tornado异步模型

在系统中，cup的速度远高于i/o速度，i/o包括网络访问、数据库访问、本地文件访问等，我们常见requests、urllib、pymysql等库都是同步i/o；在i/o期间cpu处于等待状态，但又不能执行其他操作，这就是一般系统效率低下的原因。而tornado可以很好解决这个问题。

1、基本概念

- 阻塞

  调用函数的时候，当前线程被挂起

- 非阻塞

  调用函数时，当前线程不挂起，会立即返回

以做饭为例，

- 同步就是，让你老婆做饭，然后一直等她做好饭，然后在吃饭
- 异步就是，告诉你老婆做饭，老婆同意做饭，你可以上上网、玩玩游戏，做好饭后你老婆会通知你；如果不同意，她也会立刻告诉你。

同步和异步更关心的是**获取结果的方式**，同步是获取结果后才会进行下一步操作，异步会等通知，得到通知后才会进行下一步；**阻塞和非阻塞更关注当前的线程状态**。所以同步可以调用阻塞也可以调用非阻塞接口，异步调用非阻塞接口。

- i/o多路复用

  i/o多路复用是一种机制，一个进程可以监视多个描述符，一旦某个描述符就绪（可读或可写）能够通知程序进行相应操作。

  i/o多路复用好比是找一个宿管大妈来帮你监视下楼的女生, 这个期间你可以些其他的事情. 例如可以顺便看看其他妹子,玩玩王者荣耀, 上个厕所等等i/o复用又包括 select, poll, epoll 模式. 那么它们的区别是什么?

  - **select大妈** 每一个女生下楼, select大妈都不知道这个是不是你的女神, 她需要一个一个询问, 并且select大妈能力还有限, 最多一次帮你监视1024个妹子
  - **poll大妈**不限制盯着女生的数量, 只要是经过宿舍楼门口的女生, 都会帮你去问是不是你女神

- **epoll大妈**不限制盯着女生的数量, 并且也不需要一个一个去问. 那么如何做呢? epoll大妈会为每个进宿舍楼的女生脸上贴上一个大字条,上面写上女生自己的名字, 只要女生下楼了, epoll大妈就知道这个是不是你女神了, 然后大妈再通知你

2.tornado中异步

之前，我们的程序并没有对RequestHandler中的get或post方法进行异步化处理，这就意味着，一旦在get或post方法中出现了耗时间的操作，不仅仅是当前请求被阻塞，按照Tornado框架的工作模式，其他的请求也会被阻塞，所以我们需要对耗时间的操作进行异步化处理，以前有两种方式：

- 在Tornado稍早一些的版本中，可以用装饰器实现请求方法的异步化或协程化来解决这个问题。
- 给`RequestHandler`的请求处理函数添加`@tornado.web.asynchronous`装饰器，如下所示：

```
class AsyncReqHandler(RequestHandler):

    @tornado.web.asynchronous
    def get(self):
        http = httpclient.AsyncHTTPClient()
        http.fetch("http://example.com/", self._on_download)

    def _on_download(self, response):
        do_something_with_response(response)
        self.render("template.html")
```

- 给`RequestHandler`的请求处理函数添加`@tornado.gen.coroutine`装饰器，如下所示：

```
class GenAsyncHandler(RequestHandler):

    @tornado.gen.coroutine
    def get(self):
        http_client = AsyncHTTPClient()
        response = yield http_client.fetch("http://example.com")
        do_something_with_response(response)
        self.render("template.html")
```

- 使用`@return_future`装饰器，如下所示：

```
@return_future
def future_func(arg1, arg2, callback):
    # Do stuff (possibly asynchronous)
    callback(result)

async def caller():
    await future_func(arg1, arg2)
```

在Tornado 5.x版本中，这几个装饰器都被标记为**deprcated**（过时），我们可以通过Python 3.5中引入的`async`和`await`（在Python 3.7中已经成为正式的关键字）来达到同样的效果。当然，要实现异步化还得靠其他的支持异步操作的三方库来支持，如果请求处理函数中用到了不支持异步操作的三方库，就需要靠自己写包装类来支持异步化。经常使用异步化封装的第三方库：

- 访问Web服务器：requests → aiohttp
- 操作MySQL：pymysql → aiomysql
- 操作Redis：redis → aioredis
- 操作MongoDB：pymongo → motor
-

```
# tornado异步化
import tornado.web
import tornado.ioloop
import asyncio
class IndexHandler(tornado.web.RequestHandler):
     async def get(self):
        await  asyncio.sleep(5)
        self.write(time.ctime())

def main():
```

```
    app = tornado.web.Application([(r'/',IndexHandler)],debug=True)
    app.listen(8888)
    print("server start")
    tornado.ioloop.IOLoop.current().start()

if __name__ == '__main__':
    main()
```

我们可以使用ab压力测试工具来比较一下异步和同步的效率。

```
# 安装ab
sudo apt-get install apache2-utils

#测试:
# 异步测试
(tornadoenv) python@ubuntu:~/Desktop/course/tornado$ ab -n 3 -c 3
http://127.0.0.1:8888/
This is ApacheBench, Version 2.3 <$Revision: 1706008 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd,
http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking 127.0.0.1 (be patient).....done


Server Software:        TornadoServer/6.0.3
Server Hostname:        127.0.0.1
Server Port:            8888

Document Path:          /
Document Length:        24 bytes

Concurrency Level:      3
Time taken for tests:   5.008 seconds   # 3个并发请求耗时
Complete requests:      3
Failed requests:        0
Total transferred:      657 bytes
HTML transferred:       72 bytes
Requests per second:    0.60 [#/sec] (mean)
Time per request:       5007.624 [ms] (mean)
Time per request:       1669.208 [ms] (mean, across all concurrent
requests)
```

```
Transfer rate:          0.13 [Kbytes/sec] received
```

```
import tornado.web
import tornado.ioloop
import time
import asyncio
class IndexHandler(tornado.web.RequestHandler):
     def get(self):
        time.sleep(5)
        self.write(time.ctime())

def main():
    app = tornado.web.Application([(r'/',IndexHandler)],debug=True)
    app.listen(8888)
    print("server start")
    tornado.ioloop.IOLoop.current().start()

if __name__ == '__main__':
    main()
```

```
# 同步测试
(tornadoenv) python@ubuntu:~/Desktop/course/tornado$ ab -n 3 -c 3
http://127.0.0.1:8888/
This is ApacheBench, Version 2.3 <$Revision: 1706008 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd,
http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking 127.0.0.1 (be patient)...
Benchmarking 127.0.0.1 (be patient).....done


Server Software:        TornadoServer/6.0.3
Server Hostname:        127.0.0.1
Server Port:            8888

Document Path:          /
Document Length:        24 bytes

Concurrency Level:      3
Time taken for tests:   15.021 seconds   # 耗时15秒，是异步的3倍
```

```
Complete requests:      3
Failed requests:        0
Total transferred:      657 bytes
HTML transferred:       72 bytes
Requests per second:    0.20 [#/sec] (mean)
Time per request:       15021.130 [ms] (mean)
Time per request:       5007.043 [ms] (mean, across all concurrent
requests)
```

- tornado使用aiomysql

```
# 如果没有安装iomysql,安装先
#pip install aiomysql

from tornado.web import StaticFileHandler, RedirectHandler
from aiomysql import create_pool
import time

from tornado import web
import tornado
from tornado.web import template


class MainHandler(web.RequestHandler):
    def initialize(self, db):
        self.db = db
    # 异步访问
    async def get(self, *args, **kwargs):
        uid = ""
        username = ""
        password = ""

        async with create_pool(host=self.db["host"],
port=self.db["port"],
                               user=self.db["user"],
password=self.db["password"],
                               db=self.db["name"], charset="utf8") as
pool:
            async with pool.acquire() as conn:
                async with conn.cursor() as cur:
                    await cur.execute("SELECT id, username, password
from user")
```

```
                    try:
                        uid, username, password = await
cur.fetchone()
                    except Exception as e:
                        print(e)
                        pass
        self.render("message.html", id=uid,  username=username,
password=password)
# 配置
settings = {
    "static_path": "C:/projects/tornado_overview/chapter03/static",#
静态资源路径
    "template_path": "templates",   #模板路径
    "db": {    #数据库配置
        "host": "127.0.0.1",
        "user": "root",
        "password": "123",
        "name": "db5",
        "port": 3306
    }
}

if __name__ == "__main__":
    app = web.Application([
        ("/", MainHandler, {"db": settings["db"]}),], debug=True,
**settings)
    app.listen(8888)
    tornado.ioloop.IOLoop.current().start()
```