,
 'weeknight': 7.413649806241077,
 '60-minutes-or-less': 9.413654300607185,
 'time-to-make': 9.278520775418526,
 'course': 9.274676657987765,
 'cuisine': 9.17006030766978,
 'preparation': 9.29343297028588,
 'occasion': 9.136277173913044,
 'north-american': 8.187200866921525,
 'desserts': 9.023456628905357,
 'dinner-party': 8.233824137497079,
 'holiday-event': 8.308282439870316,
```

4. Решите задачу 3, распараллелив вычисления с помощью модуля `multiprocessing` . Для обработки каждого файла создайте свой собственный процесс. Определите, за какое время задача решается для всех файлов.

In [3]:
```
import os

directory = r'C:\Users\sunya\Desktop\6семестр\интернетвещей\parallelcomputing\sem'


paths = []
for filename in os.listdir(directory):
        if filename.endswith('.csv'):
            filepath = os.path.join(directory, filename)
            paths.append(filepath)
#             paths.append(filename)
paths
```

Out[3]:
```
['C:\\Users\\sunya\\Desktop\\6семестр\\интернетвещей\\parallelcomputing\\sem\\id_tag_nsteps_1.csv',
 'C:\\Users\\sunya\\Desktop\\6семестр\\интернетвещей\\parallelcomputing\\sem\\id_tag_nsteps_2.csv',
 'C:\\Users\\sunya\\Desktop\\6семестр\\интернетвещей\\parallelcomputing\\sem\\id_tag_nsteps_3.csv',
 'C:\\Users\\sunya\\Desktop\\6семестр\\интернетвещей\\parallelcomputing\\sem\\id_tag_nsteps_4.csv',
 'C:\\Users\\sunya\\Desktop\\6семестр\\интернетвещей\\parallelcomputing\\sem\\id_tag_nsteps_5.csv',
 'C:\\Users\\sunya\\Desktop\\6семестр\\интернетвещей\\parallelcomputing\\sem\\id_tag_nsteps_6.csv',
 'C:\\Users\\sunya\\Desktop\\6семестр\\интернетвещей\\parallelcomputing\\sem\\id_tag_nsteps_7.csv',
 'C:\\Users\\sunya\\Desktop\\6семестр\\интернетвещей\\parallelcomputing\\sem\\id_tag_nsteps_8.csv']
```

In [1]:
```
import multiprocessing as mp
import csv
import numpy as np
from ast import literal_eval

paths = ['C:\\Users\\sunya\\Desktop\\6семестр\\интернетвещей\\parallelcomputing\\sem\\id_tag
 'C:\\Users\\sunya\\Desktop\\6семестр\\интернетвещей\\parallelcomputing\\sem\\id_tag_2.csv'
 'C:\\Users\\sunya\\Desktop\\6семестр\\интернетвещей\\parallelcomputing\\sem\\id_tag_3.csv'
 'C:\\Users\\sunya\\Desktop\\6семестр\\интернетвещей\\parallelcomputing\\sem\\id_tag_4.csv'
 'C:\\Users\\sunya\\Desktop\\6семестр\\интернетвещей\\parallelcomputing\\sem\\id_tag_5.csv'
 'C:\\Users\\sunya\\Desktop\\6семестр\\интернетвещей\\parallelcomputing\\sem\\id_tag_6.csv'
 'C:\\Users\\sunya\\Desktop\\6семестр\\интернетвещей\\parallelcomputing\\sem\\id_tag_7.csv'
 'C:\\Users\\sunya\\Desktop\\6семестр\\интернетвещей\\parallelcomputing\\sem\\id_tag_8.csv'
```

In [4]:
```
for i in range(8):
    with open(f'id_tag_nsteps_{i+1}.csv', 'r', newline='',  encoding='utf-8') as f:
        read = csv.reader(f, delimiter=';')

        f.seek(0)

        next(read)

        with open(f'id_tag_{i+1}.csv', 'w', newline='',  encoding='utf-8') as new_f:
            write = csv.writer(new_f, delimiter=';')
            write.writerow(['id','tag','n_steps'])


            for j in range(1000):
                try:
                    row = next(read)
                    write.writerow(row)
                except StopIteration:
                    break
```

In [3]:
```
%%file file_tags_.py

import csv

def file_tags(filename, tags_steps):
#     tags_steps = {}
    print('ok')

    with open(filename, 'r', encoding='utf-8') as f:
        read = csv.reader(f, delimiter=';')

        next(read)
        print('file read')

        for row in read:

            tag = row[1]
            n_steps = int(row[2])

            if tag in tags_steps:
                tags_steps[tag][0] += int(n_steps)
                tags_steps[tag][1] += 1
            else:
                tags_steps[tag]=[int(n_steps), 1]
            print(tags_steps[tag])
        print('file compiled')

#     output.put(tags_steps)
```

```
Overwriting file_tags_.py
```

In [26]:
```
# from multiprocessing import Process, Manager

import file_tags_

def merged(result):
    tags_av ={}
    ret = {}

    for tag, arr in result.items():
        if tag in tags_av:
            tags_av[tag][0] += arr[0]
            tags_av[tag][1] += arr[1]
        else:
            tags_av[tag] = [arr[0], arr[1]]
    for tag, arr in tags_av.items():
        ret[tag] = arr[0]/arr[1]

    return ret

def all_files(paths):

    manager = mp.Manager()
    tags_steps = manager.dict()
    processes = []
#     processes = [mp.Process(target=file_tags_.file_tags, args=(filename, tags_steps)) for

    for filepath in paths:
        p = mp.Process(target=file_tags_.file_tags, args=(filepath, tags_steps))
        processes.append(p)
        print('starting processes')
        p.start()

#     for p in processes:
#         p.start()

    for p in processes:
        print('joining processes')
        p.join()


#     results = [output.get() for p in processes]

#     return join(tags_steps)
    new_ar = {}
    for tag, arr in tags_steps.items():
        new_ar[tag] = arr
    print('processes joined')
    return merged(new_ar)
#     return join_res(tags_steps)
```

In [29]:
```
%%time

all_files(paths)
```

```
 'lentils': 5.0,
 'sudanese': 3.0,
 'tarts': 3.0,
 'soy-tofu': 19.0,
 'super-bowl': 5.0,
 'broccoli': 19.0,
 'pasta-rice-and-grains': 4.0,
 'heirloom-historical-recipes': 4.0,
 'avocado': 3.0,
 'cauliflower': 12.0,
 'south-american': 4.0,
 'pears': 3.0,
 'white-rice': 4.0,
 'elk': 4.0,
 'water-bath': 3.0,
 'squid': 1.0,
 'oranges': 1.0,
 'curries': 1.0,
 'bread-pudding': 3.0,
 'hidden-valley-ranch': 4.0,
```

In [4]:
```
%%file rand_string_.py

import random
import string
def rand_string(length, output):
    rand_str = ''.join(random.choice(
     string.ascii_lowercase + string.ascii_uppercase + string.digits)
                        for i in range(length))
    output.put(rand_str)
```

```
Overwriting rand_string_.py
```

In [7]:
```
import multiprocessing as mp
import random
import string
random.seed(123)

import rand_string_

# Определить очередь вывода
output = mp.Queue()
# Настраиваем список процессов, которые мы хотим запустить
processes = [mp.Process(target=rand_string_.rand_string, args=(5, output)) \
 for x in range(4)]
# Запуск процессов
for p in processes:
    p.start()
# Выйти (дождаться выхода) завершенных процессов
for p in processes:
     p.join()
# Получить результаты процесса из очереди вывода
results = [output.get() for p in processes]
print(results)
```

```
['4m16t', 'aluP1', '5FkbQ', 'DrEt1']
```

5. (*) Решите задачу 3, распараллелив вычисления с помощью модуля `multiprocessing` . Создайте фиксированное количество процессов (равное половине количества ядер на компьютере). При помощи очереди передайте названия файлов для обработки процессам и при помощи другой очереди заберите от них ответы.

In [12]:
```
%%file file_tags_q_.py

import csv

def file_tags_q(filename, tags_steps):
#     tags_steps = {}
    print('ok')

    tags_new = {}

    with open(filename, 'r', encoding='utf-8') as f:
        read = csv.reader(f, delimiter=';')

        next(read)
        print('file read')

        for row in read:

            tag = row[1]
            n_steps = int(row[2])

            if tag in tags_new:
                tags_new[tag][0] += int(n_steps)
                tags_new[tag][1] += 1
            else:
                tags_new[tag]=[int(n_steps), 1]
#             print(tags_new[tag])
        print('file compiled')
    tags_steps.put(tags_new)
```

```
Overwriting file_tags_q_.py
```

In [5]:
```
# import os

# os.environ['PYDEVD_DISABLE_FILE_VALIDATION']='1'
```

In [2]:
```
import multiprocessing as mp
import csv
import numpy as np
from ast import literal_eval

paths = ['C:\\Users\\sunya\\Desktop\\6семестр\\интернетвещей\\parallelcomputing\\sem\\id_tag
 'C:\\Users\\sunya\\Desktop\\6семестр\\интернетвещей\\parallelcomputing\\sem\\id_tag_2.csv'
 'C:\\Users\\sunya\\Desktop\\6семестр\\интернетвещей\\parallelcomputing\\sem\\id_tag_3.csv'
 'C:\\Users\\sunya\\Desktop\\6семестр\\интернетвещей\\parallelcomputing\\sem\\id_tag_4.csv'
 'C:\\Users\\sunya\\Desktop\\6семестр\\интернетвещей\\parallelcomputing\\sem\\id_tag_5.csv'
 'C:\\Users\\sunya\\Desktop\\6семестр\\интернетвещей\\parallelcomputing\\sem\\id_tag_6.csv'
 'C:\\Users\\sunya\\Desktop\\6семестр\\интернетвещей\\parallelcomputing\\sem\\id_tag_7.csv'
 'C:\\Users\\sunya\\Desktop\\6семестр\\интернетвещей\\parallelcomputing\\sem\\id_tag_8.csv'
```

In [4]:
```
# from multiprocessing import Process, Manager

import file_tags_q_

def all_files_q(paths):

    tags_steps = mp.Queue()

#     manager = mp.Manager()
#     tags_steps = manager.dict()
    processes = []
#     processes = [mp.Process(target=file_tags_.file_tags, args=(filename, tags_steps)) for

    for filepath in paths:
        p = mp.Process(target=file_tags_q_.file_tags_q, args=(filepath, tags_steps))
        processes.append(p)
        print('starting processes')
        p.start()

#     for p in processes:
#         p.start()

    for p in processes:
        print('joining processes')
        p.join()


    results = [tags_steps.get() for p in processes]

#     new_ar = {}
#     for tag, arr in tags_steps.items():
#         new_ar[tag] = arr
#     print('processes joined')
#     return merged(new_ar)

#     return results[0]
    return join_res(results)
```

In [*]:
```
all_files_q(paths)
```

```
starting processes
joining processes
```