



Demystifying Python's `async` and `await` Keywords

Michael Kennedy

Founder, Talk Python Training

@mkennedy

Take the code with you

The screenshot shows a GitHub repository page for 'mikekennedy/async-await-jetbrains-webcast'. The repository has 2 commits, 1 branch, 0 releases, and 1 contributor. The latest commit was made 18 seconds ago. The repository description is: "Webinar: 'Demystifying Python's async and await Keywords' with Michael Kennedy".

mikekennedy / **async-await-jetbrains-webcast**

No description, website, or topics provided.

2 commits 1 branch 0 releases 1 contributor

Branch: master New pull request Create new file Upload files Find file Clone or download

mikekennedy A few bits of cosmetics. 18 seconds ago

readme_resources A few bits of cosmetics. a minute ago

readme.md A few bits of cosmetics. 18 seconds ago

readme.md

Webinar: "Demystifying Python's async and await Keywords" with Michael Kennedy

<https://github.com/mikekennedy/async-await-jetbrains-webcast>

What is asynchronous programming?

Asynchrony, in computer programming, refers to the occurrence of events independent of the main program flow and ways to deal with such events.

These may be "outside" events such as the arrival of signals, or actions instigated by a program that take place concurrently with program execution, without the program blocking to wait for results.

-- Wikipedia

[https://en.wikipedia.org/wiki/Asynchrony_\(computer_programming\)](https://en.wikipedia.org/wiki/Asynchrony_(computer_programming))



async for speed

CPUs are multi-core

End of Moore's Law? <https://www.slideshare.net/Funk98/end-of-moores-law-or-a-change-to-something-else>

4 people clipped this slide

Power and Heat Problems Led to Multiple Cores and Prevent Further Improvements in Speed

Original data collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond and C. Batten
Dotted line extrapolations by C. Moore
Source: Chuck Moore, Data Processing in Exascale-Class Systems; April 27, 2011. Salishan Conference on High-Speed Computing

Transistors (thousands)

Single-thread Performance (SpecINT)

Frequency (MHz)

Typical Power (Watts)

Number of Cores

Recommended

- Computing Beyond Moore's Law: Architecture and Device Innovations Fujitsu Global
- Intro to course module: How do new Technologies Become Economically Feasible Jeffrey Funk
- Kyle Galler - Europe 2020, the innovation union and horizon 2020: how it all ... Universities UK
- MUVE your pupils: Best Practices in Multi-User Virtual Environments for Learning Sabine Reljic
- Practical Crypto Attacks Against Web Applications

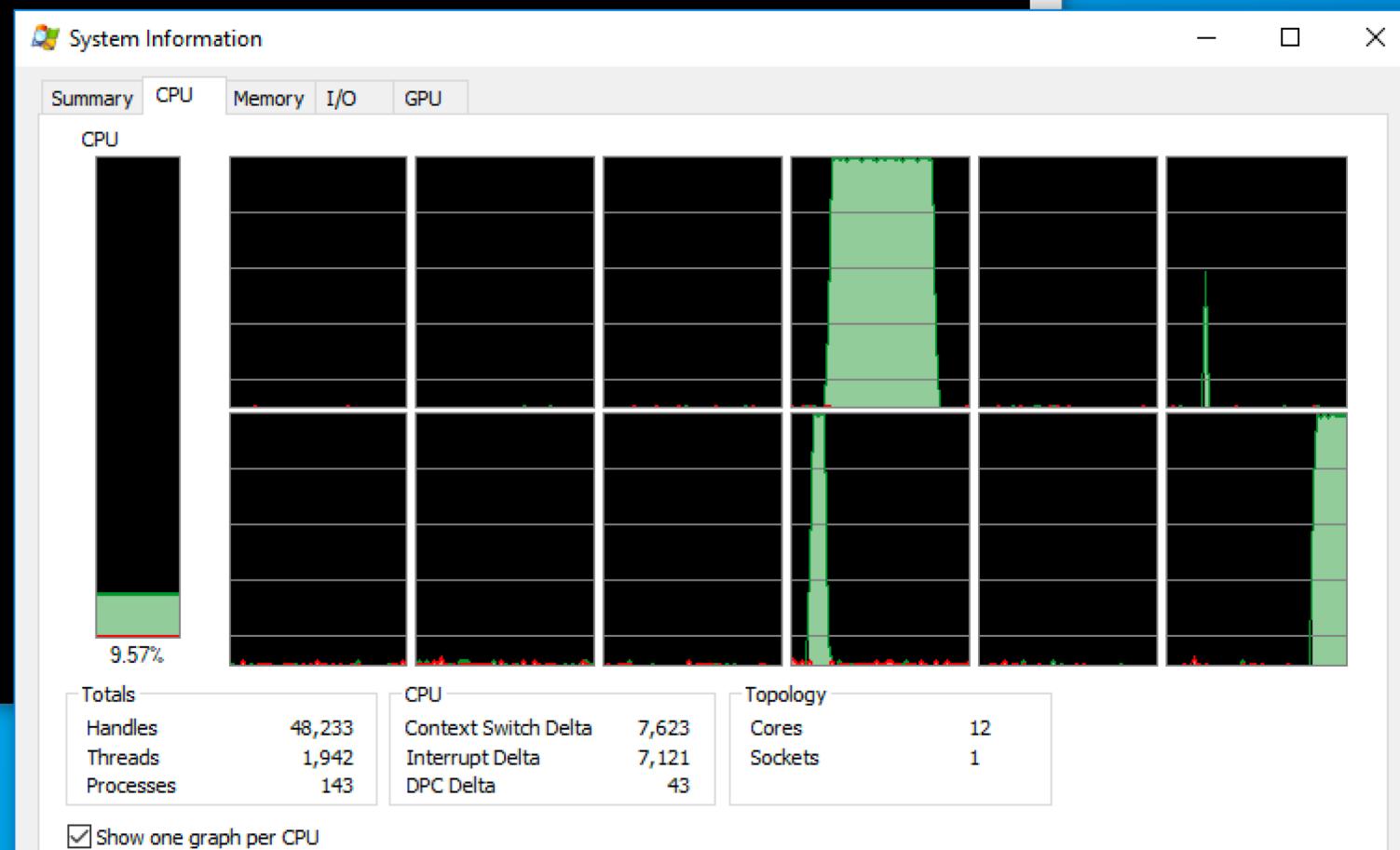
via Jeffrey Funk,

<https://www.slideshare.net/Funk98/end-of-moores-law-or-a-change-to-something-else>

Single core performance is poor

```
C:\Users\mkennedy>python
Python 3.7.0 (v3.7.0:1bf9cc5093, Jun 27 2018, 04:59:51) [MSC v.1914 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> x = 1
>>> while True:
...     x += 1
...

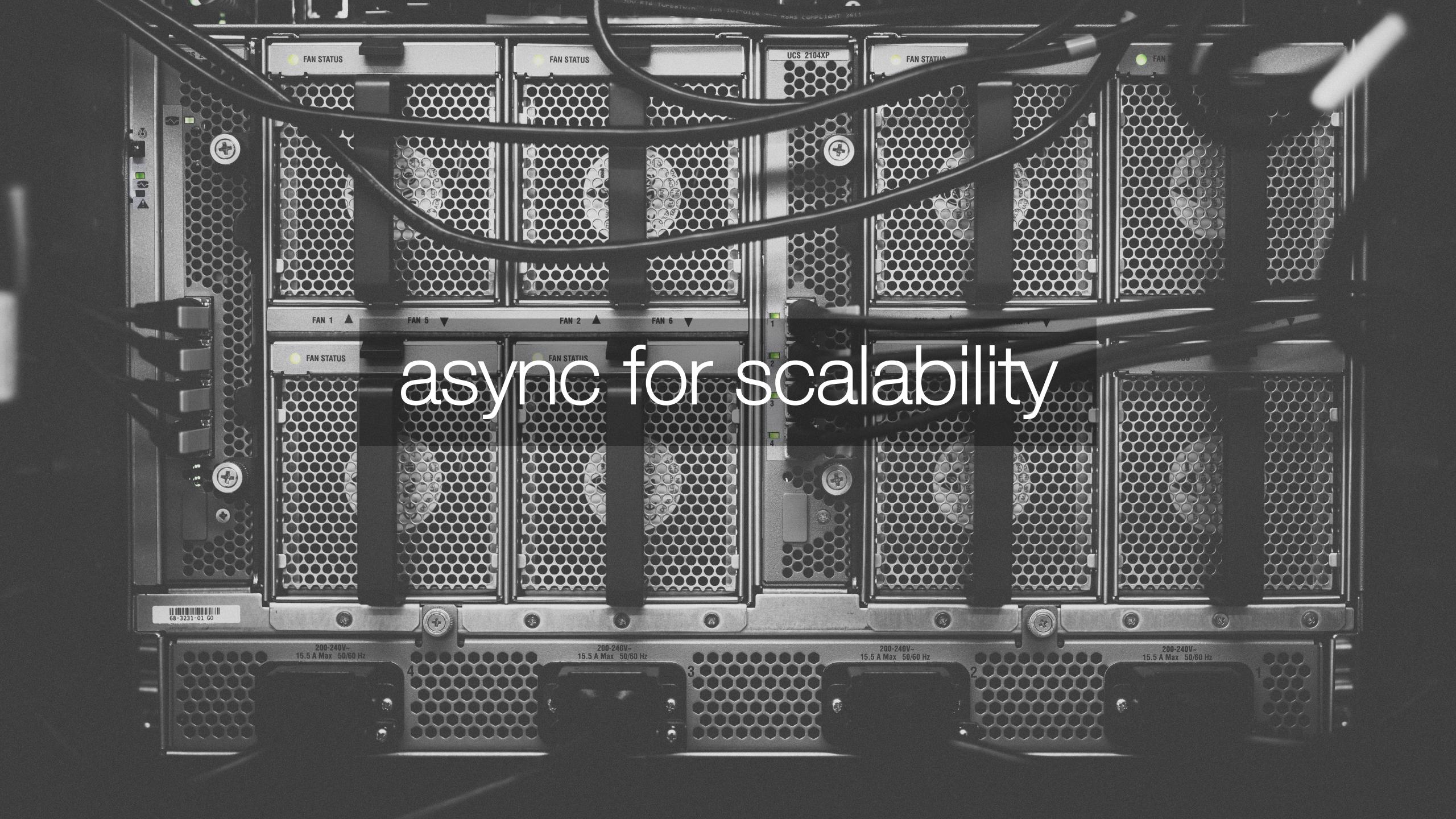
```



An upper bound for improvement

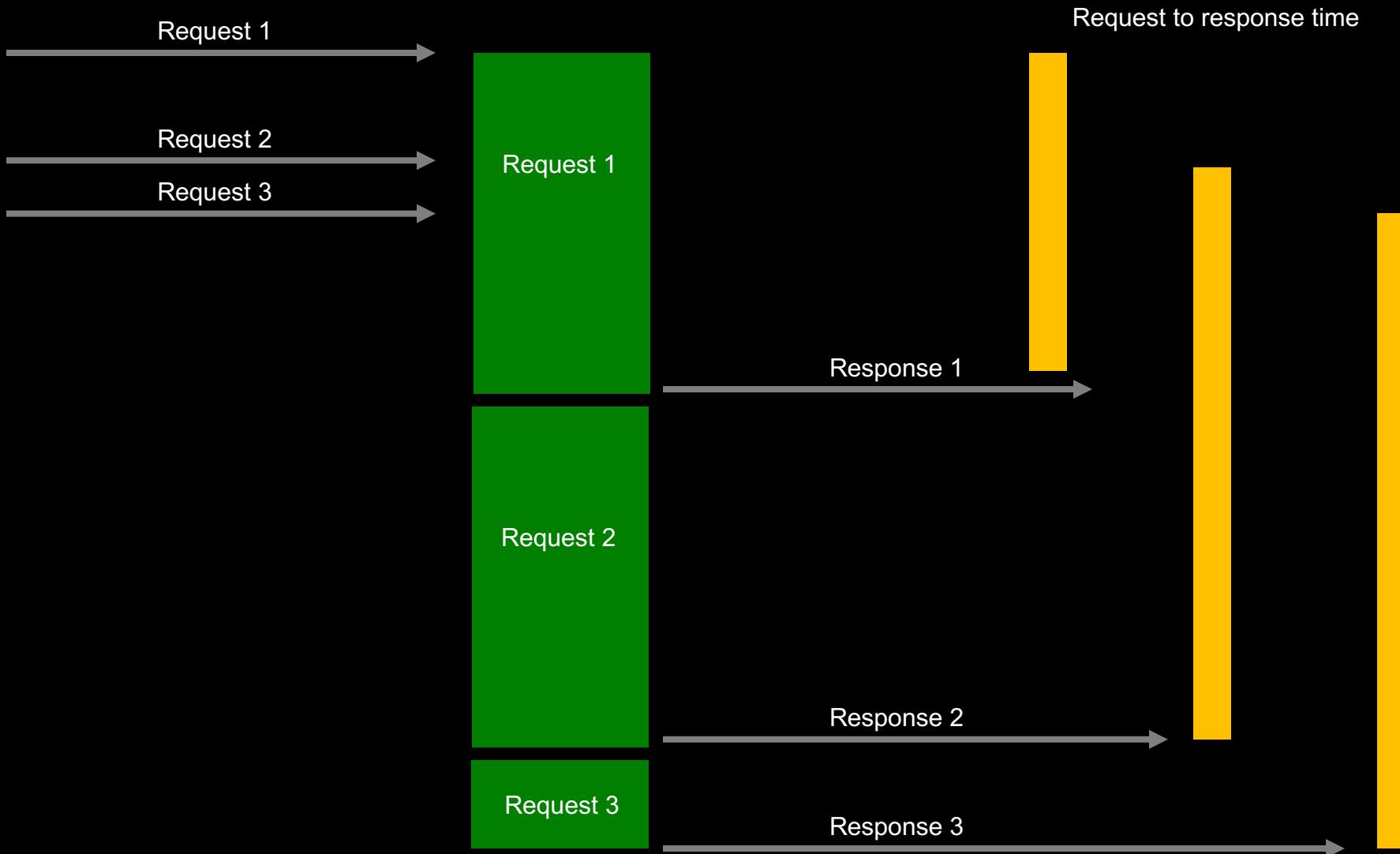


Absolute performance
improvement = **20%**

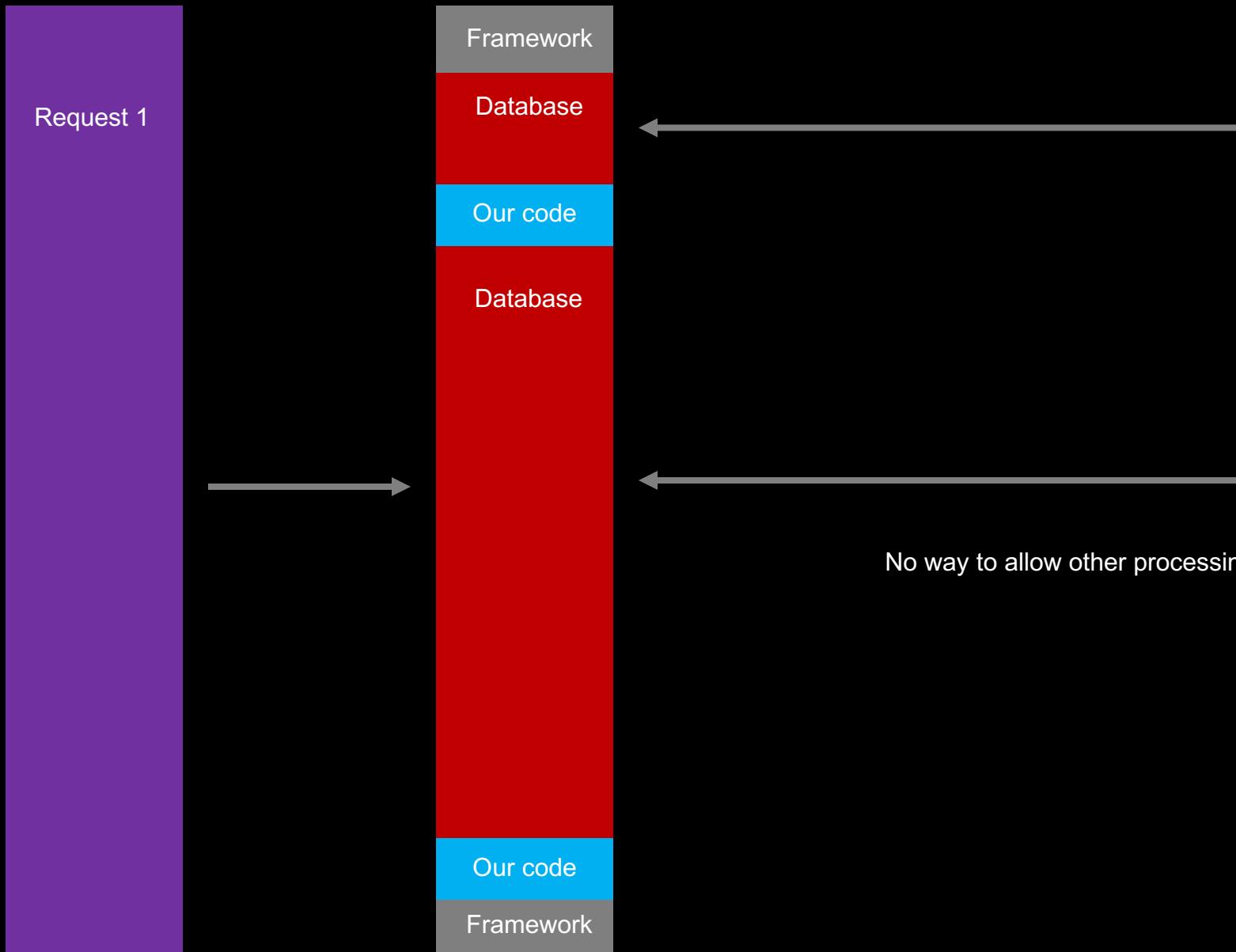


async for scalability

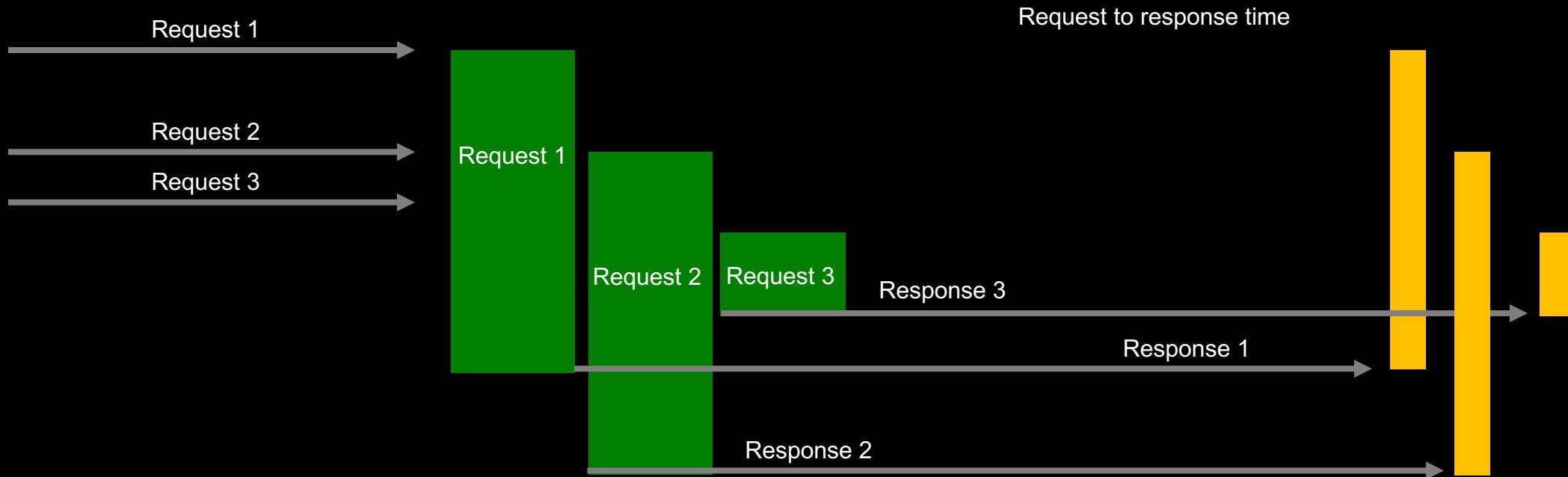
Synchronous execution



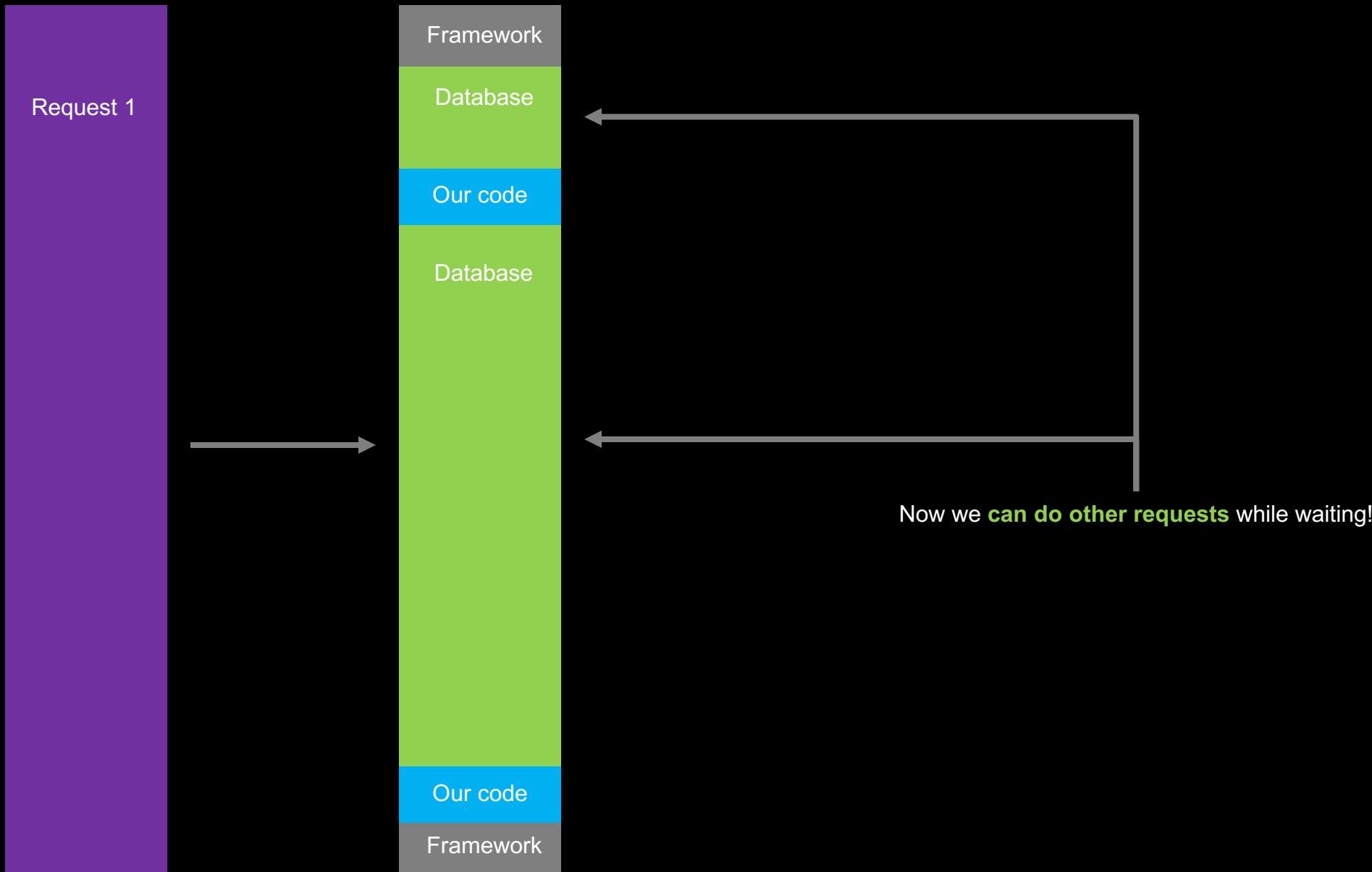
Synchronous execution – zoom in



Asynchronous execution



Asynchronous execution – zoom in



Async in Python (techniques)

Do more **at once**

`asyncio`

`threads`

Do things **faster**

`multiprocessing`

`C / Cython`

Do these **easier**

`trio`

`unsync`

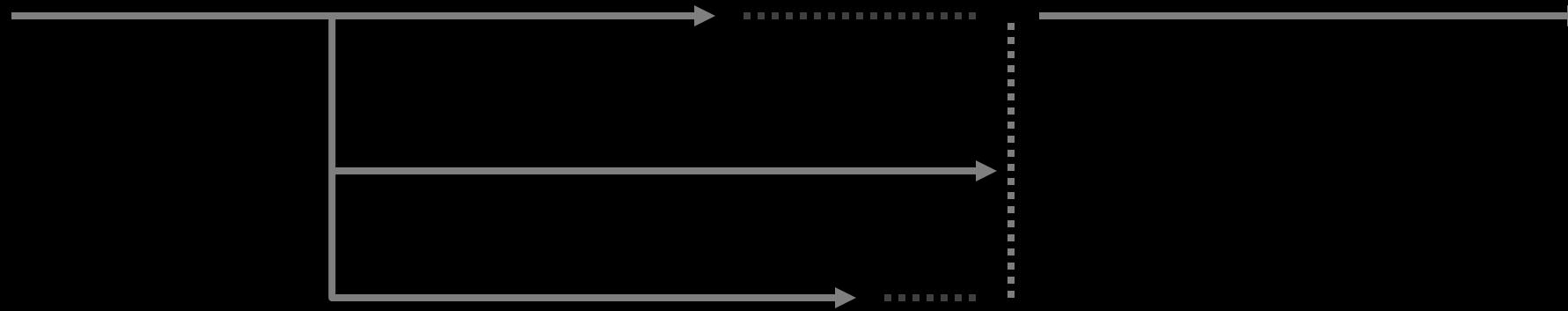
Why don't threads add computational speed?

Python has a memory management feature called **the GIL**, or **Global Interpreter Lock**

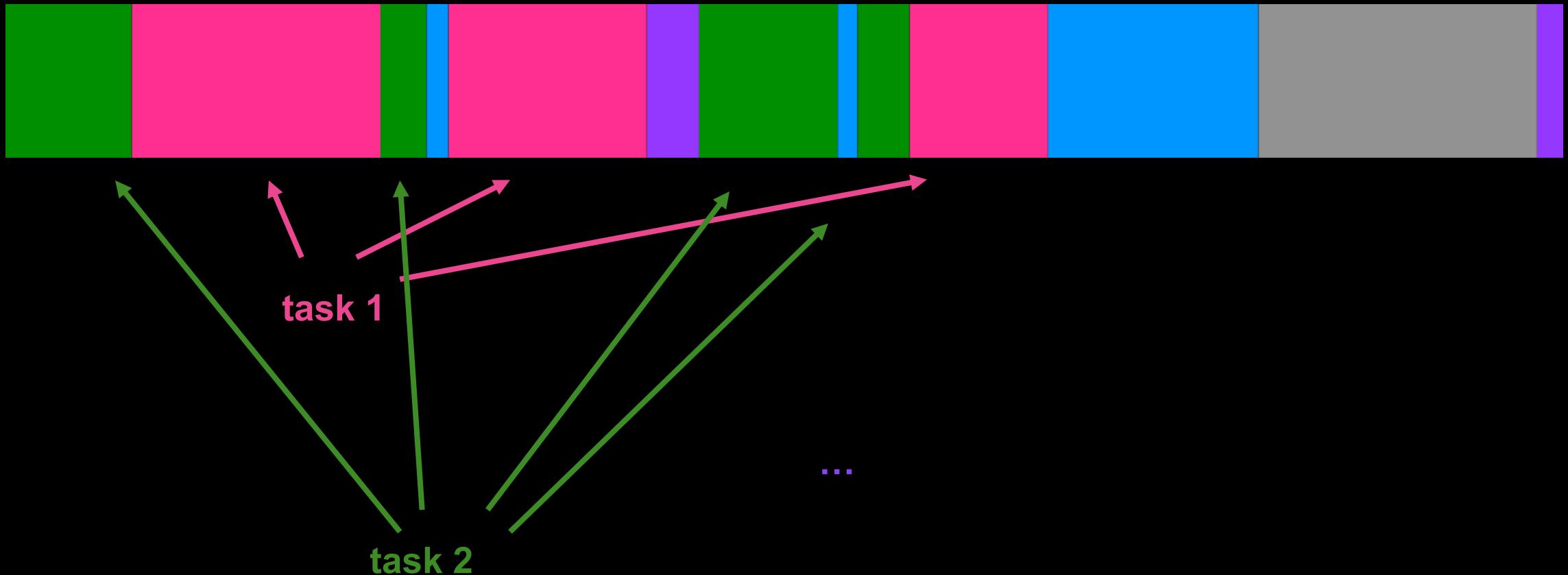


<https://realpython.com/python-gil/>

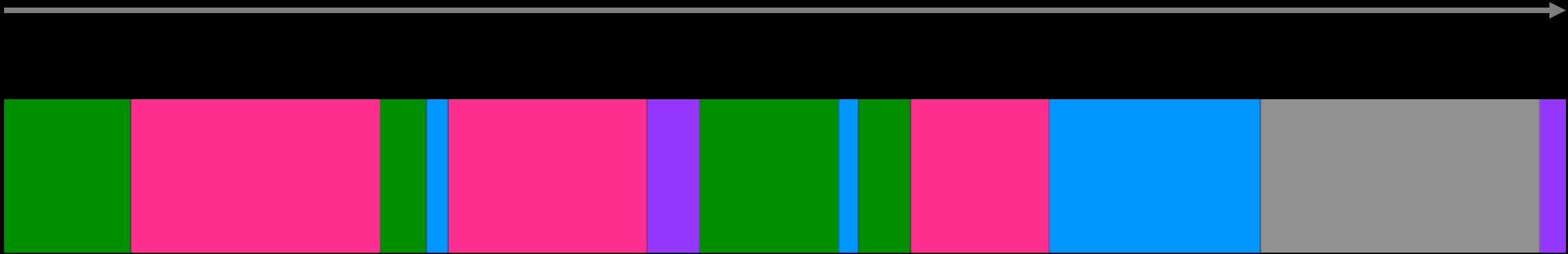
Typical concurrent programming



I/O driven concurrency



async event loop



Demo: Very simple example
of Fibonacci generator with yield

Demo producer consumer

Anatomy of an async method

Begin by making method **async**

```
async def process_data(num: int, data: asyncio.Queue):  
    processed = 0  
  
    while processed < num:  
        item = await data.get()  
        # work with item
```

await all async methods you call.

Performance improvements: Sync version

```
→ python3 sync_producer.py
-- generated item 15
-- generated item 16
-- generated item 17
-- generated item 18
-- generated item 19
-- generated item 20
+++ Processed value 1 after 19.75 sec.
+++ Processed value 4 after 19.61 sec.
+++ Processed value 9 after 18.70 sec.
+++ Processed value 16 after 18.04 sec.
+++ Processed value 25 after 17.72 sec.
+++ Processed value 36 after 17.43 sec.
Total time: 29.81 sec, ave latency: 15 sec
```

Performance improvements: async version

```
→ python3 sync_producer.py
-- generated item 7
+++ Processed value 49 after 0.24 sec.
-- generated item 6
-- generated item 8
+++ Processed value 36 after 0.23 sec.
+++ Processed value 64 after 0.59 sec.
-- generated item 7
-- generated item 9
+++ Processed value 49 after 0.15 sec.
+++ Processed value 81 after 0.60 sec.
-- generated item 10
Total time: 21.30 sec, ave latency: 0.25 sec
```

8.51 seconds faster, 60x less latency

Fast loops

uvloop

uvloop is an ultra fast
implementation of the asyncio
event loop on top of libuv.

<https://uvloop.readthedocs.io/>

Using uvloop

```
import uvloop

asyncio.set_event_loop_policy(uvloop.EventLoopPolicy())

loop = asyncio.get_event_loop()
# ...
```

A photograph of a person wearing a green t-shirt and safety goggles, working in a workshop. They are using a power tool on a piece of metal, creating numerous bright orange sparks that fly outwards. The workshop is filled with various industrial equipment, including a large green machine on the left and a lathe in the foreground. The background shows windows with a view of trees outside.

How about some real work?

Demo web scraper

Async web requests

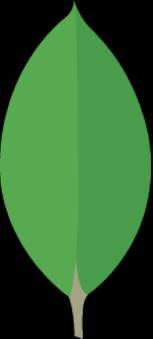
Use an **async with block** to start
the request

```
async def get_html(url: str):  
  
    async with aiohttp.ClientSession() as session:  
        async with session.get(url) as resp:  
            resp.raise_for_status()  
  
    return await resp.text()
```

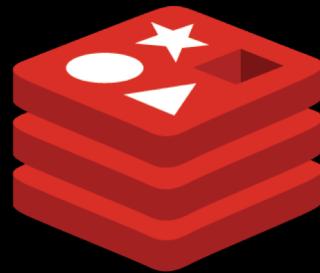
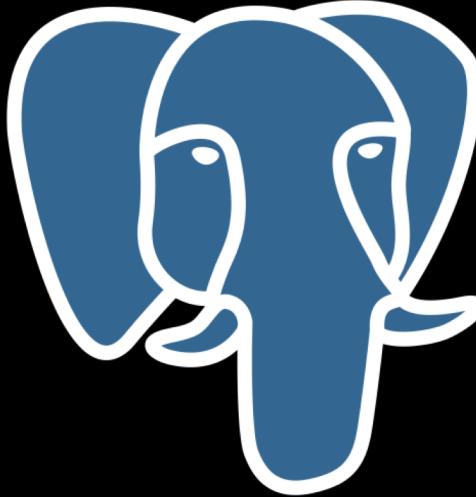


await the network read operation

Other async capable libraries



mongoDB



redis

File I/O

A screenshot of a GitHub repository page for 'Tinche/aiofiles'. The page title is 'Tinche/aiofiles: File support for asyncio'. The repository has 28 stars, 544 forks, and 35 open issues. It features 84 commits, 2 branches, 5 releases, and 12 contributors. The license is Apache-2.0. The latest commit was 14 days ago. The repository is open for business. A pull request was opened to simplify code by removing pre 3.5 compatibility checks.

Tinche/aiofiles: File support for asyncio

asyncio

84 commits | 2 branches | 5 releases | 12 contributors | Apache-2.0

Branch: master | New pull request | Create new file | Upload files | Find file | Clone or download

Tinche Add Python versions badge. | Latest commit 80d2f5d 14 days ago

aiofiles | 0.5.0 open for business. | 14 days ago

tests | simplify code by remove pre 3.5 compat checks | 2 months ago

<https://github.com/Tinche/aiofiles>

MongoDB

A screenshot of a GitHub repository page for the project "Scille / umongo". The page shows the repository's main statistics and recent activity.

Repository Information:

- Owner: Scille
- Name: umongo
- Description: sync/async MongoDB ODM, yes.
- Code: 377 commits
- Issues: 21
- Pull requests: 6
- Projects: 0
- Wiki
- Insights
- Watched by 13 users
- Unstarred by 119 users
- Forked by 22 users

Recent Activity:

- touilleMan Merge pull request #136 from elyssonmr/master ... Latest commit 4b4322b 23 days ago
- docs Fix grand-child inheritance (#66) 2 years ago
- examples Merge branch 'master' of https://github.com/mandarup/umongo a month ago

Branch Selection: Branch: master

Actions: New pull request, Create new file, Upload files, Find file, Clone or download

<https://github.com/Scille/umongo>

Postgres

The screenshot shows a GitHub repository page for `MagicStack/asyncpg`. The page title is "MagicStack / asyncpg". The repository description is "A fast PostgreSQL Database Client Library for Python/asyncio.". Key statistics displayed include 577 commits, 9 branches, 23 releases, 19 contributors, and Apache-2.0 licensing. The "Code" tab is selected. The repository has 124 watchers, 2,816 stars, and 124 forks. The last commit was made 17 days ago. Recent commits listed are "creativ and 1st1 Typo fix in transaction description.", "asyncpg v0.17.0", and "Fix the CI release system".

MagicStack / **asyncpg**

A fast PostgreSQL Database Client Library for Python/asyncio.

577 commits 9 branches 23 releases 19 contributors Apache-2.0

Branch: master ▾ New pull request Create new file Upload files Find file Clone or download ▾

creativ and 1st1 Typo fix in transaction description. Latest commit a6fdf5c 17 days ago

.ci asyncpg v0.17.0 2 months ago

.github Fix the CI release system 3 months ago

<https://github.com/MagicStack/asyncpg>

Redis

A screenshot of a GitHub repository page for 'jonathanslenders/asyncio-redis'. The page shows the repository's details, including 312 commits, 3 branches, 1 release, and 20 contributors. The latest commit was made 27 days ago. The repository is a Redis client for Python asyncio, with documentation available at <http://asyncio-redis.readthedocs.org/>.

jonathanslenders / asyncio-redis

312 commits 3 branches 1 release 20 contributors

Branch: master ▾ New pull request Create new file Upload files Find file Clone or download ▾

Commit	Message	Date
jonathanslenders	Release 0.15.1	Latest commit abe2089 27 days ago
asyncio_redis	Merge pull request #110 from kdub0/master	a month ago
docs	Improved docs.	4 years ago
examples	Comments in examples/benchmarks/hiredis_test.py added.	4 years ago

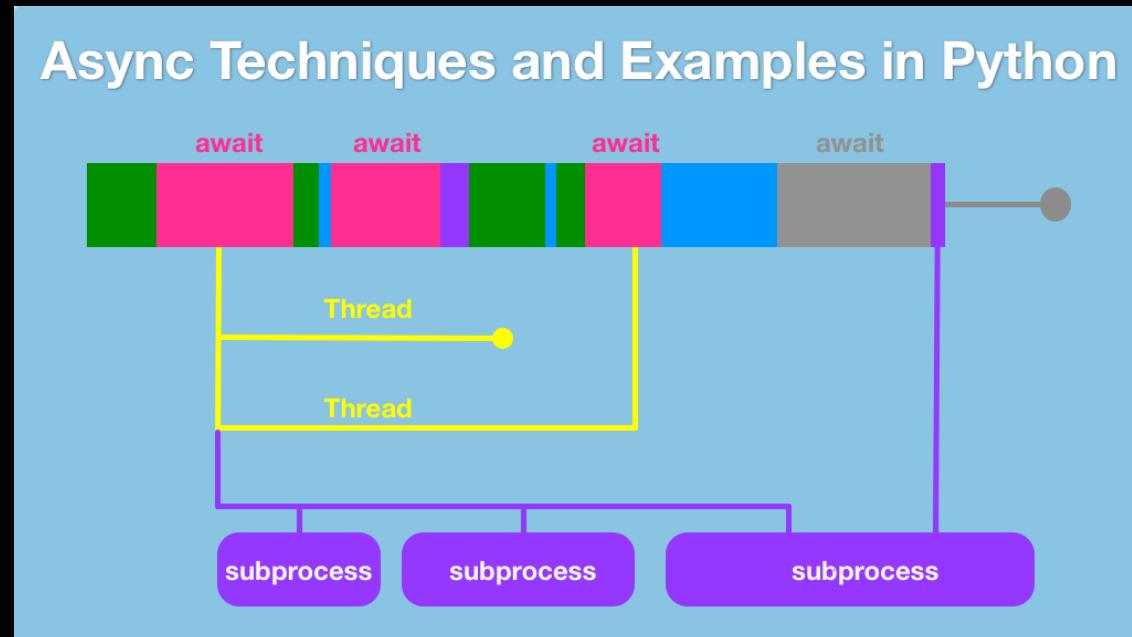
<https://github.com/jonathanslenders/asyncio-redis>

THANK YOU

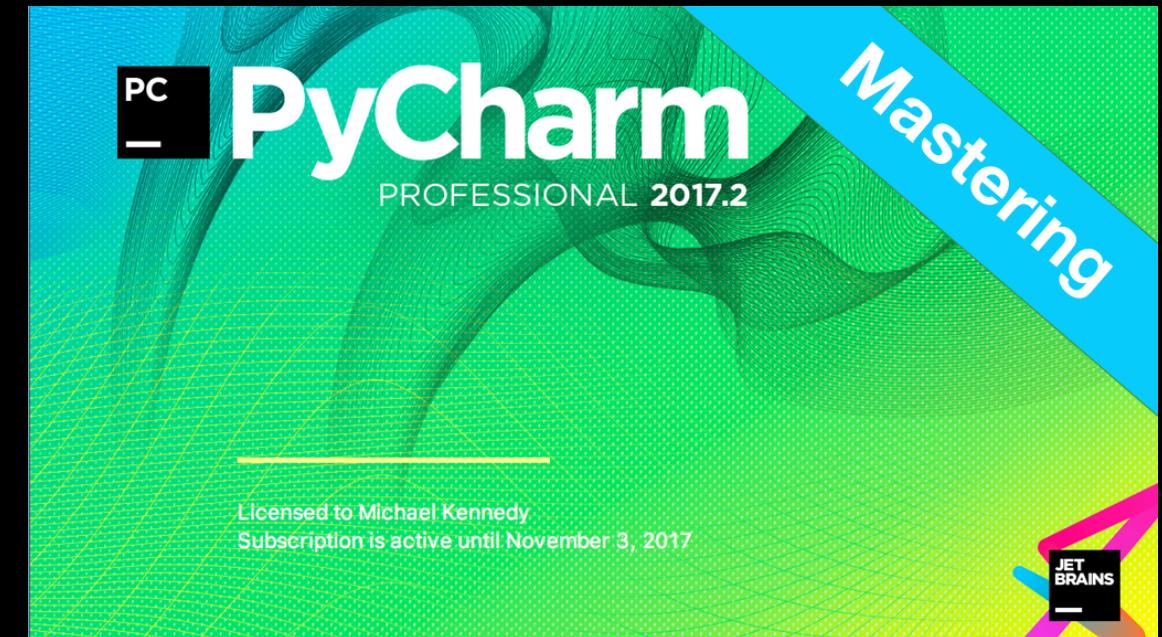


Michael Kennedy
@mkennedy

Want to go deep? Check out my online courses



talkpython.fm/async



talkpython.fm/pycharm