AsyncIO in Production War Stories

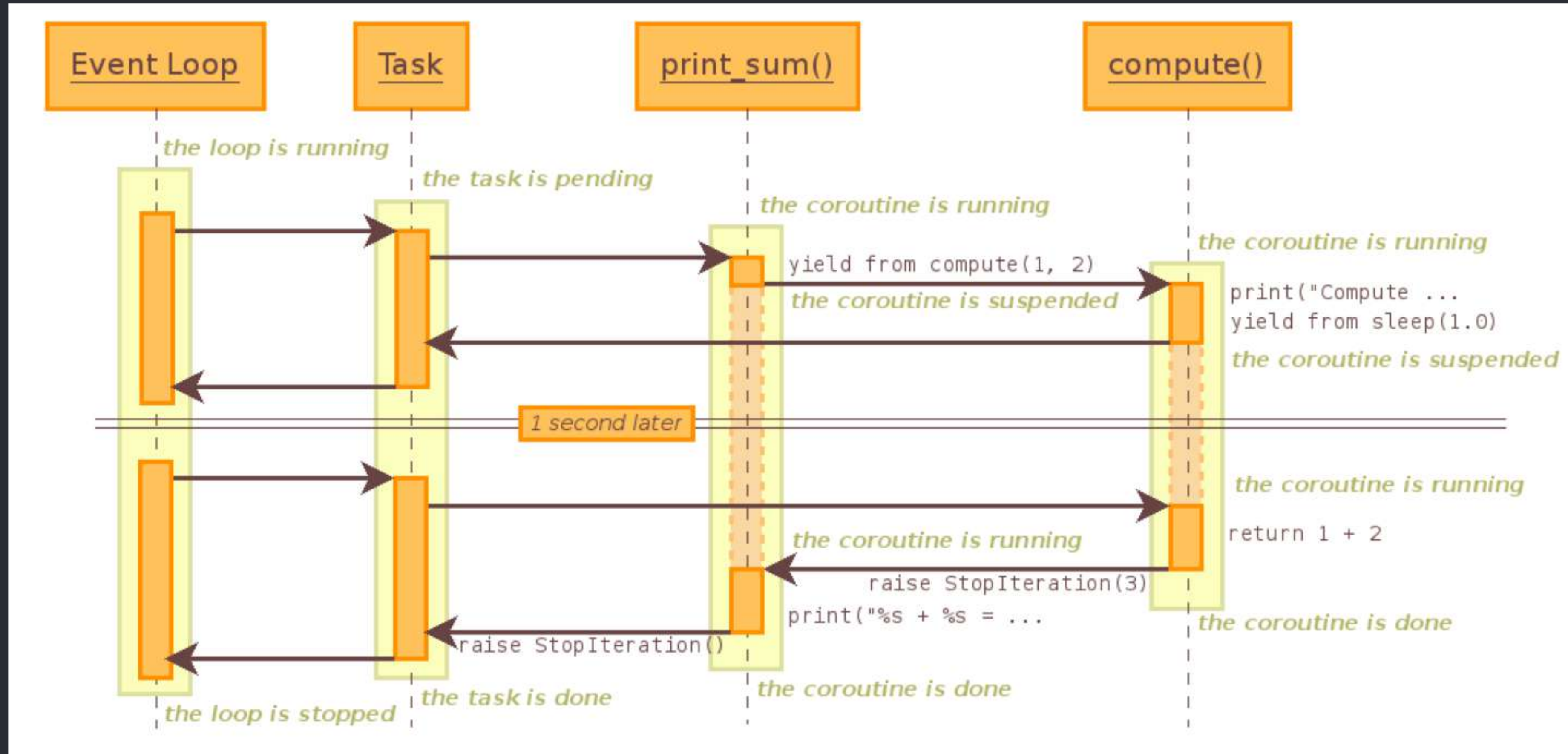
Michał Wysokiński

17.09.2018

Akamai

- is a content delivery network (CDN) and cloud services provider
- has deployed the most pervasive, highly-distributed CDN with more than 250,000 servers in over 130 countries and within more than 4,000 networks around the world, which are responsible for serving between 10 and 30% of all web traffic
- protects against web attacks such as SQL injection, XSS and RFI
- has 16 offices in EMEA (Europe, Middle East and Africa)
- and 1 office in Poland (Kraków)

How does it work



Akamai Experience the Edge

A bit of history

- September 22nd 2012
  asyncore: included batteries don't fit – submitted to Python ideas
- December 12th 2012
  PEP 3156
- March 16th 2014 [3.4] (provisional)
  AsyncIO module release with Python 3.4 (as provisional API)
  @asyncio.coroutine/yield from (introduced in 3.3)
- September 13th 2015 [3.5]
  async/await keywords introduced with Python 3.5
  asynchronous iteration, asynchronous context managers
- December 23rd 2016 [3.6] (stable)
  asynchronous generators, asynchronous comprehensions

Akamai Experience the Edge

How asyncio code looks like

```
for exec_uuid in tasks:
  task_data = await self.task_queries.get_data(exec_uuid)
  asyncio.ensure_future(self._run_task(exec_uuid))


exec_statuses = await asyncio.gather(
  *[self._run_task(ip, task_data) for ip in ips]
)
```


Akamai Experience the Edge

How asyncio code looks like – with timeouts

```
await asyncio.wait_for(
    ioloop.run_in_executor(
        self.processing_pool, lambda: plugin.run(
            ip, **exec_data.args
        )
    ),
    timeout=getattr(plugin, 'timeout', self.default_plugin_timeout)
)

async with timeout(1.5):  # async-timeout
    await inner()
```


Akamai Experience the Edge

Reasons for its existence

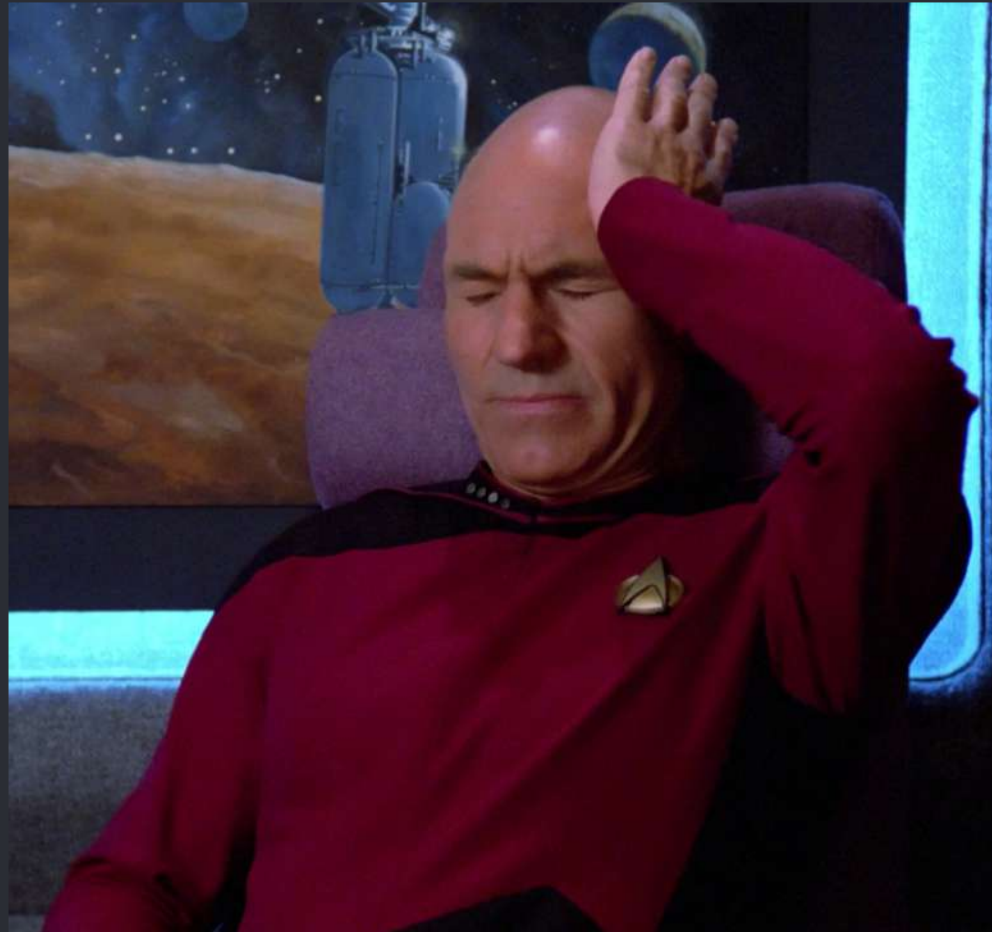
- It’s useful for handling independent tasks (similar to threads*)
- Everyone started doing it in their own way
- Modifying CPython was giving hope of better performance

Akamai Experience the Edge

A story of mixing AsyncIO and threads

Akamai Experience the Edge

We started with a bit of a…



Akamai Experience the Edge

Yyy, what just happened?

```
Task was destroyed but it is pending!
task: <Task pending create() done at run.py:5
wait_for=<Future pending cb=[Task._wakeup()]>>
```

```
ERROR:asyncio:Task exception future:
<Task finished coro=<SSHConnection._run_task() done,
exception=CancelledError()>
concurrent.futures._base.CancelledError
was never retrieved
```


Akamai Experience the Edge

Dependencies nightmare

- Tornado (ioloop is a wrapper for asyncio loop)
- Momoko (async wrapper for psycopg2)
- uvloop (wrapper for libuv, replacement for asyncio loop)
- async_test (to get rid of the standard library low level testing code)

Akamai Experience the Edge

Dependencies nightmare

- Tornado (ioloop is a wrapper for asyncio loop)
- Momoko (async wrapper for psycopg2) – PRETTY DEAD
- uvloop (wrapper for libuv, replacement for asyncio loop) basically rewrites asyncio loop which sometimes causes unexpected results
- async_test (to get rid of the standard library low level testing code) – single developer, not super stable (resource allocation), not compatible with uvloop

Akamai Experience the Edge

@asyncio.coroutine/yield from* -> async/await

Akamai Experience the Edge

@tornado.gen.coroutine/yield -> async/await

Akamai Experience the Edge

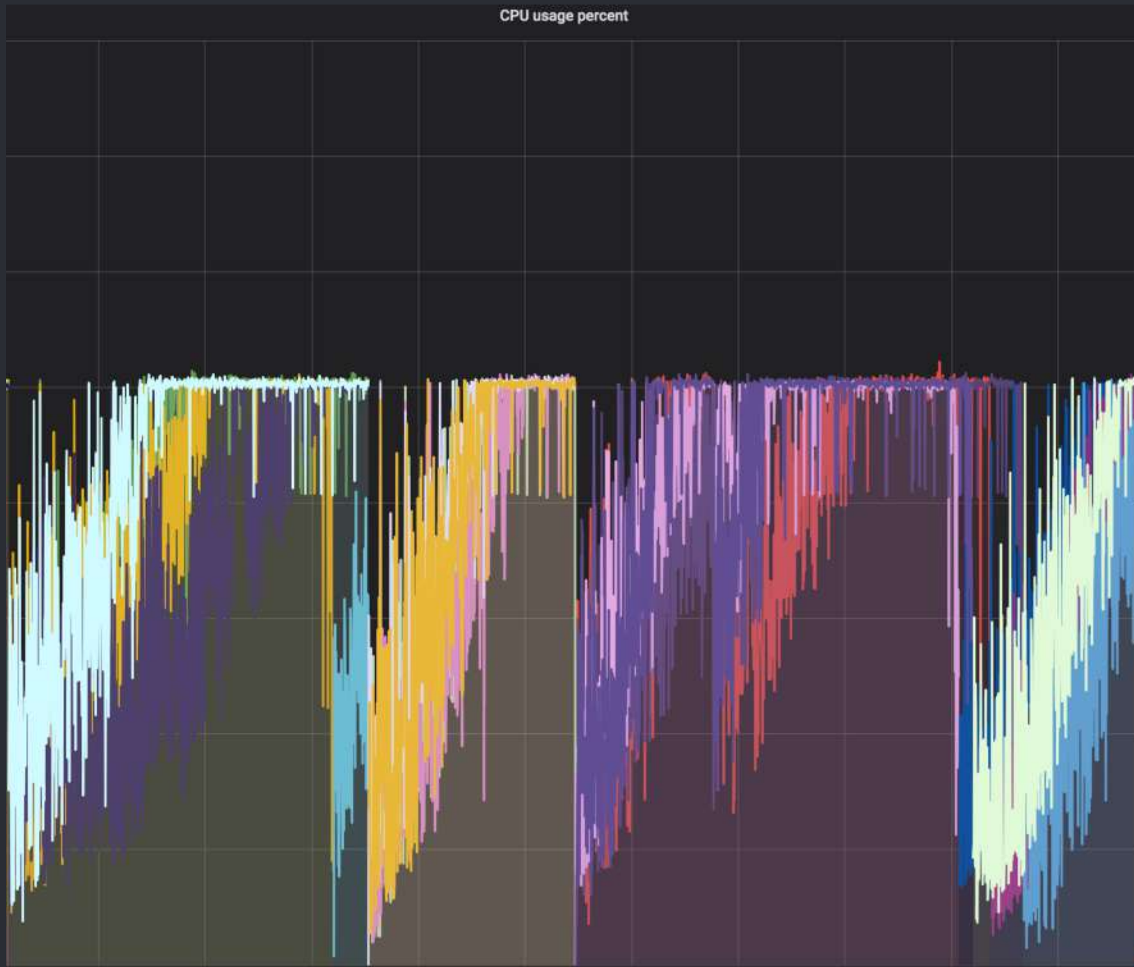
A story of an asynchronous http client

Akamai Experience the Edge

The good, the bad and the ugly

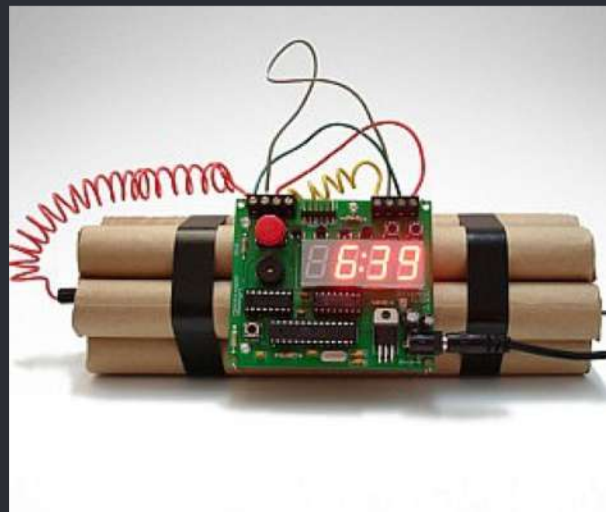
```
class A:
    def get_data(self):
        with ###.TCPConn(ssl_context, limit=1) as tcp_conn:
            async with ###.HttpClient(tcp_conn) as http_client:
                async with http_client.get(self.streamer_url.encoded) as response:
                    content_iterator = response.content.__aiter__()  # no aiter()
                    while not self.stop_streaming:
                        async with async_timeout.timeout(60):
                            try:
                                data = await content_iterator.__anext__()  # no anext()
                            except StopAsyncIteration:
                                logger.debug('Stop aiteration for stream: %s', str(self))
                                break
```


Akamai *Experience the Edge*



Akamai Experience the Edge

A story of an ElasticSearch client



Akamai Experience the Edge

Fixing async bomb

```
import aiojobs

scheduler = await aiojobs.create_scheduler(
    limit=MAX_TASKS_SIZE,
    pending_limit=WAITING_DOCS_BUFFER_SIZE,
    close_timeout=5
)
await scheduler.spawn(
    es_index_wrapper(document, es_client)
)
```


Akamai Experience the Edge

A story of a really simple microservice that gets one and only one job done

```
ips = await dns_resolver.query(edns[domain_info.name], 'A')
# with this line we reduced running time from 12 hours to 8 minutes
```


Akamai Experience the Edge

A story of a group of services communicating only with messages

Akamai Experience the Edge

PROS

- Performance gain for applications relying on IO (network, DB)
- Better resource utilization (less time spent on communication and synchronization)
- Being on the technological edge gives you new ways to solve old problems
- It makes you follow Python progress and contribute

CONS

- Still many missing features or know issues
  - async iterators have messy indeterministic cleanups
  - itertools for async is missing
  - very early implementations or complete lack of modules for interacting with popular services (zookeeper, ElasticSearch, requests, and more)
- Still young implementation with many bugs and incompatibilities
- The community becomes more and more divided

Akamai Experience the Edge

What projects are best suited for AsyncIO (IMHO)

- MICROservices
- Projects with a small list of dependencies
- Simple http APIs
- Projects with big load but light processing
- Projects where threads are not enough
- Projects where the rest of technology stack is well understood

Akamai Experience the Edge

What projects are not suited for AsyncIO (IMHO)

- Projects heavily relying on threads
- Projects with dependencies heavily using threads, unless you know their implementation really well
- Projects where processing of a single task takes a lot of time minutes/hours
- Projects doing uncommon stuff

Akamai Experience the Edge

Q&A

Akamai Experience the Edge

michal.a.wysokinski@gmail.com

Akamai Experience the Edge