An Introduction to gevent

Jordan Woehr

PyYYC - September 4, 2013

What is gevent

“gevent is a coroutine-based Python networking library that uses greenlet to provide a high-level synchronous API on top of the libevent event loop.”

- http://www.gevent.org/

Advantages of gevent

- Ideal for IO bound applications
  - No process/thread creation overhead

- Single threaded
  - Avoid GIL problems
  - Avoid locking problems

libevent

- C library
- Asynchronous event notification for
  - file descriptors
  - timeouts
  - signals
- libev
  - gevent 1.0 will use

greenlet

- Cooperatively scheduled pseudo-thread
  - a.k.a. Tasklet

```
from greenlet import greenlet

def test1():
    print 12
    gr2.switch()
    print 34

def test2():
    print 56
    gr1.switch()
    print 78

gr1 = greenlet(test1)
gr2 = greenlet(test2)
gr1.switch()
```

Output:

```
12
56
34
```

Non-blocking Synchronous IO

EchoServer.py

```
import gevent
from gevent import socket

def handle_client(conn, addr):
    while True:
        data = conn.recv(1024)
        if not data:
            print 'Client closed connection.'
            break
        print 'Received the string \'%s\'. Echoing it back.' % data
        conn.send(data)
    conn.close()

def main():
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    s.bind(('', 80))
    s.listen(1)
    while True:
        conn, addr = s.accept()
        gevent.spawn(lambda: handle_client(conn, addr))

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

EchoClient.py

```
from gevent import socket

def main():
    s = socket.socket()
    s.connect(('localhost', 80))
    s.send('Hello world!')
    print s.recv(1024)
    s.close()

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

Demo: RandomServer

```
import gevent
import random

from flask import Flask
app = Flask('Random')

@app.route('/random')
def handle_random():
    # Sleep for 0-1 seconds
    r = random.random()
    gevent.sleep(r)
    return '%f' % r

def main():
    from gevent.pywsgi import WSGIServer

    s = WSGIServer(('', 45678), app)
    try:
        s.serve_forever()
    except KeyboardInterrupt:
        pass
    except Exception as e:
        print('Unexpected exception:', e)
    finally:
        print('Stopping login server...')
        s.stop()

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

RandomServer.py

Your task

- Create a client that makes requests in parallel
- Hints:
  - Use gevent to make urllib2 “green”
    - `from gevent import monkey`
    - `monkey.patch_socket()`
  - Use gevent.spawn() to create a new greenlet
    - `g = gevent.spawn(some_func)`
  - Join greenlets by either:
    - `g.join()`
    - `gevent.joinall([g])`
  - Hammer my AWS instance in the follow way:

```
req = urllib2.urlopen(
'http://ec2-54-213-134-62.us-west-2.compute.amazonaws.com :45678/random'
)
num = float(req.read())
```