Application Caching: The Hidden Microservice

Scott Mansfield (@sgmansfield)
Senior Software Engineer
EVCache
Watch TV shows & movies anytime, anywhere.
Sign in

Get ready to watch thousands of TV shows and movies.

Enter your password

 tester_pres@netflix.com

Forgot Password? ✓ Remember Email

Who's watching Netflix?

Tester

Kids

Add Profile
Tester, choose 3 you liked
It will help us find movies & TV shows you'll love!
Tester, choose 3 you liked
It will help us find movies & TV shows you'll love!

Continue 3 chosen
Is it true that absolute power corrupts absolutely? Congressman Frank Underwood absolutely intends to find out.
He served 18 years for a crime he didn't commit. Now he's on the line again, and some want to see him put away for good.
NETFLIX ORIGINAL

STRANGER THINGS

*** 2016 TV-14 1 Season HD

A lost boy. A government lab steeped in terrible secrets. A mother who won't rest until her son is found.

This nostalgic nod to 1980s sci-fi/horror classics pays homage to "E.T.", "Poltergeist" and the novels of Stephen King.

Trending Now

STRANGER THINGS

GREY'S ANATOMY

MAD MEN

NURSE JACKIE

THE WEST WING

Comedies

HOME

THE DO-OVER

NETFLIX
Titles with: Adam Sandler

Saturday Night Live: The...
Patch Adams
Adam
Full Circle: A Nate Adams...
Adam's Apples

Adam Sandler
Adam Scott
Adam Brody
Adam Levine
Amy Adams
Adam Driver
Adam Saunders
Adam West
Adam Devine
Adam Faraizl
First they got the coke. Then they got the money. Now the Colombian cartels want the power. Let the drug wars begin.

Director José Padilha and star Wagner Moura first teamed up in "Elite Squad," a 2008 Berlin Film Festival award winner.
I am Pablo Emilio Escobar Gaviria.
or lead.
Season 2 Coming September 2
The true story of Colombia's infamously violent and powerful drug cartels fuels this gritty new gangster drama series.

Wagner Moura, Boyd Holbrook, Pedro Pascal
TV Shows, Crime TV Shows
Season 2 Coming September 2
The true story of Colombia's infamously violent and powerful drug cartels fuels this gritty new gangster drama series.

Wagner Moura, Boyd Holbrook, Pedro Pascal
TV Shows, Crime TV Shows
“Descenso”
Chilean drug chemist Cockroach brings his product to Colombian smuggler Pablo Escobar. DEA agent Steve Murphy joins the war on drugs in Bogota.
(57m)

“The Sword of Simón Bolívar”
Communist radical group M-19 makes a move against the narcos, while Murphy gets an education in Colombian law enforcement from his new partner Peña.
(46m)

“The Men of Always”
Murphy encounters the depths of government corruption when he and Peña try to derail Escobar’s political ambitions by proving he’s a narco.
(46m)

“The Palace in Flames”
Despite a new extradition treaty, the U.S. poor...
90 seconds
<table>
<thead>
<tr>
<th>What do caches touch?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Signing up*</td>
</tr>
<tr>
<td>Logging in</td>
</tr>
<tr>
<td>Choosing a profile</td>
</tr>
<tr>
<td>Picking liked videos</td>
</tr>
<tr>
<td>Personalization*</td>
</tr>
<tr>
<td>Loading home page*</td>
</tr>
<tr>
<td>Scrolling home page*</td>
</tr>
<tr>
<td>A/B tests</td>
</tr>
<tr>
<td>Video image selection</td>
</tr>
<tr>
<td>Searching*</td>
</tr>
<tr>
<td>Viewing title details</td>
</tr>
<tr>
<td>Playing a title*</td>
</tr>
<tr>
<td>Subtitle / language prefs</td>
</tr>
<tr>
<td>Rating a title</td>
</tr>
<tr>
<td>My List</td>
</tr>
<tr>
<td>Video history*</td>
</tr>
<tr>
<td>UI strings</td>
</tr>
<tr>
<td>Video production*</td>
</tr>
</tbody>
</table>

* multiple caches involved
Ephemeral Volatile Cache

Key-Value store optimized for AWS and tuned for Netflix use cases
What is EVCache?

Distributed, sharded, replicated key-value store
Tunable in-region and global replication
Based on Memcached
Resilient to failure
Topology aware
Linearly scalable
Seamless deployments
Why Optimize for AWS

Instances disappear
Zones fail
Regions become unstable
Network is lossy
Customer requests bounce between regions

Failures happen and we test all the time
EVCache Use @ Netflix

Hundreds of terabytes of data
Trillions of ops / day
Tens of billions of items stored
Tens of millions of ops / sec
Millions of replications / sec
Thousands of servers
Hundreds of instances per cluster
Hundreds of microservice clients
Tens of distinct clusters
3 regions
4 engineers
Architecture

Application

Client Library

Client Library

Eureka (Service Discovery)

Memcached

Prana (Sidecar)

EVCache Server

EVCache Client
Architecture

us-west-2a

Client

us-west-2b

Client

us-west-2c

Client
Reading (get)

us-west-2a

us-west-2b

us-west-2c

Client

Primary

Secondary
Writing (set, delete, add, etc.)

us-west-2a

Client

us-west-2b

Client

us-west-2c

Client
```java
EVCache evCache = new EVCache.Builder()
    .setAppName("EVCACHE_TEST")
    .setCachePrefix("prefix")
    .setDefaultTTL(900) // 15 minutes
    .build();

evCache.set("key", "value");
evCache.get("key");
evCache.delete("key");
```
Use Case: Lookaside Cache

Data Flow
Use Case: Transient Data Store

[Diagram of application and client library connections over time]
Use Case: Primary Store

Online Application
Client Library

EVCache Client

Data Flow

Online Services

Offeline Services

Offline / Nearline Precomputes for Recommendations
Use Case: Versioned Primary Store

Online Application

Client Library

EVCache Client

Archaius (Dynamic Properties)

Offline / Nearline Precomputes for Recommendations

Control System (Valhalla)

Online Services

Offline Services

Data Flow
Use Case: High Volume && High Availability

Application

Client Library

EVCache In-memory
Remote
Ribbon Client

Optional

Compute & Publish on schedule

Data Flow
Pipeline of Personalization

Data Flow

Compute A → Online 1 → Compute D → Compute C → Compute E

Online 1 → Compute B → Online 2

Online Services

Offline Services
Polyglot Clients

Java Client
- APP
  - EVCache

Local Proxy
- APP
  - Prana
  - EVCache

HTTP Proxy
- APP
  - HTTP
  - EVCache

Memcached Proxy
- APP
  - Memcached
  - EVCache
Use Case: Polyglot
Additional Features

Kafka
- Global data replication
- Consistency metrics

Key Iteration
- Cache warming
- Lost instance recovery
- Backup (and restore)
Additional Features (Kafka)

Global data replication
Consistency metrics
Cross-Region Replication

Region A

1. mutate

2. send metadata

3. poll msg

4. get data for set

5. https send msg

Region B

6. mutate

7. read

1. APP

2. Kafka

3. Repl Relay

4. Repl Proxy

5. Repl Proxy

6. Repl Relay

7. Kafka
Region A

1. APP mutate to Consistency Checker
2. Send metadata from APP to Consistency Checker
3. Poll message from Kafka to Consistency Checker
4. Pull data from Consistency Checker to Atlas (Metrics Backend)
5. Report from Consistency Checker to Client Dashboards
Additional Features (Key Iteration)

Cache **warming**
Lost instance **recovery**
Backup (and restore)
Cache Warming

- Application
- Client Library
- EVCache Client
- S3
- Data Flow
- Control Flow
- Metadata Flow
- Control
- Cache Warmer (Spark)
Lost Instance Recovery

Application

Client Library

EVCache Client

Zone A

Control

Zone B

Data Flow

Partial Data Flow

Control Flow

Metadata Flow

S3

Cache Warmer (Spark)
Backup (and Restore)
What is EVCache bad at?

Perfect consistency
High security
Hot keys
Very large objects
Moneta

Next-generation EV Cache server
Moneta

Moneta: The Goddess of Memory
Juno Moneta: The Protectress of Funds for Juno

• Evolution of the EVCache server
• Cost optimization
• EVCache on SSD
• Ongoing lower EVCache cost per stream
• Takes advantage of global request patterns
Old Server

- Stock Memcached and Prana (Netflix sidecar)
- All data stored in RAM in Memcached
- Expensive with global expansion / N+1 architecture
Optimization

- **Global data** means many copies
- Access patterns are heavily region-oriented
- In one region:
  - Hot data is used often
  - Cold data is almost never touched
- Keep hot data in **RAM**, cold data on **SSD**
- Size **RAM** for working set, **SSD** for overall dataset
**New Server**

- Adds Rend and Mnemonic
- Still looks like Memcached
- Unlocks cost-efficient storage & server-side intelligence

```
Rend

Memcached (RAM)

Mnemonic (SSD)

Prana
```

![Diagram showing external and internal connections]
Rend

go get github.com/netflix/rend
Rend

- **High-performance** Memcached proxy & server
- **Written in** Go
  - Powerful concurrency primitives
  - Productive and fast
- **Manages the L1/L2 relationship**
- **Tens of thousands** of connections
Rend

- Modular to allow future changes / expansion of scope
  - Set of libraries and a default `main()`
- Manages connections, request orchestration, and communication
- Low-overhead metrics library
- Multiple orchestrators
- Parallel locking for data integrity
Mnemonic

Open source soon™
Mnemonic

- Manages data storage on SSD
- Reuses Rend server libraries
- Maps Memcached ops to RocksDB ops

<table>
<thead>
<tr>
<th>Library/Component</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rend Server Core Lib (Go)</td>
</tr>
<tr>
<td>Mnemonic Op Handler (Go)</td>
</tr>
<tr>
<td>Mnemonic Core (C++)</td>
</tr>
<tr>
<td>RocksDB (C++)</td>
</tr>
</tbody>
</table>
Why RocksDB?

- Fast at medium to high write load
  - Goal was 99% read latency below 20ms
- Log-Structured Merge minimizes random writes to SSD
  - Writes are buffered in memory
- Immutable Static Sorted Tables
How we use RocksDB

- Records sharded across many RocksDBs per instance
  - Reduces number of files checked, decreasing latency
- FIFO "Compaction"
  - More suitable for our precompute use cases
- Bloom filters and indices pinned in memory
  - Trade L1 space for faster L2
FIFO Limitation

- FIFO compaction not suitable for all use cases
  - Very frequently updated records may push out valid records
- Future: custom compaction or level compaction
Moneta in Production

- Serving all of our *personalization* data
- Rend runs with two ports:
  - One for *standard* users (read heavy or active data management)
  - Another for *async and batch* users: Replication and Precompute
- Maintains working set in RAM
- Optimized for *precomputes*
  - Smartly replaces data in L1
Moneta Performance in Production

**Get:** 229 µs (139 µs) avg

Percentiles:
- 50ᵗʰ: 140 µs (113 µs)
- 75ᵗʰ: 199 µs (139 µs)
- 90ᵗʰ: 318 µs (195 µs)
- 95ᵗʰ: 486 µs (231 µs)
- 99ᵗʰ: 1.67 ms (508 µs)
- 99.9ᵗʰ: 9.68 ms (1.97 ms)

**Set:** 367 µs (200 µs) avg

Percentiles:
- 50ᵗʰ: 227 µs (172 µs)
- 75ᵗʰ: 301 µs (202 µs)
- 90ᵗʰ: 411 µs (243 µs)
- 95ᵗʰ: 579 µs (295 µs)
- 99ᵗʰ: 3.25 ms (502 µs)
- 99.9ᵗʰ: 18.5 ms (4.73 ms)

Latencies: peak (trough)
70% Reduction in cost*
Open Source

https://github.com/netflix/EVCache

https://github.com/netflix/rend
Thank You

@sgmansfield
smansfield@netflix.com
techblog.netflix.com
Failure Resilience in Client

- Operation Fast Failure
- Tunable Retries
- Operation Queues
- Tunable Latch for Mutations
- Async Replication through Kafka