Hazelcast for Java Developers

FUAD MALIKOV
CO-FOUNDER
What Is Hazelcast?

Hazelcast is a distributed, highly available and scalable Open Source In-Memory Data Grid.
In Memory Data Grid

In Memory Data Storage + In Memory Data Messaging + In Memory Data Computing
import java.util.HashMap;
import java.util.Map;

public static void main(String[] args) {
    Map<Integer, String> map = new HashMap<>();
    map.put(1, "Paris");
    map.put(2, "London");
    map.put(3, "San Francisco");

    String oldValue = map.remove(2);
}

```java
import java.util.concurrent.ConcurrentHashMap;
import java.util.concurrent.ConcurrentMap;

public static void main(String[] args) {
    ConcurrentMap<Integer, String> map = new ConcurrentHashMap<>();
    map.put(1, "Paris");
    map.put(2, "London");
    map.put(3, "San Francisco");

    String oldValue = map.remove(2);
}
```
import java.util.concurrent.ConcurrentMap;
import com.hazelcast.core.Hazelcast;
import com.hazelcast.core.HazelcastInstance;

public static void main(String[] args) {
    HazelcastInstance h = Hazelcast.newHazelcastInstance();

    ConcurrentMap<Integer, String> map = h.getMap("myMap");
    map.put(1, "Paris");
    map.put(2, "London");
    map.put(3, "San Francisco");

    String oldValue = map.remove(2);
}
Ecosystem Traction

Dozens of Commercial and Open Source Projects Embed Hazelcast
Select Customers by Industry

- HIGH-TECH
  - Crunchbase
  - Doğuş Teknoloji
  - Cambridge Assessment
  - mediocean
  - MuleSoft
  - Schneider Electric
  - British Gas
  - Allianz
  - USAA
  - CGI
  - HARRIS
  - SpinSci
  - Level(3)
  - Ericsson
  - Revenue
  - Deutsche Bank
  - Morgan Stanley

- GAMING & ENTERTAINMENT
  - Disney Parks and Resorts
  - Playtech
  - Nielsen
  - QiG
  - The Limited
  - Vale
  - General Dynamics
  - Wallenius Wilhelmsen Logistics
  - Capital One
  - CBOE
  - BBVA
  - Mizuho
  - Credit Suisse
  - Bank of America

- INSURANCE
  - New York Life
  - British Gas
  - Allianz
  - USAA
  - CGI
  - HARRIS

- CONSUMER & ECOMMERCE
  - SleepIQ
  - boulanger

- TELECOMMUNICATIONS
  - HP
  - Verizon
  - AT&T
  - Barracuda

- LOGISTICS
  - FedEx
  - DHL
  - UPS

© 2016 Hazelcast Inc. Confidential & Proprietary
Demo
Why Hazelcast?

**Scale-out Computing** enables cluster capacity to be increased or decreased on-demand

**Resilience** with automatic recovery from member failures without losing data while minimizing performance impact on running applications

**Programming Model** provides a way for developers to easily program a cluster application as if it is a single process

**Fast Application Performance** enables very large data sets to be held in main memory for real-time performance
Rebalance Data on New Node
Distributed Maps

Fixed number of partitions (default 271)
Each key falls into a partition

\[ \text{partitionId} = \text{hash(keyData)} \% \text{PARTITION\_COUNT} \]

Partition ownerships are reassigned upon membership change
New Node Added
Migration
Migration

A

B

C

D

© 2015 Hazelcast Inc. Confidential & Proprietary
Migration
Migration

A

B

C

D

© 2015 Hazelcast Inc. Confidential & Proprietary
Migration Complete
Data Safety when Node Dies
Node Crashes

A

B

C

D

© 2015 Hazelcast Inc. Confidential & Proprietary
Backups Are Restored
Backups Are Restored

A

B

C

D

Crash
Backups Are Restored
Backups Are Restored
Backups Are Restored

A

B

C

D

© 2015 Hazelcast Inc. Confidential & Proprietary
Backups Are Restored

A

B

C

D
Backups Are Restored
Backups Are Restored

A

B

C

D

Crash
Recovery Is Complete

A

C

D
Deployment Strategies
Deployment Options

**Embedded Hazelcast**

- Great for early stages of rapid application development and iteration

**Client-Server Mode**

- Necessary for scale up or scale out deployments – decouples upgrading of clients and cluster for long term TCO
Easy API

// Creating a new Hazelcast node
HazelcastInstance hz = Hazelcast.newHazelcastInstance();

// Getting a Map, Queue, Topic, ...
Map map = hz.getMap("my-map");
Queue queue = hz.getQueue("my-queue");
ITopic topic = hz.getTopic("my-topic");

// Creating a Hazelcast Client
HazelcastInstance client = HazelcastClient.newHazelcastClient();

// Shutting down the node
hz.shutdown();
Feature Overview
public class SomeTestCase {

    private HazelcastInstance[] instances;

    @Before
    public void before() throws Exception {
        // Multiple instances on the same JVM
        instances = new HazelcastInstance[2];
        instances[0] = Hazelcast.newHazelcastInstance();
        instances[1] = Hazelcast.newHazelcastInstance();
    }

    @After
    public void after() throws Exception {
        Hazelcast.shutdownAll();
    }
}
IM Data Store (Caching) Features

Java Collection API: Map, List, Set, Queue

JCache

High Density Memory Store

Hibernate 2nd Level Cache

Web Session Replication: Tomcat, Jetty

Predicate API: Indexes, SQL Query

Persistence: Map/Queue Store & Loader. Write Behind/Through

Eviction

Near Cache

Transactions: Local & XA

WAN & DR Replication

Memcached Interface
interface com.hazelcast.core.IMap<K, V>
   extends java.util.Map, java.util.ConcurrentMap

HazelcastInstance hz = getHazelcastInstance();

//java.util.concurrent.ConcurrentMap implementation
IMap<String, User> hzMap = hz.getMap("users");
hzMap.put("Peter", new User("Peter", "Veentjer");

hzMap.putIfAbsent("Peter", new User("Peter", "Veentjer");

//Distributed Lock
hzMap.lock("Peter");

User peter = map.get("Peter");
public class MapStorage implements MapStore<String, User>, MapLoader<String, User> {

    // Some methods missing ...
    @Override public User load(String key) { return loadValueDB(key); }
    @Override public Set<String> loadAllKeys() { return loadKeysDB(); }
    @Override public void delete(String key) { deleteDB(key); }
    @Override public void store(String key, User value) {
        storeToDatabase(key, value);
    }
}

<map name="users">
    <map-store enabled="true">
        <class-name>com.hazelcast.example.MapStorage</class-name>
        <write-delay-seconds>0</write-delay-seconds>
    </map-store>
</map>
// Retrieve the CachingProvider which is automatically baced by
// the chosen Hazelcast server or client provider
CachingProvider cachingProvider = Caching.getCachingProvider();

// Create a CacheManager
CacheManager cacheManager = cachingProvider.getCacheManager();

// Cache<String, String> cache = cacheManager
//     .getCache( name, String.class, String.class );

// Create a simple but typesafe configuration for the cache
CompleteConfiguration<String, String> config =
    new MutableConfiguration<String, String>()
        .setTypes( String.class, String.class );
// Create and get the cache
Cache<String, String> cache = cacheManager.createCache( "example", config );

// Alternatively to request an already existing cache
Cache<String, String> cache = cacheManager.getCache( name, String.class, String.class );

// Put a value into the cache
cache.put( "world", "Hello World" );

// Retrieve the value again from the cache
String value = cache.get( "world" );

System.out.println( value );
High Density Caching

APIs
- JCache (ICache)
- Map (IMap)

Memory Stores
- Member
- Client (Near Cache)

On-Heap Memory Store
(Objects Stored as Objects)

High-Density Memory Store
(Objects Serialized and Stored as Bytes)

On-Heap SLAB Allocator*
On-Heap SLAB Allocator*

RAM in JVM Process

2-4GB
(Limited by Garbage Collection)

0-1TB
(Limited by Machine RAM)

* coming in 3.6
# On Heap Vs. High-Density Memory Management

<table>
<thead>
<tr>
<th>On Heap Memory</th>
<th>HD Memory v2</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>0 MB</strong></td>
<td><strong>3.3 GB</strong></td>
</tr>
<tr>
<td><strong>Native</strong></td>
<td><strong>Heap Storage</strong></td>
</tr>
<tr>
<td><strong>3.9 GB</strong></td>
<td><strong>0.6 GB</strong></td>
</tr>
<tr>
<td><strong>Heap Storage</strong></td>
<td><strong>Major GC</strong></td>
</tr>
<tr>
<td><strong>9 (4900 ms)</strong></td>
<td><strong>0 (0 ms)</strong></td>
</tr>
<tr>
<td><strong>Minor GC</strong></td>
<td><strong>356 (349 ms)</strong></td>
</tr>
<tr>
<td><strong>31 (4200 ms)</strong></td>
<td></td>
</tr>
</tbody>
</table>

**Example: On Heap Memory**

**Example: HD Memory v2**
IM Distributed Computing Feature

Java Concurrency API
(Lock, Semaphore, AtomicLong, AtomicReference, Executor Service, Blocking Queue)

Entry and Item Listeners

Entry Processor

Aggregators

Map/Reduce

Data Affinity

Continues Query

Map Interceptors

Delta Update
HazelcastInstance hz = getHazelcastInstance();

// Distributed Reentrant
Lock lock = hz.getLock("myLock");
lock.lock();
try {
  // Do something
} finally {
  lock.unlock();
}
public interface com.hazelcast.core.IExecutorService extends java.util.concurrent.ExecutorService

HazelcastInstance hz = getHazelcastInstance();

//java.util.concurrent.ExecutorService implementation
IExecutorService es = hz.getExecutorService("name");
es.executeOnAllMembers(buildRunnable());
es.executeOnKeyOwner(buildRunnable(), "Peter");
es.execute(buildRunnable());

Map<?, ?> futures = es.submitToAllMembers(buildCallable());
Future<?, ?> future = es.submitToKeyOwner(buildCallable(), "Peter");

es.submitToAllMembers(buildCallable(), buildCallback());
es.submitToKeyOwner(buildCallable(), "Peter", buildCallback());
HazelcastInstance hz = getHazelcastInstance();

Map users = hz.getMap("users");
JobTracker tracker = hz.getJobTracker("default");

KeyValueSource source = KeyValueSource.fromMap(users);
Job job = tracker.newJob(source);

ICompleteFuture future = job.mapper(new MyMapper())
   .reducer(new MyReducer())
   .submit();

Map result = future.get();
HazelcastInstance hz = getHazelcastInstance();

Map users = hz.getMap("users");

int sum = users.aggregate(
    Supplier.all((user) -> user.getSalary()),
    Aggregations.longSum()
);
IM Distributed Messaging Use Case

Hazelcast Node 1 ➔ MSG ➔ Hazelcast Distributed Topic Bus ➔ Hazelcast Node 2
          |                     |                      | Hazelcast Node 3
          |                     |                      |
          | Subscribes         | Delivers            |
          |                   |                      |
          |                   |                      |
          | Subscribes         | Delivers            |

Hazelcast Topic

© 2015 Hazelcast Inc. Confidential & Proprietary
IM Distributed Messaging Features

Queue

Topic (Pub/Sub)

Event Listeners

Ring Buffers
Queue API

```java
interface com.hazelcast.core.IQueue<E>
    extends java.util.concurrent.BlockingQueue

HazelcastInstance hz = getHazelcastInstance();

//java.util.concurrent.BlockingQueue implementation
IQueue<Task> queue = hz.getQueue("tasks");

queue.offer(newTask());
queue.offer(newTask(), 500, TimeUnit.MILLISECONDS);

Task task = queue.poll();
Task task = queue.poll(100, TimeUnit.MILLISECONDS);
Task task = queue.take();
```
public class Example implements MessageListener<String> {
    public void sendMessage {
        HazelcastInstance hz = getHazelcastInstance();
        ITopic<String> topic = hz.getTopic("topic");
        topic.addMessageListener(this);
        topic.publish("Hello World");
    }

    @Override
    public void onMessage(Message<String> message) {
        System.out.println("Got message: ", message.getMessageObject());
    }
}
Thank you

@fuadm, @hazelcast
hazelcast@googlegroups.com
http://www.hazelcast.com
http://github.com/hazelcast/hazelcast