

Eva_J

程序媛

[博客园](#)
[首页](#)
[新随笔](#)
[联系](#)
[管理](#)
[订阅](#)
[SML](#)

[随笔- 34](#)
[文章- 219](#)
[评论- 359](#)

python之路——协程

阅读目录

- 一 引子
- 二 协程介绍
- 三 Greenlet模块
- 四 Gevent模块

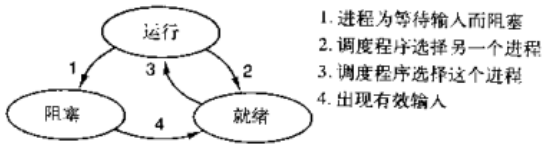
引子

之前我们学习了线程、进程的概念，了解了在操作系统中**进程是资源分配的最小单位,线程是CPU调度的最小单位**。按道理来说我们已经算是把cpu的利用率提高很多了。但是我们知道无论是创建多进程还是创建多线程来解决问题，都要消耗一定的时间来创建进程、创建线程、以及管理他们之间的切换。

随着我们对于效率的追求不断提高，**基于单线程来实现并发**又成为一个新的课题，即只用一个主线程（很明显可利用的cpu只有一个）情况下实现并发。这样就可以节省创建线程所消耗的时间。

为此我们需要先回顾下并发的本质：切换+保存状态

cpu正在运行一个任务，会在两种情况下切走去执行其他的任务（切换由操作系统强制控制），一种情况是该任务发生了阻塞，另外一种情况是该任务计算的时间过长



ps：在介绍进程理论时，提及进程的三种执行状态，而线程才是执行单位，所以也可以将上图理解为线程的三种状态

一：其中第二种情况并不能提升效率，只是为了让cpu能够雨露均沾，实现看起来所有任务都被“同时”执行的效果，如果多个任务都是纯计算的，这种切换反而会降低效率。

为此我们可以基于yield来验证。yield本身就是一种在单线程下可以保存任务运行状态的方法，我们来简单复习一下：

```
#1 yield可以保存状态，yield的状态保存与操作系统的保存线程状态很像，但是yield是代码级别控制的，更轻量级
#2 send可以把一个函数的结果传给另外一个函数，以此实现单线程内程序之间的切换

#串行执行
import time
def consumer(res):
    '''任务1:接收数据,处理数据'''
    pass

def producer():
    '''任务2:生产数据'''
    res=[]
    for i in range(10000000):
```

昵称：Eva_J
 园龄：4年9个月
 粉丝：4193
 关注：7
 +加关注

< 2020年8月 >						
日	一	二	三	四	五	六
26	27	28	29	30	31	1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	31	1	2	3	4	5

搜索

常用链接

- 我的随笔
- 我的评论
- 我的参与
- 最新评论
- 我的标签
- 更多链接

我的标签

[go\(1\)](#)
[python\(1\)](#)

随笔分类

- [python_Django\(4\)](#)
- [python基础语法\(7\)](#)
- [python面向对象\(4\)](#)
- [python网络编程\(1\)](#)
- [python线程进程与协程\(6\)](#)

随笔档案

- [2020年3月\(1\)](#)
- [2019年3月\(1\)](#)
- [2018年7月\(1\)](#)
- [2018年1月\(1\)](#)
- [2017年8月\(1\)](#)
- [2017年4月\(1\)](#)
- [2017年3月\(1\)](#)

```
        res.append(i)
    return res

start=time.time()
#串行执行
res=producer()
consumer(res) #写成consumer(producer())会降低执行效率
stop=time.time()
print(stop-start) #1.5536692142486572

#基于yield并发执行
import time
def consumer():
    '''任务1:接收数据,处理数据'''
    while True:
        x=yield

def producer():
    '''任务2:生产数据'''
    g=consumer()
    next(g)
    for i in range(10000000):
        g.send(i)

start=time.time()
#基于yield保存状态,实现两个任务直接来回切换,即并发的效果
#PS:如果每个任务中都加上打印,那么明显地看到两个任务的打印是你一次我一次,即并发执行的。
producer()

stop=time.time()
print(stop-start) #2.0272178649902344
```



二：第一种情况的切换。在任务一遇到io情况下，切到任务二去执行，这样就可以利用任务一阻塞的时间完成任务二的计算，效率的提升就在于此。

```
import time
def consumer():
    '''任务1:接收数据,处理数据'''
    while True:
        x=yield

def producer():
    '''任务2:生产数据'''
    g=consumer()
    next(g)
    for i in range(10000000):
        g.send(i)
        time.sleep(2)

start=time.time()
producer() #并发执行,但是任务producer遇到io就会阻塞住,并不会切到该线程内的其他任务去执行

stop=time.time()
print(stop-start)
```



对于单线程下，我们不可避免程序中出现io操作，但如果我们能在自己的程序中（即用户程序级别，而非操作系统级别）控制单线程下的多个任务能在一个任务遇到io阻塞时就切换到另外一个任务去计算，这样就保证了该线程能够最大限度地处于就绪态，即随时都可以被cpu执行的状态，相当于我们在用户程序级别将自己的io操作最大限度地隐藏起来，从而可以迷惑操作系统，让其看到：该线程好像是一直在计算，io比较少，从而更多的将cpu的执行权限分配给我们的线程。

协程的本质就是在单线程下，由用户自己控制一个任务遇到io阻塞了就切换另外一个任务去执行，以此来提升效率。为了实现它，我们需要找寻一种可以同时满足以下条件的解决方案：

- #1. 可以控制多个任务之间的切换，切换之前将任务的状态保存下来，以便重新运行时，可以基于暂停的位置继续执行。
- #2. 作为1的补充：可以检测io操作，在遇到io操作的情况下才发生切换

2016年6月(2)
2016年4月(1)
2016年3月(1)
2016年2月(1)
2016年1月(10)
2015年12月(6)
2015年11月(6)

文章分类

flask(1)
go(3)
mysql(7)
python宣讲专用课件(3)
python之路(16)
数据库相关(7)
周末班(3)

友链

银角大王
学霸yuan先生
冷先生

最新评论

1. Re:多表查询
查询每个部门最新入职的那位员工: select post,emp_name from empl
oyee where hire_date in (select
max(hire_date) from e...
--仰望夜空
2. Re:mysql索引原理
写的很细，很好，赞
--15927797249
3. Re:python——有一种线程池叫
做自己写的线程池
武sir的版本有完整的代码和例子吗？
或者原链接.....
--littlesnaka
4. Re:git操作备忘
sf~
--Chris2357
5. Re:python之路——博客目录
停更了1年多了呢
--Chris2357

阅读排行榜

1. python之路——博客目录(1450
38)
2. python——赋值与深浅拷贝(330
35)
3. python——SQL基本使用(3071
6)

协程介绍

协程：是单线程下的并发，又称微线程，纤程。英文名Coroutine。一句话说明什么是协程：**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。**、

需要强调的是：

```
#1. python的线程属于内核级别的，即由操作系统控制调度（如单线程遇到io或执行时间过长就会被迫交出cpu执行权限，切换其他线程）
#2. 单线程内开启协程，一旦遇到io，就会从应用程序级别（而非操作系统）控制切换，以此来提升效率（!!! 非io操作的切换与效率）
```

对比操作系统控制线程的切换，用户在单线程内控制协程的切换

优点如下：

```
#1. 协程的切换开销更小，属于程序级别的切换，操作系统完全感知不到，因而更加轻量级
#2. 单线程内就可以实现并发的效果，最大限度地利用cpu
```

缺点如下：

```
#1. 协程的本质是单线程下，无法利用多核，可以是一个程序开启多个进程，每个进程内开启多个线程，每个线程内开启协程
#2. 协程指的是单个线程，因而一旦协程出现阻塞，将会阻塞整个线程
```

总结协程特点：

- 1. 必须在只有一个单线程里实现并发
- 2. 修改共享数据不需加锁
- 3. 用户程序里自己保存多个控制流的上下文栈
- 4. 附加：一个协程遇到IO操作自动切换到其它协程（如何实现检测IO，yield、greenlet都无法实现，就用到了gevent模块（select机制））

Greenlet模块

安装：pip3 install greenlet

```
from greenlet import greenlet

def eat(name):
    print('%s eat 1' %name)
    g2.switch('egon')
    print('%s eat 2' %name)
    g2.switch()
def play(name):
    print('%s play 1' %name)
    g1.switch()
    print('%s play 2' %name)

g1=greenlet(eat)
g2=greenlet(play)

g1.switch('egon')#可以在第一次switch时传入参数，以后都不需要
```

单纯的切换（在没有io的情况下或者没有重复开辟内存空间的操作），反而会降低程序的执行速度

```
#顺序执行
import time
def f1():
    res=1
    for i in range(100000000):
        res+=i

def f2():
    res=1
    for i in range(100000000):
        res*=i
```

- 4. python——django使用mysql数据库（一）(23337)
- 5. 前端开发的正确姿势——各种文件的目录结构规划及引用(21500)

评论排行榜

- 1. python之路——博客目录(32)
- 2. python——赋值与深浅拷贝(16)
- 3. python——进程基础(9)
- 4. python的类和对象——进阶篇(8)
- 5. python_控制台输出带颜色的文字方法(8)

推荐排行榜

- 1. python之路——博客目录(63)
- 2. python——赋值与深浅拷贝(35)
- 3. python3.7导入gevent模块报错的解决方案(6)
- 4. python_控制台输出带颜色的文字方法(5)
- 5. python——挖装饰器祖坟事件(5)

```

start=time.time()
f1()
f2()
stop=time.time()
print('run time is %s' %(stop-start)) #10.985628366470337

#切换
from greenlet import greenlet
import time
def f1():
    res=1
    for i in range(100000000):
        res+=i
        g2.switch()

def f2():
    res=1
    for i in range(100000000):
        res*=i
        g1.switch()

start=time.time()
g1=greenlet(f1)
g2=greenlet(f2)
g1.switch()
stop=time.time()
print('run time is %s' %(stop-start)) # 52.763017892837524

```



greenlet只是提供了一种比generator更加便捷的切换方式，当切到一个任务执行时如果遇到io，那就原地阻塞，仍然是没有解决遇到IO自动切换来提升效率的问题。

单线程里的这20个任务的代码通常会既有计算操作又有阻塞操作，我们完全可以在执行任务1时遇到阻塞，就利用阻塞的时间去执行任务2。。。如此，才能提高效率，这就用到了Gevent模块。

Gevent模块

安装：pip3 install gevent

Gevent 是一个第三方库，可以轻松通过gevent实现并发同步或异步编程，在gevent中用到的主要模式是**Greenlet**，它是以C扩展模块形式接入Python的轻量级协程。Greenlet全部运行在主程序操作系统进程的内部，但它们被协作式地调度。



g1=gevent.spawn(func1,1,,2,3,x=4,y=5) 创建一个协程对象g1，spawn括号内第一个参数是函数名，如eat，后面可以有多个参数

g2=gevent.spawn(func2)

g1.join() #等待g1结束

g2.join() #等待g2结束

#或者上述两步合作一步: gevent.joinall([g1,g2])

g1.value#拿到func1的返回值



```

import gevent
def eat(name):
    print('%s eat 1' %name)
    gevent.sleep(2)
    print('%s eat 2' %name)

def play(name):
    print('%s play 1' %name)
    gevent.sleep(1)
    print('%s play 2' %name)

```

```

g1=gevent.spawn(eat, 'egon')
g2=gevent.spawn(play, name='egon')
g1.join()
g2.join()
#或者gevent.joinall([g1,g2])
print('主')

```



上例`gevent.sleep(2)`模拟的是`gevent`可以识别的io阻塞,而`time.sleep(2)`或其他的阻塞,`gevent`是不能直接识别的需要下面一行代码,打补丁,就可以识别了

```

from gevent import monkey;monkey.patch_all()

import gevent
import time
def eat():
    print('eat food 1')
    time.sleep(2)
    print('eat food 2')

def play():
    print('play 1')
    time.sleep(1)
    print('play 2')

g1=gevent.spawn(eat)
g2=gevent.spawn(play)
gevent.joinall([g1,g2])
print('主')

```



我们可以用`threading.current_thread().getName()`来查看每个g1和g2, 查看的结果为DummyThread-n, 即假线程

dummy

英 ['dʌmi] ㄉㄨˋ 美 ['dʌmi] ㄉㄨˋ

- n. 仿制品; 沉默寡言的人; 笨蛋, 蠢货; 挂名代表, 傀儡
- vt. 制作样本, 制作样张; 不吭声, 缄口; 〈美俚〉装聋作哑; 替别人占领土地
- adj. 虚设的; 假的; 摆样子的, 做样品的; 挂名的

```

from gevent import monkey;monkey.patch_all()
import threading
import gevent
import time
def eat():
    print(threading.current_thread().getName())
    print('eat food 1')
    time.sleep(2)
    print('eat food 2')

def play():
    print(threading.current_thread().getName())
    print('play 1')
    time.sleep(1)
    print('play 2')

g1=gevent.spawn(eat)
g2=gevent.spawn(play)
gevent.joinall([g1,g2])
print('主')

```



Gevent之同步与异步

```

from gevent import spawn, joinall, monkey; monkey.patch_all()

import time
def task(pid):
    """
    Some non-deterministic task
    """
    time.sleep(0.5)
    print('Task %s done' % pid)

def synchronous(): # 同步
    for i in range(10):
        task(i)

def asynchronous(): # 异步
    g_l=[spawn(task,i) for i in range(10)]
    joinall(g_l)
    print('DONE')

if __name__ == '__main__':
    print('Synchronous:')
    synchronous()
    print('Asynchronous:')
    asynchronous()
# 上面程序的重要部分是将task函数封装到Greenlet内部线程的gevent.spawn。
# 初始化的greenlet列表存放在数组threads中，此数组被传给gevent.joinall 函数，
# 后者阻塞当前流程，并执行所有给定的greenlet任务。执行流程只会在 所有greenlet执行完后才会继续向下走。

```

Gevent之应用举例一

通过gevent实现单线程下的socket并发

注意： from gevent import monkey; monkey.patch_all()一定要放到导入socket模块之前，否则gevent无法识别socket的阻塞

```

from gevent import monkey; monkey.patch_all()
from socket import *
import gevent

#如果不想用monkey.patch_all()打补丁,可以用gevent自带的socket
# from gevent import socket
# s=socket.socket()

def server(server_ip,port):
    s=socket(AF_INET,SOCK_STREAM)
    s.setsockopt(SOL_SOCKET,SO_REUSEADDR,1)
    s.bind((server_ip,port))
    s.listen(5)
    while True:
        conn,addr=s.accept()
        gevent.spawn(talk,conn,addr)

def talk(conn,addr):
    try:
        while True:
            res=conn.recv(1024)
            print('client %s:%s msg: %s' % (addr[0],addr[1],res))
            conn.send(res.upper())
    except Exception as e:
        print(e)
    finally:
        conn.close()

if __name__ == '__main__':
    server('127.0.0.1',8080)

```

```
from socket import *

client=socket (AF_INET,SOCK_STREAM)
client.connect (('127.0.0.1',8080))

while True:
    msg=input('>>: ').strip()
    if not msg:continue

    client.send(msg.encode('utf-8'))
    msg=client.recv(1024)
    print(msg.decode('utf-8'))
```

```
from threading import Thread
from socket import *
import threading

def client(server_ip,port):
    c=socket (AF_INET,SOCK_STREAM) #套接字对象一定要加到函数内，即局部名称空间内，放在函数外则被所有线程共享，则大家都能操作
    c.connect ((server_ip,port))

    count=0
    while True:
        c.send(('s say hello s' %(threading.current_thread().getName(),count)).encode('utf-8'))
        msg=c.recv(1024)
        print(msg.decode('utf-8'))
        count+=1
if __name__ == '__main__':
    for i in range(500):
        t=Thread(target=client,args=('127.0.0.1',8080))
        t.start()
```

分类: python之路

好文要顶

关注我

收藏该文

 **Eva_J**
关注 - 7
粉丝 - 4193

±加关注

2

0

posted @ 2018-01-21 17:22 Eva_J 阅读(7365) 评论(8) 编辑 收藏

发表评论

#1楼 2018-07-31 19:16 | HoneyCY

回复 引用

- 蒜蓉小龙虾
- 香辣小龙虾
- 十三香小龙虾
- 水煮牛肉
- 草莓糖葫芦
- 韩国烤肉
- 泡菜饼
- 海鲜饼
- 辣牛肉汤
- 泡菜汤
- 辣豆腐汤
- 干锅牛蛙
- 麻婆豆腐
- 口水鸡
- 凉拌猪肚
- 钵钵鸡

麻辣烫
酸辣粉
毛血旺
炒花甲
糖醋排骨
酸汤肥牛
酱猪蹄
烤乳猪
虾饺
烧卖
糯米鸡
粉果
马蹄糕
蟹黄包
奶油鸡蛋卷
肠粉
干炒牛河
湿炒牛河
艇仔粥
潮州牛肉丸
菠萝咕嚕肉
蒜蓉金针菇
叉烧
滑蛋牛肉
客家甜酒鸡
客家酿豆腐
猪脚姜
烧鹅
虎皮尖椒
糖醋里脊
蒜蓉粉丝蒸扇贝
凉瓜牛肉
香煎芙蓉蛋
蒜蓉开边虾
三杯鸡
酱猪蹄
宫保鸡丁
夫妻肺片
鱼香肉丝
粉蒸牛肉
粉蒸排骨
猪肚鸡
回锅肉
火锅
西红柿炒蛋
灯影牛肉
青椒鱼
担担面
红油耳丝
串串香
老妈蹄花
冒菜
红烧排骨
老坛子
樟茶鸭
汽锅丸子
热干面
腌笃鲜
蚂蚁上树
杭椒牛柳
辣子鸡
红烧牛肉
肉夹馍
凉皮
米皮
大盘鸡
锅贴
猪豚骨拉面
海鲜拉面
日式担担面
烤鱿鱼
盐烧三文鱼
烤青花鱼
烤银鳕鱼
烤鳗鱼
无骨原味炸鸡
天妇罗
草莓冰淇淋
巧克力冰淇淋
香草冰淇淋
四川泡菜

宫保虾球
干烧大黄鱼
清蒸鲈鱼
松仁玉米
梅菜扣肉
鱼头豆腐汤
脆皮鸡
黄金咸蛋卷
椒盐鱼下巴
板栗煲蹄花
鹅肝
醉泥螺
西湖醉鱼
熏鱼
麻油萝卜
盐水煮毛豆
马兰头拌香干
东坡肉
洋葱爆腰花
冬笋炒羊肉
干菜焖肉
铁板牛蛙
粽子
红烧带鱼 都觉得很赞

支持(9) 反对(3)

#2楼 2018-09-26 21:27 | 一捅浆糊 回复 引用

@ HoneyCY
李时珍的皮

支持(3) 反对(0)

#3楼 2019-05-07 11:30 | xue_shan 回复 引用

老师讲课好清晰，真女神。

支持(0) 反对(0)

#4楼 2019-05-15 09:59 | baozhilv 回复 引用

老师看了您的 10 期的几个视频，您装模块的时候都是 cmd 安装，其实可以直接在 pycharm 上装的，方便快捷，也不会有报错问题，如果您不知道的话

支持(0) 反对(5)

#5楼 2019-05-15 16:02 | baozhilv 回复 引用

自己测试了
from gevent import monkey;monkey.patch_all()
不一定要在导入默认模块之前

支持(0) 反对(0)

#6楼 2019-05-17 15:17 | 马昌伟 回复 引用

good good good

支持(0) 反对(0)

#7楼 2019-12-15 15:29 | Mr_Riven 回复 引用

@ HoneyCY
蒜蓉小龙虾
香辣小龙虾
十三香小龙虾
水煮牛肉
草莓糖葫芦
韩国烤肉
泡菜饼
海鲜饼
辣牛肉汤
泡菜汤
辣豆腐汤
干锅牛蛙
麻婆豆腐
口水鸡
凉拌猪肚
钵钵鸡
麻辣烫
酸辣粉
毛血旺
炒花甲
糖醋排骨
酸汤肥牛
酱猪蹄

烤乳猪
虾饺
烧卖
糯米鸡
粉果
马蹄糕
蟹黄包
奶油鸡蛋卷
肠粉
干炒牛河
湿炒牛河
艇仔粥
潮州牛肉丸
菠萝咕嚕肉
蒜蓉金针菇
叉烧
滑蛋牛肉
客家甜酒鸡
客家酿豆腐
猪脚姜
烧鹅
虎皮尖椒
糖醋里脊
蒜蓉粉丝蒸扇贝
凉瓜牛肉
香煎芙蓉蛋
蒜蓉开边虾
三杯鸡
酱猪蹄
宫保鸡丁
夫妻肺片
鱼香肉丝
粉蒸牛肉
粉蒸排骨
猪肚鸡
回锅肉
火锅
西红柿炒蛋
灯影牛肉
青椒鱼
担担面
红油耳丝
串串香
老妈蹄花
冒菜
红烧排骨
老坛子
樟茶鸭
汽锅丸子
热干面
腌笃鲜
蚂蚁上树
杭椒牛柳
辣子鸡
红烧牛肉
肉夹馍
凉皮
米皮
大盘鸡
锅贴
猪豚骨拉面
海鲜拉面
日式担担面
烤鱿鱼
盐烧三文鱼
烤青花鱼
烤银鳕鱼
烤鳗鱼
无骨原味炸鸡
天妇罗
草莓冰淇淋
巧克力冰淇淋
香草冰淇淋
四川泡菜
宫保虾球
干烧大黄鱼
清蒸鲈鱼
松仁玉米
梅菜扣肉
鱼头豆腐汤
脆皮鸡

黄金咸蛋卷
椒盐鱼下巴
板栗煲蹄花
鹅肝
醉泥螺
西湖醉鱼
熏鱼
麻油萝卜
盐水煮毛豆
马兰头拌香干
东坡肉
洋葱爆腰花
冬笋炒羊肉
干菜焖肉
铁板牛蛙
粽子
红烧带鱼 都觉得很赞

支持(0) 反对(0)

#8楼 2019-12-15 15:29 | Mr_Riven

回复 引用





@_ baozhilv
瞎说什么大实话

支持(0) 反对(0)

刷新评论 刷新页面 返回顶部

发表评论

编辑 预览

B    

支持 Markdown

提交评论 退出 订阅评论

[Ctrl+Enter快捷键提交]

- 【推荐】超50万行VC++源码: 大型组态工控、电力仿真CAD与GIS源码库
- 【推荐】了不起的开发者，挡不住的华为，园子里的品牌专区
- 【推荐】有道智云周年庆，API服务大放送，注册即送100元体验金！
- 【推荐】开放下载！《15分钟打造你自己的小程序》（内附详细代码）



相关博文：

- python之路
- 我的python之路
- python之路
- Python之路
- python之路
- » 更多推荐...

最新 IT 新闻：

- 消息人士：商汤科技考虑在完成新一轮融资后在科创板上市
- 《王者荣耀》曜FMVP皮肤“云鹰飞将”来了：飞檐走壁 如履平地
- 苏宁推49元换电池 覆盖200余款机型 已卖出超10000单
- 华为最高档天才少年：刚毕业年薪201万元 全球仅4人

· 美国强迫收购TikTok、限45天、还要中间费！李开复：做法不可思议
» 更多新闻...