利用requests与Threading编写python多线程HTTP下载器

原创 Kcang 于 2019-04-30 15:17:31 发布 阅读量2.2k 收藏 7 点赞数 版权

分类专栏： 应用类 文章标签： python requests 多线程 下载

准备

最近 神经网络 研究遇到了一点瓶颈，于是想着琢磨点其他的东西缓冲一下，正好那天遇到了一些关于下载的问题，我就想着这些网络上的下载器，不是垃圾就是广告太多，还慢，既然这样，那就自己写一个呗！
基本思路就是利用**requests**这个模块向路由请求数据，然后**open**打开文件把数据写进去。考虑到 多线程 ，那得利用open函数里的seek()去指定每个线程负责写入那一块数据，然后大家一起工作最后就拼凑成一个完整的文件。
但是，既然是写一个下载器，那么，文件中断下载了，那也不能重新下啊。于是我就增加了一个data文件，用来记录本次下载的数据指针，这样就算中途断开也能从这个data文件里读取到上一次下载了那些数据，然后继续从上次断开的地方继续下载！

分段下载的 requests 请求

多线程下载的关键就是如何从服务器分段请求数据，这个我也是从别人哪里看来的：

```
headers = {'Range': 'bytes=%d-%d'%(startSeek,endSeek)}
re = requests.get(self.url, headers=headers, stream=True)
```

像这样请求，服务器就会返回你 'bytes' 标段的数据段。

分段存储open()

有了分开下载以后，自然是得分开存储：

```
with open(filePath, 'r+b') as file:
    file.seek(startSeek)
    file.write(data)
```

这样打开文件，利用**seek()**函数，将指针指向你所需要的位置，就会从你指定的位置开始存储数据。

下载情况的实时显示

首先既然要实时显示，要么就像pip下载东西一样，一个动态的进度条，要么就动态的跳数字，这里我想简单一点，打算在屏幕上动态跳几个数字来展示下载到什么地步了就行。
在python里，想要动态显示屏幕信息，我只知道两种方法(直接用python的动画模块除外，不过效果可能也不错的哦！)

1.就是利用大家常见的print！

```
print("\r 第 %d 次迭代中，第 %d 个样本的 %d 次循环迭代!"%(goNumber,i+1,cishu),end="",flush=True)
```

一个就是像这样利用 **"\r" 以及 end="", flush=True** 来告诉**print**函数你需要清屏。这样就可以动态显示你需要的画面了。

2.利用sys模块的write函数来实现，说实话，一开始看到这个函数的时候我以为我穿越到了java。

```
sys.stdout.write("\r文件下载进度：%.2f%%(%.2f M/%.2f M) - %.2f KB/s" % (getSize*100/fileSize, self.changeFormat(getSize), self.changeFormat(fileSize), self.changeFormat(speed)*1024))
sys.stdout.flush()
```

关于这个函数我就不多做解释了，效果与print函数是一样的。但是这个表达方式，乍一看，跟java还真的很像呢！

现在，我们需要显示出来的动态数据，包括下载数据大小，下载进度，以及实时网速。
下载数据大小，我们利用累加写入文件的data的长度来计算。

```
for data in re.iter_content():
    file.write(data)
    getSize = getSize + len(data)
```

利用一个for循环，将数据从response里取出来，然后累加计算每个写入文件的数据段长度，这样就有了已经下载的数据大小。

计算进度则很简单了，先根据请求头，获取文件总大小，然后除一下就有了。

```
re = requests.head(url)
fileSize = int(re.headers['content-length'])
```

利用fileSize去除我们的getSize再取百分数，就有了我们的下载进度百分比。

现在计算网速。

```
        startTime = time.time()
        headers = {'Range': 'bytes=%d-%d'%(self.startSeek,self.endSeek)}
        re = requests.get(self.url, headers=headers, stream=True)
        getSize = self.startSeek
        timeGetSize = 0
        with open(self.filePath, 'r+b') as file:
            file.seek(getSize)
            for data in re.iter_content():
                file.write(data)
                getSize = getSize + len(data)
                timeGetSize = timeGetSize + len(data)
                endTime = time.time()
                if (endTime-startTime)>=1:
                    startTime = endTime
                    speed = timeGetSize
                    timeGetSize = 0
```

这是完整代码里的一整块。其中利用**time**模块来计算经过的时间，然后新建一个变量**timeGetSize**来累加计算在一秒的时间里，我们接收到多少数据。待到**endTime-startTime**等于一秒后，输出**speed**。

好了，现在基本的问题都解决了，还剩一个，如何统计多线程的网速和下载信息?
没法，我只能想到利用全局变量的办法统计。

```
global getSizeAll
global speedAll
```

在程序的一开始，新建两个全局变量，然后在每一个线程里引用他，并且每个线程对应自己的索引：

```
getSizeAll[self.threadID] = getSize
speedAll[self.threadID] = speed
```

然后在外面的循环里统计这个全局变量，这样就有了多线程的网速和下载进度了。

断点下载问题

既然是一个完整的下载器，自然得有断点恢复下载的能力，在这里我的做法实时保存现在的下载进度，当现在的下载中断了以后，从保存的数据里恢复下载进度，继续下载。
因此，只要保存这个**getSizeAll**就行了，即每个线程自己下载到哪儿了，下次恢复的时候只要继续请求从这里开始的数据就行。
存储上，因为要存一个list，要么转二进制存文件，然后二进制解码，要么用json编码成字符，然后再json解码，这里我就用我熟悉的json编码了：

```
savaMsg = json.dumps(getSizeAll,ensure_ascii=False)
            with open(msgFilePath,'w') as f:
                f.write(savaMsg)
```

然后解码：

```
        with open(msgFilePath, 'r') as f:
            getSizeAll = json.loads(f.read())
```

遇到的坑。。。

第一个就是open函数的读写，因为涉及多线程，所以要分块读写数据，但是我一开始用的 'wb' 模式来读写，最后在下载文件以后频频出现文件损坏无法打开的问题，而且下载的文件很容易莫名其妙少了一块数据，导致损坏，后来折腾了半天才发现这个问题。

r或rt 默认模式，文本模式读
rb 二进制文件

w或wt 文本模式写，打开前文件存储被清空
wb 二进制写，文件存储同样被清空

a 追加模式，只能写在文件末尾
a+ 可读写模式，写只能写在文件末尾

完整代码

```python
#-------多线程可断点下载----------------
class HttpDownloadThreading(threading.Thread):
    def __init__(self, url, filePath, startSeek, endSeek, threadID):
        threading.Thread.__init__(self)
        self.url = url
        self.filePath = filePath
        self.startSeek = startSeek
        self.endSeek = endSeek
        self.threadID = threadID
    def run(self):
        global getSizeAll
        global speedAll
        startTime = time.time()
        headers = {'Range': 'bytes=%d-%d'%(self.startSeek,self.endSeek)}
        re = requests.get(self.url, headers=headers, stream=True)
        getSize = self.startSeek
        timeGetSize = 0
        with open(self.filePath, 'r+b') as file:
            file.seek(getSize)
            for data in re.iter_content():
                file.write(data)
                getSize = getSize + len(data)
                timeGetSize = timeGetSize + len(data)
                endTime = time.time()
                if (endTime-startTime)>=1:
                    startTime = endTime
                    speed = timeGetSize
                    timeGetSize = 0
                    getSizeAll[self.threadID] = getSize
                    speedAll[self.threadID] = speed
                    #sys.stdout.write("\r文件下载进度：%.2f%%(%.2f M/%.2f M) - %.2f KB/s" % (getSize*100/fileSize, self.changeFormat(getSize), self.changeFormat(fileSize), self.changeFormat(speed)*1024))
                    #sys.stdout.flush()
        getSizeAll[self.threadID] = self.endSeek
        speedAll[self.threadID] = 0
def changeFormat(x):
        return x/(1024*1024)
def HttpDownloadThreadingFunction(url, filePath, threadNum):
    msgFilePath = filePath + '.data'
    global getSizeAll
    global speedAll
    speedAll = []
    for i in range(0,threadNum):
        speedAll.append(0)
    re = requests.head(url)
    fileSize = int(re.headers['content-length'])
    if os.path.isfile(filePath):
        with open(msgFilePath, 'r') as f:
            getSizeAll = json.loads(f.read())
        sizeForThread = []
        for i in range(0, threadNum):
            sizeForThread.append(i * int(fileSize / threadNum))
        sizeForThread.append(fileSize + 1)
        for i in range(0,threadNum):
            h = HttpDownloadThreading(url,filePath,getSizeAll[i],sizeForThread[i+1], i)
            h.start()
        while 1:
            time.sleep(1)
            getSum = 0
            speedSum = 0
            for i in range(0,threadNum):
                getSum = getSum + getSizeAll[i] - sizeForThread[i]
                speedSum = speedSum + speedAll[i]
            sys.stdout.write("\r文件下载进度：%.2f%%(%.2f M/%.2f M) - %.2f KB/s" % (getSum*100/fileSize, changeFormat(getSum), changeFormat(fileSize), changeFormat(speedSum)*1024))
            sys.stdout.flush()
            if int((getSum+threadNum)/fileSize) >= 1:
                break
            savaMsg = json.dumps(getSizeAll,ensure_ascii=False)
            with open(msgFilePath,'w') as f:
                f.write(savaMsg)
    else:
        f =  open(filePath,'wb')
        f.truncate(fileSize)
        f.close()
        sizeForThread = []
        getSizeAll = []
        for i in range(0,threadNum):
            sizeForThread.append(i*int(fileSize/threadNum))
            getSizeAll.append(i*int(fileSize/threadNum))
        sizeForThread.append(fileSize+1)
        for i in range(0,threadNum):
            h = HttpDownloadThreading(url,filePath,sizeForThread[i],sizeForThread[i+1], i)
            h.start()
        while 1:
            time.sleep(1)
            getSum = 0
            speedSum = 0
            for i in range(0,threadNum):
                getSum = getSum + getSizeAll[i] - sizeForThread[i]
                speedSum = speedSum + speedAll[i]
            sys.stdout.write("\r文件下载进度：%.2f%%(%.2f M/%.2f M) - %.2f KB/s" % (getSum*100/fileSize, changeFormat(getSum), changeFormat(fileSize), changeFormat(speedSum)*1024))
            sys.stdout.flush()
            if int(getSum/fileSize) >= 1:
                break
            savaMsg = json.dumps(getSizeAll,ensure_ascii=False)
            with open(msgFilePath,'w') as f:
                f.write(savaMsg)
    try:
        os.remove(msgFilePath)
    except:
        pass
    print('下载完成')
```

开始下载试试：

```python
url = 'https://d2.xia12345.com/down/109/2019/04/2HgugAHm.mp4'
HttpDownloadThreadingFunction(url, 'move.mp4', 20)
```

```
E:\anaconda\python.exe C:/Users/hp/Desktop/project/new_file/internet/downloadObject/dlo.py
文件下载进度：0.14%(1.03 M/711.00 M) - 86.10 KB/s
```

效果还是不错的！另外，这个网址是个彩蛋。想看的可以下载看看哦！

结尾

最后，经过我的测试，发现这个下载器性能也一般，特别在下载大文件的时候，经常直接卡主，然后线程好像直接就没了，不知道是不是python全局锁的原因，还在研究中。
另外，既然是下载器，怎么也得能使用迅雷下载地址啊。研究中。

【Python_requests学习笔记（九）】基于requests和threading模块实现多线程爬虫 来没之昂的博客 915
此篇文章中介绍基于 requests 和 threading 模块实现多线程爬虫，并以 抓取Cocos中文社区中：热门主题下的帖子名称及id数据 为例进行讲解。

Python使用HTTP代理实现多线程/多进程网络请求 Trb401012的博客 410
在上面的代码中，我们首先导入了必要的库，并设置了代理字典。我们学习Python必然是为了找到高薪的工作，下面这些面试题是来自阿里、腾讯、字节等一线互联网大厂最新的面试资料，并且有阿里大佬给出了权威的解答，刷完这一套面试资料相信大家都能找到满意的工作。Python所有方向的技术点做的整理，形成各个领域的知识…

python 多进程 requests_python requests多进程不安全的场景-CSDN博 ... 3-20
pythonrequests是个老牌的http client库,在多线程下(threading)是安全的,在协程下自然也是安全的。值得一说的是Requets自身并没有实现连接池,而是引入了标准库urllib2的连接池。池的线程安全实现很简单,就是加锁控制边界。

requests Python中最好用的网络请求工具 基础速记+最佳实践_python 网... 4-2
pip3 install requests -i https://pypi.tuna.tsinghua.edu.cn/simple 最简单请求,发送一个get请求,获得返回值。 import requests res = requests.get("http://www.baidu.com") print(res) >>> <Response [200]> ...

在Http协议下实现多线程断点的下载 yinbucheng的博客 1302
0.使用多线程下载会提升文件下载的速度，那么多线程下载文件的过程是： (1)首先获得下载文件的长度，然后设置本地文件的长度 HttpURLConnection.getContentLength(); RandomAccessFile file = new RandomAccessFile("QQWubiSetup.exe","rwd"); file.setLengt

【Python】使用requests库实现多线程下载大文件 冰冷的希望的博客 3618
1.当我们请求下载文件的时候，可以使用head请求看一下该文件有多大，响应头里的“Content-Length”字段表示文件的字节数。使用使用requests库可以实现网络请求，但如果用于下载大文件，单线程下载确实不能很好地利用宽度，改为多线程会更好一点。2.拿到了文件大小之后，根据线程数划分为多个数据块，即每个线程都请求一部…

...多线程flask+tkinter+requests+threading_python写一个网络传送文件... 3-25
七,程序源代码 博客地址:https://blog.csdn.net/beginner2021 源代码下载地址:文件快传(使用tkinter+flask+requests+threading)此文件为使用这些模块的实例程序(加注释)-Python文档类资源-CSDN下载...

多线程调用一个接口,使用 threading_requests 多线程请求同一个接口-CS... 3-25
/user/bin/env python#coding=utf-8importrequestsimportdatetimeimporttimeimportthreadingclassurl_request():times=[]error=[]defreq(self):#请求首页接口myreq=url_request()headers={'User-Agent':'Mozilla/5.0 (iPhone; CPU ...

介绍requests+threading多线程爬虫，提取采用xpath 和正则两种，介绍线程锁 weixin_34127717的博客 555
爬虫专业的都喜欢scrapy框架，但scrapy上手需要时间，对初学者不太适合。 本文介绍使用requets爬虫，为了利于演示学习，使用了xpath解析html和完全使用正则来提取两种方法，仅供参考。 代码是爬取http://esf.sz.fang.com/，房天下网站的深圳二手房信息 import requests,json,random import re,threading f...

Python最佳实践—requests模块下载超大文件，并实时显示下载进度和速度 魏德曼的博客 1460
Python最佳实践—requests模块下载超大文件，并实时显示下载进度和速度

Requests补充,线程,进程_requests进程 3-15
一、Requests进阶 1.cookie处理->模拟浏览器登录 步骤: 1.登录 -> 得到cookie 2.带着cookie去请求到书架url -> 书架上的内容 以上步骤可以使用session进行请求:session(会话)可以被认为是一连串的请求,且在这个过程中cookie不会丢...

python多线程接口案例 12-21
项目为某内控公司要求并发测试，编写多线程访问接口，并生成Excel报告的脚本，记录基本步骤。 若有需要写UI自动化，接口自动化，多线程，服务器测试定时脚本等等，可联系本工具熊。 分五步操作实现50个或更多用户...

Python文件多线程下载工具.zip 07-10
Python 大文件多线程下载 支持断点续传，主要调用了ThreadPool 和 requests，控制线程数量，减少浪费内存资源。

python 实现多线程下载m3u8格式视频并使用fmmpeg合并 01-20
电影之类的长视频好像都用m3u8格式了，这就导致了多线程下载视频的意义不是很大，都是短视频，线不线程就没什么意义了嘛。 我们知道，m3u8的链接会下载一个文档，相当长，半小时的视频，应该有接近千行ts链接。 这些...

python 多线程将大文件分开下载后在合并的实例 12-25
废话不多说了，上代码吧： import threading import requests import time import os class Mythread(threading.Thread): def __init__(self,url,startpos,endpos,f): super(Mythread,self).__init__() self.url...

python 笔记（3）——request、爬虫、socket、多线程 @zhujj_的博客 1076
利用 requests.session() 获取到 session 对象，使用 session 对象发送请求，可以维护cookies 和 session，不必自己操作。1、find、findAll 根据标签、属性等进行查找（find 是查找第一个匹配的；解析器可以使用：html.parser、lxml、xml、html5lib。查找页面元素的通用方法：find、findAll、select。1、使用requests发送http请求。3-2-2…

python+requests_500线程接口压力测试 默金 3941
请求通过次数： 100 请求异常次数： 0 10个线程，每个线程压力请求10次,共计100次，没有请求异常 import threading import requests import time import logging logging.basicConfig(level=logging.INFO, format='%(asctime)s - %(name)s - %(levelname...

Python 使用多线程进行并发请求 最新发布 Trb701012的博客 1146
我们学习Python必然是为了找到高薪的工作，下面这些面试题是来自阿里、腾讯、字节等一线互联网大厂最新的面试资料，并且有阿里大佬给出了权威的解答，刷完这一套面试资料相信大家都能找到满意的工作。Python所有方向的技术点做的整理，形成各个领域的知识点汇总，它的用处就在于，你可以按照上面的知识点去找对应的学…

Python (HTTP多进程服务器) qq_43166246的博客 706
import socket import multiprocessing import re class Server(object): def init(self, port): “”" 在初始化中做好tcp连接的准备工作 “”" # 1创建一个tcp套接字 self.tcp_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM) #...

Python数据抓取——多线程，异步 热门推荐 公众号：瑞行AI 1万+
本文主要是为了加快数据抓取任务，考虑使用多进程、多线程、异步原理，相关概念可以参考 https://www.liaoxuefeng.com/wiki/0013747381250 95c955c1e6d8bb493182103fac9270762a000/0013868322563729e03f6905ea94f0195528e3647887415000操作系统可以同时运行多个任务。首先，考虑单核CP

python并发request,具有多线程的python-requests weixin_42362003的博客 3041
I am working on creating a HTTP client which can generate hundreds of connections each second and send up to 10 requests on each of those connections. I am using threading so concurrency can be achiev...

python中threading模块详解 李晓蒙的博客 1946
threading提供了一个比thread模块更高层的API来提供线程的并发性。这些线程并发运行并共享内存。 下面来看threading模块的具体用法： 一、Thread的使用 目标函数可以实例化一个Thread对象，每个Thread对象代表着一个线程，可以通过start()方法，开始运行。 这里对使用多线程并发，和不适用多线程

python m3u8多线程下载器 07-09
回答1： Python是一种功能强大且易于学习的编程语言，它具有丰富的库和工具，可用于各种开发任务。M3U8是一种多媒体播放列表文件格式，常用于网络视频的流媒体传输。在Python中，我们可以使用多线程来实现一个M3U8多线程下载器。 首先，我们需要使用requests库来获取M3U8文件的内容。使用requests库发送HTTP请…

“相关推荐”对你有帮助么？
非常没帮助 没帮助 一般 有帮助 非常有帮助

Kcang
码龄9年 暂无认证
9 原创 105万+ 周排名 200万+ 总排名 2万+ 访问 等级
201 积分 8 粉丝 15 获赞 27 评论 67 收藏
私信 关注

搜博主文章

热门文章

python微服务，使用nacos，实现注册中心，服务注册，加载多个配置文件，负载均衡客户端以及跨组调用服务 11052
利用python爬取某直播网站实时弹幕并分析 4642
利用requests与Threading编写python多线程HTTP下载器 2278
python 熔断器，限流器，实现服务的熔断和限流，也可以应用与其他场景的函数超时处理和异常错误处理。 1301
利用netty实现跨局域网间接口通信工具 1105

分类专栏

应用类 8篇
算法 1篇

最新评论

python微服务，使用nacos，实现注册中…
Kcang: 就是一个dict，对应配置中心的json，会把配置中心的json按照键值对放到m…
python 熔断器，限流器，实现服务的熔…
CSDN-Ada助手: 多亏了你这篇博客, 解决了问题: https://ask.csdn.net/questions/801…
python微服务，使用nacos，实现注册中…
小鱼鱼噢: myConfig这个怎么写呀
利用python爬取某直播网站实时弹幕并分析
Hearty_07: 能获取到用户的uid么

最新文章

一个基于python的rpc框架BroachRpc，开箱即用，无注册中心微服务模式

手动编码实现spring框架IOC和AOP功能

利用netty实现跨局域网间接口通信工具

2022年 3篇 2021年 2篇

2020年 1篇 2019年 3篇

目录