INSTITUT
POLYTECHNIQUE
DE PARIS

A Key-Value Store from Scratch

CSC5004

Cloud Computing Infrastructures

Authors:

Nevena Vasilevska
Teodor Cvijovic

Introduction

During the lab session, a basic form of NoSQL data storage, specifically a key-value store (KVS) utilizing the JGroups Library in Java, was implemented. We will provide an overview of the fundamental features of the key-value store, the strategy employed for distributing keys among nodes, and a comparative analysis of the implemented solution with an alternative strategy. Source code of the implementation can be found in the following address: https://github.com/teodorcvijovic/distributed-kv-store .

01 The JGroups Library

To comprehend JGroups, the online tutorial was followed. The tutorial encompassed the implementation of SimpleChat for communication between views in a JChannel. Noteworthy sections of the code, such as connecting to a cluster and handling messages, were emphasized.

02 Implementation of the Key-Value Store

The ConsistentHash class was completed to encapsulate a strategy based on consistent hashing. This technique assigns positions on a hash ring to servers, allowing scalability without a system-wide impact. The implementation involved utilizing the Strategy pattern and creating strategy objects for each node. The StoreImpl class extended ReceiverAdapter, and methods for handling local and remote calls were developed.

03 Handling Remote Calls

A field named 'workers' was introduced to StoreImpl, initialized, and ExecutorService was used for handling remote calls in a separate thread. The send method was implemented to push a command to a destination node. A private CmdHandler was created to handle received messages and execute commands. The call method was implemented to manage both local and remote calls.

04 Concurrent Access and Data Migration

The management of remote and local calls was merged into a single execute method, synchronized to prevent concurrent accesses. ConcurrentHashMap was employed for thread safety in data access. A data migration mechanism was proposed and implemented upon a view change, considering the addition of a node. The implementation was validated with tests.

05 Strategy Comparison

A round-robin strategy for data distribution was implemented, and a test was created to compare the advantages of consistent hashing over round-robin. The test involved inserting keys with consistent hashing, adding a node, and measuring the number of migrated keys. Consistent hashing demonstrated superior performance in terms of the number of keys migrated compared to round-robin.

Conclusion

The achievements of the lab session were summarized, emphasizing the successful implementation of a key-value store, performance testing with different strategies, and ensuring the functionality of each system component.