Cache - Key of Performance

Ghergu
Nicolae-Marius

Single-Level Cache, Distributed Cache & Performance Testing

CONTENTS

01

What is Cache?

Single-Level Cache and Distributed Cache

Definitions

Single-Level Cache

Refers to a caching mechanism where data is stored and accessed from a single cache instance. In this type of cache, all cached data is stored in a single location, typically within the memory of the application. When data is requested, it is first checked in the cache, and if found, it is return directly.

Distributed Cache

Is a caching mechanism that spans across multiple nodes and servers in a distributed system. Instead of relying on a single cache instance, the data is partitioned and stored in multiple cache nodes.
Benefits? Scalability, High availability, Improved performance and data consistency.

02

Cache Solutions

EhCache and Caffeine for Spring Boot Applications

EhCache

EhCache is a widely used, open-source Java-based cache. It features memory and disk stores, listeners, cache loaders, RESTful and SOAP APIs and other very useful features.

Pro

- Widely used, making it a mature and stable caching solution
- Offers a comprehensive set of features, including support for distributed caching, caching annotations and cache event listeners
- Has good integration with popular frameworks like Spring and Hibernate
- Provides support for cache persistence, allowing cached data to be stored on disk or in database for durability.

Cons

- Configuration can be complex, especially when dealing with advanced features like distributed caching
- In certain scenarios, might introduce some performance overhead

Caffeine

Caffeine is a high-performance caching library for Java.

Pro

- Designed to be highly performant, offering fast in-memory caching with low latency and high throughput
- Has a simple API, making it easy to use and integrate into applications
- Provides various cache eviction strategies, allowing to automatically remove less frequently used or expired items from the cache
- Offers flexibility in configuring cache behavior and allows customization of eviction policies, cache loading and cache statistics.

Cons

- No built-in support for distributed caching
- While offers most of the essential caching features, it may lack some advanced features by Ehcache
- Does not have as many built-in integrations with popular frameworks compare

03

Microservices and Data-Intensive Apps

Real Problems and Examples

Microservices and Data-Intensive Apps

Dealing with requests

While microservices are sometimes a benefit for applications, other times can be painful.
Why? Because the amount of requests. You need to build a reliable, scalable and maintainable systems.

Traffic

Traffic means a lot. Actually, the whole thing. Every request can make a lot of traffic and maybe trigger other 20 applications in order to get the final data.

Problems

- Slow response time
- High resource utilization
- Database performance issues

Problems

- Suboptimal code and algorithms
- Network latency and bottlenecks
- A lot of traffic for resources

REAL PROBLEMS

Database DTO & Views

Let's say you want to get data from a lot of databases and you increased performance using views. But is that enough?

Unavailable services

Let's say a microservice needs to restart for some seconds. In those seconds, we loss availability.

Big Complexity of Code

Let's say you need to compute a lot of data in order to get the result. For example, a csv and re-build again and again csv, either if you only need another field for that object.

Memory usage

While we have to compute data, we use memory. For example, some programming languages are well-known for memory usage.

03

Hazelcast

And how to boost your app by 1000% by Distributed Cache

Hazelcast

Open-source, distributed, and highly scalable in-memory data grid platform.

Pro

- Offers a robust distributed caching solution, allowing to cache data across multiple nodes in a cluster
- Is designed to scale horizontally, allowing to add more nodes to the cluster when application's load increases
- Provides built-in fault-tolerance mechanisms
- Offers distributed computing capabilities through its distributed data structures and APIs
- Supports publish-subscribe messaging and eventing mechanism

Cons

- Provides comprehensive documentation and resources
- Comes with inherent complexities, including managing data consistency and network latency
- Stores data in-memory and it doesn't provide built-in support for persistence storage

03

Gatling

Performance testing using Java

Gatling

Open-source load testing tool used for performance testing and stress testing web applications. Provides a range of features and capabilities to simulate high loads and measure the performance of applications.

Pro

- Is designed to handle high loads and simulate thousands of concurrent users
- Leverages an asynchronous, non-blocking architecture that allows it to generate high level of virtual users without significant resource consumption
- Allows to define complex simulation scenarios using domain-specific language (DSL)
- Provides real-time metrics and details reports
- Integrates well with popular CI tools like Jenkins, Teamcity and Bamboo

Cons

- No Built-in browser simulation. It focuses on performance at the protocol level, making it less suitable for testing specific browser-based behaviors or client-side interactions

THANKS!

Do you have any questions?

DEMO TIME!!!

https://github.com/nicugnm/single-distributed-testing-presentation

Bibliography

- https://www.baeldung.com/cs/
- https://www.baeldung.com
- https://hazelcast.com/