docs.hazelcast.com

What is Hazelcast IMDG?

5-6 minutes

Hazelcast IMDG is an open-source distributed in-memory object store supporting a wide variety of data structures.

You can use Hazelcast IMDG to store your data in RAM, spread and replicate it across your cluster of machines, and perform computations on it. Replication gives you the resilience to failures of cluster members.

Hazelcast IMDG is highly scalable and available. Distributed applications can use it for distributed caching, synchronization, clustering, processing, pub/sub messaging, etc.

It is implemented in Java language and has clients for Java, C++, .NET, REST, Python, Go and Node.js. Hazelcast IMDG also speaks Memcached and REST protocols. It plugs into Hibernate and can easily be used with any existing database system.

Hazelcast IMDG makes distributed computing simple by offering distributed implementations of many developer-friendly interfaces. For example, the Map interface provides an In-Memory Key Value store which confers many of the advantages of NoSQL in terms of developer friendliness and developer productivity.

Your cloud-native applications can easily use Hazelcast IMDG. It is flexible enough to use as a data and computing platform out-of-the-box or as a framework for your own cloud-native applications and microservices.

Hazelcast IMDG is designed to be lightweight and easy to use. Since it is delivered as a compact library (JAR) and has no external dependencies other than Java, it easily plugs into your software solution and provides distributed data structures and computing utilities.

It is designed to scale up to hundreds of members and thousands of clients. When you add new members, they automatically discover the cluster and linearly increase both the memory and processing capacity. The members maintain a TCP connection between each other and all communication is performed through this layer. Each cluster member is configured to be the same in terms of functionality. The oldest member (the first member created in the cluster) automatically performs the data assignment to cluster members. If the oldest member dies, the second oldest member takes over.

> You can come across with the term "master" or "master member" in some sections of this manual. They are used for contextual clarification purposes; please remember that they refer to the "oldest member" which is explained in the above paragraph.

Hazelcast IMDG offers simple scalability, partitioning (sharding), and re-balancing out-of-the-box. It does not require any extra coordination processes. NoSQL and traditional databases are difficult to scale out and manage. They require additional processes for coordination and high availability. With Hazelcast IMDG, when you start another process to add more capacity, data and backups are automatically and evenly balanced.

Hazelcast's Distinctive Strengths

- It is open source.
- It is only a JAR file. You do not need to install software other than

Java.

- Hazelcast IMDG stores everything in-memory (RAM). It is designed to perform fast reads and updates.
- Hazelcast IMDG is peer-to-peer; there is no single point of failure in a Hazelcast IMDG cluster; each member in the cluster is configured to be functionally the same. They all store equal amounts of data and do equal amounts of processing. You can embed Hazelcast IMDG in your existing application or use it in client and server mode where your application is a client to Hazelcast members.
- When the size of your memory and compute requirements increase, new members can be dynamically joined to the Hazelcast IMDG cluster to scale elastically.
- Data is resilient to member failure. Data backups are distributed across the cluster. This is a big benefit when a member in the cluster crashes as data is not lost. Hazelcast keeps the backup of each data entry on multiple members. On a member failure, the data is restored from the backup and the cluster continues to operate without downtime.
- Members are always aware of each other unlike in traditional key-value caching solutions.
- Hazelcast provides out-of-the-box distributed data structures.

  Finally, Hazelcast has a vibrant open source community enabling it to be continuously developed.

  Hazelcast is a fit when you need:

- analytic applications requiring big data processing by partitioning the data
- to retain frequently accessed data in the grid
- a cache, particularly an open source JCache provider with elastic distributed scalability

- a primary data store for applications with utmost performance, scalability and low-latency requirements
- an In-Memory NoSQL Key Value Store
- publish/subscribe communication at highest speed and scalability between applications
- applications that need to scale elastically in distributed and cloud environments
- a highly available distributed cache for applications
- an alternative to Coherence and Terracotta.