ASYNC WEB SERVERS

WHY DO YOU NEED THEM?

Andrew Svetlov

http://asvetlov.blogspot.com

andrew.svetlov@gmail.com

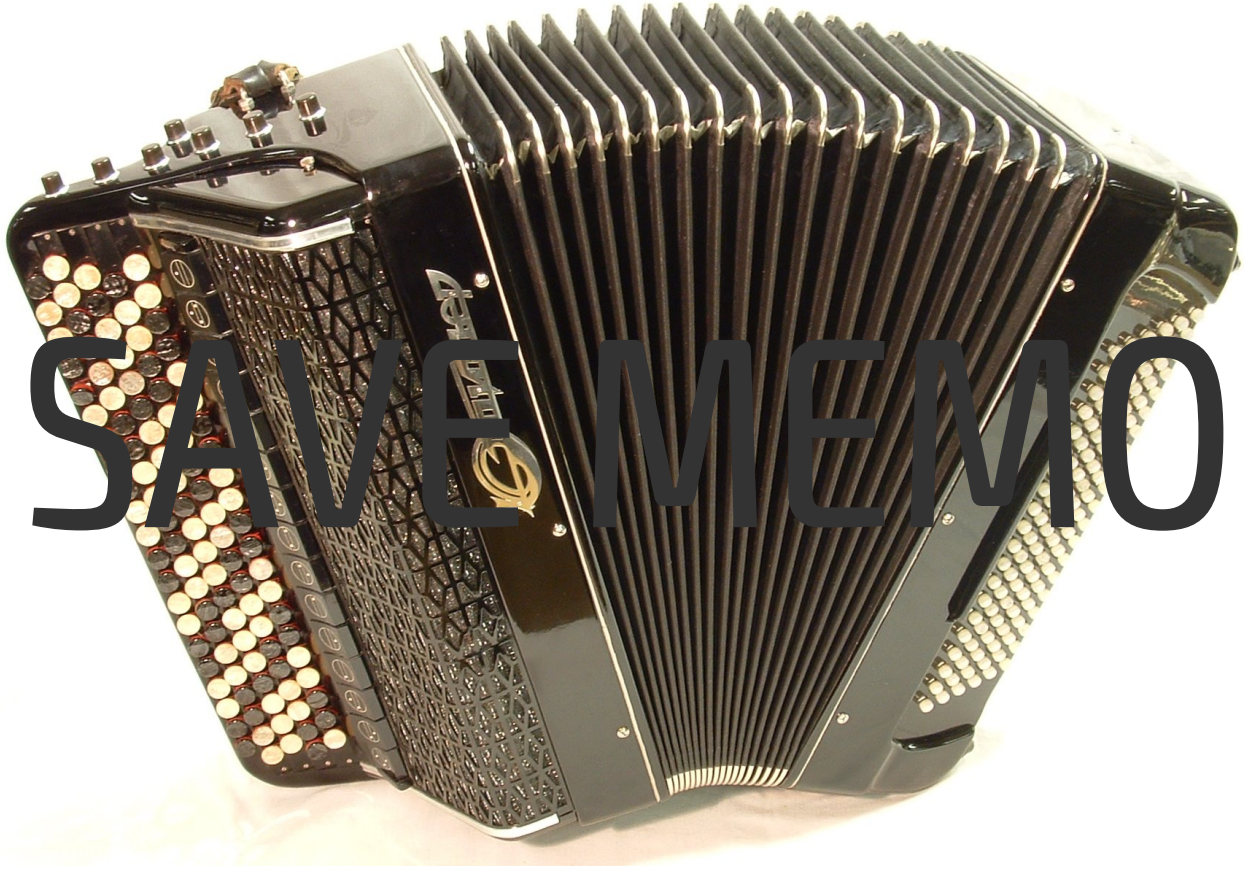
http://asvetlov.github.io/aiohttp-krasnoyarsk-2016/

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

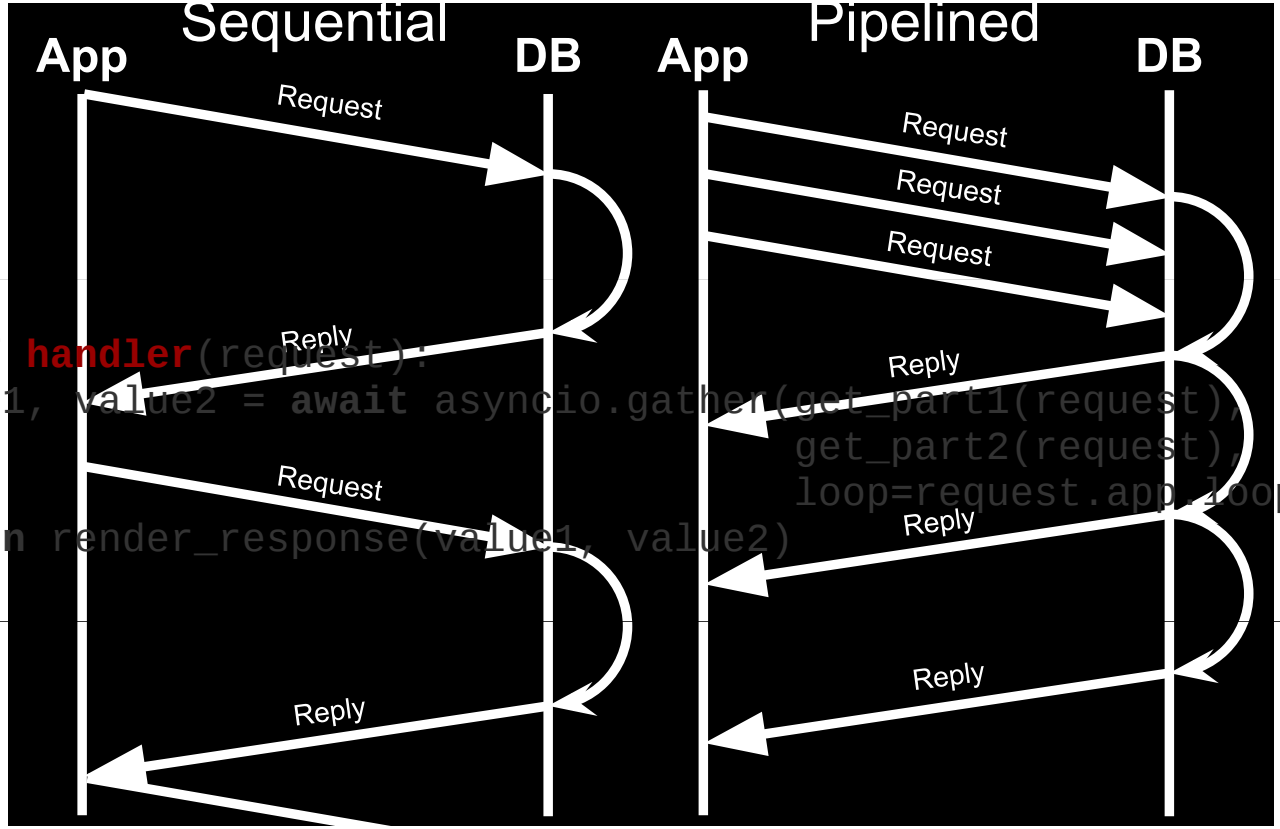
CONCURRENT EXECUTION

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)
```




```
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

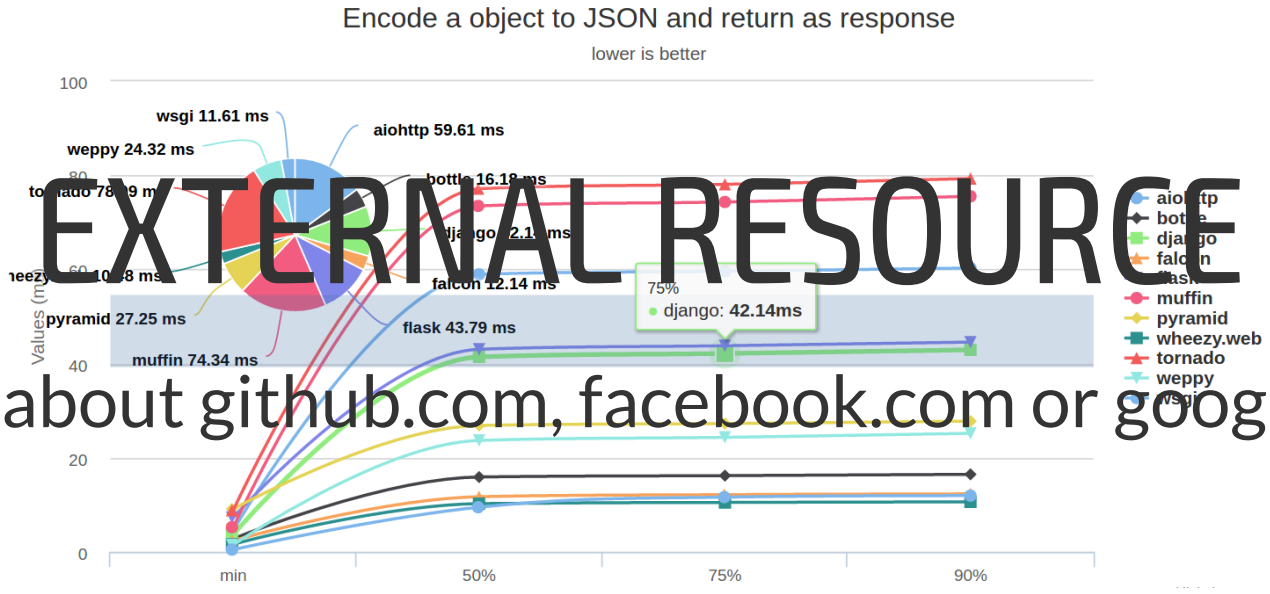
```
async def handler(request):
    async with request.app['db'] as conn:
        await conn.execute('SELECT * FROM ...')
```

TIMEOUTS

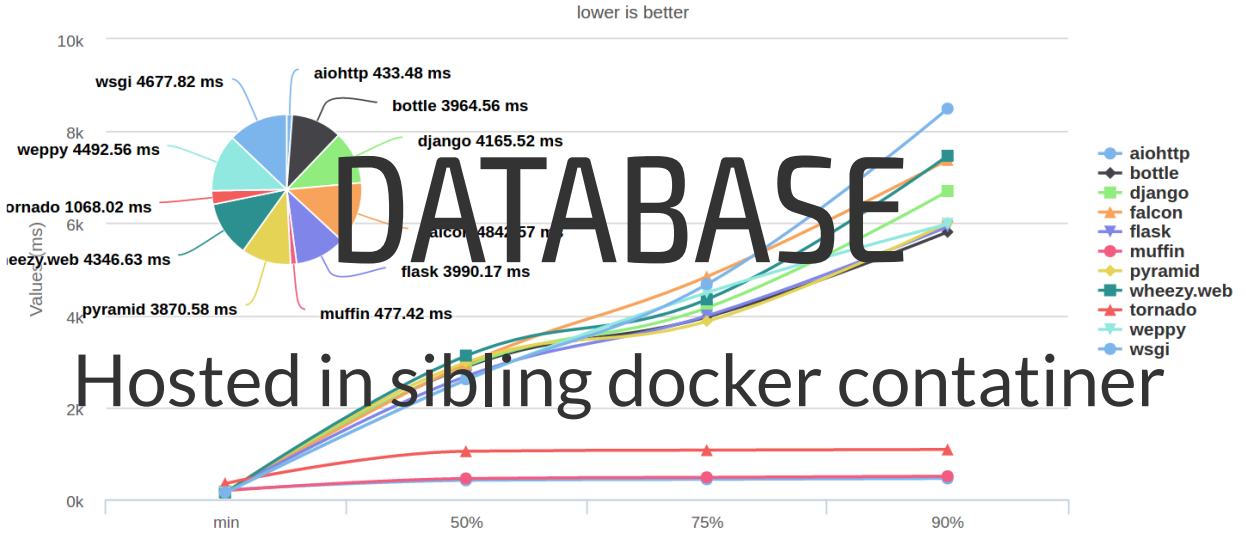
```
with asyncio_timeout.timeout(10):
    async with session.get(url) as response:
        assert response.status == 200
        return await response.read()
```

PERFORMANCE

JSON REQUESTS



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
django 4165.52 ms
weppy 4492.56 ms
ornado 1068.02 ms
eezy.web 4346.63 ms
flask 3990.17 ms
pyramid 3870.58 ms
muffin 477.42 ms
DATABASE
Hosted in sibling docker contatiner
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/