



Asyncio in production

Hrafn Eiriksson



quests with python

python

3x faster Flask apps

Quart as a upgrade to Flask

Making 1 aiohttp

Apr 22, 2016 - by Pawel

Python has evolved since Flask was



Async/await in Python 3.5 and why it is awesome

Python 3's Killer Feature: asyncio

Jun 21, 2017 | By Michael Flaxman, Principal Engineer



12 FEBRUARY 2018 / PYTHON

Scaling a polling Python application with async

Why you no *asyncio*?!



Why you no *asyncio*?!

Asynchronous programming is different



Why you no *asyncio*?!

asyncio is relatively new



Why you no *asyncio*?!

Converting existing Python apps to use asyncio is not simple



Why you no *asyncio*?!

The community has built multiple concurrency libraries



Why you no *asyncio*?!

Asynchronous programming is not always what you want



My goal today

1. Discuss why I went all in on asyncio (and try to convince you to do the same)
2. Migrating to asyncio and the inevitable issues you run into
3. Asyncio in production: A before/after comparison

Not my goal today

- ~~4. An introduction into asyncio~~



Part 1: Why bother with *asyncio*?

My software already works!



A bit of background...



A bit of background...

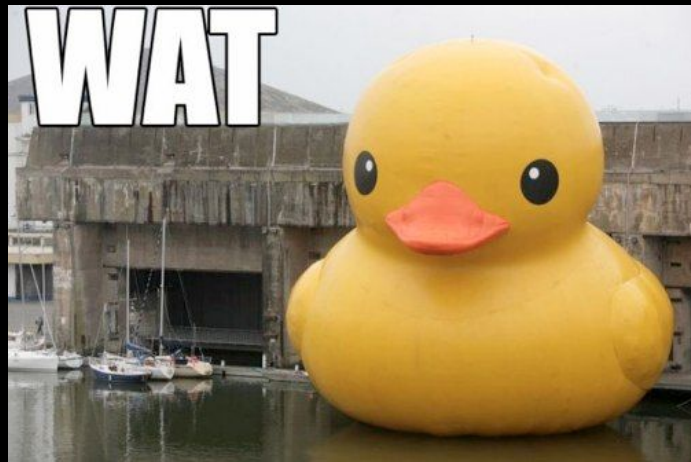
```
krummi@krummi:~$ node

> [] + []
''

> [] + {}
'[object Object]'

> {} + []
0

> {} + {}
'[object Object][object Object]'
```



Credit: <https://www.destroyallsoftware.com/talks/wat>

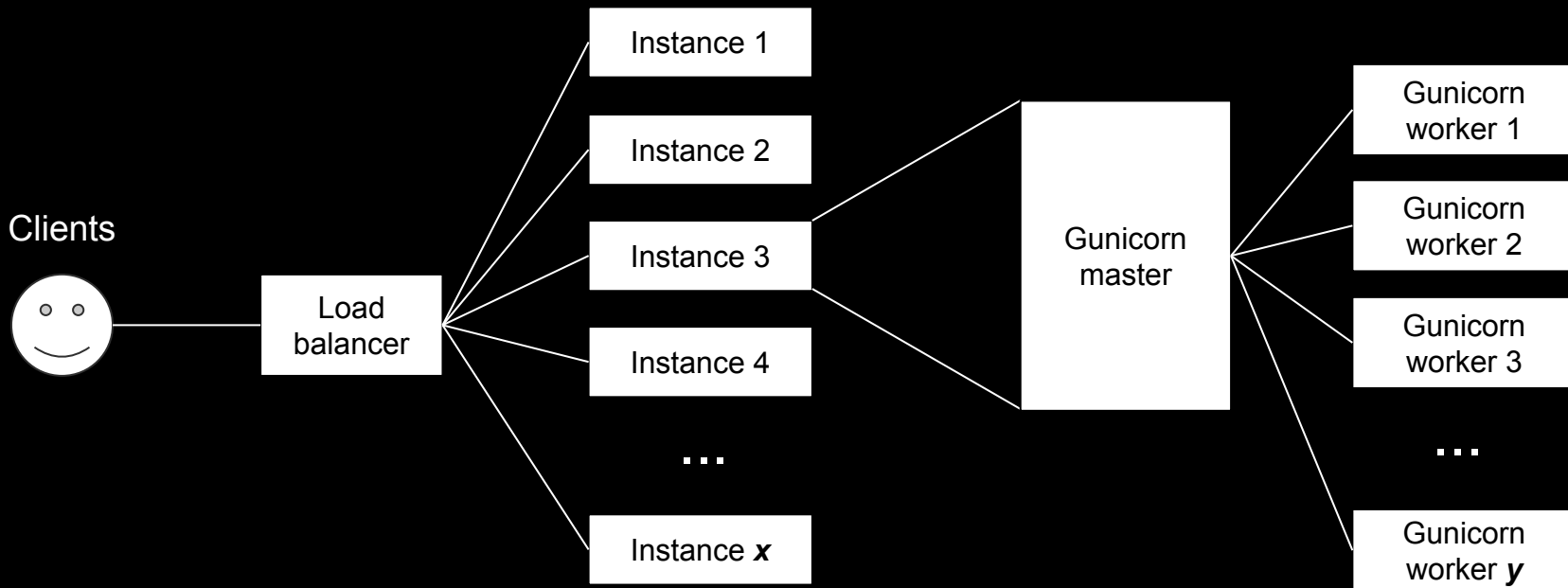


A bit of background...

\$markets



How we (typically) scale our services





```
import eventlet
```

```
eventlet.monkey_patch( )
```



```
import eventlet  
  
eventlet.monkey_patch( )
```



TLDR

What makes asyncio so attractive is that it's:

- Explicit
- Part of the language



Part 2: Migrating to *asyncio*



The asyncio ecosystem

Previously people relied on monkey patching

Now it seems to be becoming quite mature:

- Dozens of web frameworks (aiohttp, Sanic, Quart)
- Loads of database drivers (asyncpg, aiomysql, aioredis, etc)
- ...and way more [1]

[1] <https://github.com/python/asyncio/wiki/ThirdParty>

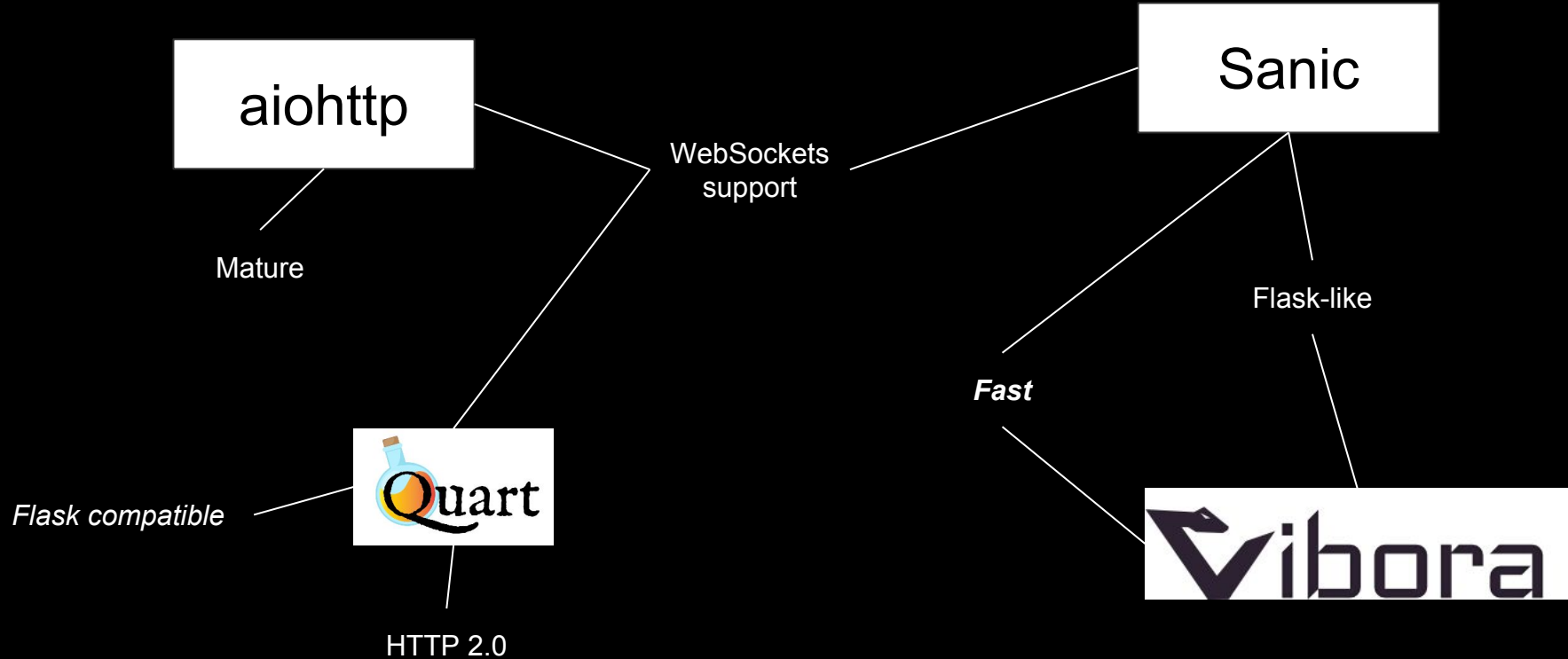


A microservice migration

Based on a true story



Asyncio web frameworks



An example: Quart

Flask



```
from flask import Flask
app = Flask(__name__)

@app.route('/')
def hello_world():
    return 'Hello, World!'
```

Quart [2]



```
from quart import Quart
app = Quart(__name__)

@app.route('/')
async def hello_world():
    return 'Hello, World!'
```

Migrating from Flask

It should be possible to migrate to Quart from Flask by a find and replace of `flask` to `quart` and then adding `async` and `await` keywords. See the [docs](#) for full details.



aiohttp



```
from aiohttp import web

async def handle(request):
    name = request.match_info.get('name', "Anonymous")
    text = "Hello, " + name
    return web.Response(text=text)

app = web.Application()
app.add_routes([web.get('/', handle),
                 web.get('/{name}', handle)])

web.run_app(app)
```



A migration example: Sentry

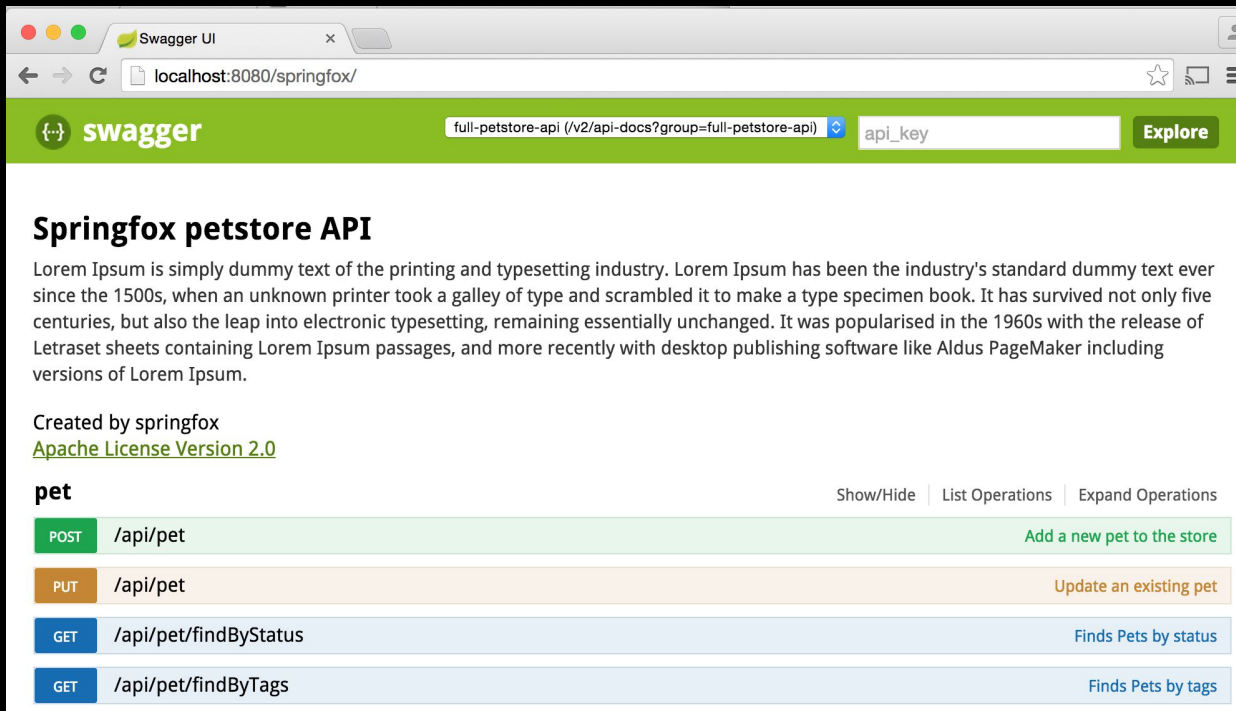
```
from raven import Client as SentryClient
from aiohttp import web
from raven_aiohttp import AioHttpTransport

@web.middleware
async def sentry_middleware(request, handler):
    try:
        return await handler(request)
    except Exception:
        request.app.sentry_client.captureException()
        raise

app = web.Application(middlewares=[sentry_middleware])
app.sentry_client = SentryClient(
    sentry_dsn,
    transport=AioHttpTransport,
)
```



Then problems hit...



The screenshot shows the Swagger UI interface in a web browser. The browser's address bar displays 'localhost:8080/springfox/'. The Swagger UI header is green and contains the 'swagger' logo, a dropdown menu for the API ('full-petstore-api (v2/api-docs?group=full-petstore-api)'), an 'api_key' input field, and an 'Explore' button.

Springfox petstore API

Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book. It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentially unchanged. It was popularised in the 1960s with the release of Letraset sheets containing Lorem Ipsum passages, and more recently with desktop publishing software like Aldus PageMaker including versions of Lorem Ipsum.

Created by springfox
[Apache License Version 2.0](#)

pet Show/Hide List Operations Expand Operations

POST	/api/pet	Add a new pet to the store
PUT	/api/pet	Update an existing pet
GET	/api/pet/findByStatus	Finds Pets by status
GET	/api/pet/findByTags	Finds Pets by tags



Issues with asyncio

- A lot of new concepts to wrap your head around
- `async/await` everywhere
- Debugging asyncio code can be problematic
- Be wary of running synchronous code in async functions



TLDR

1. Map out your dependencies to see if asyncio-compatible versions exist
2. Experiment with asyncio versions of your dependencies
3. Watch out for asyncio gotchas
4. Profit!



Part 3: *asyncio* in production



Before/after comparison

Before: Flask + psycpg2 + eventlet

VS

After: aiohttp + asyncpg + uvloop

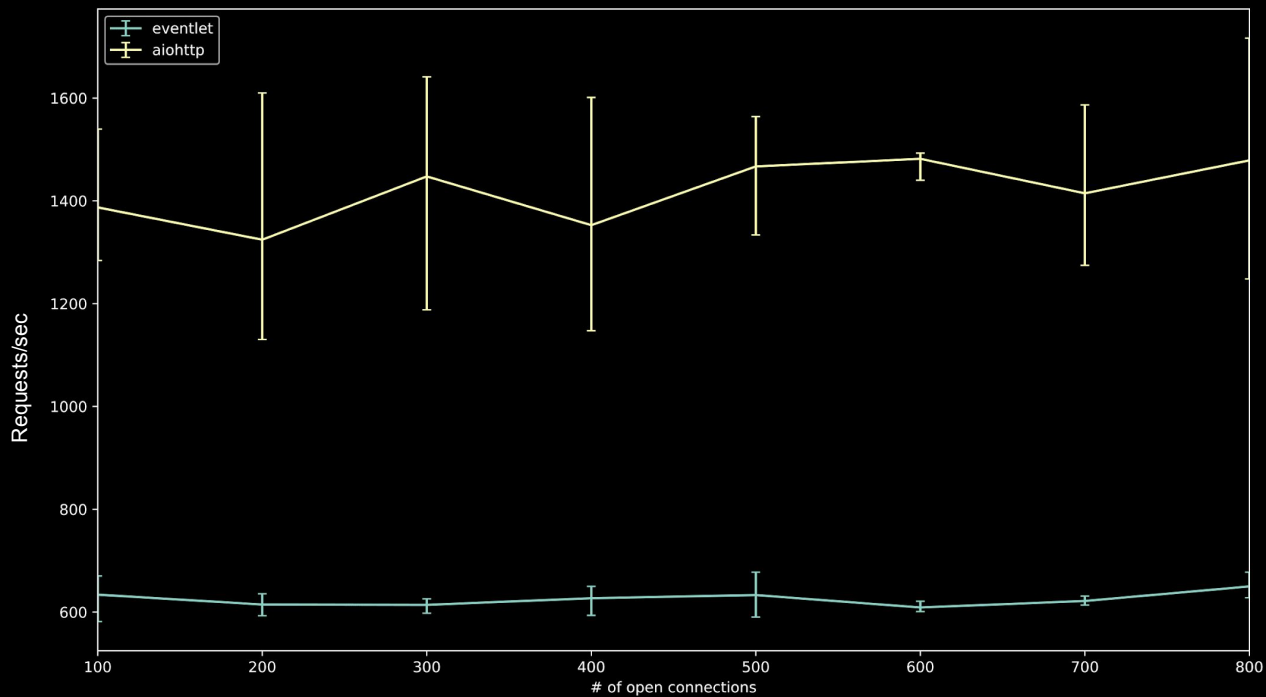


Methodology

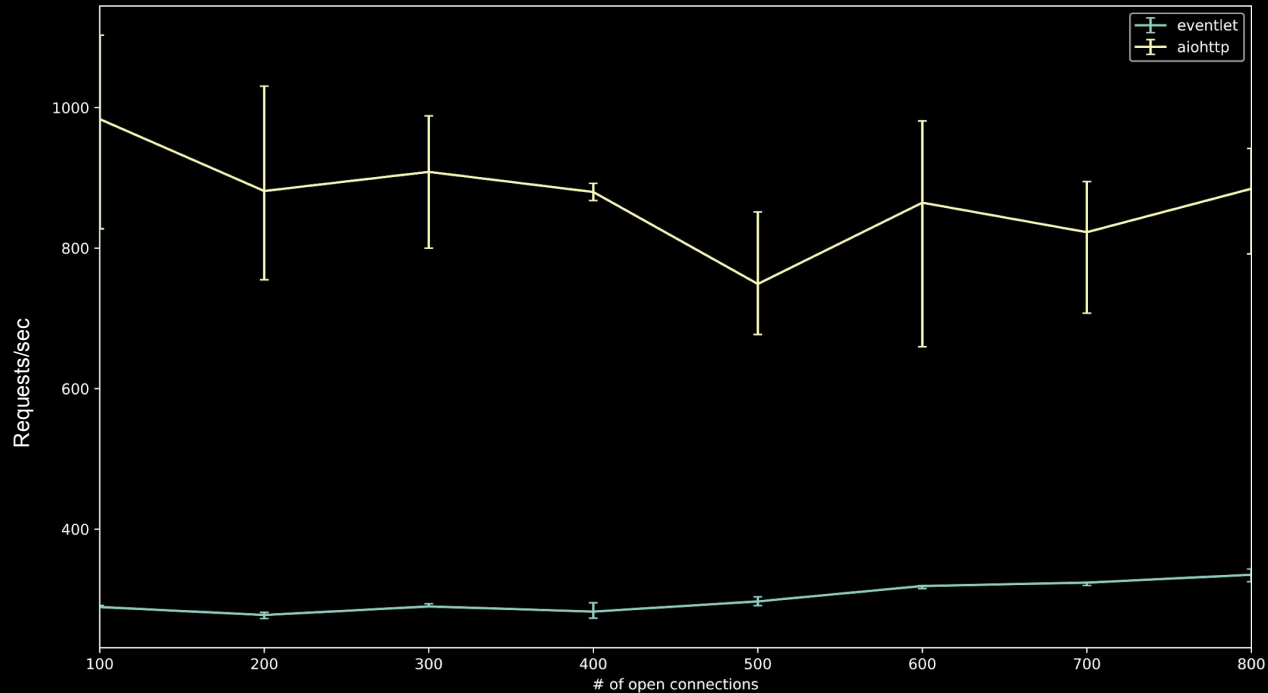
- Use wrk (<https://github.com/wg/wrk>) to do HTTP benchmarking.
- Ran each configuration of the benchmark:
 - For 30 seconds
 - 10 times
 - Using a variable number of open HTTP connections
 - Noted the median and the 25%/75% latency for each run
 - 10 seconds of sleep between runs



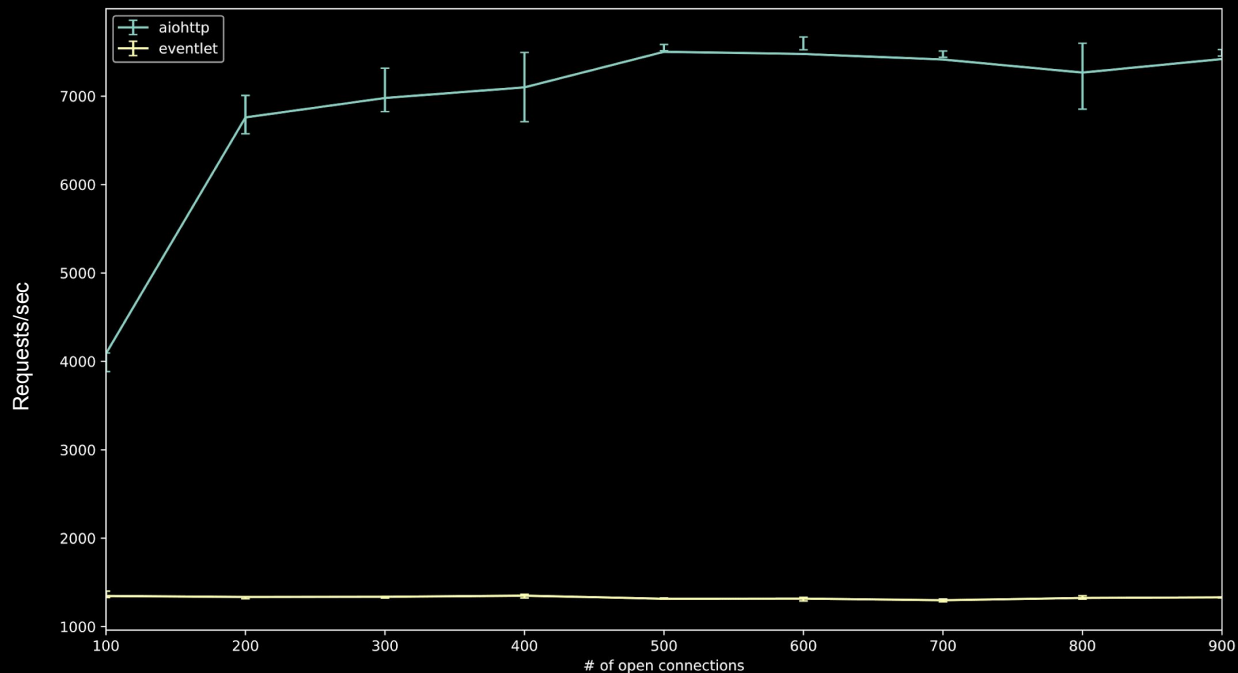
Local comparison: Simple ping



Local comparison: Database access



In production comparison



Conclusion

So is asyncio worth the effort?



Thank you!

