Creating and using a distributed lock

Snowcamp 2018 - Katia Aresti
Distributed cache (in memory datagrid)
key/value

Uses peer-to-peer communication

No masterslave
No bottleneck or SPOF

Search, Streams, integration with Spark, Kafka, Hadoop, Vert.x...

Embedded and Client/Server
Clustering with Infinispan
Clustered Vert.x application example

1. Send to address **IDS**

2. Consume from address **IDS**

3. Send to address **REBOOT**

4. Consume from address **REBOOT**

https://github.com/karesti/vertx-api/tree/snowcamp/clustered
https://github.com/karesti/vertx-api/tree/snowcamp-withlock
Clustered Lock

- Non reentrant
- Non blocking
- Node level Owner (!thread)
- Auto-release on Failure, Split-brain
@Experimental
public interface ClusteredLockManager {

    boolean defineLock(String name);

    boolean defineLock(String name, ClusteredLockConfiguration configuration);

    ClusteredLock get(String name);

    ClusteredLockConfiguration getConfiguration(String name);

    boolean isDefined(String name);

    CompletableFuture<Boolean> remove(String name);

    CompletableFuture<Boolean> forceRelease(String name);
}
Clustered Lock - **Infinispan 9.2**

```java
@Experimental
public interface ClusteredLock {

    CompletableFuture<Void> lock();

    CompletableFuture<Boolean> tryLock();

    CompletableFuture<Boolean> tryLock(long time, TimeUnit unit);

    CompletableFuture<Void> unlock();

    CompletableFuture<Boolean> isLocked();

    CompletableFuture<Boolean> isLockedByMe();

}
Replicated Cache

Key → Name

Value → { Owner, Request Id, Status } → A

Lock

SnowCamp

Node 1  Node 2  Node 3  ...  Node N
RPC calls to change lock state

Clustered Lock Manager

Clustered Lock

Node 1

Node 2

Node 3

...  

Node N

Replicated Cache

(RPC calls to change lock state)

Address (not a thread)
Clustered Lock - Infinispan 9.2

```java
@Experimental
public interface ClusteredLock {

    CompletableFuture<Void> lock();

    CompletableFuture<Boolean> tryLock();

    CompletableFuture<Boolean> tryLock(long time, TimeUnit unit);

    CompletableFuture<Void> unlock();

    CompletableFuture<Boolean> isLocked();

    CompletableFuture<Boolean> isLockedByMe();
}
```
Using Infinispan API

```java
lock.tryLock()
 .thenCompose(result -> {
   if (result) {
     try {
       // manipulate protected state
     } finally {
       return lock.unlock();
     }
   } else {
     // Do something else
   }

```
Comparing with the JDK

```java
lock.tryLock()
  .thenCompose(result -> {
    if (result) {
      try {
        // manipulate protected state
      } finally {
        return lock.unlock();
      }
    } else {
      // Do something else
    }
  });
```

```java
Lock lock = ...;
if (lock.tryLock()) {
  try {
    // manipulate protected state
  } finally {
    lock.unlock();
  }
} else {
  // perform alternative actions
}
```
Coding `tryLock()`, `unlock()` and `tryLock(time, unit)`!
Final thoughts and roadmap

The **power of distributed caches don’t stop on caching!**

Concurrency is not simple (at all)

Creating **reentrant** locks too

Adding ownership levels for **locking (dealing concurrency in each node too)**

Locks for **client / server** mode

Other structures available: **Multimap, Counters ...**

Coming: **Topics, Sorted Sets**
Feedback please! ❤️

Clustered Lock

https://github.com/infinispan/infinispan/tree/master/lock

Vert.x Cluster Manager

https://github.com/vert-x3/vertx-infinispan
@karesti
@infinispan