In-Memory Computing SUMMIT EUROPE 2019

java.util.concurrent for distributed coordination

Ensar Basri Kahveci
Hazelcast

Hazelcast

The leading open source Java IMDG

- Distributed Java collections, concurrency primitives, messaging
- Caching, application scaling, distributed coordination 🎉

Hazelcast Cloud

- https://hazelcast.cloud

Hazelcast Jet: In-memory stream and fast batch processing

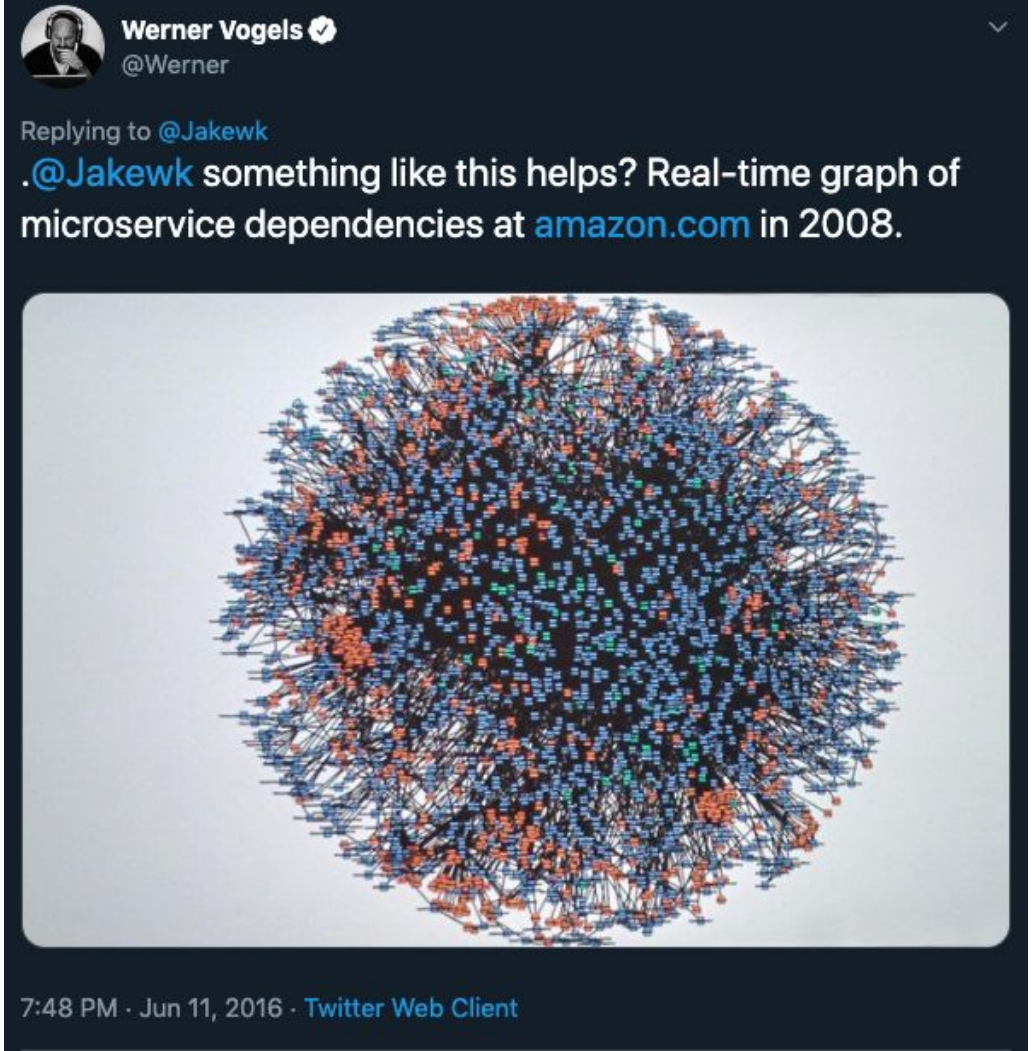
Agenda

What is distributed coordination?

How distributed coordination APIs evolved over time?

`java.util.concurrent.*` for distributed coordination

Demo on Hazelcast IMDG 3.12


Werner Vogels
@Werner
Replying to @Jakewk
.@Jakewk something like this helps? Real-time graph of microservice dependencies at amazon.com in 2008.
7:48 PM · Jun 11, 2016 · Twitter Web Client

#

Distributed Coordination

Leader election

Synchronization

Group membership

Configuration and metadata management


~~DO IT~~
~~YOURSELF~~

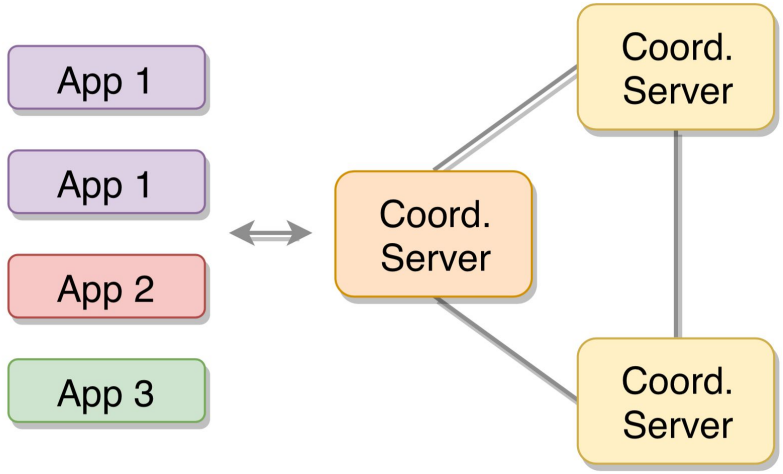
Distributed Coordination Systems

Consensus algorithms under the hood

CP with respect to CAP

Deployed as a central repository

APIs for coordination tasks



Google Chubby (Paxos)

Google Chubby (Paxos)

Apache ZooKeeper (ZAB)

Google Chubby (Paxos)

Apache ZooKeeper (ZAB)

etcd (Raft)

Chubby & ZooKeeper

```
/services
   /payment
   /product
      /photo
```

etcd

```
/services
/services/payment
/services/product
/services/product/photo
```

Chubby	ZooKeeper	etcd
Locking APIs	Recipes	

A Simple Locking Recipe for ZooKeeper

1. create an ephemeral znode “ `/lock` ”
2. if success, enter to the critical section
3. else, register a watch on “ `/lock` ”
4. when the watch is notified, i.e., the lock is released, retry step #1

Chubby

Locking APIs

ZooKeeper

Recipes

“Friends don't let friends write ZK recipes.”
Apache Curator Tech Notes #6

etcd

Leader election and distributed lock primitives

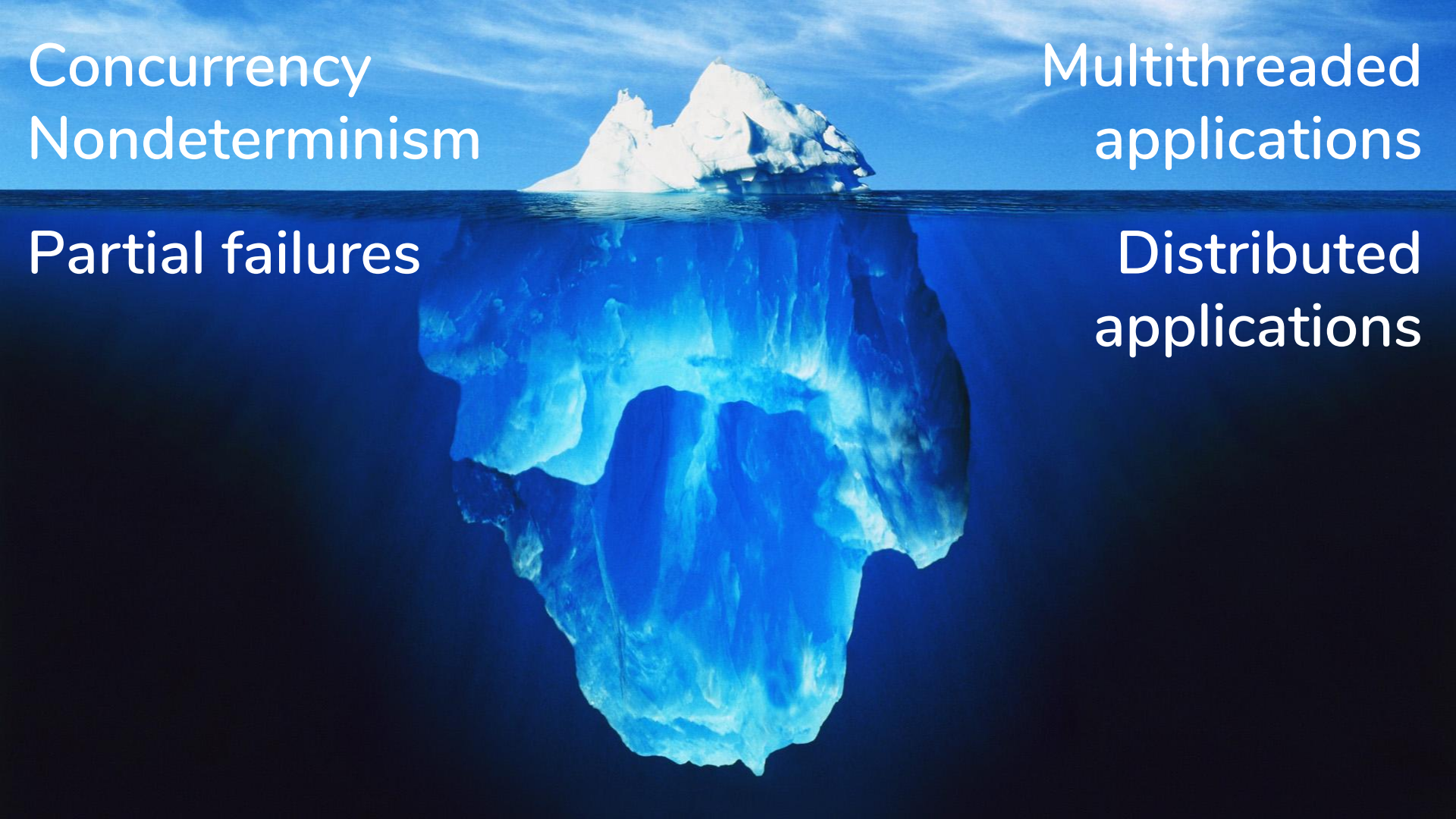
High-level APIs

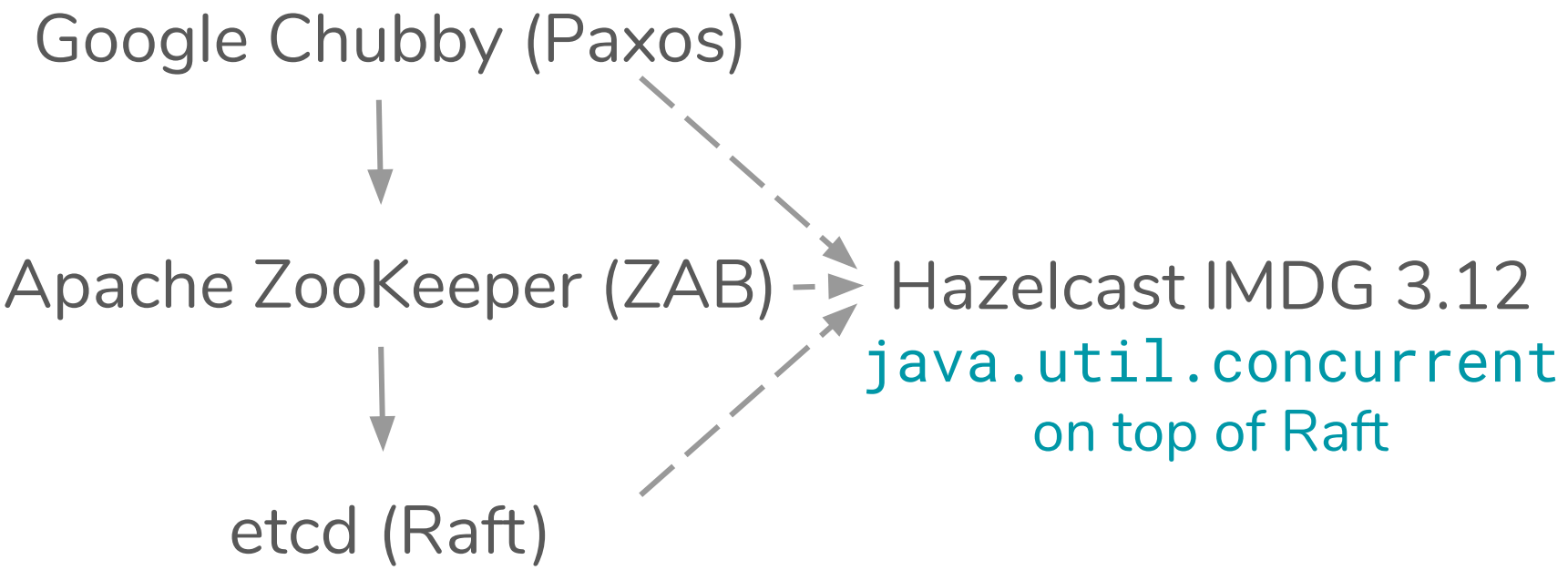
A low-level file-system / KV store API is

- easy to misuse,
- not suitable for all coordination tasks.

High-level APIs minimise guesswork and development effort.

`java.util.concurrent.*` in JDK


Concurrency
Nondeterminism
Multithreaded
applications
Partial failures
Distributed
applications


Google Chubby (Paxos)
Apache ZooKeeper (ZAB)
etcd (Raft)
Hazelcast IMDG 3.12
java.util.concurrent
on top of Raft

An Opinionated & High-Level Framework

`IAtomicLong, IAtomicReference,`

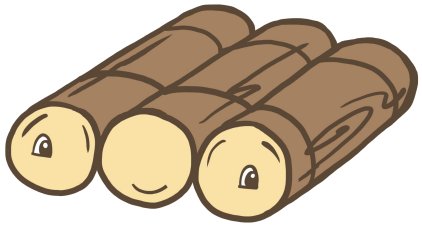
`ICountDownLatch, ISemaphore, FencedLock`

Well-defined failure semantics

CP with respect to CAP

DIY-style tested with Jepsen

Why Raft?



Understandability as a primary goal

Handles crash failures and network failures.

Operational as long as the majority is up.

Runtime concerns (snapshotting, dynamic membership)

Performance optimizations (fast reads, batching)

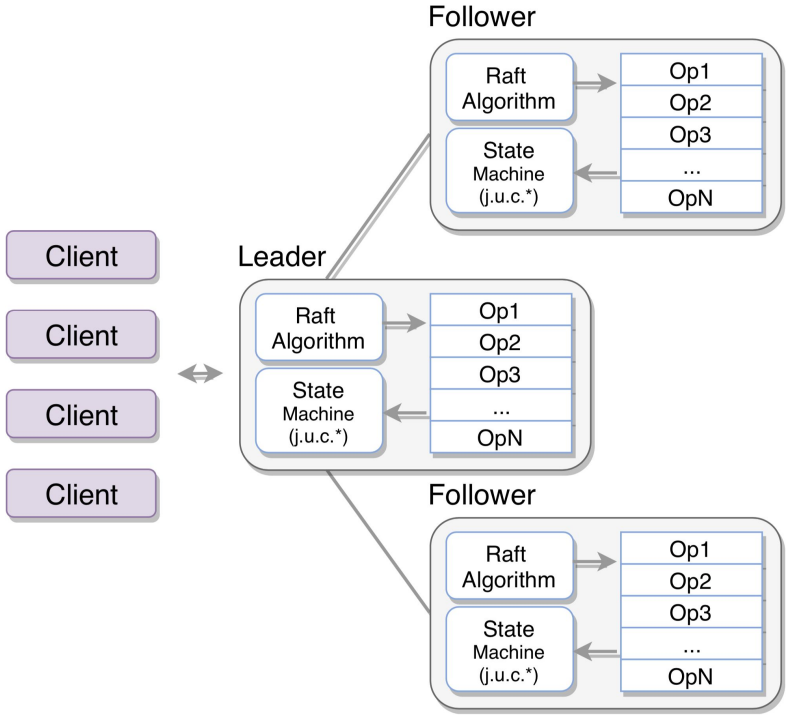
https://raft.github.io

Replicated State Machines

A leader is elected among the nodes.

The leader replicates ops to the followers.

All nodes run the ops in the same order.



CP Subsystem

Minimal configuration

CP primitives and AP data structures in the same cluster

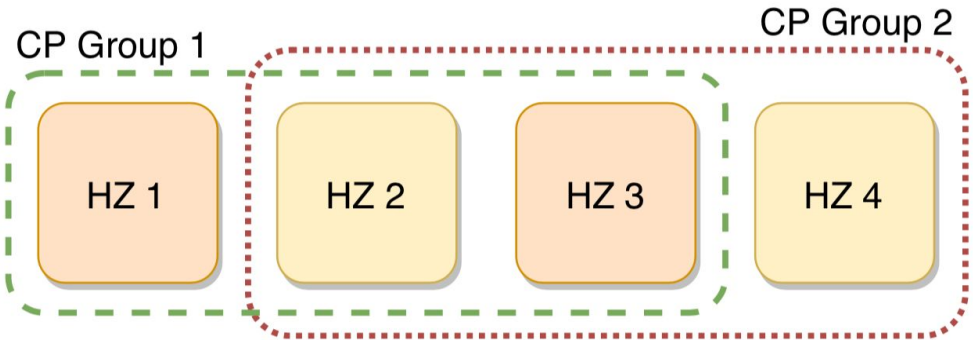
Dynamic clustering programmatically or via REST API

Horizontal Scalability

Each CP group runs the Raft algorithm independently.

CP primitives can be distributed to multiple CP groups.

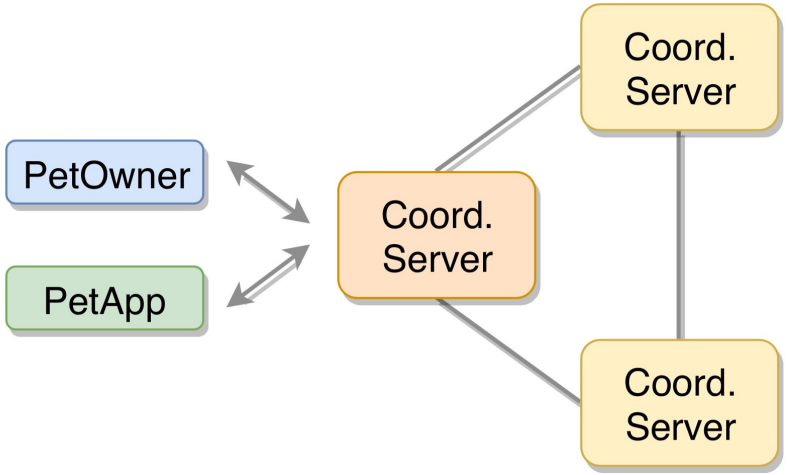
CP groups can be distributed to CP members.




ENOUGH TALK
LET’S DEMO

DEMO #1: Configuration management

https://github.com/metanet/juc-talk



FencedLock

Linearizable distributed impl of `java.util.concurrent.locks.Lock`

Suitable for both fine-grained and coarse-grained locking

CP Sessions

A session starts on the first lock / semaphore request.

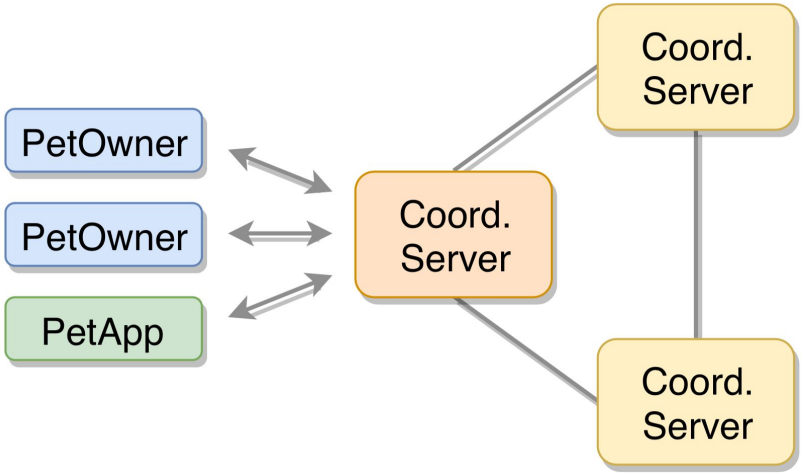
Session heartbeats are periodically committed in the background.

If no heartbeat for some time *(session TTL)*, the session is closed.

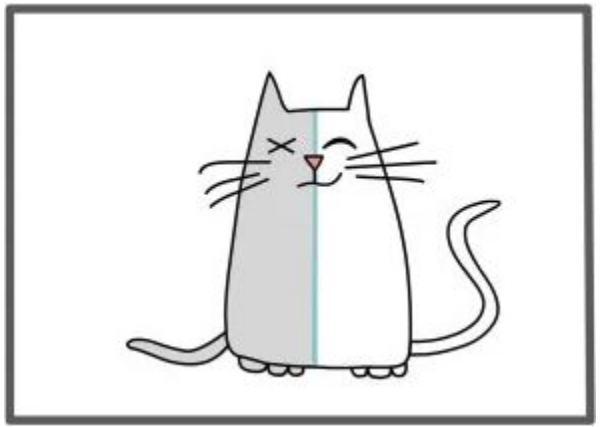
Auto-release mechanism for `FencedLock` and `ISemaphore`

DEMO #2: Adding Redundancy

We use FencedLock for leader election.



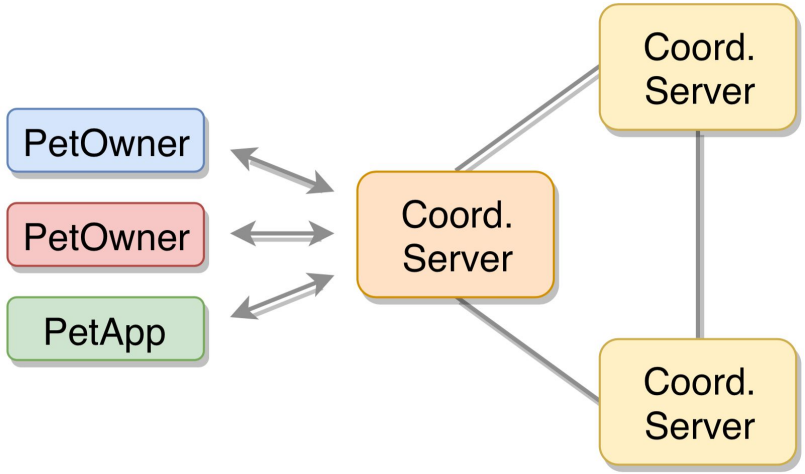
CP sessions offer a trade-off between safety and liveness.



DEMO #3: Fencing-off Stale Lock Holders

"How to do distributed locking"

"Distributed locks are dead; long live distributed locks!"



Recap

Avoid writing your own implementations for coordination.

High-level APIs minimise guesswork and development effort.

- `java.util.concurrent.*` FTW!

Operational simplicity matters.

- Dynamic clustering
- Horizontal scalability

Future Plans

KV Store

Event Listeners

Disk persistence

Tooling

Resources

https://github.com/metanet/juc-talk (demos)

Hazelcast IMDG Docs

CP Subsystem Code Samples

https://hazelcast.com/blog/author/ensarbasri

Hazelcast IMDG 3.12

Thanks!

In-Memory
Computing Summit
Europe 2019

Ensar Basri Kahveci
Distributed Systems Engineer @ Hazelcast
@ metanet