怎么吧一个同步的函数变成一个 异步的函数呢？？ 请使用mysql的异步客户端pymysql等配合tornado

怎么把一个同步的函数变成一个 异步的函数呢？？

这样才能在tornado里面使用 该异步函数来实现并发 ！

目前我知道的几种做法：

1.使用 @threadExecutor 装饰器来使用多线程在背后跑，使得当前的同步函数变成了异步函数来跑

（但本质上不就是变成了多线程了吗，那还要多协程干嘛？？）

2.使用 @tornado-celery装饰器， 使用任务队列来把同步函数丢到 celery队列里去执行，

这样对于tornado来说该函数就是异步执行的了（结合celery等任务队列来实现异步，不是很懂）

3.tornado已经给我们写好了一个 AsyncHttpClient库了，

其实常用的异步函数基本本质上都是走到发起http请求那一步，比如访问mysql数据库拿数据等，本质上就是http请求而已，

所以，完全可以基于tornado的AsyncHttpClient来封装实现各种自己的异步函数http请求库！！

（比如自己写个Async_mysql_query等【其实tornado已经写好了,现成的数据库异步库（如pymysql/motor等）网上也很多】），

所以，异步http请求函数AsyncHttpClient是我们的一个利器，一定要好好使用它，基本就可以实现几乎所有需要的异步http请求函数，

就算不自己写，网上也有很多异步库可以使用！

（只是movoto偏偏使用了一个奇葩的mysql同步库，还是不走pytohn的socket而走c的socket的，所以连gevent也没法用，

注定movoto连接mysql这一块并发不起来。。）

----------异步mysql库使用介绍----------

咱们经常使用的mysql库，MySQL-Python库是用C写的，很遗憾它是阻塞的，要实现异步的MySQL驱动必须用Python版本的MySQL驱动！

现在社区里面有两个纯python实现的mysql驱动。一个是 myconnpy 另一个是PyMysql ~ 这两个mysql驱动文档相当的少呀，好在他们的用法和MySQldb相当的像，不然就要头疼的看代码了。。。 实现的方式是用socket来交互，不像mysqldb封装了libmysqlclient那样 ！

myconnpy中国的tornado大牛推荐过，但是也评价过，貌似有些bug的样子。
我这边就用Mozilla公司 也在用的pymysql ~

咱们先创建数据库和数据表

```
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> create database rui;
Query OK, 1 row affected (0.00 sec)

mysql> use rui;
Database changed
mysql> create table kkk(id int(10) not null primary key auto_increment,
    -> name char(20) not null default '0',
    -> info char(20) not null default '0',
    -> ranstr char(20) not null default '0');
Query OK, 0 rows affected (0.04 sec)

mysql>
mysql> desc kkk;
+--------+----------+------+-----+---------+----------------+
| Field  | Type     | Null | Key | Default | Extra          |
+--------+----------+------+-----+---------+----------------+
| id     | int(10)  | NO   | PRI | NULL    | auto_increment |
| name   | char(20) | NO   |     | 0       |                |
| info   | char(20) | NO   |     | 0       |                |
| ranstr | char(20) | NO   |     | 0       |                |
+--------+----------+------+-----+---------+----------------+
4 rows in set (0.00 sec)

mysql>
```


201006358.jpg

安装python的mysql模块~

```
[root@101 ~]#
[root@101 ~]# pip install PyMySQL
Downloading/unpacking PyMySQL
  Downloading PyMySQL-0.6.1.tar.gz (51kB): 51kB downloaded
  Running setup.py egg_info for package PyMySQL
Installing collected packages: PyMySQL
  Running setup.py install for PyMySQL
Successfully installed PyMySQL
Cleaning up...
[root@101 ~]#
```


200829712.jpg

PySQL针对mysql操作demo还算简单的~

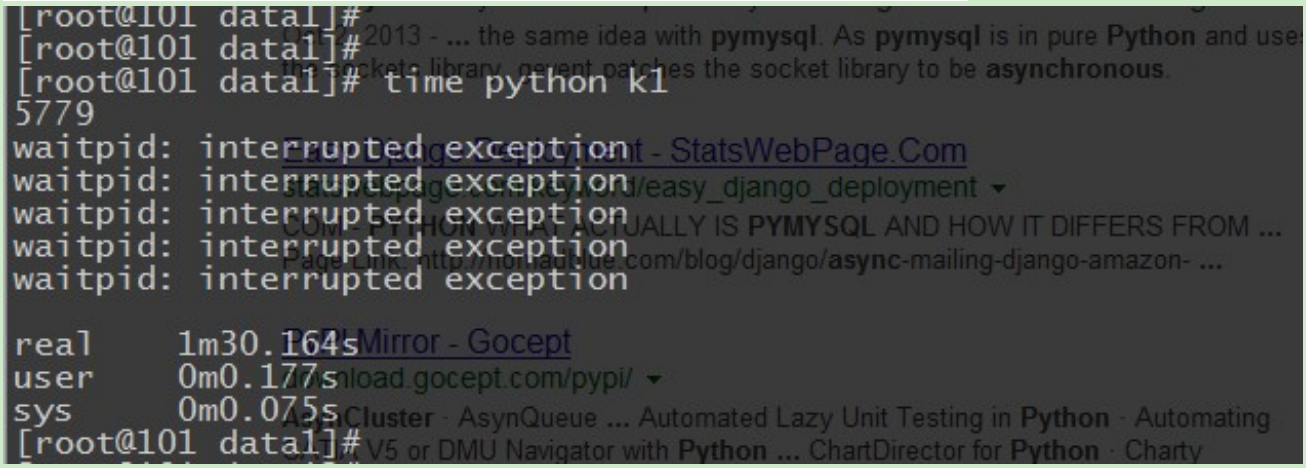
```
import pymysql
db = pymysql.connect(host = 'localhost', passwd = '123123', user = 'root', db = 'ru
i')
cursor = db.cursor()
sql='select count(*) from kkk'
data = cursor.execute(sql)
print cursor
cursor.close()
db.close()
```

其实咱们也可以模拟apache那样prefork模式，来派生任务执行对象。

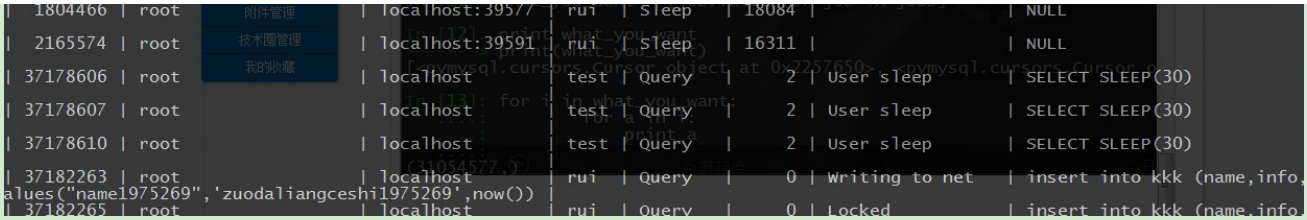
prefork采用预派生子进程方式，用单独的子进程来处理 不同的请求，进程之间彼此独立。我这边测试是mysql堵塞方式，大家也可以利用这种方案模拟多个任务执行。有点类似多进程的样子，消耗比较大的~

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
#xiaorui.cc
import MySQLdb, pymysql
import signal, os, sys

workers = {}

def run():
con = MySQLdb.connect(host='localhost', db='rui', user='root', passwd='123123')
signal.signal(signal.SIGTERM, lambda sig, status: sys.exit(0))
cur = con.cursor()
cur.execute("SELECT SLEEP(30)")

def killall(sig, status):
for pid in workers.keys():
os.kill(pid, signal.SIGTERM)

def waitall():
for pid in workers.keys():
try:
os.waitpid(pid, 0)
except:
print "waitpid: interrupted exception"

def main():
print os.getpid()
signal.signal(signal.SIGTERM, killall)
for i in range(3):
pid = os.fork()
if pid == 0:
try:
run()
except:
print "run: interrupted exception"
sys.exit(0)
else:
workers[pid] = 1
waitall()

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

看到这三个sleep30都在跑吗？ 看起来生效了，但三个进程还是消耗了130秒。



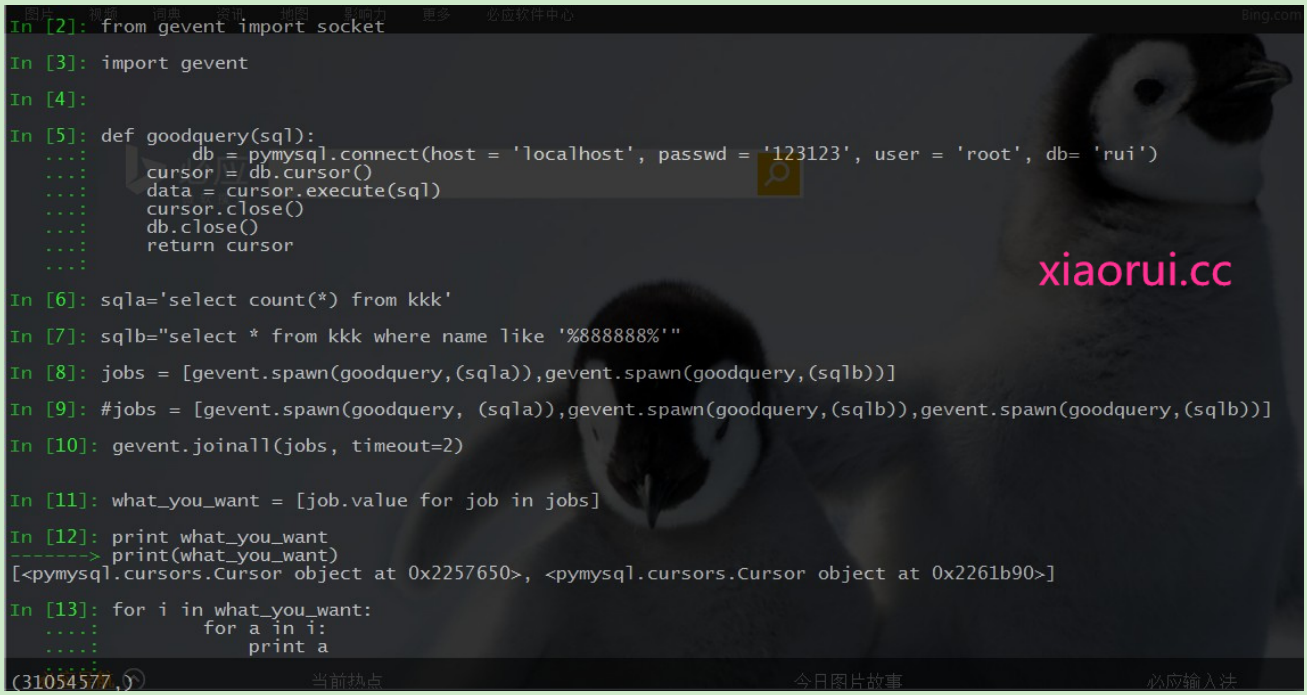
230057221.jpg



225729957.jpg

好了说正题，今天怎么主要说的就是gevent和python下的mysql驱动测试，分享个gevent和pymysql的在一起使用的实例demo ~

```
def goodquery(sql):
db = pymysql.connect(host = 'localhost', passwd = '123123', user = 'root', db= 'rui')
cursor = db.cursor()
data = cursor.execute(sql)
cursor.close()
db.close()
return cursor
sqla='select count(*) from kkk'
sqlb="select * from kkk where name like '%888888%'"
jobs = [gevent.spawn(goodquery, (sqla)), gevent.spawn(goodquery, (sqlb))]
#jobs = [gevent.spawn(goodquery, (sqla)),gevent.spawn(goodquery,(sqlb)),gevent.spawn(
goodquery,(sqlb))]
gevent.joinall(jobs, timeout=2)
what_you_want = [job.value for job in jobs]
print what_you_want
for i in what_you_want:
for a in i:
print a
```



221013629.jpg

哎，还是有点堵塞。。。 跑了6个耗时3s的sql，共用了18秒的时间。。。。

```
(17513532, 'name888888', 'zuodaliangceshi88888', '2013-12-08 20:19:01')
(17515709, 'name888888', 'zuodaliangceshi88888', '2013-12-08 20:19:02')
(17553440, 'name888888', 'zuodaliangceshi88888', '2013-12-08 20:19:21')
(17555994, 'name888888', 'zuodaliangceshi88888', '2013-12-08 20:19:22')
(17556854, 'name888888', 'zuodaliangceshi88888', '2013-12-08 20:19:23')
(17560812, 'name888888', 'zuodaliangceshi88888', '2013-12-08 20:19:25')
(17561852, 'name888888', 'zuodaliangceshi88888', '2013-12-08 20:19:25')
(17563912, 'name888888', 'zuodaliangceshi88888', '2013-12-08 20:19:26')
(17564174, 'name888888', 'zuodaliangceshi88888', '2013-12-08 20:19:27')
(17567541, 'name888888', 'zuodaliangceshi88888', '2013-12-08 20:19:28')
(17569163, 'name888888', 'zuodaliangceshi88888', '2013-12-08 20:19:29')
(17569395, 'name888888', 'zuodaliangceshi88888', '2013-12-08 20:19:29')
(17572455, 'name888888', 'zuodaliangceshi88888', '2013-12-08 20:19:31')

real    0m18.259s
user    0m0.073s
sys     0m0.013s
[root@101 ~]#
```

231823673.jpg

经过一上午的折腾，得知gevent的版本没有用对，只有gevent 1.0 才完美支持socket，然后需要在引入模块的后面，打上别的补丁!

```
In [1]: import gevent

In [2]: print ge
getattr  gevent

In [2]: print gevent.version_info
------> print(gevent.version_info)
(1, 0, 0, 'final', 0)

In [3]:
```

143638790.png

```
1  gevent.monkey.patch_socket()
```

```
1  real 0m8.993s
2  user 0m0.071s
3  sys 0m0.016s
```

在这里我再测试下多线程的版本：

```
import pymysql
import threading
def goodquery(sql):
db = pymysql.connect(host = 'localhost', passwd = '123123', user = 'root', db= 'rui')
cursor = db.cursor()
data = cursor.execute(sql)
cursor.close()
db.close()
print cursor
return cursor
sqla='select count(*) from kkk'
sqlb="select * from kkk where name like '%888888%'"
#jobs = [gevent.spawn(goodquery,(sqla)),gevent.spawn(goodquery,(sqlb)),gevent.spawn(goodquery,(s
vent.spawn(goodquery,(sqlb))]
#jobs = [gevent.spawn(goodquery, (sqla)),gevent.spawn(goodquery,(sqlb)),gevent.spawn(goodquery,
#gevent.joinall(jobs, timeout=30)
#what_you_want = [job.value for job in jobs]
threads=[]
for i in range(5):
threads.append( threading.Thread( target=goodquery,args=(sqlb,) ) )
for t in threads:
t.start()
for t in threads:
t.join()
```

```
mysql> show processlist;
+----------+------+-----------------+------+---------+------+-------------+-----------------------------------------------+
| Id       | User | Host            | db   | Command | Time | State       | Info                                          |
+----------+------+-----------------+------+---------+------+-------------+-----------------------------------------------+
| 51552924 | root | localhost       | rui  | Query   |    0 | NULL        | show processlist                              |
| 51552930 | root | localhost:43010 | rui  | Query   |    5 | Sending data | select * from kkk where name like '%888888%' |
| 51552931 | root | localhost:43011 | rui  | Query   |    5 | Sending data | select * from kkk where name like '%888888%' |
| 51552932 | root | localhost:43012 | rui  | Query   |    5 | Sending data | select * from kkk where name like '%888888%' |
| 51552933 | root | localhost:43015 | rui  | Query   |    5 | Sending data | select * from kkk where name like '%888888%' |
| 51552934 | root | localhost:43016 | rui  | Query   |    5 | Sending data | select * from kkk where name like '%888888%' |
+----------+------+-----------------+------+---------+------+-------------+-----------------------------------------------+
6 rows in set (0.00 sec)

mysql> show processlist;
+----------+------+-----------------+------+---------+------+-------------+-----------------------------------------------+
| Id       | User | Host            | db   | Command | Time | State       | Info                                          |
+----------+------+-----------------+------+---------+------+-------------+-----------------------------------------------+
| 51552924 | root | localhost       | rui  | Query   |    0 | NULL        | show processlist                              |
| 51552930 | root | localhost:43010 | rui  | Query   |    6 | Sending data | select * from kkk where name like '%888888%' |
| 51552931 | root | localhost:43011 | rui  | Query   |    6 | Sending data | select * from kkk where name like '%888888%' |
| 51552932 | root | localhost:43012 | rui  | Query   |    6 | Sending data | select * from kkk where name like '%888888%' |
| 51552933 | root | localhost:43015 | rui  | Query   |    6 | Sending data | select * from kkk where name like '%888888%' |
| 51552934 | root | localhost:43016 | rui  | Query   |    6 | Sending data | select * from kkk where name like '%888888%' |
+----------+------+-----------------+------+---------+------+-------------+-----------------------------------------------+
6 rows in set (0.00 sec)

mysql> show processlist;
+----------+------+-----------------+------+---------+------+-------------+-----------------------------------------------+
| Id       | User | Host            | db   | Command | Time | State       | Info                                          |
+----------+------+-----------------+------+---------+------+-------------+-----------------------------------------------+
| 51552924 | root | localhost       | rui  | Query   |    0 | NULL        | show processlist                              |
| 51552930 | root | localhost:43010 | rui  | Query   |    7 | Sending data | select * from kkk where name like '%888888%' |
| 51552931 | root | localhost:43011 | rui  | Query   |    7 | Sending data | select * from kkk where name like '%888888%' |
| 51552933 | root | localhost:43015 | rui  | Query   |    7 | Sending data | select * from kkk where name like '%888888%' |
| 51552934 | root | localhost:43016 | rui  | Query   |    7 | Sending data | select * from kkk where name like '%888888%' |
+----------+------+-----------------+------+---------+------+-------------+-----------------------------------------------+
```


034003590.jpg

他的测试结果要比gevent慢点 ~但也是并发的执行，可以在mysql进程里面看到执行的记录 ~

```
[root@101 ~]# time python t.py
real 0m11.122s
user 0m0.095s
sys 0m0.026s
```

总结下：

gevent pymysql 或者是 threading pymysql 是靠谱的~ 是可以解决大数据下的mysql读写堵塞的问题的~

但是和sohu、腾讯的朋友讨论了下我的这个方案，mysql堵塞是在与事务处理时发生的。看来mysql的堵塞不是这么搞解决的，以后有环境后，会继续的追踪这事 ~