**[Python的进程和线程（二）——IO密集型任务](https://www.cnblogs.com/yimiaoyikan/p/10485207.html)**

**一、什么是IO密集型任务？**

主要的核心任务是进行IO操作，比如写文件，进行磁盘的读写等等。

上一篇博客，对计算密集型任务，多线程并没有体现它的好处，反而话费的时间更长。对IO密集型任务多线程会有明显的优势

**二、举例：**

任务：爬取韩寒的博客

1、获取urls，

2、根据文章的url，爬取内容，保存文件

3、将urls分给不同的进程/线程

4、多进程/多线程爬取

**step1：爬取urls**

思路：用requests库来爬取页面，用beautifulSoup库来获取目标的urls值

[复制代码](javascript:void(0);)

import requests

from bs4 import BeautifulSoup

def get\_urls(url):

'''

获取目标URL

:param url: 列表页面url

:return: 目标url列表

'''

res=requests.get(url).text

soup = BeautifulSoup(res, features="html.parser")

#通过css selector解析页面，获取元素

artile\_urls=soup.select(".atc\_title > a")

url\_list=list(i.get("href") for i in artile\_urls)

return(url\_list)

[复制代码](javascript:void(0);)

**step2：根据文章的url，爬取内容，保存文件**

[复制代码](javascript:void(0);)

import time,os

import requests

from bs4 import BeautifulSoup

def get\_content(urls,dirpath):

'''

获取文章内容

:param urls: 要获取文章的url列表

:param dirpath: 文章内容文件保存路径

:return:

'''

for url in urls:

# print("要抓取的url是%s" % url)

res = requests.get(url).content.decode("utf-8")

soup = BeautifulSoup(res, features="html.parser")

paragraphs = soup.select("#sina\_keyword\_ad\_area2 > p")

content=""

for i in paragraphs:

content+=i.get\_text()

if not os.path.exists(dirpath):

os.makedirs(dirpath)

open(dirpath + r'/' + url[-26:], 'w').write(content)

[复制代码](javascript:void(0);)

**step3：将urls分给不同的进程/线程**

思路：

　　假设我们启动n个进程，一共要爬取的url列表是urls,列表的长度为url\_len

　　我们先把列表整除n得到长度为L（不一定能够整除，所以最后一个进程为总数-前面n-1个进程的），则第1个进程要爬取的url列表是urls[0:L],第2个进程要爬取的url列表是url[L: 2L],依次类推。。。。，最后一个进程要爬取的url列表是url[i\*n:url\_len]

[复制代码](javascript:void(0);)

for n in [8, 4, 2, 1]:

# 将urls分割到url\_list

url\_list = []

url\_split\_len = url\_len // n

for i in range(n):

if i == n - 1:

url\_list.append(urls[i \* url\_split\_len:url\_len])

else:

url\_list.append(urls[i \* url\_split\_len:(i + 1) \* url\_split\_len])

[复制代码](javascript:void(0);)

参照上一篇博客，多进程和多线程函数。

我们的目标函数是，get\_content，这个函数需要2个参数，一个是url列表，一个是保存文件的路径

[复制代码](javascript:void(0);)

import time

def thread\_process\_job(n, Thread\_or\_Process, url\_list, job):

"""

n: 多线程或多进程数

Thread\_Process: Thread／Process类

job: countdown任务

"""

local\_time = time.time()

# 实例化多线程或多进程

threads\_or\_processes = [Thread\_or\_Process(target=job, args=(url\_list[j],str(n)+Thread\_or\_Process.\_\_name\_\_)) for j in range(n)]

for t in threads\_or\_processes:

t.start() # 开始线程或进程，必须调用

for t in threads\_or\_processes:

t.join() # 等待直到该线程或进程结束

print(n, Thread\_or\_Process.\_\_name\_\_, " run job need ", time.time() - local\_time)

[复制代码](javascript:void(0);)

**step4:多进程或者多线程爬取**

我们爬取前6页的数据，代码如下

[复制代码](javascript:void(0);)

if \_\_name\_\_=="\_\_main\_\_":

t = time.time()

urls = []

for i in range(7):

url='http://blog.sina.com.cn/s/articlelist\_1191258123\_0\_' + str(i + 1) + '.html'

page\_urls=get\_urls(url)

urls.extend(page\_urls)

url\_len = len(urls)

print("total urls number is ", url\_len)

for n in [8, 4, 2, 1]:

# 将urls分割到url\_list

url\_list = []

url\_split\_len = url\_len // n

for i in range(n):

if i == n - 1:

url\_list.append(urls[i \* url\_split\_len:url\_len])

else:

url\_list.append(urls[i \* url\_split\_len:(i + 1) \* url\_split\_len])

# 分割任务后创建线程

thread\_process\_job(n, Thread, url\_list, get\_content)

thread\_process\_job(n, Process, url\_list, get\_content)

print("All done in ", time.time() - t)

[复制代码](javascript:void(0);)

**代码可能存在的问题：**

代码在运行中，可能会存在的问题是，调用get\_content函数时，可能一个进程正在创建文件夹，一个进程正好在判断文件不存在要创建，在创建的时候会报错文件已存在。