多线程爬虫

进程、线程对比

功能

进程， 能够完成多任务， 比如 在一台电脑上能够同时运行多个 QQ

线程， 能够完成多任务， 比如 一个 QQ 中的多个聊天窗口

定义的不同

进程是系统进行资源分配和调度的一个独立单位.

线程是进程的一个实体,是 CPU 调度和分派的基本单位,它是比进程更小的能独立运行的基本单位.线程自己基本上不拥有系统资源,只拥有一点在运行中必不可少的资源(如程序计数器,一组寄存器和栈),但是它可与同属一个进程的其他的线程共享进程所拥有的全部资源.

区别

一个程序至少有一个进程,一个进程至少有一个线程.

线程的划分尺度小于进程(资源比进程少)， 使得多线程程序的并发性高。

进程在执行过程中拥有独立的内存单元， 而多个线程共享内存， 从而极大地提高了程序的运行效率

线线程不能够独立执行， 必须依存在进程中

优缺点

线程和进程在使用上各有优缺点： 线程执行开销小， 但不利于资源的管理和保护； 而进程正相反。

队列对象 Queue

python 中的标准库，队列是基本的 FIFO 容器，线程间最常用的交换数据的形式，FIFO 先进先出

引用：from queue import Queue

python 原生的 list,dict 等，都是 not thread safe 的。而 Queue，是线程安全的，多线程环境下资源的管理，建议使用队列

常用方法

Queue.qsize() 返回队列的大小

Queue.empty() 如果队列为空，返回 True,反之 False

Queue.full() 如果队列满了，返回 True,反之 False

Queue.full 与 maxsize 大小对应

Queue.get([block[, timeout]])获取队列，timeout 等待时间

put(item[, block[, timeout]]) 将 item 放入队列中。

1. 如果可选的参数 block 为 True 且 timeout 为空对象，阻塞调用，无超时（默认的情况）。
2. 如果 timeout 是个正整数，阻塞调用进程最多 timeout 秒，如果一直无空空间可用，抛出 Full 异常（带超时的阻塞调用）。
3. 如果 block 为 False，如果有空闲空间可用将数据放入队列，否则立即抛出 Full 异常

示例

#创建一个“队列”对象

from queue import Queue

```
myqueue =Queue(maxsize = 10) #队列中能存放的数据个数的上限
#将一个值放入队列中
myqueue.put(10)
#将一个值从队列中取出
myqueue.get()
```

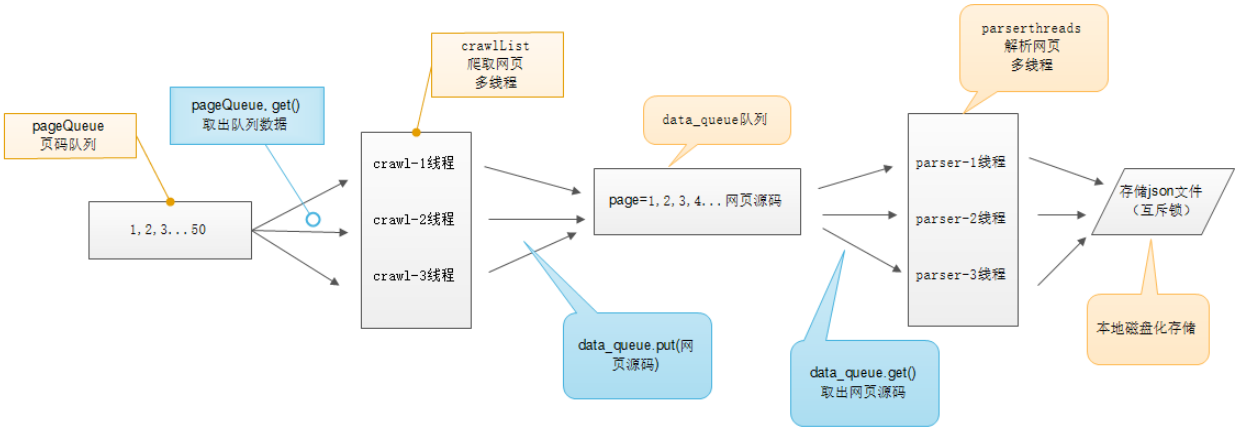
案例：糗事百科多线程爬虫

需求

设计多线程爬虫爬取糗事百科

1、用三个线程爬取 10 页页面内容，放入队列
2、用三个线程解析队列中的页面内容
3、把提取的内容存入 json 文件

多线程示意图



代码实现

```
import requests
from lxml import etree
from queue import Queue
import threading
import json
```

```
class thread_crawl(threading.Thread):
    '''
    抓取线程类
    '''
    def __init__(self, threadID):
        threading.Thread.__init__(self)
        self.threadID = threadID
        self.headers = {
            'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/52.0.2743.116 Safari/537.36',
            'Accept-Language': 'zh-CN,zh;q=0.8'}

    def run(self):
        print("Starting " + self.threadID)
        self.qiushi_spider()
        print("Exiting ", self.threadID)

    def qiushi_spider(self):
        while not page_queue.empty():
            page = page_queue.get()
            url = 'http://www.qiushibaike.com/8hr/page/' + str(page) + '/'
            print('spider:', self.threadID, ',page:', str(page))
            # 多次尝试失败结束、防止死循环
            timeout = 4
            while timeout > 0:
                timeout -= 1
                try:
                    content = requests.get(url, headers=self.headers,timeout=0.5)
                    data_queue.put(content.text)
                    break
                except Exception as e:
                    print('qiushi_spider', e)


class Thread_Parser(threading.Thread):
    '''
    页面解析类;
    '''
    def __init__(self, threadID, file):
        threading.Thread.__init__(self)
        self.threadID = threadID
        self.file = file
```

```
    def run(self):
        print('starting ', self.threadID)
        while not exitFlag_Parser:
            try:
                '''
                调用队列对象的 get()方法从队头删除并返回一个项目。可选参数为 block，默认为 True。
                如果队列为空且 block 为 True，get()就使调用线程暂停，直至有项目可用。
                如果队列为空且 block 为 False，队列将引发 Empty 异常。
                '''
                item = data_queue.get(False)
                if not item:
                    pass
                self.parse_data(item)
                data_queue.task_done() #提示线程 join()是否停止阻塞
            except:
                pass
        print('Exiting ', self.threadID)

    def parse_data(self, item):
        '''
        解析网页函数
        :param item: 网页内容
        :return:
        '''
        try:
            html = etree.HTML(item)
            result = html.xpath('//div[contains(@id,"qiushi_tag")]')
            for site in result:
                try:
                    imgUrl = site.xpath('.//img/@src')[0]
                    print('imgUrl:',imgUrl)
                    title = site.xpath('.//h2')[0].text.strip()
                    print('title:',title)
                    content = site.xpath('.//div[@class="content"]/span')[0].text.strip()
                    print('content:',content)
                    vote = None
                    comments = None
                    try:
                        vote = site.xpath('.//i')[0].text
                        comments = site.xpath('.//i')[1].text
                        print("vote:",vote)
                        print("comments:",comments)
                    except:
```

```
                            pass
                        data = {
                            'imgUrl': imgUrl,
                            'title': title,
                            'content': content,
                            'vote': vote,
                            'comments': comments,
                        }

                        if mutex.acquire():
                            data = json.dumps(data, ensure_ascii=False)
                            print('save....',data)
                            self.file.write(data + "\n")
                            mutex.release()
                    except Exception as e:
                        print('site in result', e)
            except Exception as e:
                print('parse_data', e)

def main():
    output = open('./data/qiushibaike.json', 'a',encoding='utf-8')
    #初始化网页页码 page 从 1-10 个页面
    for page in range(1, 11):
        page_queue.put(page)
    #初始化采集线程
    crawlthreads = []
    crawlList = ["crawl-1", "crawl-2", "crawl-3"]
    for threadID in crawlList:
        thread = thread_crawl(threadID)
        thread.start()
        crawlthreads.append(thread)
    #初始化解析线程 parserList
    parserthreads = []
    parserList = ["parser-1", "parser-2", "parser-3"]
    #分别启动 parserList
    for threadID in parserList:
        thread = Thread_Parser(threadID, output)
        thread.start()
        parserthreads.append(thread)

    # 等待队列清空
    while not page_queue.empty():
        pass
```

```
    # 等待所有线程完成
    for t in crawlthreads:
        t.join()

    while not data_queue.empty():
        pass
    # 通知线程是时候退出
    global exitFlag_Parser
    exitFlag_Parser = True

    for t in parserthreads:
        t.join()
    print("Exiting Main Thread")
    if mutex.acquire():
        output.close()


if __name__ == '__main__':

    data_queue = Queue()
    page_queue = Queue(50)
    exitFlag_Parser = False
    mutex = threading.Lock()

    main()
```