gevent

376次阅读

Python通过yield提供了对协程的基本支持，但是不完全。而第三方的gevent为Python提供了比较完善的协程支持。

gevent是第三方库，通过greenlet实现协程，其基本思想是：

当一个greenlet遇到IO操作时，比如访问网络，就自动切换到其他的greenlet，等到IO操作完成，再在适当的时候切换回来继续执行。由于IO操作非常耗时，经常使程序处于等待状态，有了gevent为我们自动切换协程，就保证总有greenlet在运行，而不是等待IO。

由于切换是在IO操作时自动完成，所以gevent需要修改Python自带的一些标准库，这一过程在启动时通过monkey patch完成：

```
from gevent import monkey; monkey.patch_socket()
import gevent

def f(n):
    for i in range(n):
        print gevent.getcurrent(), i

g1 = gevent.spawn(f, 5)
g2 = gevent.spawn(f, 5)
g3 = gevent.spawn(f, 5)
g1.join()
g2.join()
g3.join()
```

运行结果：

```
<Greenlet at 0x10e49f550: f(5)> 0
<Greenlet at 0x10e49f550: f(5)> 1
<Greenlet at 0x10e49f550: f(5)> 2
<Greenlet at 0x10e49f550: f(5)> 3
<Greenlet at 0x10e49f550: f(5)> 4
<Greenlet at 0x10e49f910: f(5)> 0
<Greenlet at 0x10e49f910: f(5)> 1
<Greenlet at 0x10e49f910: f(5)> 2
<Greenlet at 0x10e49f910: f(5)> 3
<Greenlet at 0x10e49f910: f(5)> 4
<Greenlet at 0x10e49f4b0: f(5)> 0
<Greenlet at 0x10e49f4b0: f(5)> 1
<Greenlet at 0x10e49f4b0: f(5)> 2
<Greenlet at 0x10e49f4b0: f(5)> 3
<Greenlet at 0x10e49f4b0: f(5)> 4
```

可以看到，3个greenlet是依次运行而不是交替运行。

要让greenlet交替运行，可以通过gevent.sleep()交出控制权：

```
def f(n):
    for i in range(n):
        print gevent.getcurrent(), i
        gevent.sleep(0)
```

执行结果：

```
<Greenlet at 0x10cd58550: f(5)> 0
```

```
<Greenlet at 0x10cd58910: f(5)> 0
<Greenlet at 0x10cd584b0: f(5)> 0
<Greenlet at 0x10cd58550: f(5)> 1
<Greenlet at 0x10cd584b0: f(5)> 1
<Greenlet at 0x10cd58910: f(5)> 1
<Greenlet at 0x10cd58550: f(5)> 2
<Greenlet at 0x10cd58910: f(5)> 2
<Greenlet at 0x10cd584b0: f(5)> 2
<Greenlet at 0x10cd58550: f(5)> 3
<Greenlet at 0x10cd584b0: f(5)> 3
<Greenlet at 0x10cd58910: f(5)> 3
<Greenlet at 0x10cd58550: f(5)> 4
<Greenlet at 0x10cd58910: f(5)> 4
<Greenlet at 0x10cd584b0: f(5)> 4
```

3个greenlet交替运行，

把循环次数改为500000，让它们的运行时间长一点，然后在操作系统的进程管理器中看，线程数只有1个。

当然，实际代码里，我们不会用gevent.sleep()去切换协程，而是在执行到IO操作时，gevent自动切换，代码如下：

```
from gevent import monkey; monkey.patch_all()
import gevent
import urllib2

def f(url):
    print('GET: %s' % url)
    resp = urllib2.urlopen(url)
    data = resp.read()
    print('%d bytes received from %s.' % (len(data), url))

gevent.joinall([
        gevent.spawn(f, 'https://www.python.org/'),
        gevent.spawn(f, 'https://www.yahoo.com/'),
        gevent.spawn(f, 'https://github.com/'),
])
```

运行结果：

```
GET: https://www.python.org/
GET: https://www.yahoo.com/
GET: https://github.com/
45661 bytes received from https://www.python.org/.
14823 bytes received from https://github.com/.
304034 bytes received from https://www.yahoo.com/.
```

从结果看，3个网络操作是并发执行的，而且结束顺序不同，但只有一个线程。

小结

使用gevent，可以获得极高的并发性能，但gevent只能在Unix/Linux下运行，在Windows下不保证正常安装和运行。

由于gevent是基于IO切换的协程，所以最神奇的是，我们编写的Web App代码，不需要引入gevent的包，也不需要改任何代码，仅仅在部署的时候，用一个支持gevent的WSGI服务器，立刻就获得了数倍的性能提升。具体部署方式可以参考后续“实战”-“部署Web App”一节。
