Hazelcast

by Alexander Edinger, Jonas Eppard and Fernand Hoffmann

TODO: General Introduction

1 Key-Value-Stores

TODO:

1.1 History of Key-Value-Stores

TODO: Change

1.2 Working principle of Key-Value-Stores

TODO:

1.3 Comparison of Key-Value-Stores to relational databases

TODO:

2 Hazelcast in theory

Hazelcast, or formerly known as Hazelcast IMDG, is an In-Memory Data grid written in Java by the Hazelcast corporation. It is a cluster storage and combines "Data at Rest" with "Data in Motion" by providing a streaming architecture for dynamic data together with low-latency storage for static data. (Hazelcast, 2022; "Hazelcast IMDG — Leading open source in-memory data grid • Hazelcast", n.d.)

2.1 Architecture of Hazelcast

At the core of the Hazelcast platform stands the distributed architecture of cluster nodes with the streaming engine and the low-latency storage. A cluster consists of at least one node. The streaming engine is in the form of a Kappa-Architecture which, compared to a Lambda-Architecture, provides the same methodology for data processing. The low-latency storage provides availability and partition tolerance with possible consistency through the use of the CP-subsystem. Hazelcast offers different possibilities for a client to connect to the cluster through the use of APIs and Connectors. Implementations exist for multiple languages, including Java, C++, Node.js, Python and Go. ("Getting Started with a Hazelcast Client", n.d.; Hazelcast, 2022) Encapsulating the Connectors and the cluster are the security features of Hazelcast for clients and nodes. This includes the authentication of new nodes and clients, as well as the encryption of traffic between the nodes or between a client and the cluster.

2.2 Hazelcast in CAP-Theorem

The CAP-Theorem states, that a distributed system can not be partition tolerant, consistent and highly available at the same time, but can only fulfil two of the three properties. A 2010 article further adds, that partition tolerance cannot be sacrificed, because sacrificing partition tolerance would require a network to never drop any messages, which in reality cannot exist. (Brewer, 2000; "You Can't Sacrifice Partition Tolerance", 2010)

Hazelcast is by default an AP-system, favouring availability over consistency. Since each client interacts with the cluster through a gateway node and node failures can happen in a distributed system, Hazelcast replicates the data on backup nodes to ensure availability. ("CAP Theorem", n.d.; "HazelVision Episode 06 — CAP Theorem — YouTube", n.d.)

Some distributed data structures, for example a Lock or a Semaphore, require strong consistency rather than high availability. For such cases, Hazelcast offers a CP-subsystem, which is built upon the cluster and allows for consistent storage of specified distributed objects. The subsystem is based on the RAFT (see Woos et al. (2016)) consensus algorithm, through which the participating nodes in the subsystem can reach a consensus on correct data by periodic heartbeats and votings. (“CAP Theorem”, n.d.; “CP Subsystem”, n.d.; Woos et al., 2016)

Since Hazelcast is an In-Memory Data Grid, no persistence happens by default. To strengthen consistency of data structures, persistence can be configured for nodes in the CP-subsystem to persist the data to a non-volatile storage. (“CP Subsystem”, n.d.)

2.3 Hazelcast is BASE

The acronym BASE stands for Basically Available, Soft-State and Eventual Consistency and is used to describe a system with those properties (Brewer, 2000). Hazelcast fits all three properties due to the following reasons: Due to Hazelcast being an AP-system, it prioritizes availability and can thus be considered basically available. Through periodic heartbeats between the cluster nodes, a failure in nodes can be detected and data from backup nodes can be fetched. (“Failure Detector Configuration”, n.d.)
Due to the high availability offered by Hazelcast, an exact state of the cluster cannot be determined to a certain point in time. This is further amplified when choosing one-phase commits for transactions, where the commit log is not copied to other nodes. Consequently, a node in the process of writing data can be interrupted due to a node failure and leave the system in an inconsistent state. (“Creating a Transaction Interface”, n.d.)

To prevent increasing entropy in the cluster, Hazelcast synchronizes the nodes in periodic intervals. The primary node, the node which is delegated to store the data, sends a summary of the data to its backup nodes. The backup nodes compare the data with their version and in the case of an inconsistency, the synchronization process is triggered. (“Hazelcast’s Replication Algorithm”, n.d.)

3 Hazelcast in Practice

TODO:

3.1 Installation of Hazelcast

TODO:

3.2 Usage and API of Hazelcast

TODO:

3.3 Examples in Hazelcast

TODO:

4 Evaluation of Hazelcast

TODO:

4.1 Advantages of Hazelcast

TODO:

4.2 Disadvantages of Hazelcast

TODO:

4.3 Lessons learned

TODO:

5 Conclusion and Recommendations

TODO:

References

Brewer, D. E. A. (2000). Towards robust distributed systems.

CAP theorem [Hazelcast]. (n.d.). Retrieved April 10, 2023, from https://hazelcast.com/glossary/cap-theorem/

CP subsystem. (n.d.). Retrieved April 10, 2023, from https://docs.hazelcast.com/imdg/4.2/cp-subsystem/cp-subsystem

Creating a transaction interface. (n.d.). Retrieved April 10, 2023, from https://docs.hazelcast.com/imdg/4.2/transactions/creating-a-transaction-interface

Failure Detector Configuration. (n.d.). Retrieved April 12, 2023, from https://docs.hazelcast.com/imdg/4.2/clusters/failure-detector-configuration

Getting Started with a Hazelcast Client. (n.d.). Retrieved April 12, 2023, from https://docs.hazelcast.com/hazelcast/latest/clients/hazelcast-clients

Hazelcast. (2022, November 3). *What is the hazelcast platform? | hazelcast explainer*. Retrieved April 10, 2023, from https://www.youtube.com/watch?v=UpIgHzKbMp0

Hazelcast IMDG — leading open source in-memory data grid • hazelcast [Hazelcast]. (n.d.). Retrieved April 10, 2023, from https://hazelcast.org/imdg/

Hazelcast's replication algorithm. (n.d.). Retrieved April 10, 2023, from https://docs.hazelcast.com/hazelcast/5.0/consistency-and-replication/replication-algorithm

HazelVision episode 06 — CAP theorem — YouTube. (n.d.). Retrieved April 10, 2023, from https://www.youtube.com/watch?v=6DHxX_-fA8Y&list=PLhAaDrEJmCb3lpFQ6kDOkf_9wSdRp1cgx&index=10

Woos, D., Wilcox, J. R., Anton, S., Tatlock, Z., Ernst, M. D., & Anderson, T. (2016). Planning for change in a formal verification of the raft consensus protocol. *Proceedings of the 5th ACM SIGPLAN Conference on Certified Programs and Proofs*, 154–165. https://doi.org/10.1145/2854065.2854081

You can't sacrifice partition tolerance [Codahale.com]. (2010, October 7). Retrieved April 10, 2023, from https://codahale.com//you-cant-sacrifice-partition-tolerance/