Caching Best Practices

Moshe Zadka – https://cobordism.com

2021

Acknowledgement of Country

Belmont (in San Francisco Bay Area Peninsula)
Ancestral homeland of the Ramaytush Ohlone

What is a Cache

(For the purposes of this talk)

- Key/value
- In-memory
- Not pesistent

Not a Cache

- Reliable
- Communication

Why Cache? Latency

Long computations

Why Cache? Resources

Expensive computations

Why Cache? Examples

- Latency: Username retrieved from different datacenter
- Resources: Indexed DB join

Memcache

- Popular cache
- Fast
- Does one thing
- Focus of this talk

Memcache interface

```
# Simplified from pymemcahce API
class MemcacheClient:
    def set(self, key, value):
        ...
    def get(self, key):
        ...
    def get_many(self, keys):
        ...
    def delete(self, key):
        ...
```

Memcache key lifecycle

- set
- get
- expire/delete/evict

Memcache routing

(Like HTTP proxies)

Memcache routing

(Like HTTP proxies)
Examples:

Memcache routing

(Like HTTP proxies)
Examples:
mcrouter

Memcache routing

(Like HTTP proxies)
Examples:
mcrouter
twemproxy

Memcache client-side routing

Client responsible for:

Memcache client-side routing

Client responsible for:

- Detecting server outage and fail-over
- "Fair" sharding of keys

Memcache: Routing is necessary

Redundancy
Scaling

Caching points

- Use: optimize latency or resources
- Keys can vanish
- Routing necessary (client or server)

Caching tier example

- McRouter layer
- Memcache layer
- TCP/DNS load-balancing for McRouter

Caching complicates code

Potentially introduces new bugs

Caching adds corner cases

Code needs to account for them.

Network: Disconnection

Persistent connection disconnects

Network: Timeouts

Packets go missing
Servers go offline

Keys: Missing

Server rebooted
Expiry
...and more

Values: Invalid

Key collision
Problem in previous version

Values: Outdated

Things change

Cache invalidation

Hard

Cache expiration

Upper limit on staleness

Cache delete

Mixed blessing

Cache adds complexity

More ways to fail...

Cache adds complexity

More ways to fail...
especially under pressure.

Test cache hit path

Often missed in unit tests

Test cache miss path

...in realistic scenarios.

Test bad cache value

If you update your software, can you detect an old cache value?

Test cache networking: disconnection

Will happen at the weirdest times.

Test cache networking: timeouts

How much can the cache slow you down?

Cache is an optimization

Treat it as one

Measure cache effectiveness

Is it helpful? How much?

Measure ratio by code path

Each code path/object type should have its own counter
Counters are good

Measure timer histogram by hit/miss

Is there a difference? Enough?

Configuration: Dummy cache

Cache object that always "miss"

- Debugging
- A/B performance
- Remediation

Performance: A/B

By key hash

What to cache?

Granularity?

What to cache?

Granularity?
Think + Test

Cache: Size concerns

Caching can cause eviction

Cache: Serialization

JSON?

Cache: Serialization

JSON?Pickle?

Cache: Socket options

`TCP_NODELAY?`

Cache: Socket options

```
TCP_NODELAY?
TCP_QUICKACK?
```

Cache: Socket options

```
TCP_NODELAY?
TCP_QUICKACK?
TCP_CORK?
```

Cache: Multiget

Refactoring might be required

Takeaways

- Caching is an optimization
- Test correctness
- Monitor performance
- Apply best practices