Distributed Caching With Redis For SpringBoot Apps

- Sasi Peri

Safe Harbor Statement

This session is intended to outline the general direction and is **my view** intended for information purposes only. You are encouraged to do **your own thorough analysis and research** before the use of the technology in your production grade applications.

Session Overview

- **Part - I**
  1. Distributed Caching Overview
  2. Why Redis?
  3. Popular Caching Patterns; Use Cases; Common Pit-falls
  4. Demo: Implement few R/W cache patterns in SpringBootApp (with Spring-Data-Redis)
     - RDMS (mysql) as primary source
     - 3rd party api as primary source
- **Part - 2**
  1. Security (ACLs)
  2. Introduce security into SpringBoot Client built in "Part-1#4"
  3. Quick look into modules RedisJSON and RediSearch
- **Part - 3**
  1. Redis Cache to store Web Session Data
  2. Demo with SpringBoot App (with Spring-Session-Data-Redis)
  3. Cache Eviction and TTL

What more?

- Where is the presentation and/any material ? (blogs? Website?)
- Where is final code? (e.g. Github)
- Redis reference links?
- How do I get started?

Tools and Software I will use

- Docker
- Redis (regular)
- Redis Stack Server (for additional modules)
- RedisInsight (GUI)
- Redis CLI and CLI Monitor
- Java 17
- SpringBoot (3.0.0-SNAPSHOT)
- Quick look at Redis Cloud (will not use in demo)

PART -1

Distributed Caching?

Cache that can be shared by multiple apps or multiple instances of same app, lives external to the app.

With Micro Services going main stream, it’s even more of a need to share the cache, to be able to auto-scale/auto-recover. *(Building "stateless process” → Factor - 6 of 12 factor apps)*

Some expectations and possibilities?

- Can span multiple servers, regions, zones, nodes?
- Can grow elastically?
- Highly scalable and available?
- High performance, Data Integrity
- And the ** **Brewer's** CAP theorem** **&&** How well the 3rd factor is handled?

- ….. Needs are endless…

Why Redis

Redis designed to be a NOSQL database, with memory first.

It can do many other things.

Beyond caching …

- NOSQL DB
- Streams & Pub-Sub
- Search capabilities (RediSearch)
- As document store (e.g. RedisJSON)
- As Graph DB (RedisGraph)
- RedisTimeSeries s(monitor)
- RedisAI (AI & TensorFlows)
- …. And more

How does that matter if I use it for distributed caching only?

Popular Caching Needs & Patterns

1. Read
    1. Read-Through (inline)
    2. Look-Aside
2. Write
    1. Write-Through (inline)
    2. Write-Behind
3. Session Caching

Details on subtle differences in the various patterns

- ✓ VMWare documentation on Inline and look-aside cache types for Read and Write
- ✓ Spring Documentation

Demo time!

What we will do?

In a SpringBoot Rest API

1. Implement a Read cache pattern
   - When MySQL as main data store
   - 3rd party API as source
2. Implement a Write cache pattern

Client: Cache Providers

Popular java cache managers

1. Lettuce
2. Redisson (OSS & JSR107 compliant)
3. Jedis

Information about Redis Cache pro

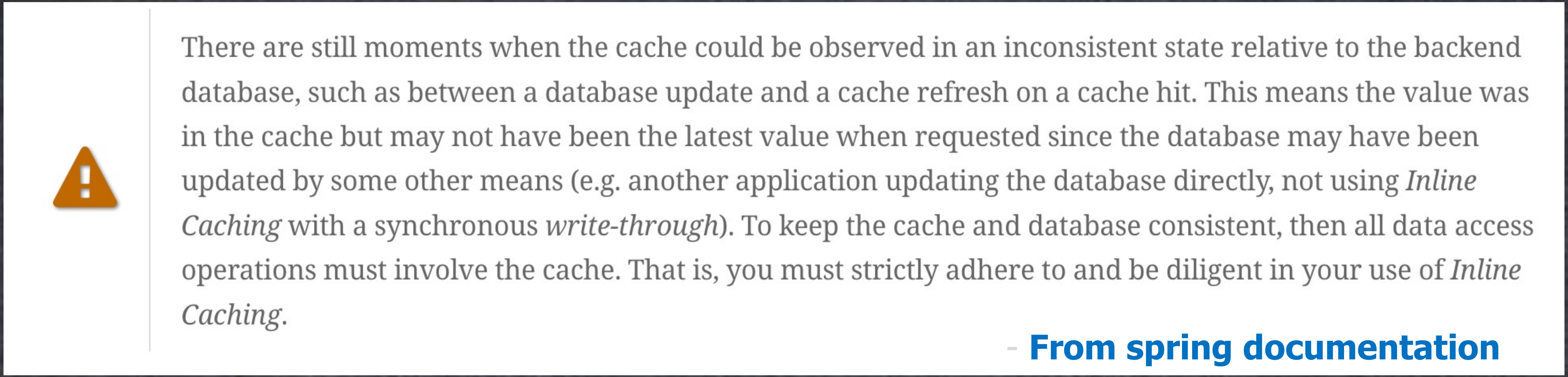
Points worth noting

The fundamental premise of caching is, when *given the same arguments, if a service call yields the same results* every time, and *source is not changing too often*.

Examples:

- Searching for a customer in the *Database* using an account returns same customer.
- Pulling a doctor info from *3rd party API*, using and NPI, not changing frequently, pulls the same doctor every time.

Pit-Falls



There are still moments when the cache could be observed in an inconsistent state relative to the backend database, such as between a database update and a cache refresh on a cache hit. This means the value was in the cache but may not have been the latest value when requested since the database may have been updated by some other means (e.g. another application updating the database directly, not using *Inline Caching* with a synchronous *write-through*). To keep the cache and database consistent, then all data access operations must involve the cache. That is, you must strictly adhere to and be diligent in your use of *Inline Caching*.

- **From spring documentation**

Examples:

- Legacy apps updating database directly
- Human intervention and manual updates
- Batch and/or house keeping jobs.
- And more…

Few Ideas :

- Proper expiry, eviction policies
- Hot loading the cache

Pit-Falls

Examples:

- Legacy apps updating database directly
- Human intervention and manual updates
- Batch and/or house keeping jobs.
- And more…

Few Ideas to work around:

- Proper expiry, eviction policies
- Hot loading

PART-2

Session Overview

- **Part - 1**
  1. Distributed Caching Overview
  2. Why Redis?
  3. Popular Caching Patterns; Use Cases; Common Pit-falls
  4. Demo: Implement few R/W cache patterns in SpringBootApp (with Spring-Data-Redis)
     - RDMS (mysql) as primary source
     - 3rd party api as primary source
- **Part - 2**
  1. Security (ACLs)
  2. Introduce security into SpringBoot Client built in "Part-1#4"
  3. Quick look into modules RedisJSON and RediSearch
- **Part - 3**
  1. Redis Cache to store Web Session Data
  2. Demo with SpringBoot App (with Spring-Session-Data-Redis)
  3. Cache Eviction and TTL

Security – Server side

ACL Based Security

Example

- "user admin on nopass ~* &* +@all"
- Parts: (1) user (2) admin (3)on (4)nopass (5) ~* (6) &* (7) +@all
    1. User
    2. User name
    3. Active or inactive (on/off)
    4. Password (nopass is no password set; Or password supplied with (supplied with prefix = >)
    5. Keys (prefix = ~)
    6. Streams (pub/sub channel access; prefix &)
    7. Commands (Type of commands R/W and such; prefix = +)

Security – Client side

Managed via few application properties

RedisJSON & RediSearch

Demo time!

PART -3

Session Overview

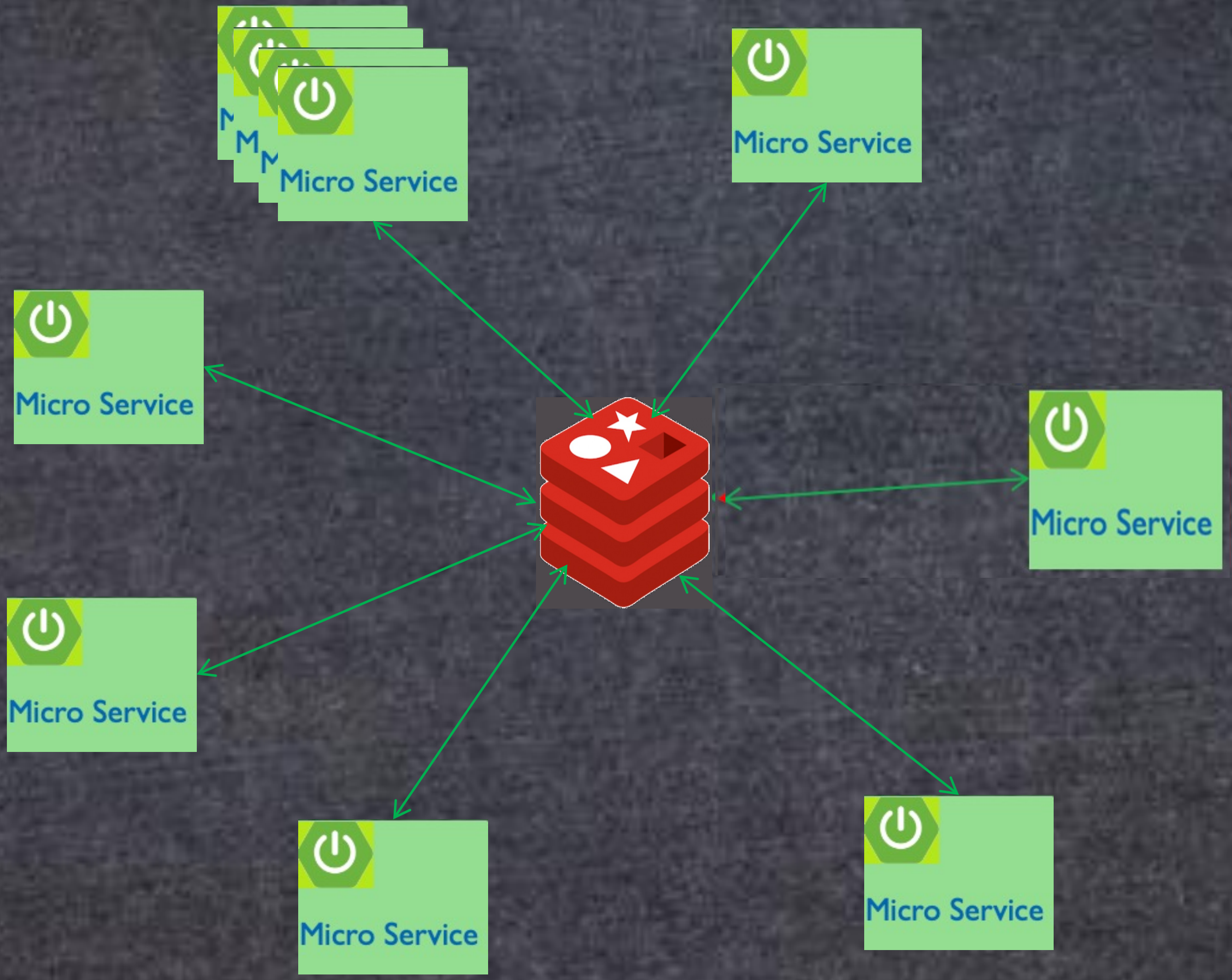
- **Part - I**
  1. Distributed Caching Overview
  2. Why Redis?
  3. Popular Caching Patterns; Use Cases; Common Pit-falls
  4. Demo: Implement few R/W cache patterns in SpringBootApp (with Spring-Data-Redis)
     - RDMS (mysql) as primary source
     - 3rd party api as primary source
- **Part - 2**
  1. Security (ACLs)
  2. Introduce security into SpringBoot Client built in "Part-I#4"
  3. Quick look into modules RedisJSON and RediSearch
- **Part - 3**
  1. Redis Cache to store Web Session Data
  2. Demo with SpringBoot App (with Spring-Session-Data-Redis)
  3. Cache Eviction and TTL

What’s session data?

Data available with in the scope of a user session

Why cache session data?

- Fast access to session state
- Business continuity and resiliency to failures



High Availability

Horizontal Scaling

Zero Downtime Updates

- Why is web session cache is more tricky?

- How does Spring Session and Redis make it easy (spring-session-data-redis)

Demo time!

- Cache eviction policies that Redis supports
- Spring client
  - ✓ Time-To-Live (Part-1/API)
  - ✓ Session cleanup (Part-3/Webapp)

Demo time!

Links and References

Code and Deck

- ✓ Code/GitHub for final code Part1, Part2 & Part3: source code
- ✓ Presentation on my home page (by GithubPages): sasiperi.io

Reference Links

- ✓ Why your MySQL needs Redis?
- ✓ What happens when my Redis runs out of memory?
- ✓ VMWare doc on inline, look-aside, write-behind caches
- ✓ Spring Data Redis
- ✓ Spring Session Data Redis
- ✓ HttpSession with Redis. (Srping docs)
- ✓ Samples and code. (Spring/Redis)

Other Redis Ref links

- ✓ Try Redis Cloud for free
- ✓ Watch this video on the benefits of Redis Cloud over other Redis providers
- ✓ Redis Developer Hub - tools, guides, and tutorials about Redis
- ✓ RedisInsight Desktop GUI

Need help getting started?

Here are more links to documents, tutorials and videos

1. Sign up for a free Redis Cloud account using this link and use the Redis Stack database in the cloud.
    1. Create a free Redis Stack database, or use TIGER200 to get a $200 credit on a paid account with a larger storage **(Use this if you have a large dataset)**
    2. Redis Stack : Redis Stack on Docker
2. Based on the language/framework you want to use, you will find the following client libraries:
    1. Redis OM .NET (C#)
        1. Watch this getting started video
        2. Follow this getting started guide
    2. Redis OM Node (JS)
        1. Watch this getting started video
        2. Follow this getting started guide
    3. Redis OM Python
        1. Watch this getting started video
        2. Follow this getting started guide
    4. Redis OM Spring (Java)
        1. Watch this getting started video
        2. Follow this getting started guide
3. The above videos and guides should be enough to get you started in your desired language/framework. From there you can expand and develop your app. Use the resources below to help guide you further.
    1. Developer Hub - Our main developer page where you can find information on building using Redis with sample projects, guides, and tutorials.
    2. Redis Stack getting started page - Lists all the Redis Stack features. From there you can find relevant docs and tutorials for all the capabilities of Redis Stack.
    3. Redis Rediscover - Provides use-cases for Redis as well as real-world examples and educational material
    4. RedisInsight - Desktop GUI tool - Use this to connect to Redis to visually see the data. It also has a CLI inside it that lets you send Redis CLI commands. It also has a profiler so you can see commands that are run on your Redis instance in real-time
4. Youtube Videos
    1. Official Redis Youtube channel
    2. Redis Stack videos - Help you get started modeling data, using Redis OM, and exploring Redis Stack
    3. Redis Stack Real-Time Stock App from Ahmad Bazzi
    4. Build a Fullstack Next.js app with Fireship.io
    5. Microservices with Redis Course by Scalable Scripts on freeCodeCamp
5. Public datasets - Contains **1000s of public datasets**
6. Redis datasets - Contains some pre-made Redis Datasets
7. Random Data API - Contains APIs to get random data from several different types of models
8. Redis Ebooks - Contains several free e-books

Thank You!