Screaming Fast API Clients

Moshe Zadka – https://cobordism.com

2020

I live in the San Francisco Bay Area. I want to acknowledge that the San Francisco Bay Area is the ancestral home of the Muweka Ohlone.

Acknowledgement of Country
San Francisco Bay Area
Ancestral home of the Muweka Ohlone.

Latency is the Site Killer
Every 100ms of latency in your site lose more customers

(Micro)service Architecture
Layers

(Micro)service Architecture
Fan-out

Lognormal Black Swans

- Lognormal: $1/x$ (kinda)
- Normal: e^{-x^2}

You are extremely unlikely to meet someone over seven feet, even if you are invited to an NBA team after party – but some of my posts are 4000 words long, about 5 standard deviations from my average of 500 words: this is like meeting someone who's 7 feet two inches in your local grocery store!

Averages Lie
Only good for normal distributions

Your Backend is Slow
Lognormal, not normal

Multiplicity Magnifies Outliers
With 5 queries:

- P90 becomes P50
- P99 becomes P90

Measure
Histograms, not averages

Measure
All layers

Let's Write Some Code

```
@app.route('/')
def hello_world():
    all_values = sum(
        CLIENT.get(URL).json()["value"]
        for x in range(FANOUT)
    )
    return json.dumps(dict(total=all_values))
```

This is simplified code that shows what usually goes on in the "middle tier": makes some queries to the backend, do some local computation, and return a JSON.

How long does it take? The sum of all the times. This means that a backend that is *occasionally* slow will make this function *almost always* slow.

Let's Write Some Code

```
@app.route('/')
async def hello_world(request):
    all_values = await defer.gatherResults([
        CLIENT.get(URL).addCallback(treq.json_content)
        for x in range(FANOUT)
    ])
    total = sum(res["value"] for res in all_values)
    return f'Total {total}'
```

Parallelizing clients means taking a maximum, not the sum. What's the difference? If you are almost sure to get a single slow hit, now the timing is constant: it does not matter if it is 2 or 1.

Being slowest as the slowest request, and not the sum of the two slowest requests, is progress.

Let's Simulate

With fanout of 10:

- P50: each: 0.04 seq: 0.82 par 0.3
- P90: each: 0.23 seq: 1.8 par 0.98
- P99: each: 1.04 seq: 4.33 par 3.05

While it does not improve the P99 by much, parallelization gives us 3x-2x speedups at P50 and P99.

Timing Out and Retry

Temporary slow-downs

When backends are slow, this is often transient. A single host might be overloaded. The disk might be busy. A packet might be lost. Giving up quickly and retrying can often mean the difference.

Let's Write Some Code

```
def get_with_timeout(url):
    def try(_ign=None):
        return CLIENT.get(URL).addCallback(treq.json_content)
    d = try()
    d.addTimeout(0.1)
    d.addErrback(try)
    return d
```

Let's Simulate

- P50: 0.18
- P90: 0.51
- P99: 1.66

Let's Simulate

Retried requests: 25

Let's Write Some Code

```
def get_with_timeout(url):
    def try(_ign=None):
        return CLIENT.get(URL).addCallback(treq.json_content)
    d = try()
    d.addTimeout(0.1)
    d.addErrback(try)
    d.addTimeout(0.4)
    return d
```

Let's Simulate

- P50: 0.19
- P90: 0.53
- P99: 0.6

Summary

- Latency
- Backend latency
- SLA
- Measurement
- Simulation