Concurrent Python

Cody Soyland

Austin Web Python User Group - Feb 28, 2013

UMBEL™

Getting started

- Notes on codysoyland.com
- Install gevent 1.0 (release candidate)
- Load my iPython Notebooks

The “Real-time” Web

- Technologies that enable delivery of information to users the instant the servers know about it
- Interesting scaling challenges
- Lots of open connections

High concurrency creates new challenges

- Non-blocking I/O
- Low resource overhead
- Distributed

Concurrency is not the same thing as Parallelism

- Concurrency is about the composition of independent processes
- Parallelism is about the simultaneous execution of independent processes

Building concurrent systems

- Processes (ie. CGI, mpm_prefork)
- Threads (ie. mpm_worker, most common)
- Non-blocking I/O
  - Callbacks (CPS, Reactor Pattern)
  - Coroutines

Threads

- Pre-emptive scheduling (non-deterministic)
  - Race conditions and locks/mutexes
  - Memory overhead
  - Readable, synchronous interface
  - Guaranteed cooperation

Callbacks

- Call stack not preserved
- Simple things are intuitive
- Complex things become confusing

Coroutines

- Call stack preserved
- Synchronous API
- Benefits of threads without the non-determinism

Greenlet

- True coroutines in Python
- Call switching from Stackless Python implemented as a C extension module
- Call stack slicing to preserve context
  - Portions of the stack are copied to the heap and vice-versa

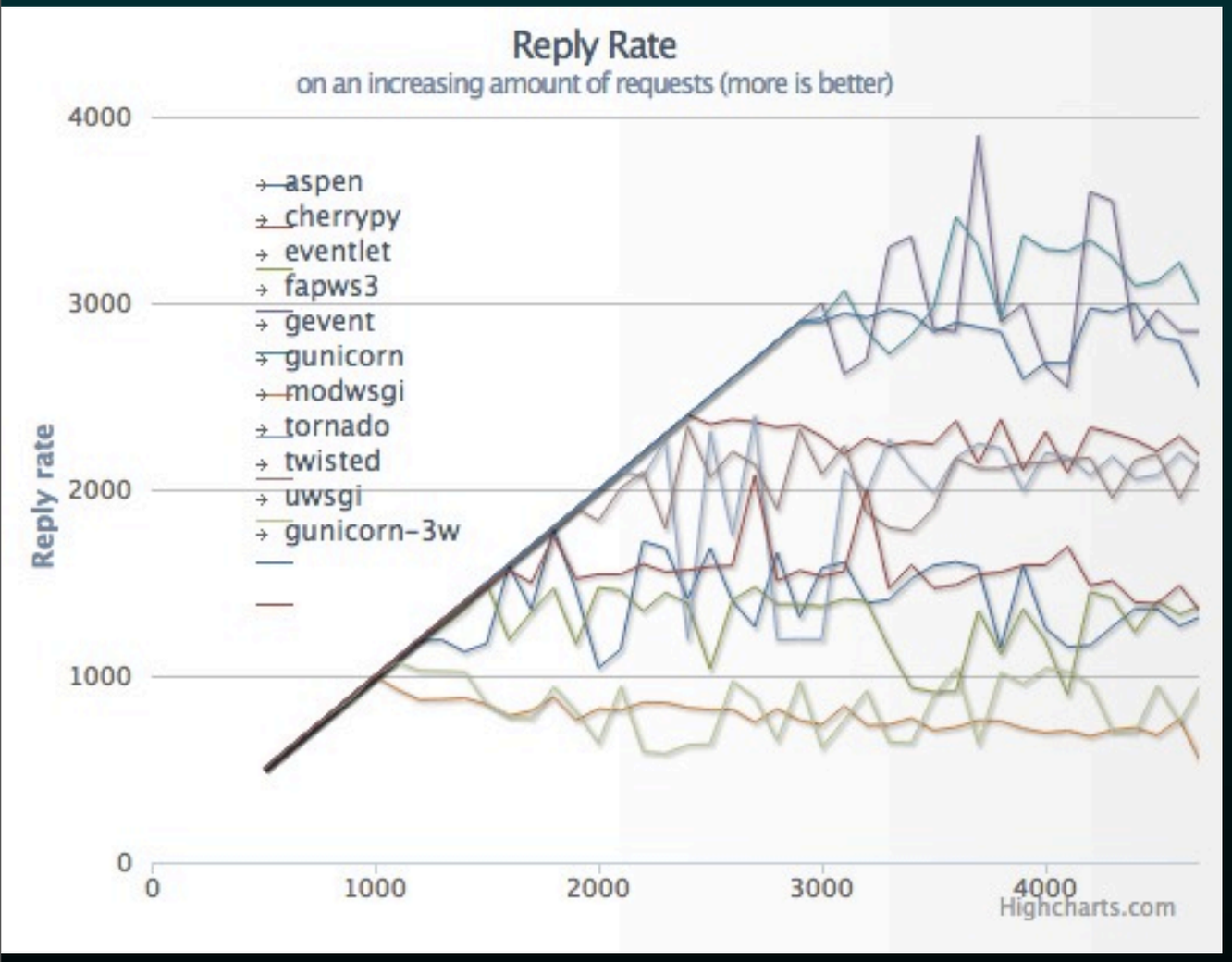
Gevent

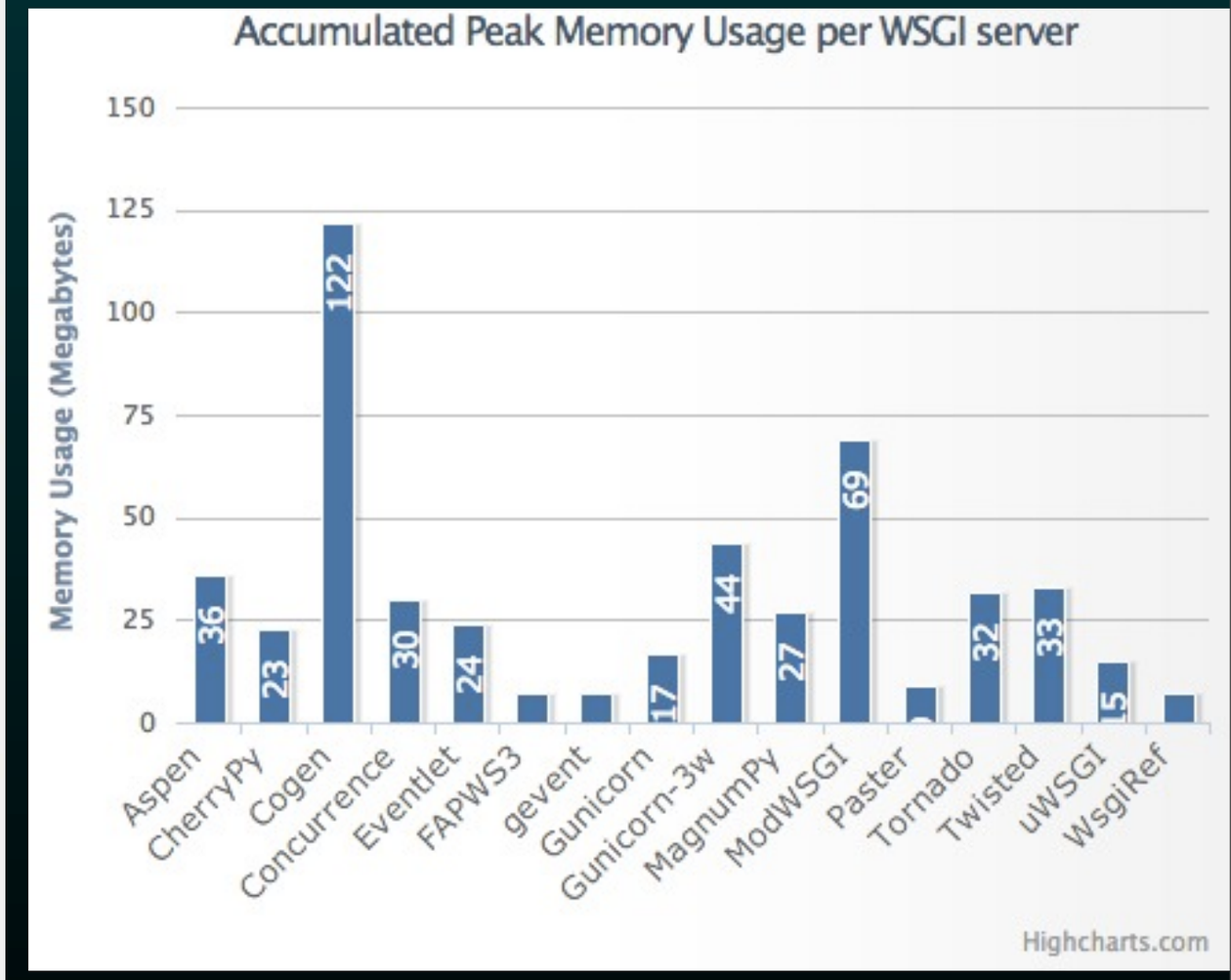
- Expands upon greenlet to provide “green threads”
- Provides an event loop (libev with 1.0, libevent on current pre-1.0)

Green threads

- Similar programming style to POSIX threads
- POSIX threads are pre-emptive
- Green threads are cooperative
- Many green threads can exist within a single POSIX thread
- Green threads are very lightweight

Crazy Fast!





http://nichol.as/benchmark-of-python-web-servers

gevent.server

```
>>> from gevent.server import StreamServer
>>> def handle(socket, address):
...     socket.send('Your address is %s\n' % address[0])
...
>>> server = StreamServer(('127.0.0.1', 1234), handle)
>>> server.serve_forever()
```

```
$ nc localhost 1234
Your address is 127.0.0.1
```

gevent.socket

- Cooperative socket implementation
- When used with monkey patching, brings asynchronous network I/O to an abundance of third-party libraries:
  - memcached
  - redis-py
  - boto
  - requests

Wait, monkey patching?

- Yes!
- Uncooperative libraries are “patched” to cooperate with event loop
- Creates a large ecosystem of gevent-compatible libraries

gevent.monkey

- from gevent.monkey import patch_all; patch_all()
- Patches standard library:
  - socket
  - ssl
  - os
  - time
  - select
  - thread/threading

Deploying

- Use Gunicorn!
- In gunicorn.conf:
  - worker_class = “gevent”
- That’s it! (well, almost)

Websockets

- gevent-websocket
  - worker_class = geventwebsocket.gunicorn.workers.GeventWebSocketWorker
- gevent-socketio
  - worker_class = socketio.sgunicorn.GeventSocketIOWorker

Green database library

```
# gunicorn.conf


# Postgres
def post_fork(server, worker):
    from psycogreen.gevent.psyco_gevent import patch_psycopg
    patch_psycopg()


# MySQL
def post_fork(server, worker):
    import pymysql
    pymysql.install_as_MySQLdb()
```

Scaling

- Messaging (backend)
  - ZeroMQ: zmq.green
  - Redis (monkey-patch compatible)
- Load balancing (frontend)
  - HAProxy
  - Varnish
  - Nginx 1.3+

Further Resources

- gevent For the Working Python Developer
  - http://sdiehl.github.com/gevent-tutorial/
- A Curious on Coroutines and Concurrency - Beazley
  - www.dabeaz.com/coroutines/Coroutines.pdf
- Concurrency is not Parallelism - Rob Pike
  - http://vimeo.com/49718712

Thanks! Questions?

- codysoyland.com
- cody@soyland.com
- twitter.com/codysoyland
- github.com/codysoyland