

第 11 关、协程

1、协程

协程指的是单线程下的并发，又称微线程，协程是一种用户态的轻量级线程，即协程是由用户程序自己控制调度的。

- (1) 异步：在一个任务未完成时，就可以执行其他多个任务，彼此不受影响；
- (2) 同步：一个任务结束才能启动下一个。



2、多协程

每台计算机都靠着CPU（中央处理器）干活。在过去，单核CPU的计算机在处理多任务时，会出现一个问题：每个任务都要抢占CPU，执行完了一个任务才开启下一个任务。CPU

毕竟只有一个，这会让计算机处理的效率很低。



为了解决这样的问题，一种非抢占式的异步技术被创造了出来，这种方式叫多协程（在此，多是多个的意思）

1-1、gevent 库

1-1-1、安装 gevent

- 1 window电脑：在终端输入命令：pip install gevent，按下enter键；
- 2 mac电脑：在终端输入命令：pip3 install gevent，按下enter键

1-1-2、gevent 使用

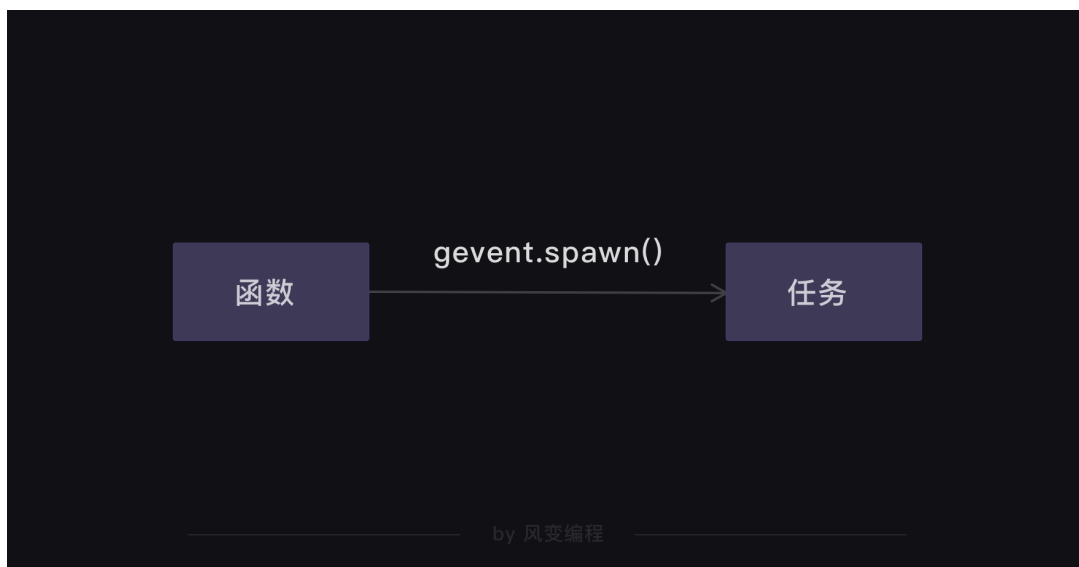
```
1 from gevent import monkey
2 #从gevent库里导入monkey模块。
3 monkey.patch_all()
4 #monkey.patch_all()能把程序变成协作式运行，就是可以帮助程序实现异步。
5 import gevent,time,requests
6 #导入gevent、time、requests。
7
8 start = time.time()
9 #记录程序开始时间。
10
11 url_list = ['https://www.baidu.com/',
12 'https://www.sina.com.cn/',
13 'http://www.sohu.com/',
14 'https://www.qq.com/',
15 'https://www.163.com/',
16 'http://www.iqiyi.com/',
17 'https://www.tmall.com/',
18 'http://www.ifeng.com/']
19 #把8个网站封装成列表。
20
21 def crawler(url):
22 #定义一个crawler()函数。
23     r = requests.get(url)
```

```

24     #用requests.get()函数爬取网站。
25     print(url,time.time()-start,r.status_code)
26     #打印网址、请求运行时间、状态码。
27
28 tasks_list = [ ]
29 #创建空的任务列表。
30
31 for url in url_list:
32     #遍历url_list。
33     task = gevent.spawn(crawler,url)
34     #用gevent.spawn()函数创建任务。
35     tasks_list.append(task)
36     #往任务列表添加任务。
37 gevent.joinall(tasks_list)
38 #执行任务列表里的所有任务，就是让爬虫开始爬取网站。
39 end = time.time()
40 #记录程序结束时间。
41 print(end-start)
42 #打印程序最终所需时间。

```

- 第 1、3 行代码：从 `gevent` 库里导入了 `monkey` 模块，这个模块能将程序转换成可异步的程序。`monkey.patch_all()` 能给程序打上补丁，让程序变成是异步模式，而不是同步模式（我们要在导入其他库和模块前，先把 `monkey` 模块导入进来，并运行 `monkey.patch_all()`。这样，才能先给程序打上补丁。可以理解成这是一个规范的写法）；
- 第 5 行代码：我们导入了 `gevent` 库来帮我们实现多协程，导入了 `time` 模块来帮我们记录爬取所需时间，导入了 `requests` 模块帮我们实现爬取8个网站；
- 第 21、23、25 行代码：我们定义了一个 `crawler` 函数，只要调用这个函数，它就会执行【用 `requests.get()` 爬取网站】和【打印网址、请求运行时间、状态码】这两个任务；
- 第 33 行代码：因为 `gevent` 只能处理 `gevent` 的任务对象，不能直接调用普通函数，所以需要借助 `gevent.spawn()` 来创建任务对象（注意：`gevent.spawn()` 的参数需为要调用的函数名及该函数的参数。比如，`gevent.spawn(crawler,url)` 就是创建一个执行 `crawler` 函数的任务，参数为 `crawler` 函数名和它自身的参数 `url`）；



- 第 35 行代码：用 `append` 函数把任务添加到 `tasks_list` 的任务列表里；
- 第 37 行代码：调用 `gevent` 库里的 `joinall` 方法，能启动执行所有的任务。`gevent.joinall(tasks_list)` 就是执行 `tasks_list` 这个任务列表里的所有任务，开始爬取。

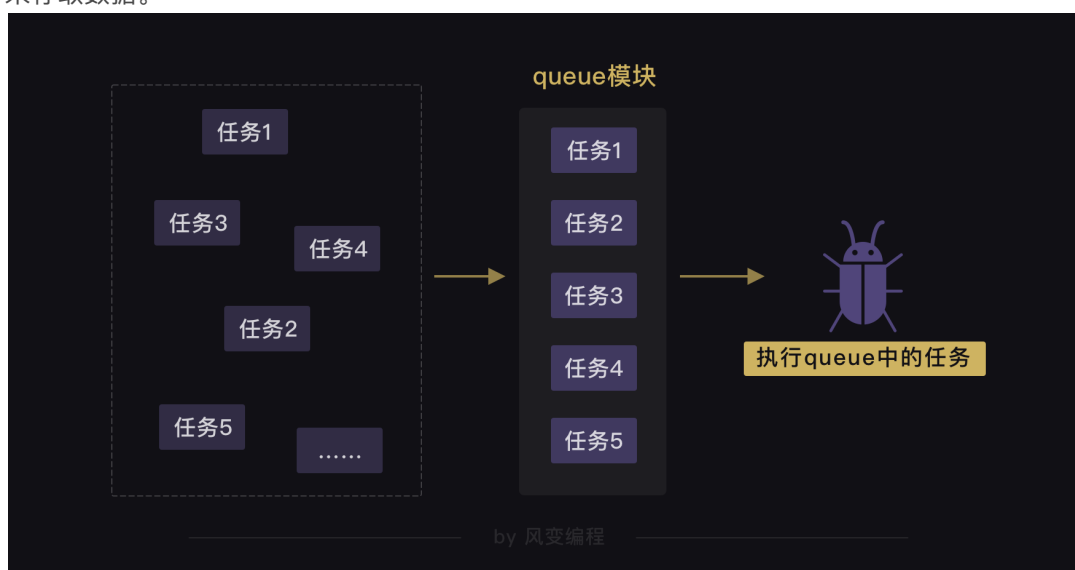
用gevent实现多协程爬取的重点

- 0 定义爬取函数
- 1 用gevent.spawn()创建任务
- 2 用gevent.joinall()执行任务

by 风变编程

1-2、queue 模块

queue 翻译成中文是队列的意思。我们可以用 queue 模块来存储任务，让任务都变成一条整齐的队列，就像银行窗口的排号做法。因为 queue 其实是一种有序的数据结构，可以用来存取数据。



```
1 from gevent import monkey
2 monkey.patch_all()
3 import gevent,time,requests
4 from gevent.queue import Queue
5
6 start = time.time()
7
8 url_list = ['https://www.baidu.com/',
9 'https://www.sina.com.cn/',
10 'http://www.sohu.com/',
11 'https://www.qq.com/',
12 'https://www.163.com/',
13 'http://www.iqiyi.com/',
14 'https://www.tmall.com/',
15 'http://www.ifeng.com/']
16
```

```

17 work = Queue()
18 for url in url_list:
19     work.put_nowait(url)
20
21 def crawler():
22     while not work.empty():
23         url = work.get_nowait()
24         r = requests.get(url)
25         print(url,work.qsize(),r.status_code)
26
27 tasks_list = [ ]
28
29 for x in range(2):
30     task = gevent.spawn(crawler)
31     tasks_list.append(task)
32 gevent.joinall(tasks_list)
33
34 end = time.time()
35 print(end-start)

```

1-2-1、导入模块

```

1 from gevent import monkey
2 #从gevent库里导入monkey模块。
3 monkey.patch_all()
4 #monkey.patch_all()能把程序变成协作式运行，就是可以帮助程序实现异步。
5 import gevent,time,requests
6 #导入gevent、time、requests
7 from gevent.queue import Queue
8 #从gevent库里导入queue模块

```

1-2-2、创建队列，并往队列存入数据

```

1 start = time.time()
2 #记录程序开始时间
3
4 url_list = ['https://www.baidu.com/',
5 'https://www.sina.com.cn/',
6 'http://www.sohu.com/',
7 'https://www.qq.com/',
8 'https://www.163.com/',
9 'http://www.iqiyi.com/',
10 'https://www.tmall.com/',
11 'http://www.ifeng.com/']
12
13 work = Queue()
14 #创建队列对象，并赋值给work。
15 for url in url_list:
16     #遍历url_list
17     work.put_nowait(url)
18     #用put_nowait()函数可以把网址都放进队列里。

```

1-2-3、定义爬取函数

```
1 def crawler():
2     while not work.empty():
3         #当队列不是空的时候，就执行下面的程序。
4         url = work.get_nowait()
5         #用get_nowait()函数可以把队列里的网址都取出。
6         r = requests.get(url)
7         #用requests.get()函数抓取网址。
8         print(url,work.qsize(),r.status_code)
9         #打印网址、队列长度、抓取请求的状态码。
```

- empty 方法，是用来判断队列是不是空了的；
- get_nowait 方法，是用来从队列里提取数据的；
- qsize 方法，是用来判断队列里还剩多少数量的。

queue对象的方法

put_nowait()	往队列里存储数据
get_nowait()	从队列里提取数据
empty()	判断队列是否为空
full()	判断队列是否为满
qsize()	判断队列还剩多少数量