ASYNC WEB SERVERS

WHY DO YOU NEED THEM?

Andrew Svetlov

http://asvetlov.blogspot.com

andrew.svetlov@gmail.com

http://asvetlov.github.io/aiohttp-krasnoyarsk-2016/

BIO

- Use Python for more than 16 years
- Python Core Developer since 2012
- *asyncio* committer
- *aiohttp* maintainer
- Author of a dozen libraries under *aio-libs* umbrella

WHY?

- It's cool!!!
- I'm an author
- Websockets out-of-the-box
- ...

TO SAVE MEMORY!

WSGI SETUP

- 500 Mb process
- 100 ms response time
- 70% serving requests
- 7 RPS
- 5% CPU usage

AIOHTTP

- 500 Mb process
- 103-105 ms response time
- 100 RPS
- 70% CPU usage

FAILURE

- Latency: 50 ms → 3-15 sec
- Response: 50 ms → 0.5-1 sec

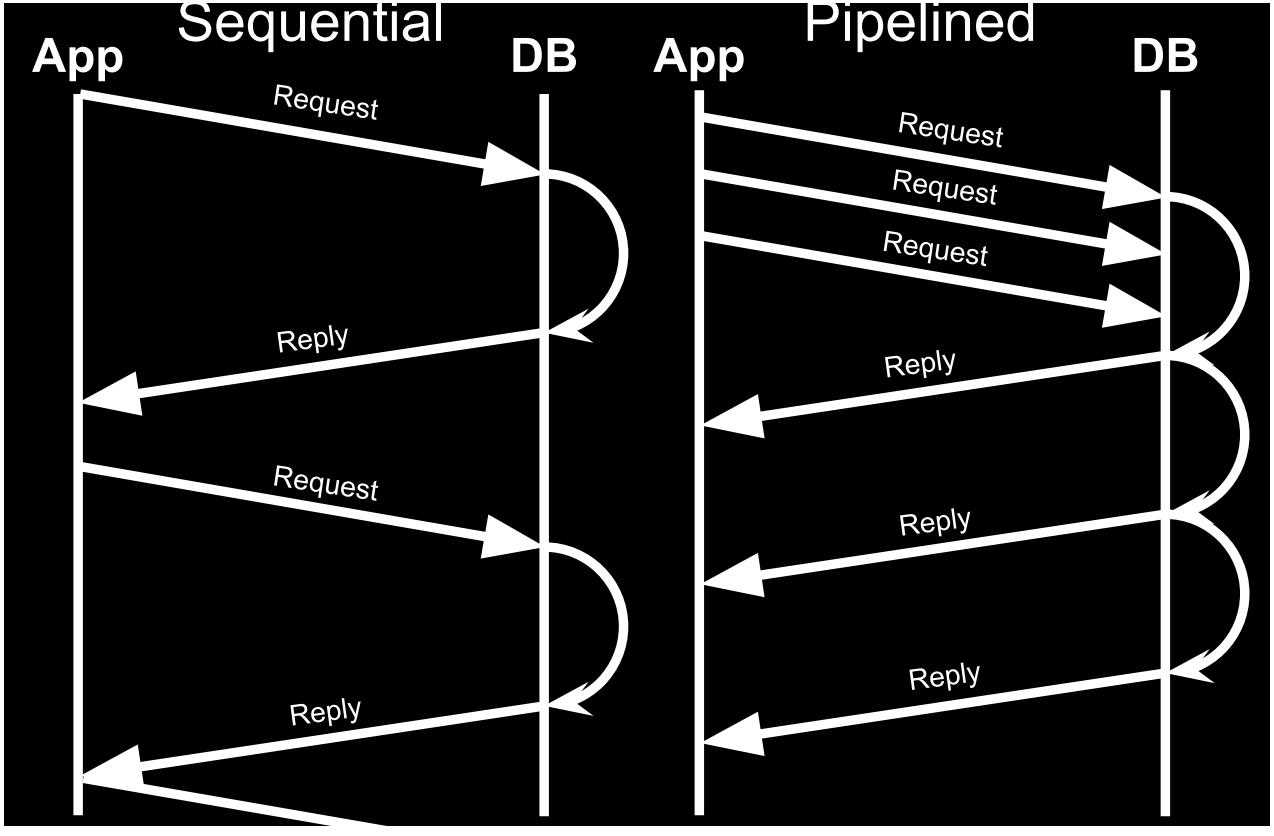
CONCURRENT EXECUTION

NAIVE APPROACH

```
async def handler(request):
    value1 = await get_part1(request)
    value2 = await get_part2(request)
    return render_response(value1, value2)
```

EXPLICIT CONCURRENCY

```
async def handler(request):
    value1, value2 = await asyncio.gather(get_part1(request),
                                          get_part2(request),
                                          loop=request.app.loop)
    return render_response(value1, value2)
```


Sequential
App
DB
Request
Reply
Request
Reply
Pipelined
App
DB
Request
Request
Request
Reply
Reply
Reply

```
async def handler(request):
    value1, value2 = await asyncio.gather(get_part1(request),
                                          get_part2(request),
                                          loop=request.app.loop)
    return render_response(value1, value2)
```

HIDDEN CONCURRENCY

```
async def handler(request):
    value1 = await get_part1(request)
    value2 = await get_part2(request)
    return render_response(value1, value2)
```

HTTP KEEP-ALIVE

SERVER-SIDE CONNECTIONS

- NGIXG
- Backend (aiohttp)

CLIENT CONNECTIONS

```
async def handler(request):
    session = request.app['client_session']
    async with session.get(url) as resp:
        body = yield from resp.json()
    return render_json(body)
```

DB CONNECTION POOLS

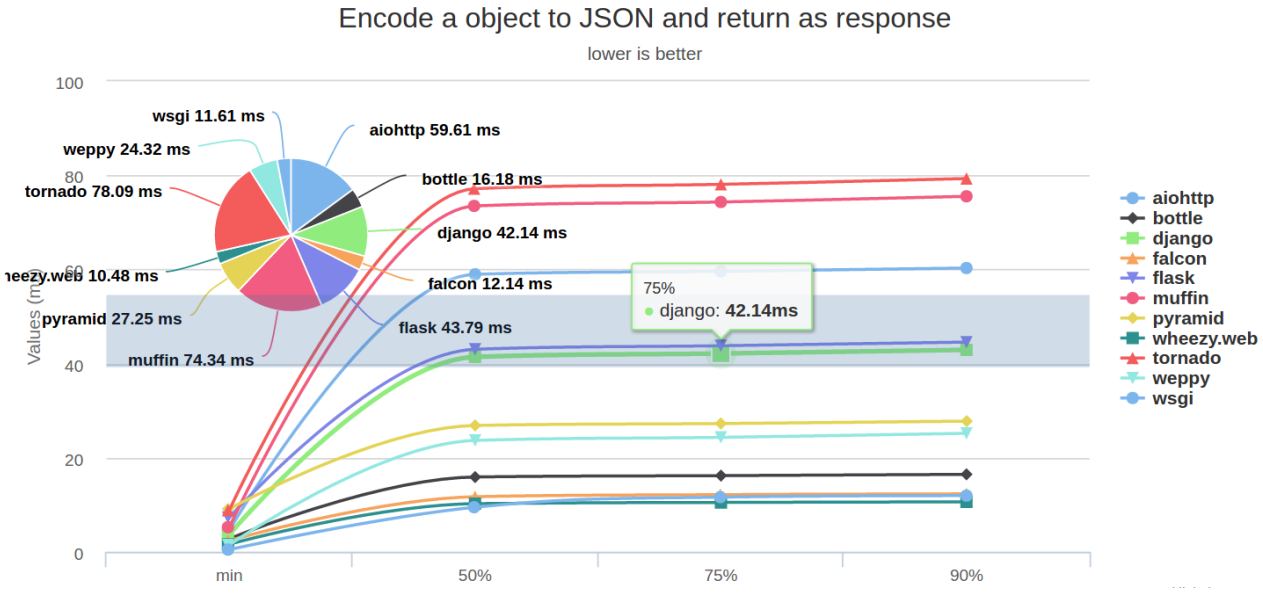
```
async def handler(request):
    async with request.app['db'] as conn:
        await conn.execute('SELECT * FROM ...')
```

TIMEOUTS

```
with asyncio_timeout.timeout(10):
    async with session.get(url) as response:
        assert response.status == 200
        return await response.read()
```

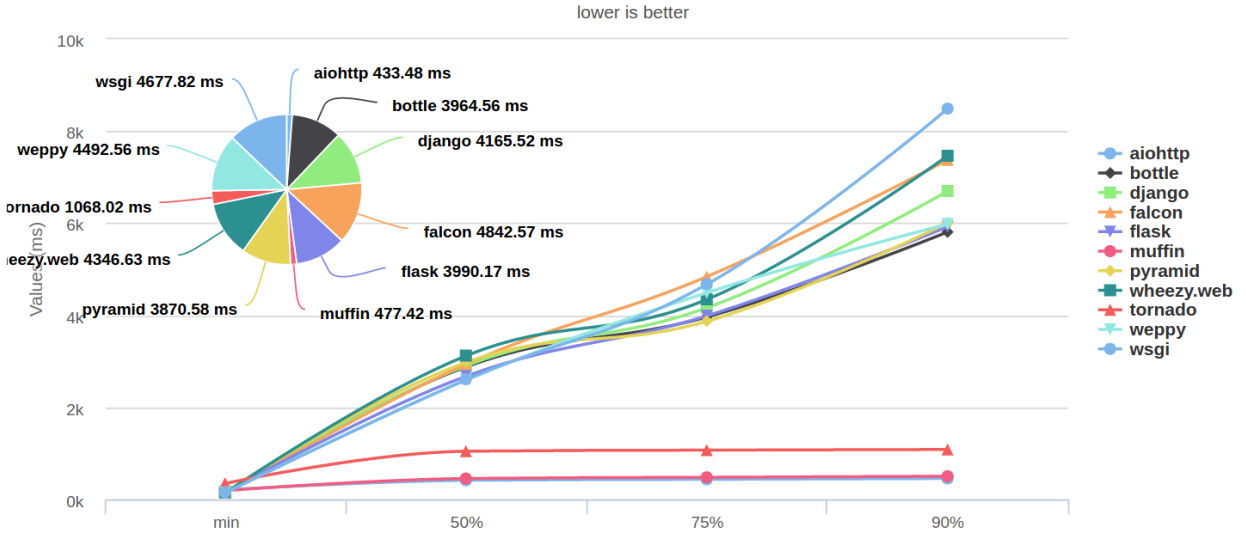
PERFORMANCE

JSON REQUESTS


Encode a object to JSON and return as response
lower is better
Values (ms)
100
80
60
40
20
0
min
50%
75%
90%
wsgi 11.61 ms
aiohttp 59.61 ms
weppy 24.32 ms
bottle 16.18 ms
tornado 78.09 ms
django 42.14 ms
heezy.web 10.48 ms
falcon 12.14 ms
pyramid 27.25 ms
flask 43.79 ms
muffin 74.34 ms
75%
django: 42.14ms
aiohttp
bottle
django
falcon
flask
muffin
pyramid
wheezy.web
tornado
weppy
wsgi

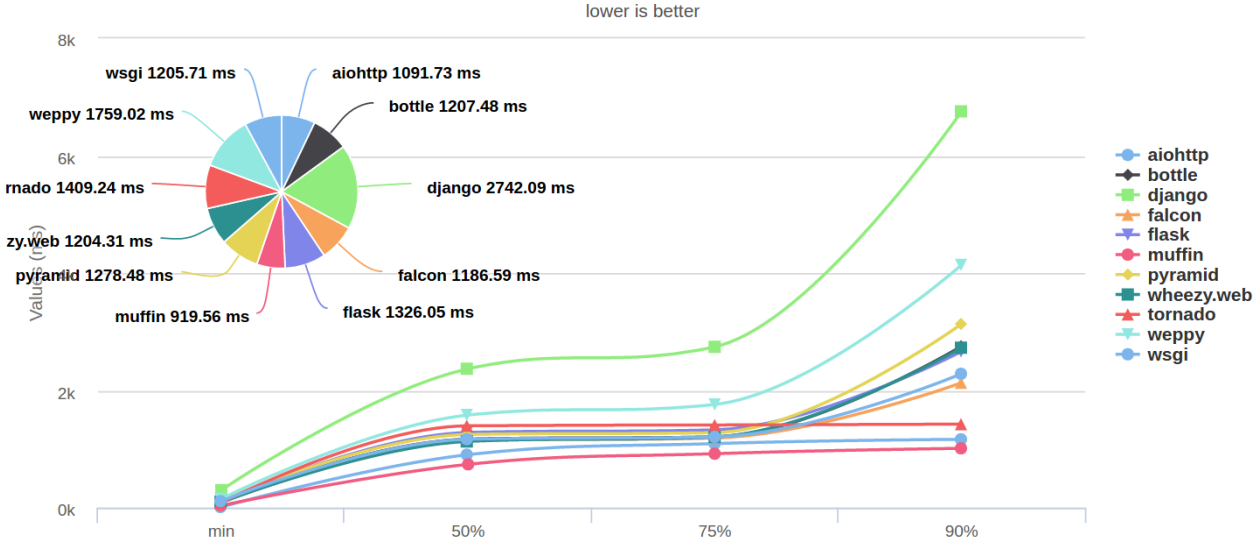
EXTERNAL RESOURCE

Think about github.com, facebook.com or google.com


lower is better
10k
8k
6k
4k
2k
0k
Values (ms)
wsgi 4677.82 ms
aiohttp 433.48 ms
bottle 3964.56 ms
weppy 4492.56 ms
django 4165.52 ms
ornado 1068.02 ms
falcon 4842.57 ms
eezy.web 4346.63 ms
flask 3990.17 ms
pyramid 3870.58 ms
muffin 477.42 ms
min
50%
75%
90%
aiohttp
bottle
django
falcon
flask
muffin
pyramid
wheezy.web
tornado
weppy
wsgi

DATABASE

Hosted in sibling docker contatiner


lower is better
8k
6k
2k
0k
wsgi 1205.71 ms
aiohttp 1091.73 ms
weppy 1759.02 ms
bottle 1207.48 ms
rnado 1409.24 ms
django 2742.09 ms
zy.web 1204.31 ms
pyramid 1278.48 ms
falcon 1186.59 ms
muffin 919.56 ms
flask 1326.05 ms
min
50%
75%
90%
aiohttp
bottle
django
falcon
flask
muffin
pyramid
wheezy.web
tornado
weppy
wsgi

QUESTIONS?

Andrew Svetlov

http://asvetlov.blogspot.com

andrew.svetlov@gmail.com

http://asvetlov.github.io/aiohttp-krasnoyarsk-2016/