

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 `<locking writeSkewCheck="false"/>`
- Pessimistic
 - high key contention
 - rollbacks are less desirable
 - more costly/more guarantees

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)

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

- Synchronization
 - cache backed by a data store
 - Transaction more efficient
 - 1PC optimisation
 - TransactionManager not writing logs
- Hibernate 2LC

```
<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

The screenshot shows the Java Monitoring & Management Console (JMX) interface. The left pane displays a tree view of MBeans, with the path `tx.recovery.admin.LocalCacheRecoveryAdminTest > Cache > "test(dist_sync)" > "DefaultCacheManager" > RecoveryAdmin` selected. The `showInDoubtTransactions` operation is highlighted. The right pane shows the details of the `showInDoubtTransactions` operation, including its name, description, and return type (`java.lang.String`). Below the main console, a separate window displays the operation return value: `120-5674-21-1174918-6974-103-3529 >], internalId = 562962838323201, status = [_PREPARED_]`. Annotations with arrows point to the `RecoveryAdmin` MBean, the `showInDoubtTransactions` operation, the return value string, the `internalId`, and the `status`.

Each cache that has recovery enabled exposes this MBean

Current status of the in-doubt transaction

XID

Internal Id to be used with other operations

JMX - force commit

The screenshot displays the Java Monitoring & Management Console (JMX) interface. The main window shows a tree view of MBeans on the left and an 'Operation invocation' panel on the right. The 'forceCommit' operation is selected in the tree and its details are shown in the right panel. A red box highlights the 'forceCommit' operation in the tree and the 'forceCommit' button in the invocation panel. A red box also highlights the parameter value '562962838323201' in the invocation panel. An 'Operation return value' dialog box is overlaid on the console, displaying 'Commit successful!' and an 'OK' button.

Connection Window Help
pid: 20420 org.codehaus.plexus.classworlds.launcher.Launcher test -Dtest=SimpleCacheRecoveryAdminTest

Overview Memory Threads Classes VM Summary **MBeans**

JMImplementation
com.sun.management
java.lang
java.util.logging
tx.recovery.admin.LocalCacheRecoveryAdminTest
Cache
 "__recoveryInfoCacheName__(local)"
 "test(dist_sync)"
 "DefaultCacheManager"
 Cache
 DistributionManager
 LockManager
 RecoveryAdmin
 Attributes
 Operations
 showInDoubtTransactions
 forceCommit
 forceCommit
 forceRollback
 forceRollback
 forget
 forget
 RpcManager
 Statistics
 Transactions
 CacheManager
tx.recovery.admin.LocalCacheRecoveryAdminTest2

Operation invocation
java.lang.String forceCommit (p1 562962838323201)

MBeanOperationInfo
Name Value
Operation:
Name forceCommit
Description Forces the commit of an in-doubt transaction
Impact UNKNOWN
ReturnType java.lang.String
Parameter-0:
Name
Description
Type

Operation return value
Commit successful!
OK

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

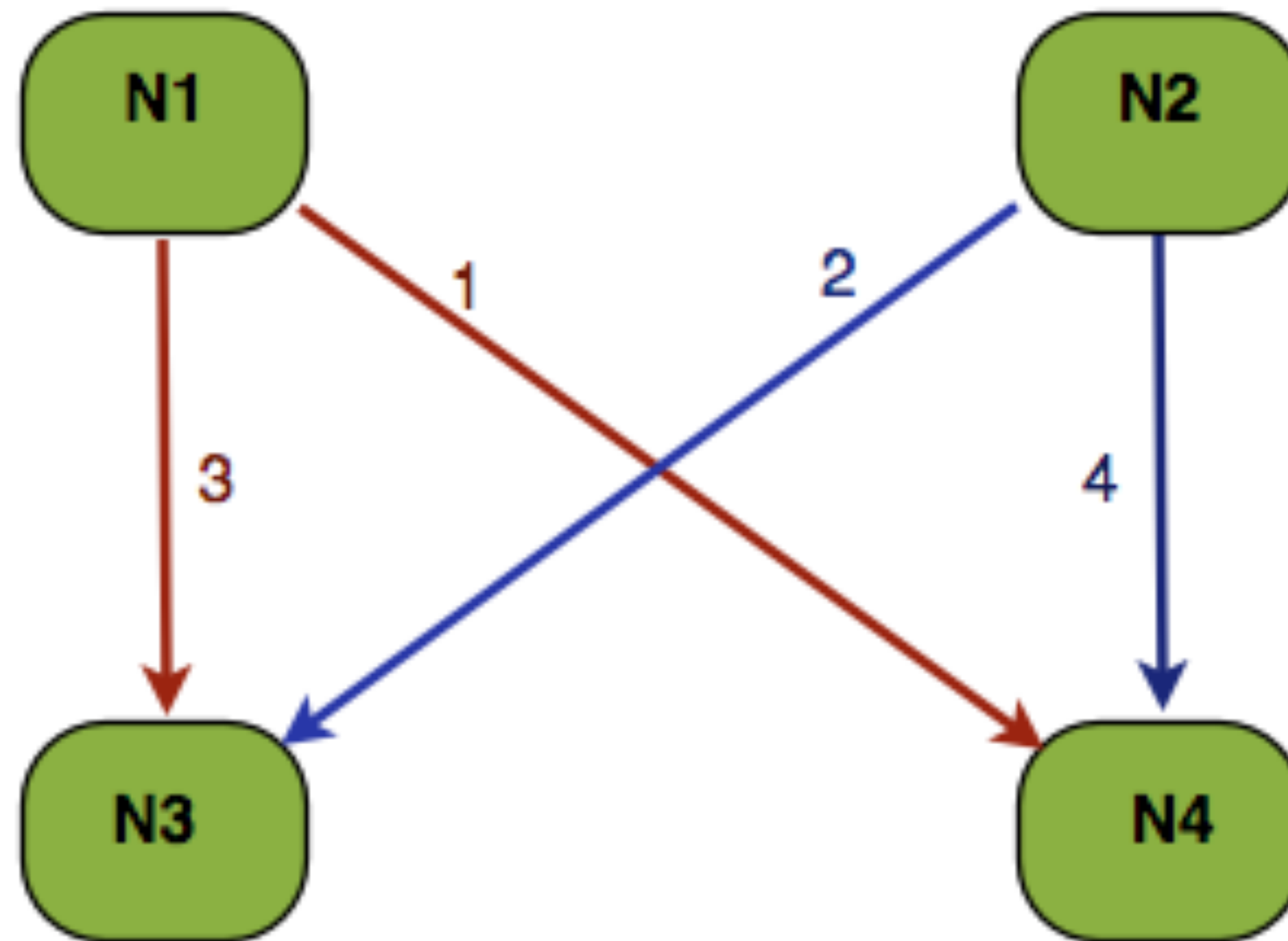
What's to be done?

- Order key
 - e.g. lexicographically
 - Tx1: a -> b
 - Tx2: a -> b
 - not always possible
- Use deadlock detection
 - fail fast `<deadlockDetection enabled="true" spinDuration="1000"/>`
 - one tx succeeds

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

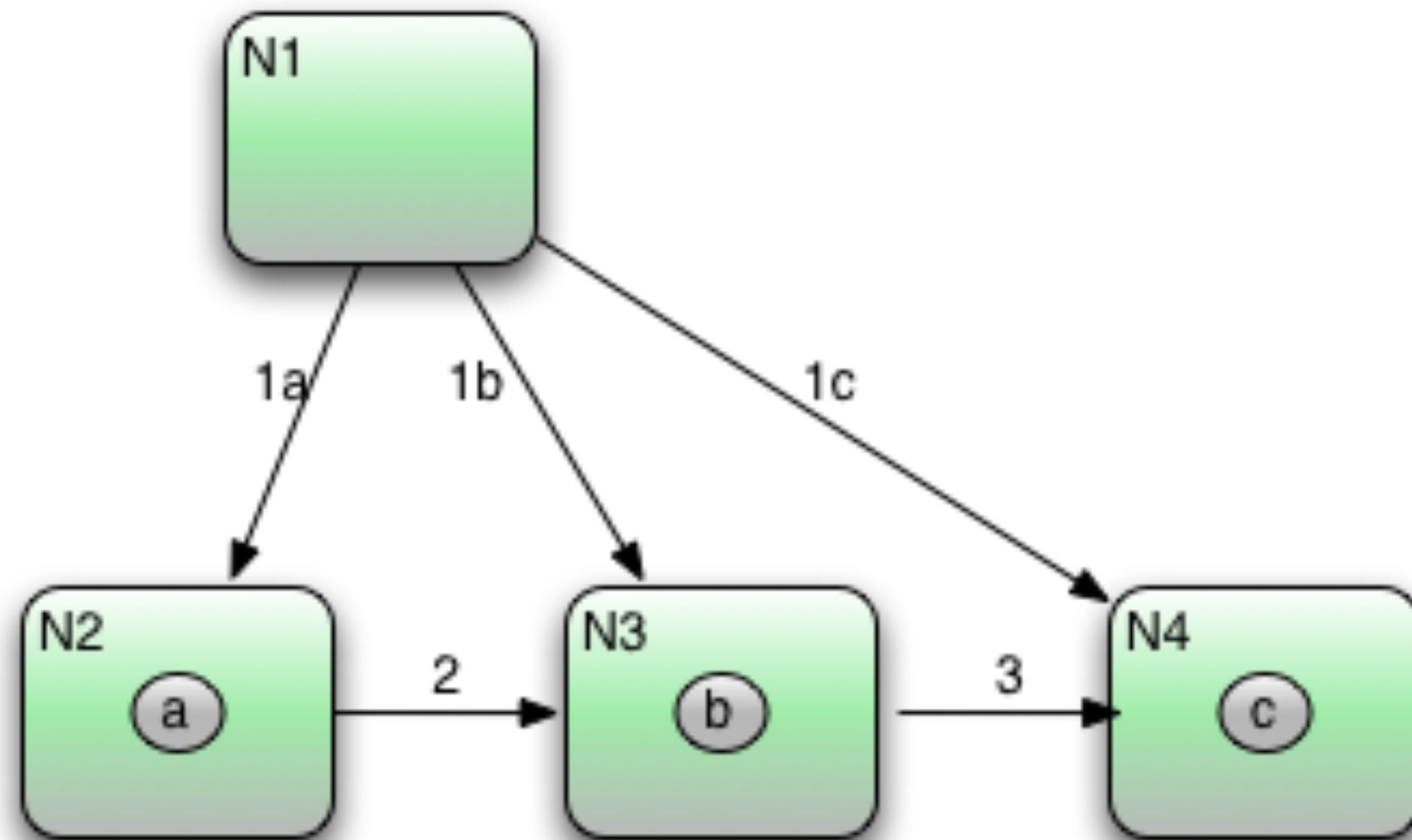
Same-key deadlock



Incremental locking - the problem

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

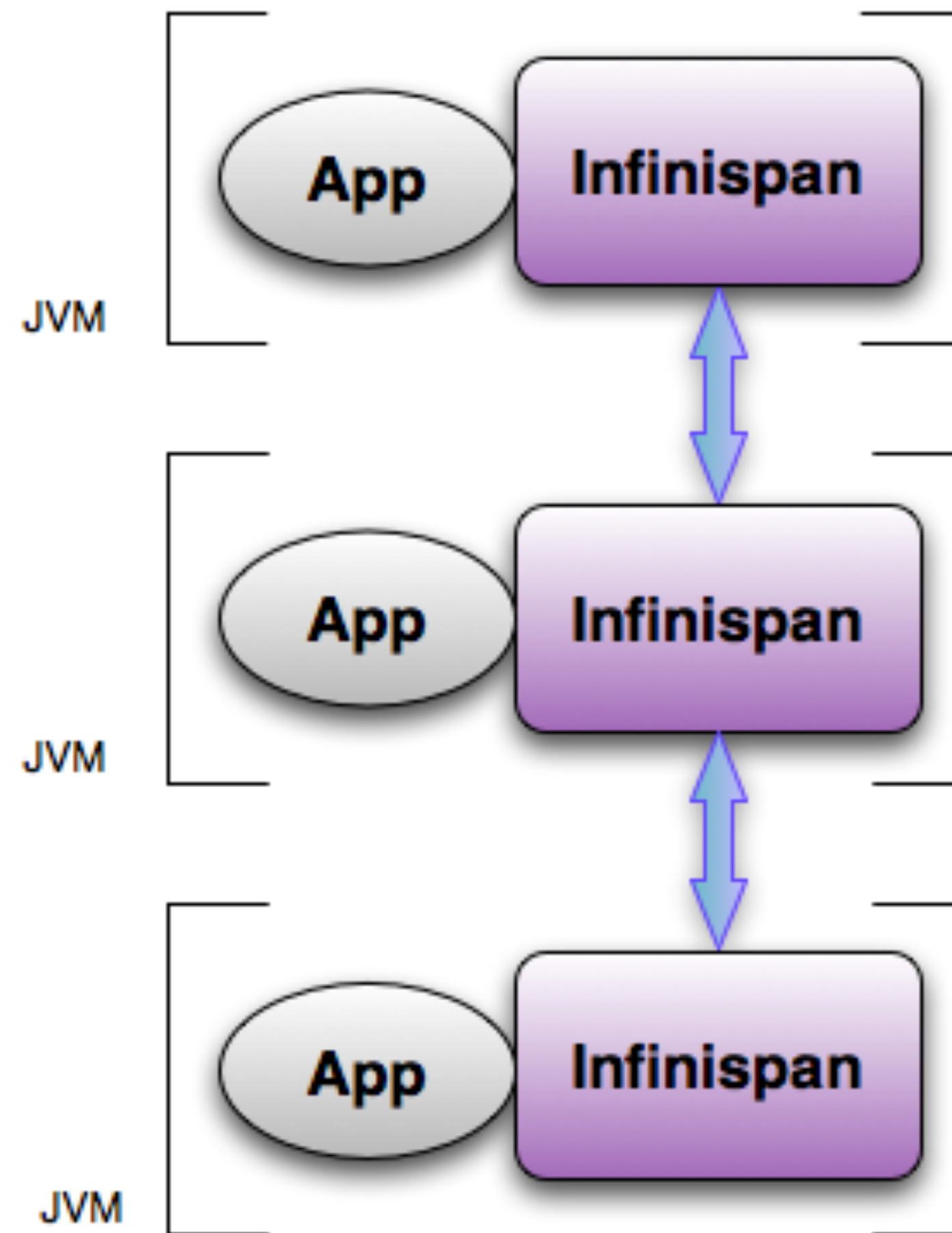
Incremental locking



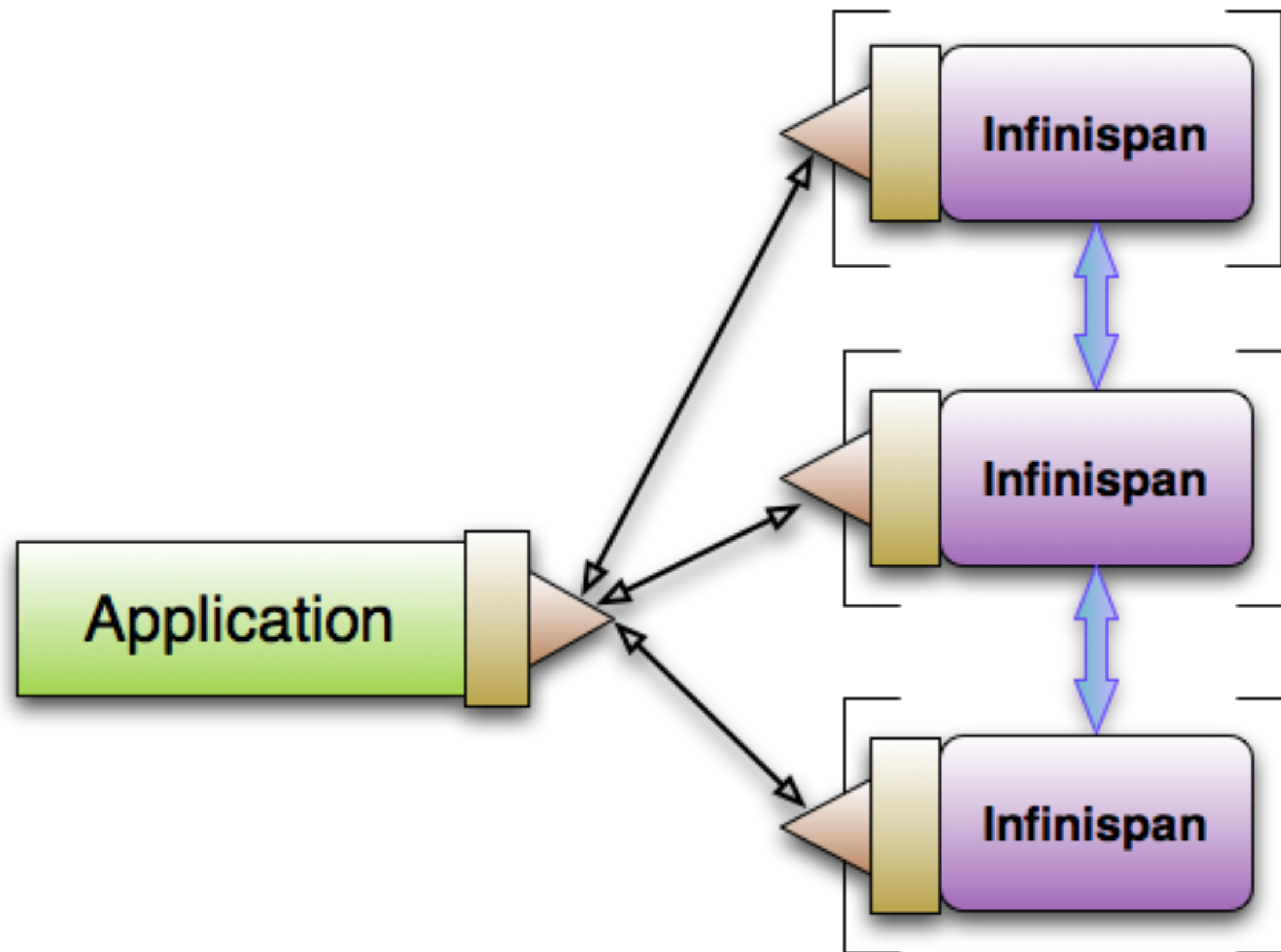
P2P vs Client-Server

- 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

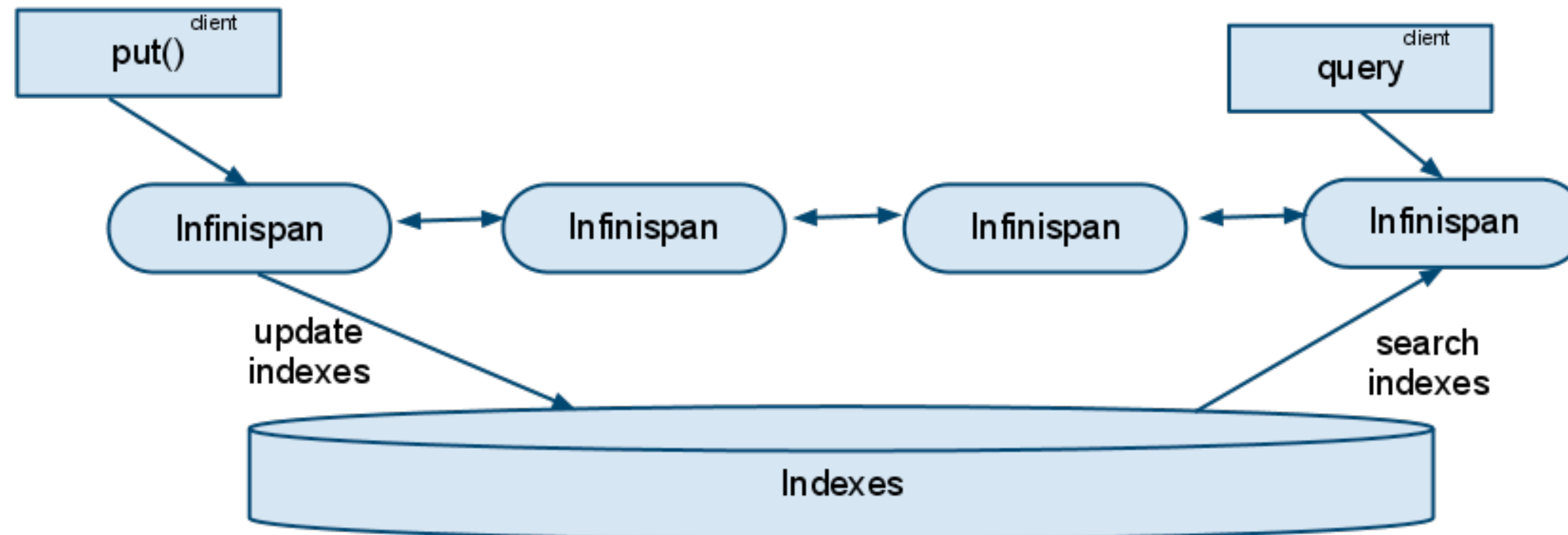
- Querying
 - based on Hibernate Search (Apache Lucene)
- API (5.0)

```
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