JUDCon

JBoss Users & Developers Conference

London:2011

Infinispan for Ninja Developers

Mircea Markus, Red Hat R&D

Who’s this guy?

- R&D JBoss Clustering @ Redhat
- JBoss clustering: JBossCache, PojoCache, jGroups,..
- Infinispan developer - day 1
- Founder Radargun

Agenda

- Transactions
  - optimistic/pessimistic 

  - JTA support
    - XA or not
    - recovery
  - deadlock avoidance 

- Client server access
- Querying and indexing
- Distribution
  - topology-aware
  - virtual nodes
- Q&A

Cache transaction modes

- Before 5.1
  - mixed
- Starting with 5.1
  - no mixed access!
  - non-transactional `<transaction transactionMode="NON_TRANSACTIONAL"/>`
  - transactional `<transaction transactionMode="TRANSACTIONAL"/>`
    - optimistic
    - pessimistic
    - requires a TransactionManager
- backward compatible `<transaction autoCommit="true"/>`

Transaction locking modes

- Transactional caches only
- Optimistic

```
<transaction lockingMode="OPTIMISTIC"/>
```

  - no locks before prepare
  - small lock scope
- Pessimistic

```
<transaction lockingMode="PESSIMISTIC"/>
```

  - lock acquired on each write
  - writes block writes
  - reads do not block
  - locks held longer

Pessimistic or Optimistic?

- Optimistic
  - low contention
  - high contention -> many rollbacks
  - disable version check
- Pessimistic
  - high key contention
  - rollbacks are less desirable
  - more costly/more guarantees

```
<locking writeSkewCheck="false"/>
```

JTA integration

- JTA transactions
  - known API
- Multiple options
    - full xa (XAResource)
    - less strict (Synchronization)
- Configurable
  - same client API

XA or not?

- XA
  - proper distributed transactions
  - recovery enabled
    - or not (default)
- Synchronization
  - cache backed by a data store
  - Transaction more efficient
    - 1PC optimisation
    - TransactionManager not writing logs
  - Hibernate 2LC

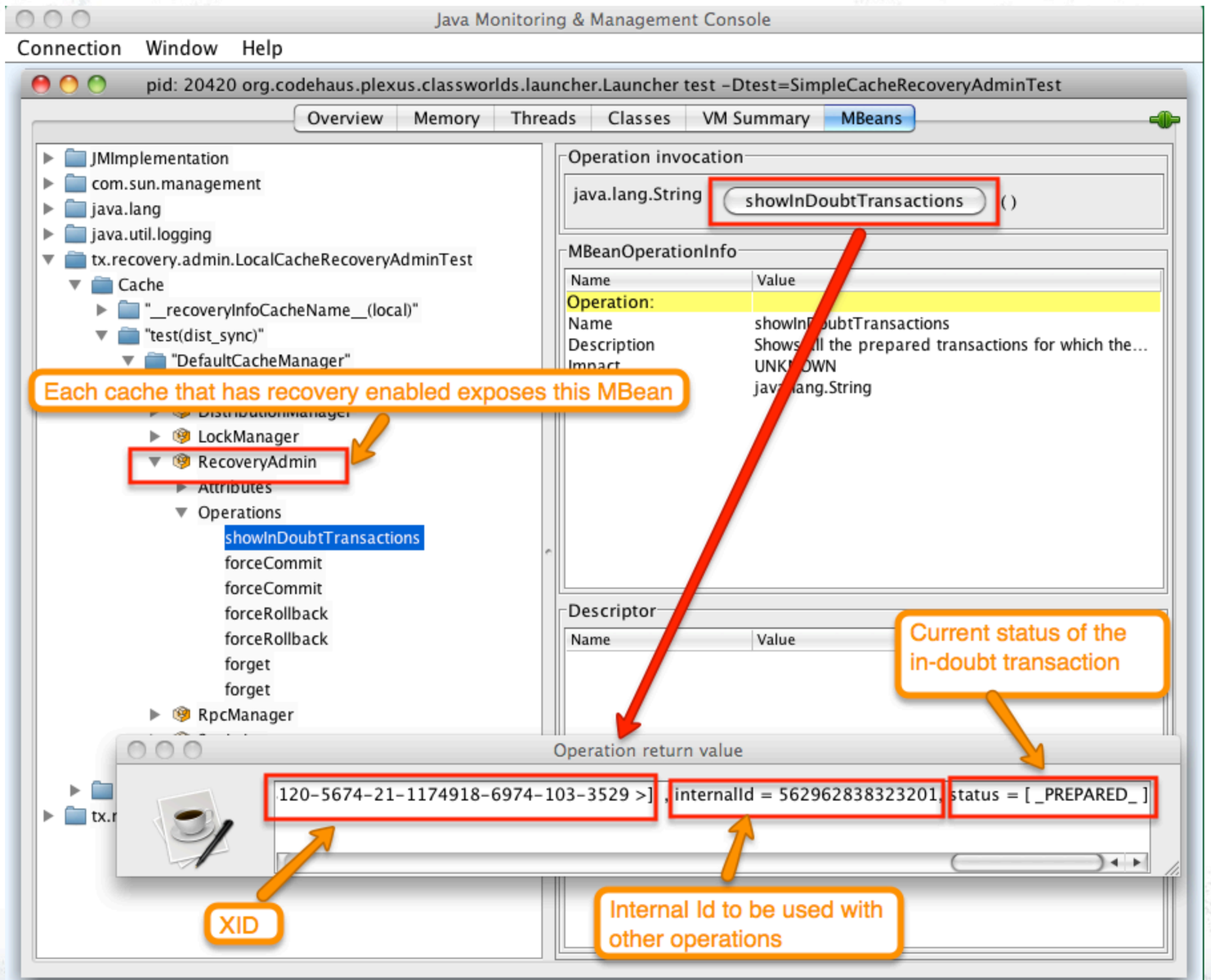
```
<transaction>
    <recovery enabled="true"/>
</transaction>
```

```
<transaction useSynchronization="true"/>
```

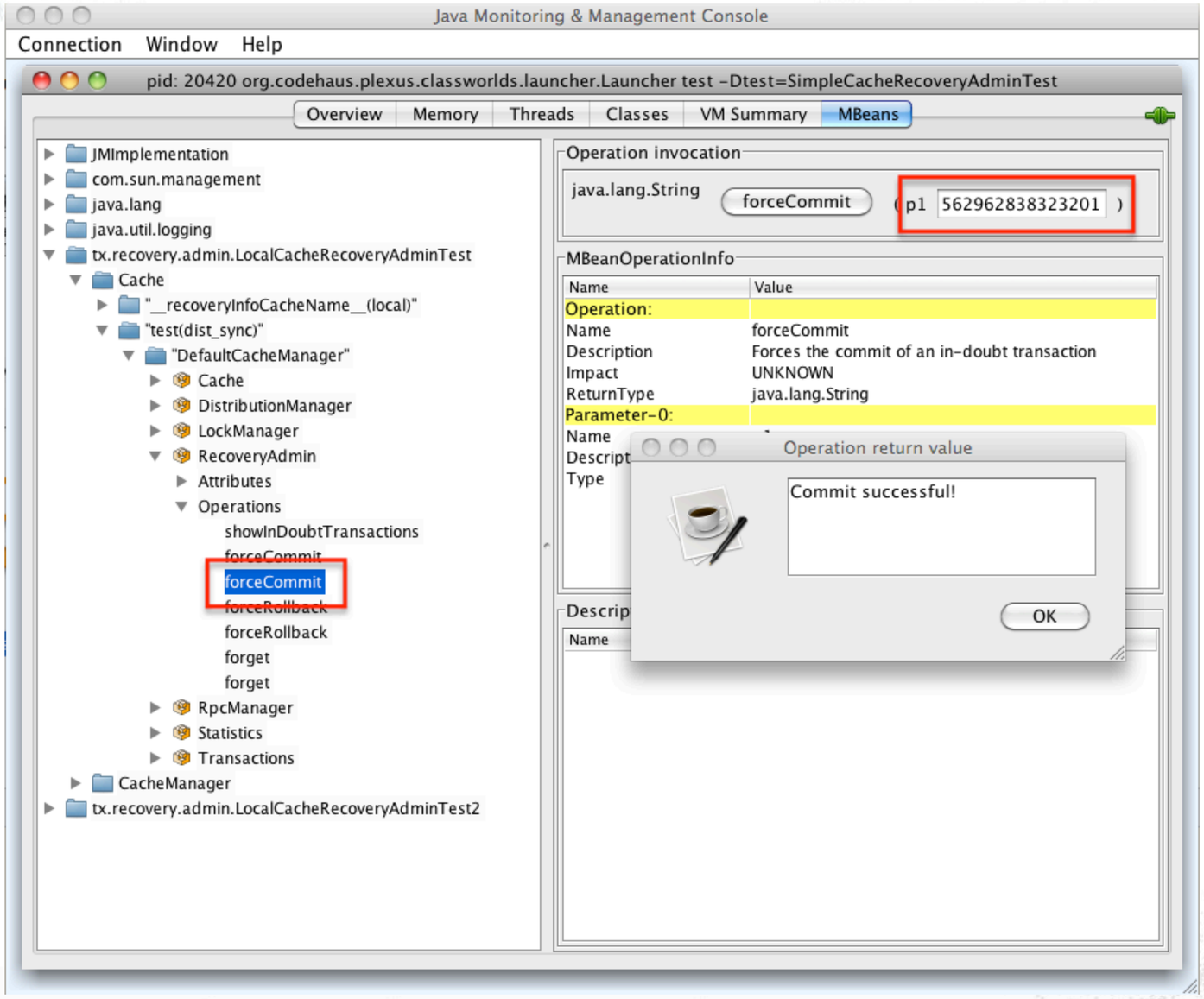
Recovery

- When is needed?
  - prepare successful, commit fails
  - inconsistent state!
- How to handle it
  - TransactionManager informs SysAdmin
  - JMX tooling exposed to
    - force commit
    - force rollback

JMX - show in-doubt tx



JMX - force commit



Deadlocks

- Deadlock
  - Tx1: a -> b
  - Tx2: b -> a
  - “right” timing
- Bad for system throughput
  - threads blocked until (one) tx timeouts
  - lockAcquisitionTimeout defaults to 10 seconds!
  - a,b are locked during this time -> potentially more deadlocks

What’s to be done?

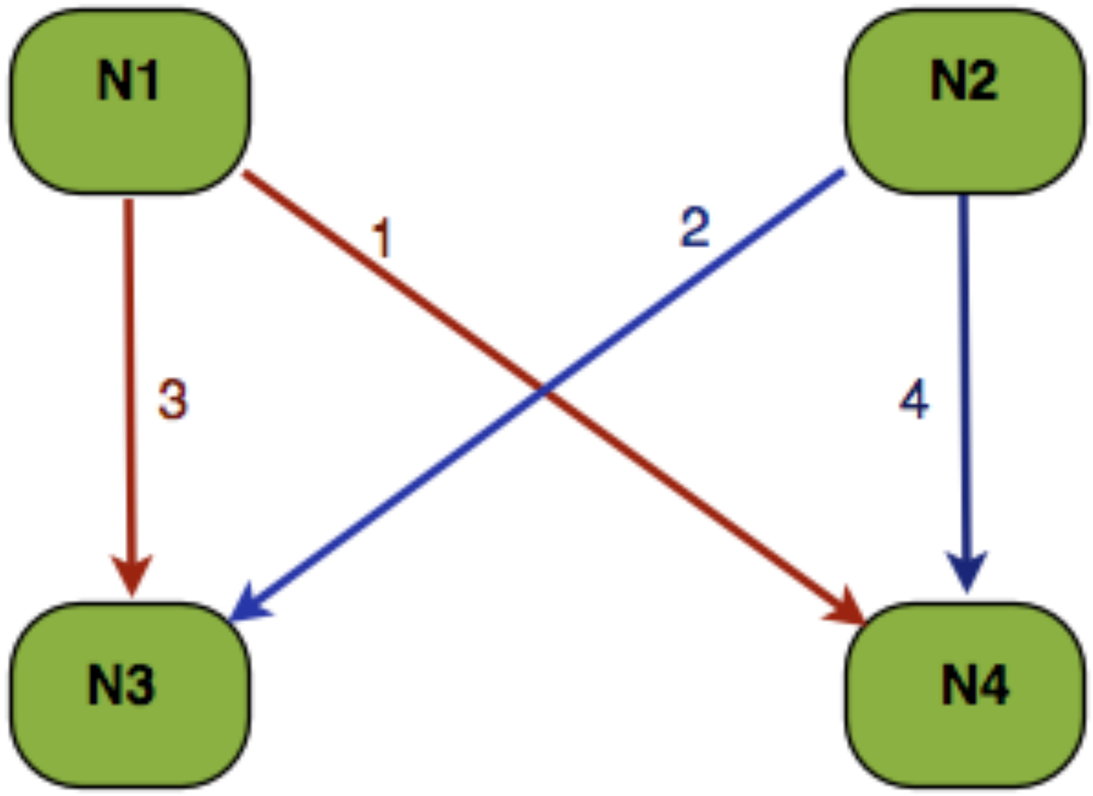
- Order key
  - e.g. lexicographically
  - Tx1: a -> b
  - Tx2: a -> b
  - not always possible
- Use deadlock detection
  - fail fast
  - one tx succeeds

```
<deadlockDetection enabled="true" spinDuration="1000"/>
```

New deadlock avoidance techniques (5.1)

- Single lock owner
  - avoid same key-deadlocks
- Optimistic only
  - Incremental locking
    - acquire locks in sequence
- Lock reordering
  - based on consistent hash

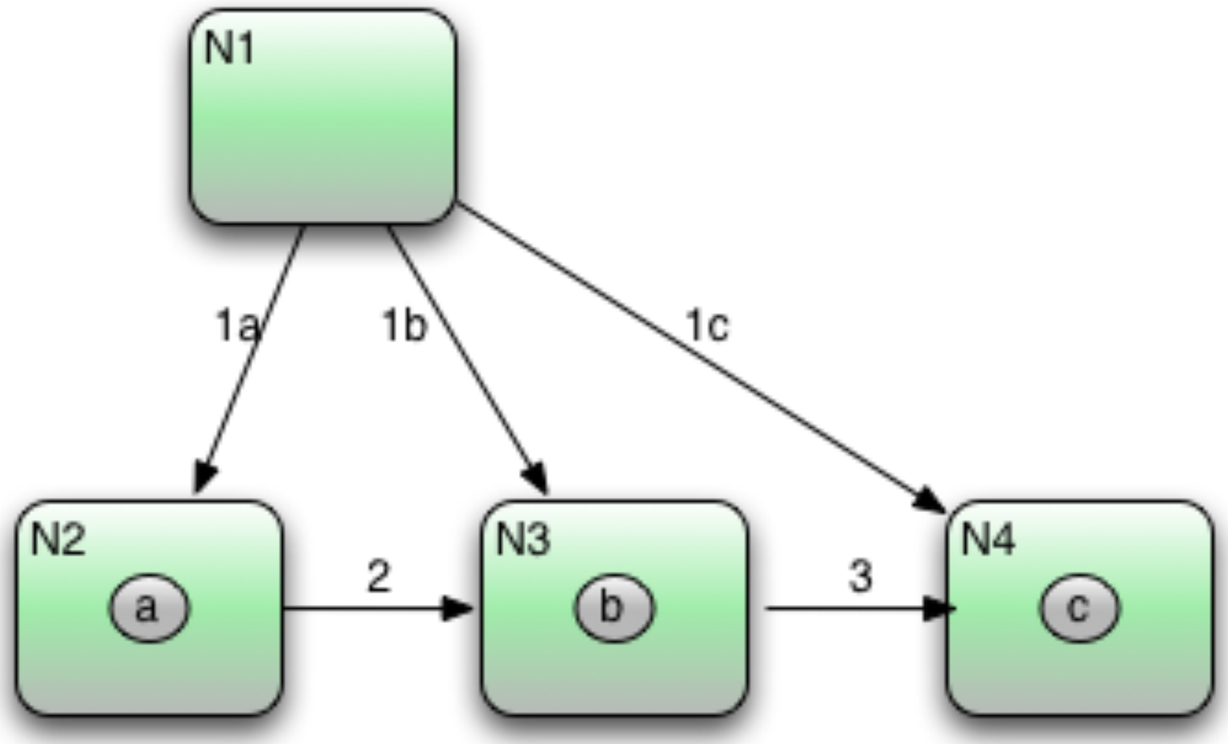
Same-key deadlock



Incremental locking - the problem

- Tx1 and Tx2 execute in parallel on two nodes N1 and N2
- Tx1 and Tx2 write keys {a,b}
- consistentHash(a) = {N3} and consistentHash(b) = {N4}
- With “right” timing:
  - Tx1 lock acquired on "a" @ N3
  - Tx2 lock acquired on "b" @ N4
  - Deadlock!

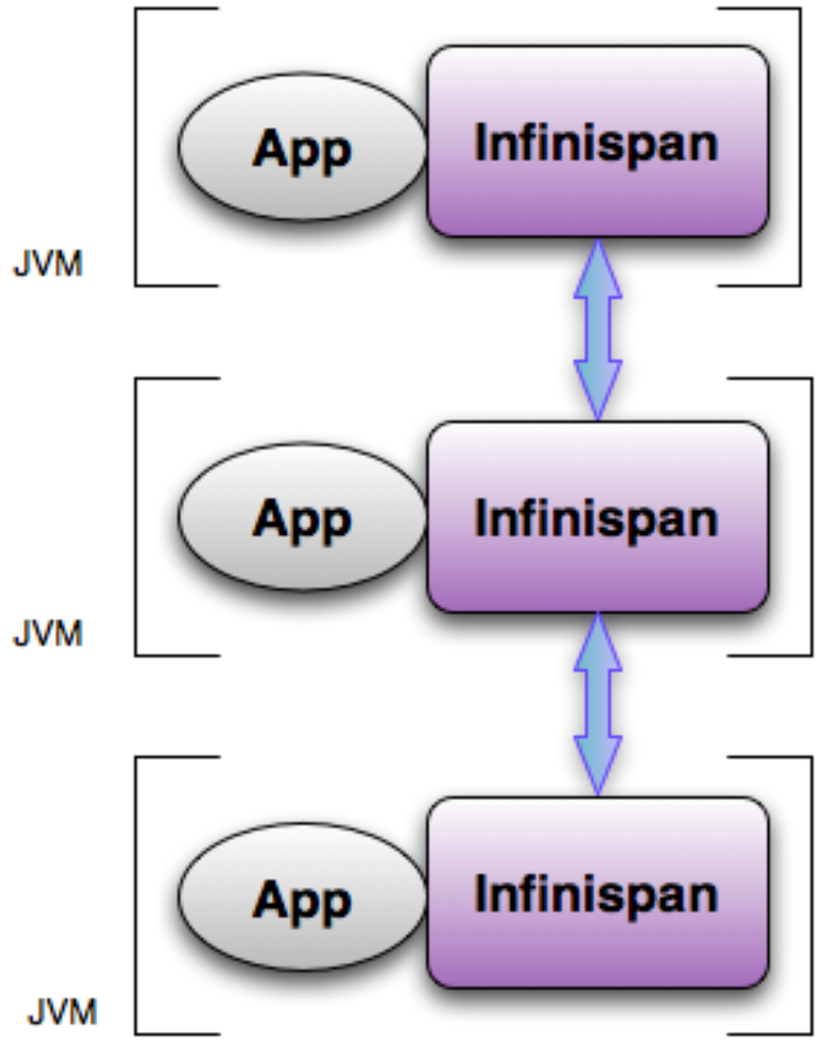
Incremental locking



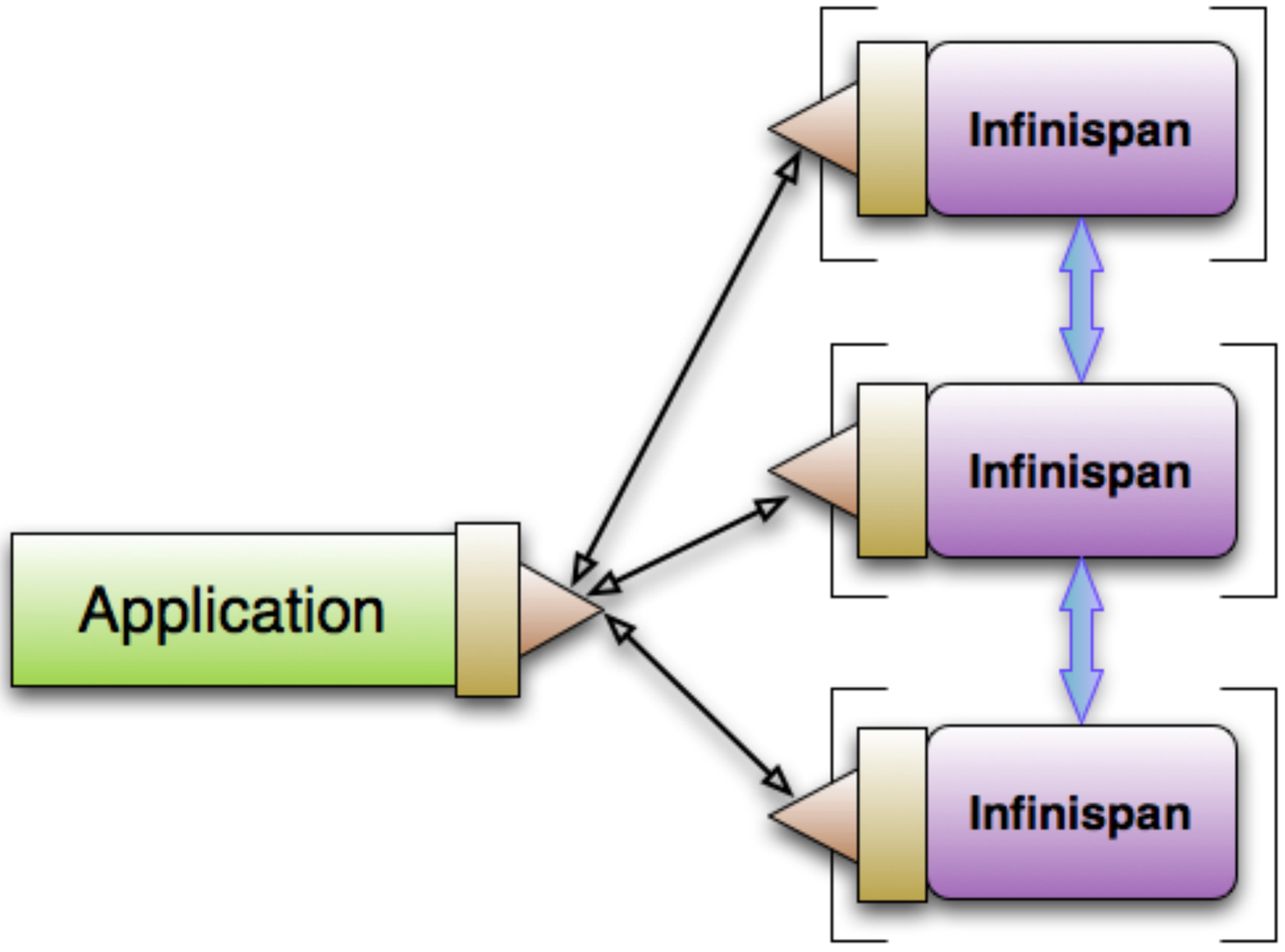
P2P vs Client-Server

- Peer to Peer
- Client-Server
- Hotrod
- Available server endpoints
- Client-Server-when?

Peer To Peer Architecture



Client/Server Architecture



Supported Protocols

- REST
- Memcached
- Hot Rod

Hotrod?!

- Wire protocol for client server communications
- Open
- Language independent
- Built-in failover and load balancing
- Smart routing
- xa support - to come

Server Endpoint Comparison

	Protocol	Client Libraries	Clustered?	Smart Routing	Load Balancing/Failover
REST	**Text**	N/A	**Yes**	**No**	Any HTTP load balancer
Memcached	**Text**	Plenty	**Yes**	**No**	Only with predefined server list
Hot Rod	**Binary**	Java, Python	**Yes**	**Yes**	Dynamic

Client/Server - when?

- Client not affected by server topology changes
- Multiple apps share the same grid
- Tier management
  - incompatible JVM tuning
  - security
- Non-JVM clients

Querying and indexing

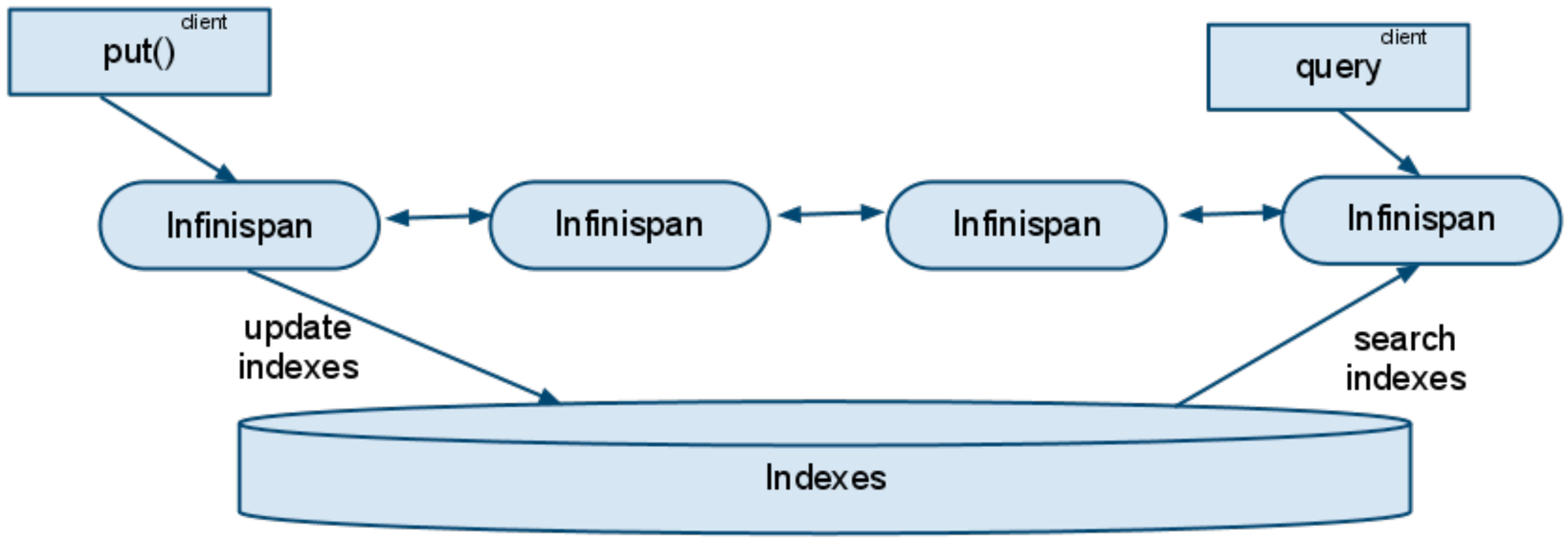
- Querying
  - based on Hibernate Search (Apache Lucene)
- API (5.0)

```
SearchManager sm = Search.getSearchManager(cache);

// luceneQuery below is an org.apache.lucene.search.Query
final CacheQuery ispnQuery= sm.getQuery(luceneQuery);

List<Object> list = ispnQuery.list();
```

Shared indexes



Sharing the index

- FSDirectory
  - shared disk
  - problematic on some file systems
- Infinispan directory provider
  - fast, reliable
  - can be persisted if needed

Distribution

- Based on consistent hash
  - horizontal scalability
  - pluggable hash function
- Topology aware consistent hash
  - improved consistency
- Virtual Nodes
  - better load balancing - it’s all about performance!
  - added in 5.0

```
<namedCache name="distributedCache">
   <clustering mode="distribution">
      <hash hashFunctionClass="foo.bar.MyChImpl" />
   </clustering>
</namedCache>
```

Topology aware consistent hash

- Each node aware about cluster topology
  - Machine, rack, site
- Distribute backups as far as possible
  - other site. rack. machine

```
<transport clusterName="infinispan-cluster" distributedSyncTimeout="50000"
           nodeName="Jalapeno" machineId="m1" rackId="r1" siteId="s1">
   ....
</transport>
```

- Collocated nodes
  - higher chance to fail together
  - cheaper access

Virtual nodes

- Physical node
  - multiple virtual nodes
  - configurable per node
- Why needed?
  - uneven load on cluster nodes
    - bad for performance!
    - worse for small clusters
- more powerful nodes can take more load
  - configure more virtual nodes

Questions?

- www.infinispan.org
- http://infinispan.blogspot.com
- http://twitter.com/infinispan
  - #infinispan
  - #JUDCon
- irc #infinispan on freenode