Async All The Way Down

An adventure in asynchrony

Aru Sahni

@IAmAru

arusahni

Canines As A Service

- People *need* a web service that serves a dog photo upon request
- We can do it!
  - Already have an internal dog microservice (http://localhost:8000)
  - Need to track requests for billing
  - Need to provide the names of the dogs

Furst Attempt

Furst Attempt: Synchronous

- Use what I'm used to
  - Flask
  - Requests
  - psycopg2 (Postgres)

```
RESULTS = Counter()

def count_filename(filename):
    RESULTS[filename] += 1

def get_dog_name(filename):
    names = # <Database query>
    return names

@app.route("/")
def get_dog():
    image = requests.get("http://canine/dog")
    filename = image.headers["X-Filename"]
    count_filename(filename)
    dog_name = get_dog_name(filename)
    return send_file(image.content,
                     attachment_filename=dog_name,
                     mimetype="image/jpeg")
```

Furst Attempt: Success!

It works! http://localhost:5000

Total Requests	90
Median Time	426ms
Avg. Time	402ms
Std. Dev	278ms

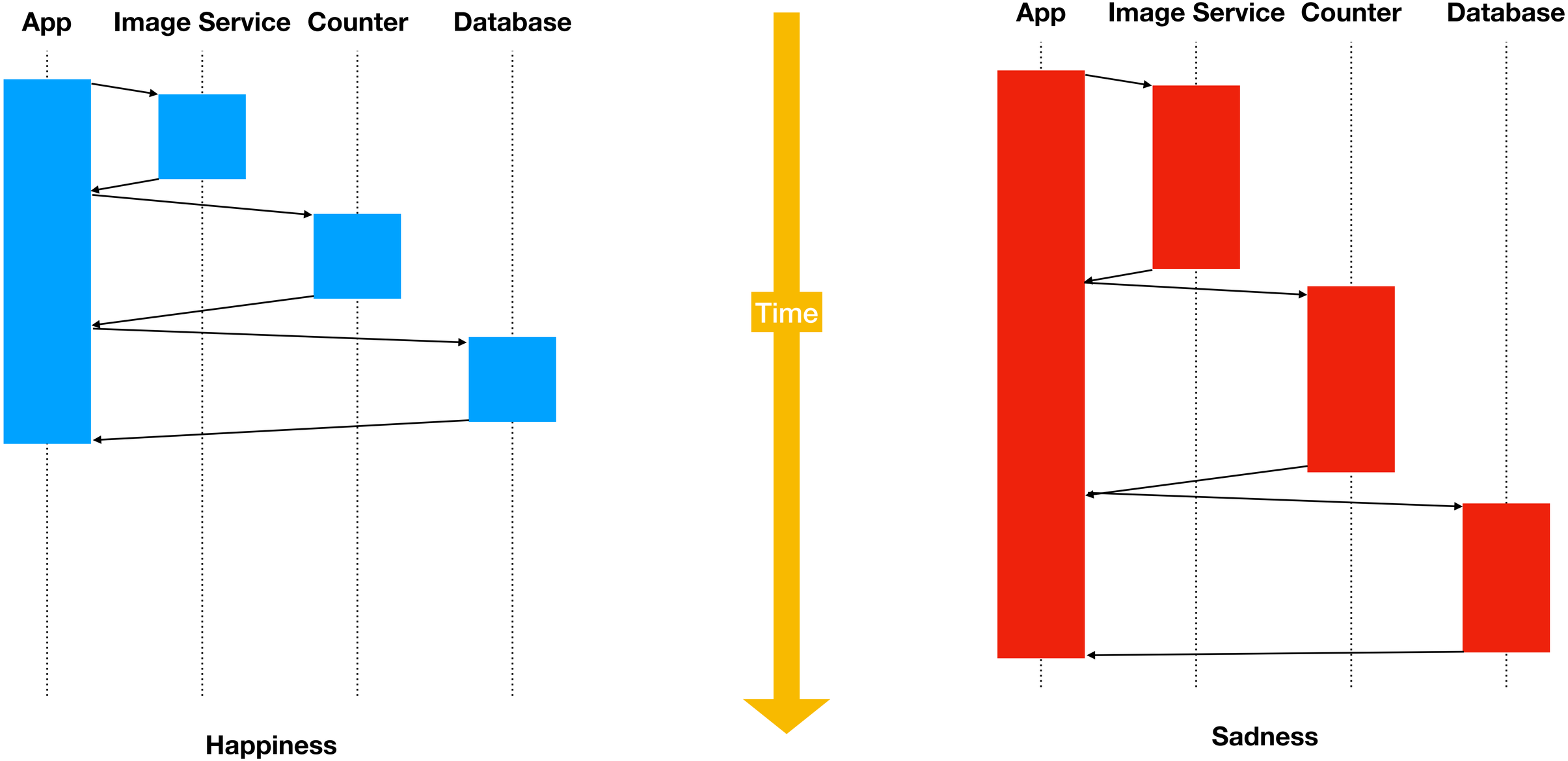
Networks are Lossy

Furst Attempt: Loss Is Pain

😞 http://localhost:5000/?latency=15

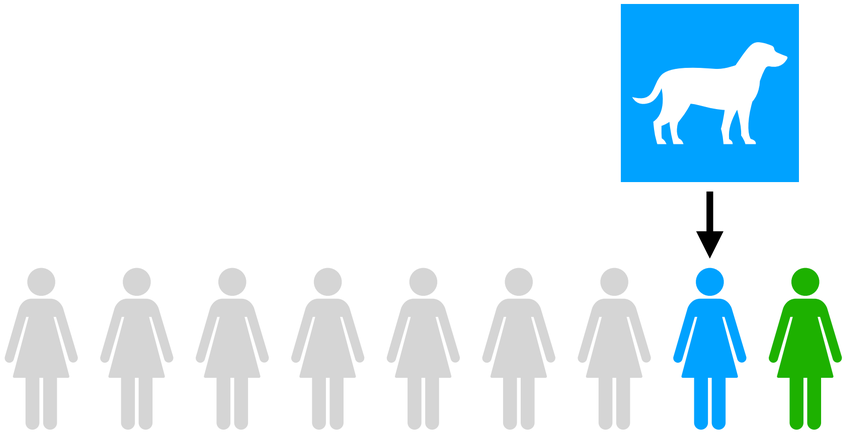
Total Requests	90
Median Time	45113ms
Avg. Time	45177ms
Std. Dev	158ms

I/O Latency's Impact

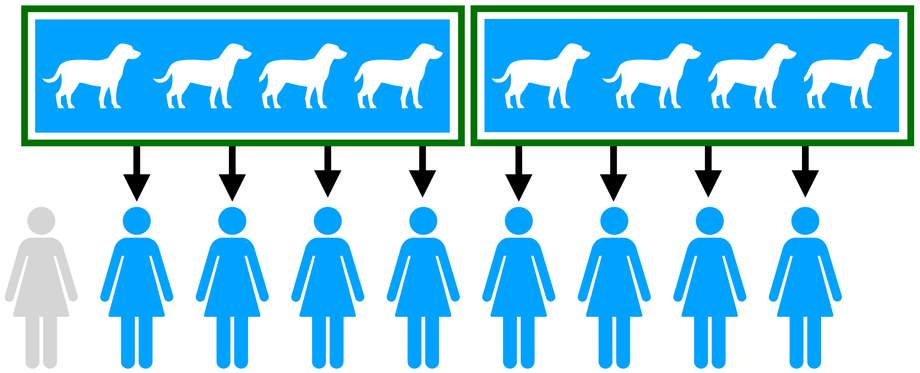


On The Server: Single Thread

On The Server: Single Thread

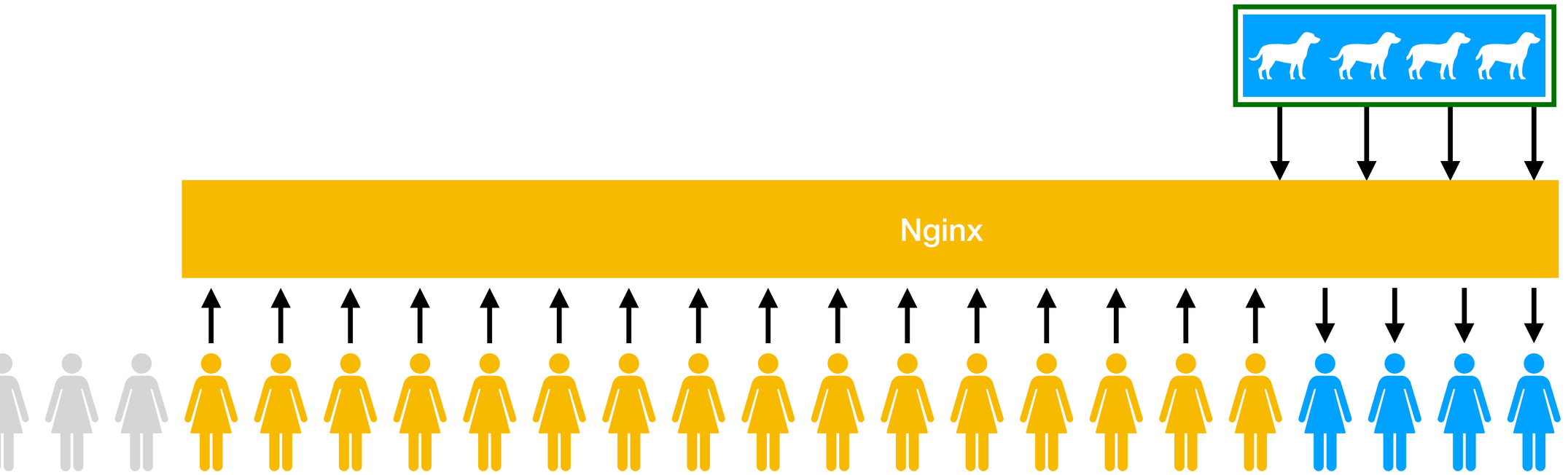


On The Server: Multi{ Thread,Process}



- uwsgi
- gunicorn

On The Server: At Scale



Enter Async

- Long requests block others
- Can add threads and processes
- Need a better way to concurrently handle I/O-heavy requests
- That’s asyncio!

Enter Async

```
async def my_task():
    x = 1
    y = await get_y()
    return x + y
```

Run

Yield

Run

https://medium.com/@gauravbaliyan/a-good-use-of-asyncio-in-python-and-django-8aa7bc401e5f

Second Attempt

Second Attempt: Paint it Async

- New hotness
  - Quart
  - aiohttp
  - asyncpg (Postgres)

```
async def count_filename(filename):
    RESULTS[filename] += 1

async def get_dog_name(filename):
    """Get the dog name(s)."""
    return (await # QUERY)

@app.route("/")
async def get_dog():
    async with aiohttp.ClientSession() as session:
        async with session.get("http://canine/any") as resp:
            filename = resp.headers["X-Filename"]
            await count_filename(filename)
            dog_name = await get_dog_name(filename)
            return await send_file(BytesIO(await resp.read()),
                attachment_filename=dog_name,
                mimetype="image/jpeg")
```

Second Attempt: I thought Async was Good?

🤔 http://localhost:5010/?latency=15

Total Requests	90
Median Time	45234ms
Avg. Time	45754ms
Std. Dev	1154ms

Blocking vs. Nonblocking

Sequential Synchronous

```
def dress():
    put_on_pants()
    put_on_shoes()
```

Sequential Synchronous

```
def commute():
    listen_to_podcasts()
    ride_the_metro()
    drink_coffee_or_beer()
```

Sequential Blocking

```
async def commute():
    await listen_to_podcasts()
    await ride_the_metro()
    drink_coffee_or_beer()
```

Sequential Nonblocking

```
async def commute():
    await asyncio.gather(
        listen_to_podcasts(),
        ride_the_metro()
    )
    drink_coffee_or_beer()
```

Third Attempt

Third Attempt: Long-Awaited

- Don’t block unless necessary

```
async def count_filename(filename):
    RESULTS[filename] += 1

async def get_dog_name(filename):
    """Get the dog name(s)."""
    return (await # QUERY)

@app.route("/")
async def get_dog():
    async with aiohttp.ClientSession() as session:
        async with session.get("http://canine/any") as resp:
            filename = resp.headers["X-Filename"]
            dog_name, _ = await asyncio.gather(
                get_dog_name(filename),
                count_filename(filename, latency)
            )
            return await send_file(BytesIO(await resp.read()),
                attachment_filename=dog_name,
                mimetype="image/jpeg")
```

Third Attempt: That's more like it.

🤘 http://localhost:5020/?latency=15

Total Requests	90
Median Time	30440ms
Avg. Time	30672ms
Std. Dev	695ms

45s

30s

Use Async

- Network Activity
  - HTTP
  - Database queries
- Disk I/O

Use {other}

- Computation
  - Data Science/ML
  - Image Processing
  - Compression
  - Cryptography
  - Fibonacci Flexing

Async Libraries

- Use Flask? Check out Quart.
- Use Django? Check out Django 3.
- Requests? aiohttp
- DRF? FastAPI
- SQLAlchemy? asyncpgsa.

`await more_resources()`

- Miguel Grinberg Asynchronous Python for the Complete Beginner
  - https://www.youtube.com/watch?v=iG6fr81xHKA
- Lynn Root - Advanced asyncio: Solving Real-world Production Problems
  - https://www.youtube.com/watch?v=bckD_GK80oY
- Brett Cannon’s blog
  - https://snarky.ca/how-the-heck-does-async-await-work-in-python-3-5/


@IAmAru
Questions?