redhat. | JBoss by Red Hat

Java. Cloud. Leadership.

Real World Infinispan

Pete Muir
Red Hat, Inc.
May 2012

Agenda

- Introduction
  - What is Infinispan?
  - Principle use cases
  - Key features
- Hands-on demo
  - build an application using infinispan
- Extras
  - Querying the Grid
  - Database - OGM
  - Performance tuning - RadarGun
- Conclusion

Lab Setup

- Download the lab zip:

  http://bit.ly/infinispan-labs

  - Unzip the lab to your disk to a location of your choice
  - If you are a git user, you can clone the repository:

```
git clone git://github.com/infinispan/infinispan-labs.git
```

  - each stage of this lab has a checkpoint which is branched, you can check out the code for each Checkpoint using:

```
git checkout -b checkpointX origin/checkpointX
```


Lab Setup

- Follow along using

  http://bit.ly/infinispan-labs

- Download JBoss AS 7.1.1 from

  http://jboss.org/jbossas/downloads

- Unzip JBoss AS to your disk to a location of your choice

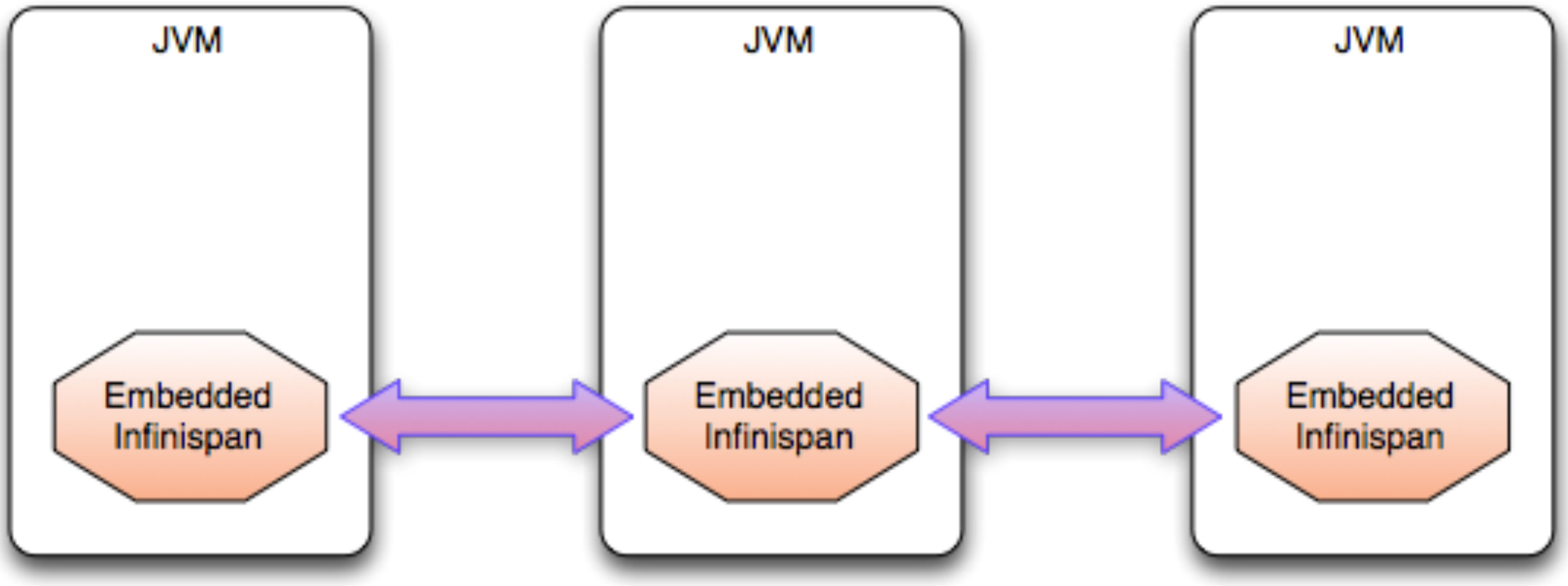
Introduction

@plmuir

So what is Infinispan?

- Distributed, in memory, data structure
- Highly available
- Elastic
- Open source

Distributed Data structure



High availability

- Memory is volatile
- Make redundant copies
  - Total replication (Replication Mode)
  - Partial replication (Distribution Mode)
- Topology changes
  - Node will crash!
  - Re-arrange state

Elasticity

- Expect
  - Node additions
  - Node removals
- Topology changes
  - are totally consistent
  - do not "stop the world"

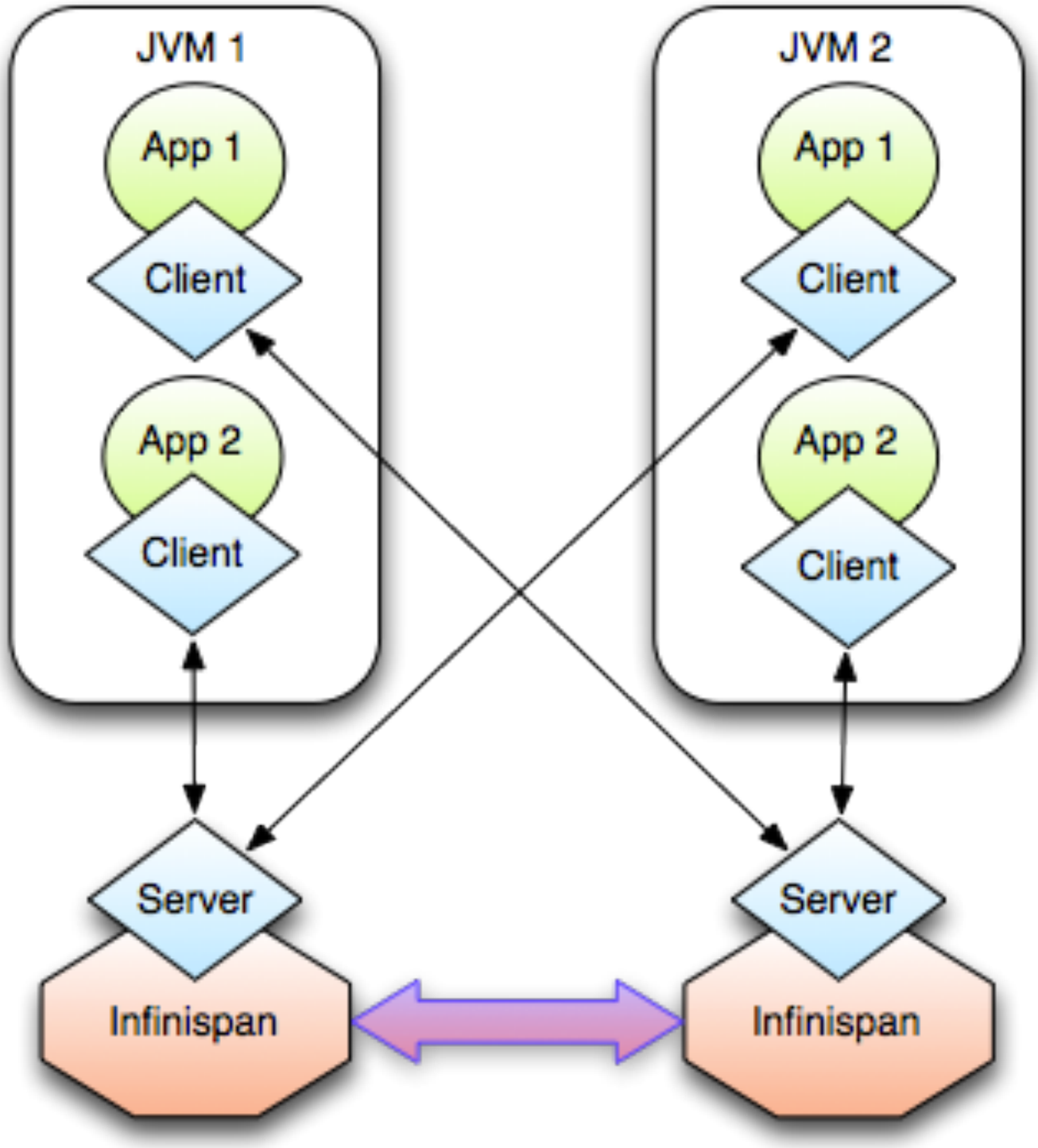
Access modes

- Embedded
  - client and node on same VM
  - fast!
- Client/server
  - different processes
  - multiple protocols
    - REST
    - Memcached
    - Hotrod

Embedded access

@plmuir

Client/server access



Server endpoints
- REST
- Memcached
- Hotrod

@plmuir

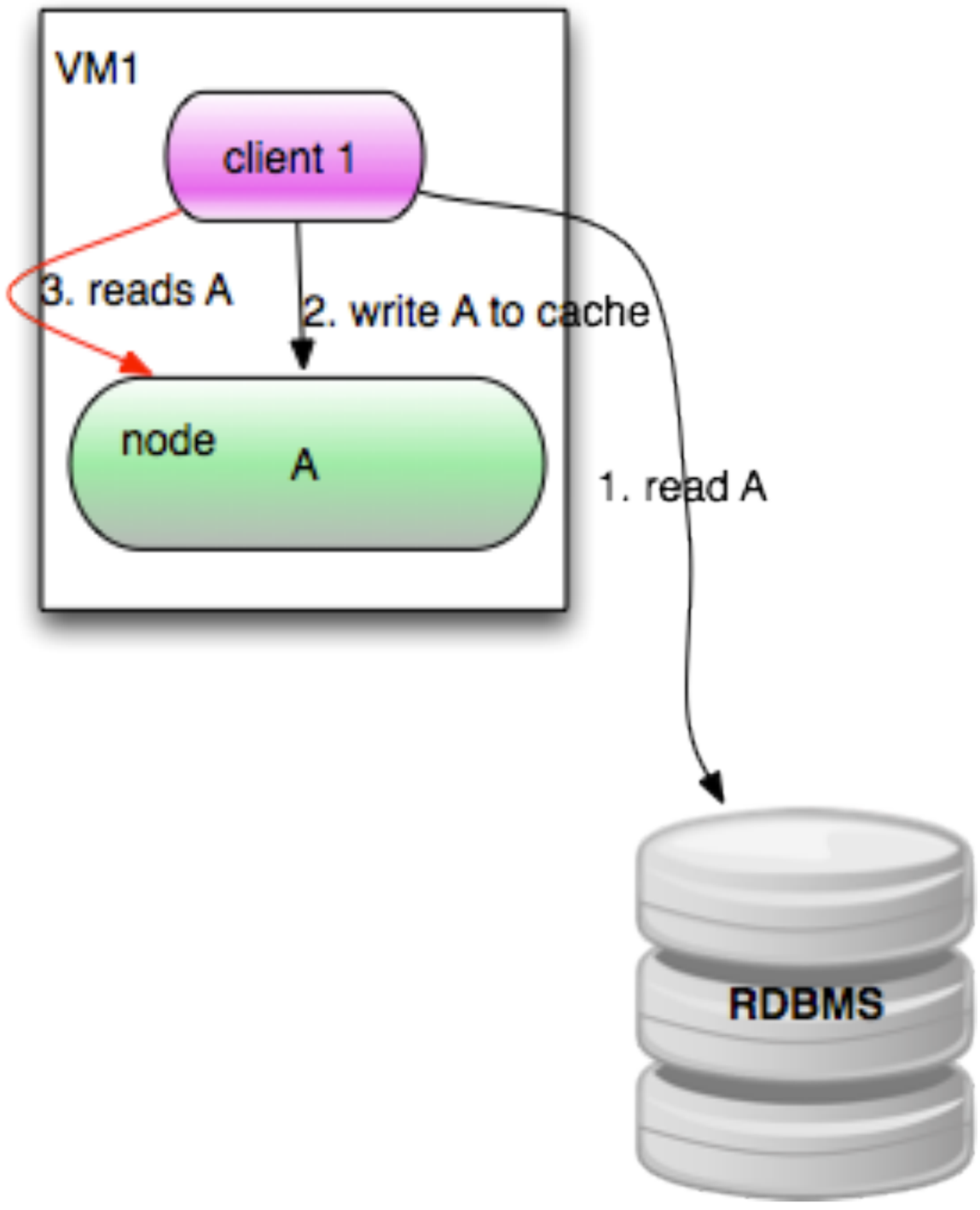
Main use cases

- Local cache
  - e.g. Hibernate 2nd level cache
- Cluster of caches
  - More caching capacity
  - Co-located clients
- Data Grid
  - dedicated cluster of servers
  - remote access

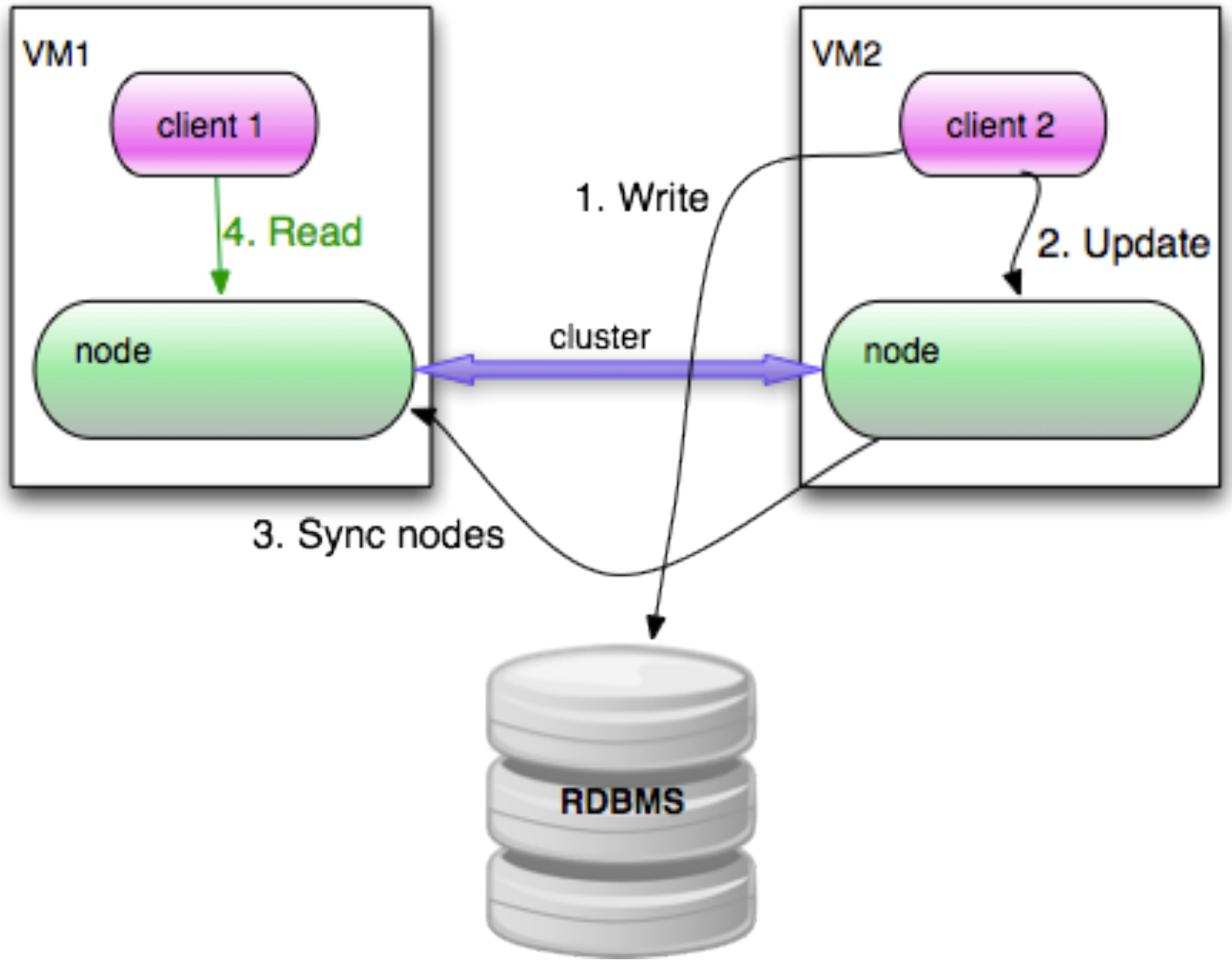
Good old caching...

- Local cache
  - java.util.Map
- And some more
  - eviction
  - expiry
  - write through/behind
  - passivation
  - preloading
  - notifications

Use Case 1: Local Cache

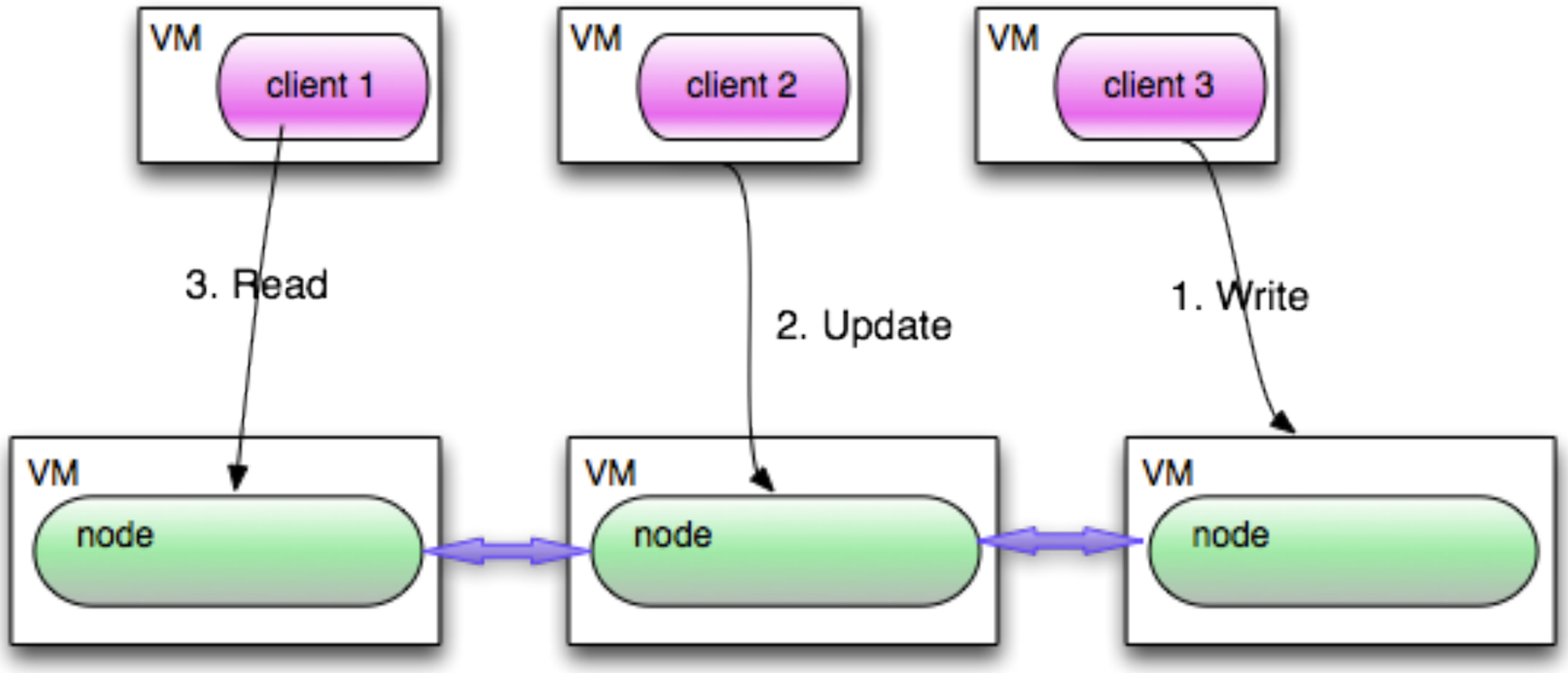


Use Case 2: Cluster of caches

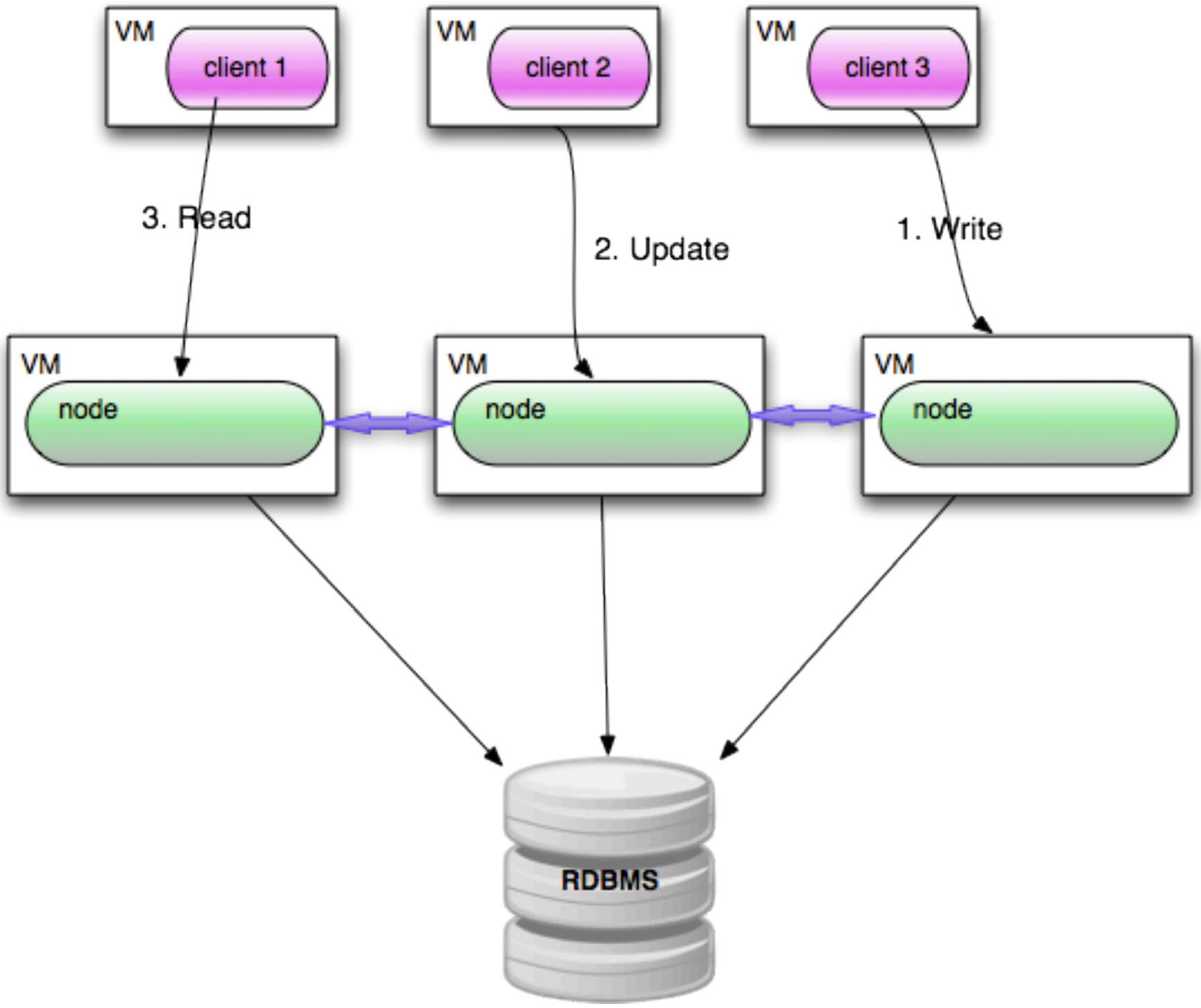


@plmuir

Use Case 3: Data grid



Use Case 3: Data grid



@plmuir

Key features

- Cloud oriented
- Transactions
- Querying
- Map/Reduce and Dist Executors
- Cache loaders
- Management
  - JMX
  - RHQ

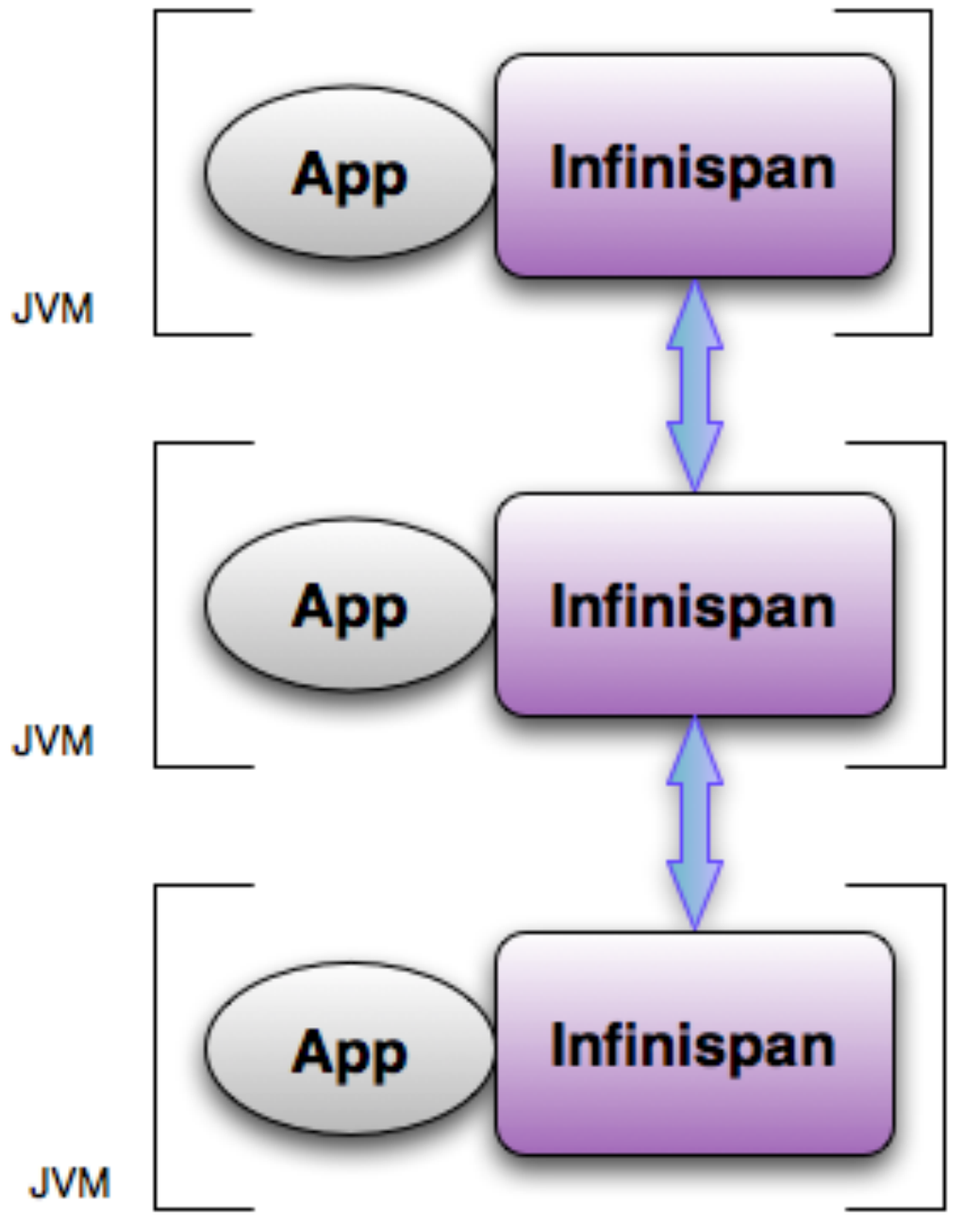
Hands on Demo

@plmuir

Reliable Multipoint Communication

@plmuir

Why do you care?



@plmuir

Shall we try it out?

- In the lab project you'll find a test script for your network. Run it!
  - LAB_HOME/nic-test
- If all goes well, you'll get two windows in which you can draw up on your screen. Draw on one, see it in both.
- Easy to try: JGroups has no dependencies!

What is unreliable ?

- Messages get
  - dropped
    - too big (UDP has a size limit), no fragmentation
    - buffer overflow at the receiver, switch
      - NIC, IP network buffer
  - reordered
- We don't know who is in a cluster (IP multicast)
  - we don't know when a new node joins, leaves, or crashes
- Fast sender might overwhelm slower receiver(s)
  - flow control

So what Is JGroups ?

- Library for reliable cluster communication
- Provides
  - Fragmentation
  - Message retransmission
  - Flow control
  - Ordering
  - Group membership, membership change notification
- LAN or WAN based
  - IP multicasting transport default for LAN
  - TCP transport default for WAN
  - Autodiscovery of cluster members

Terminology

- Message
- Address
- View
- Group topology

Discovery Protocols

- PING, MPING, BPING, ..
- TCP_PING
- JDBC_PING
- S3_PING
- CASSANDRA_PING

Eviction and expiration

@plmuir

Expiration

- Time based
  - lifespan
  - max idle
- Expired entries removed
  - from cache
  - from persistent store (if any)

Configuration

```
<namedCache name="expirationCache">
    <expiration
        wakeUpInterval="500"
        lifespan="60000"
        maxIdle="1000"
    />
</namedCache>
```


Eviction

- Memory is finite
  - something has to give!
- Evict based on data access
- Bounded caches

Eviction strategies

- None (default)
- Unordered
- FIFO
- LRU
- LIRS

LIRS

- Low Inter-reference Recency Set replacement
- Hybrid
  - frequency of access
  - time of the last access

Passivation

- Evict to external store
  - file, database...
- Cheaper than remote access (?)
- Use the right eviction policy
  - keep relevant bits in memory

Configuration

```
<namedCache name="evictionCache">
    <eviction
        maxEntries="5000"
        strategy="FIFO" wakeUpInterval="2000"/>
</namedCache>
```


Tuning eviction

- What eviction policy should I use?
- Measure, don't guess
  - Cache JMX stats
  - hits/misses ratio
- Memory issues?
  - Aggressive wakeup interval

Listeners

@plmuir

Listener types

- Cache listeners
  - data: added, remove, changed, entry loaded
  - transaction: completed, registered
  - topology: changed, data rehashed
- Cache manager listeners
  - cache started/stopped, view changed/merge

Synchronicity

- listener executes in caller's thread (default)
  - keep it short!
- Or async

```
@Listener(sync = false)
public class AuditListener {
    //...
}
```


- Listeners are local
- Can veto an operation
- Participate in transactions
- Do not work on RemoteCacheManager

Transactions

@plmuir

Agenda

- Transactions
  - optimistic/pessimistic
  - JTA support
- XA (or not)
- Recovery
- Deadlock avoidance

Cache types

- Non transactional
- Transactional
  - optimistic
  - pessimistic
  - TransactionManager required
- No mixed-access

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


Transactional caches

- 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)

XA or not?

- XA
  - proper distributed transactions
  - recovery enabled
    - or not

```
<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"/>
```

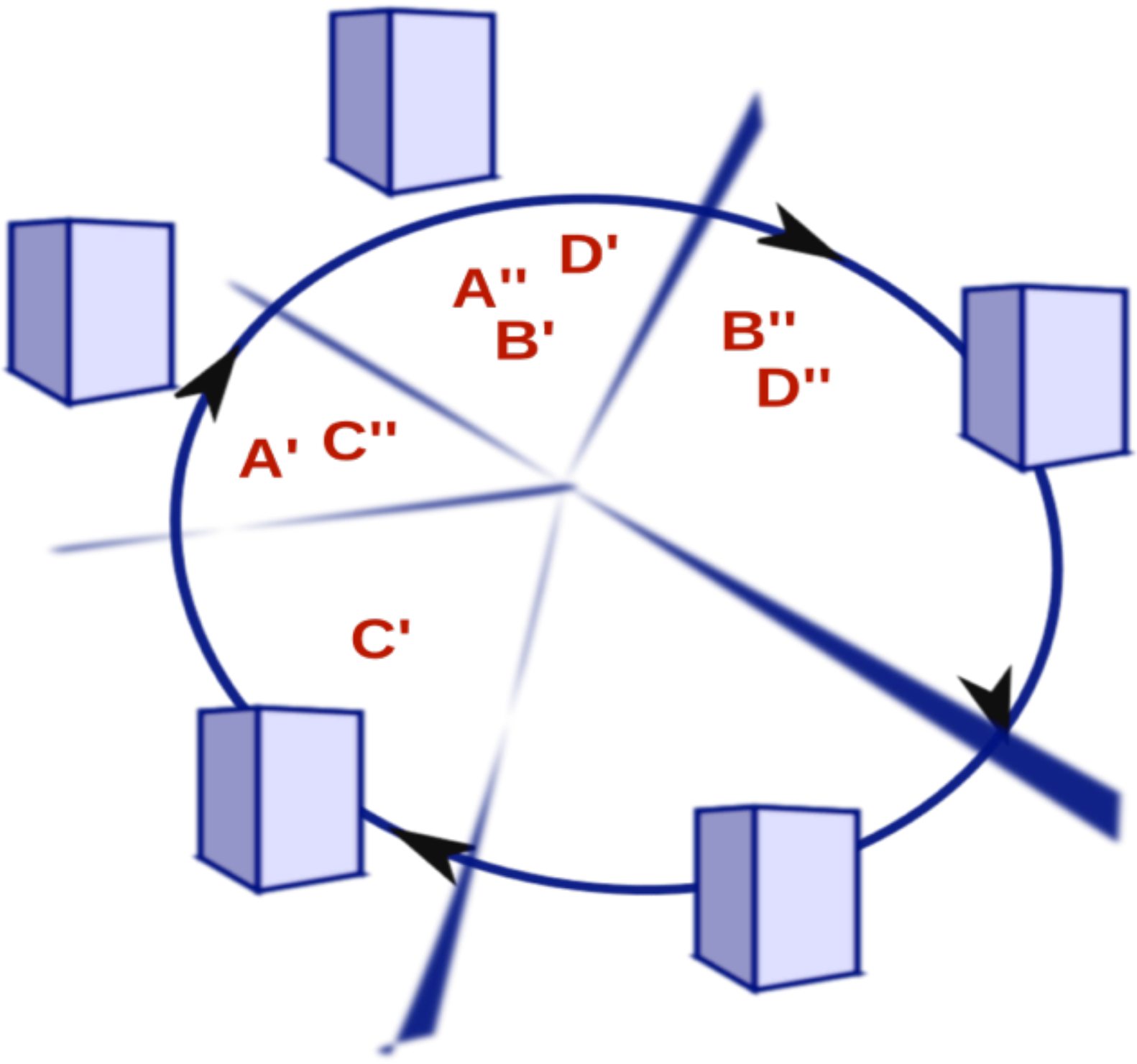

Recovery

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

Modes of Operation

@plmuir

Consistent Hashing: DIST

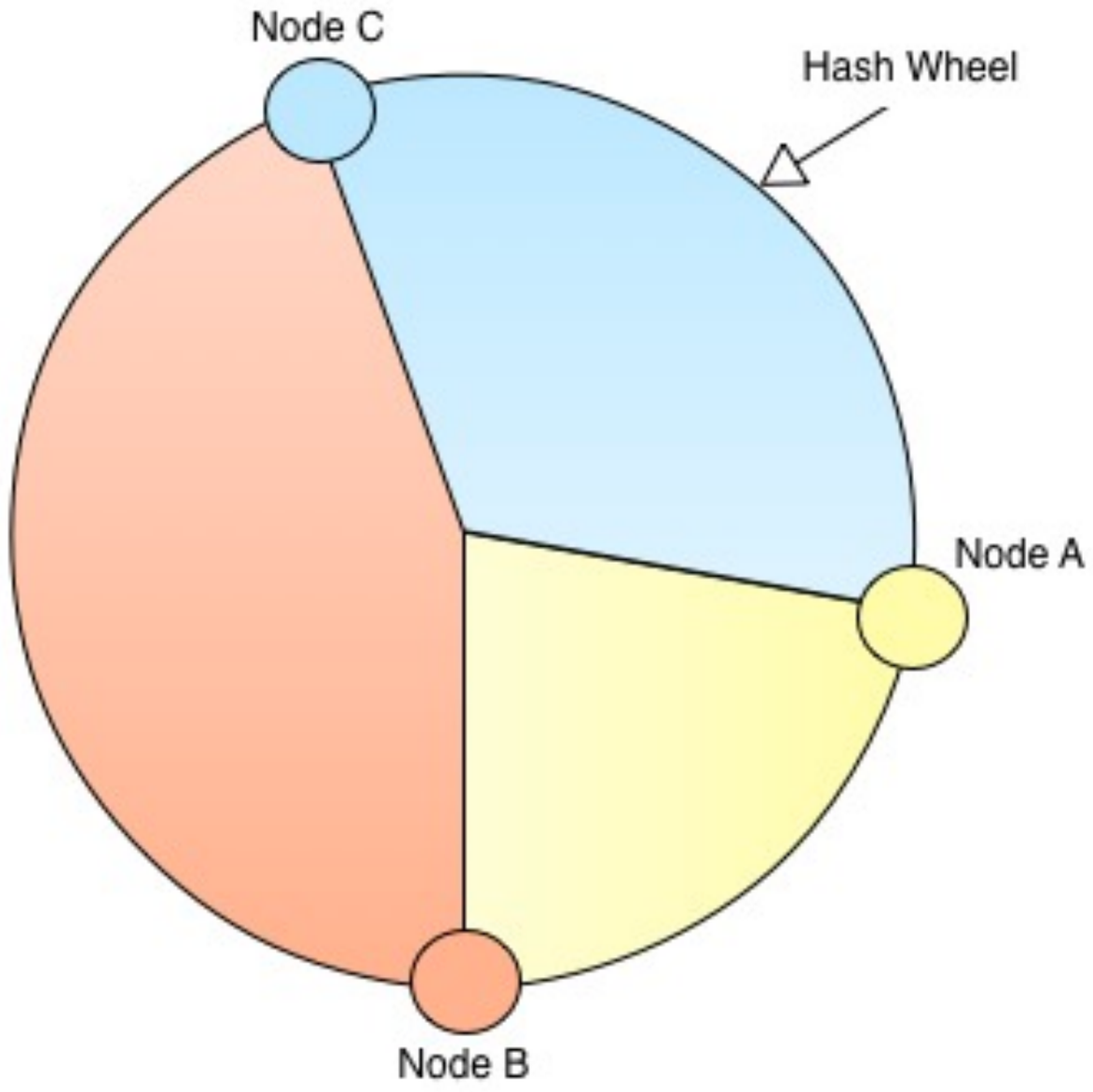


@plmuir

Clustering: Cache modes

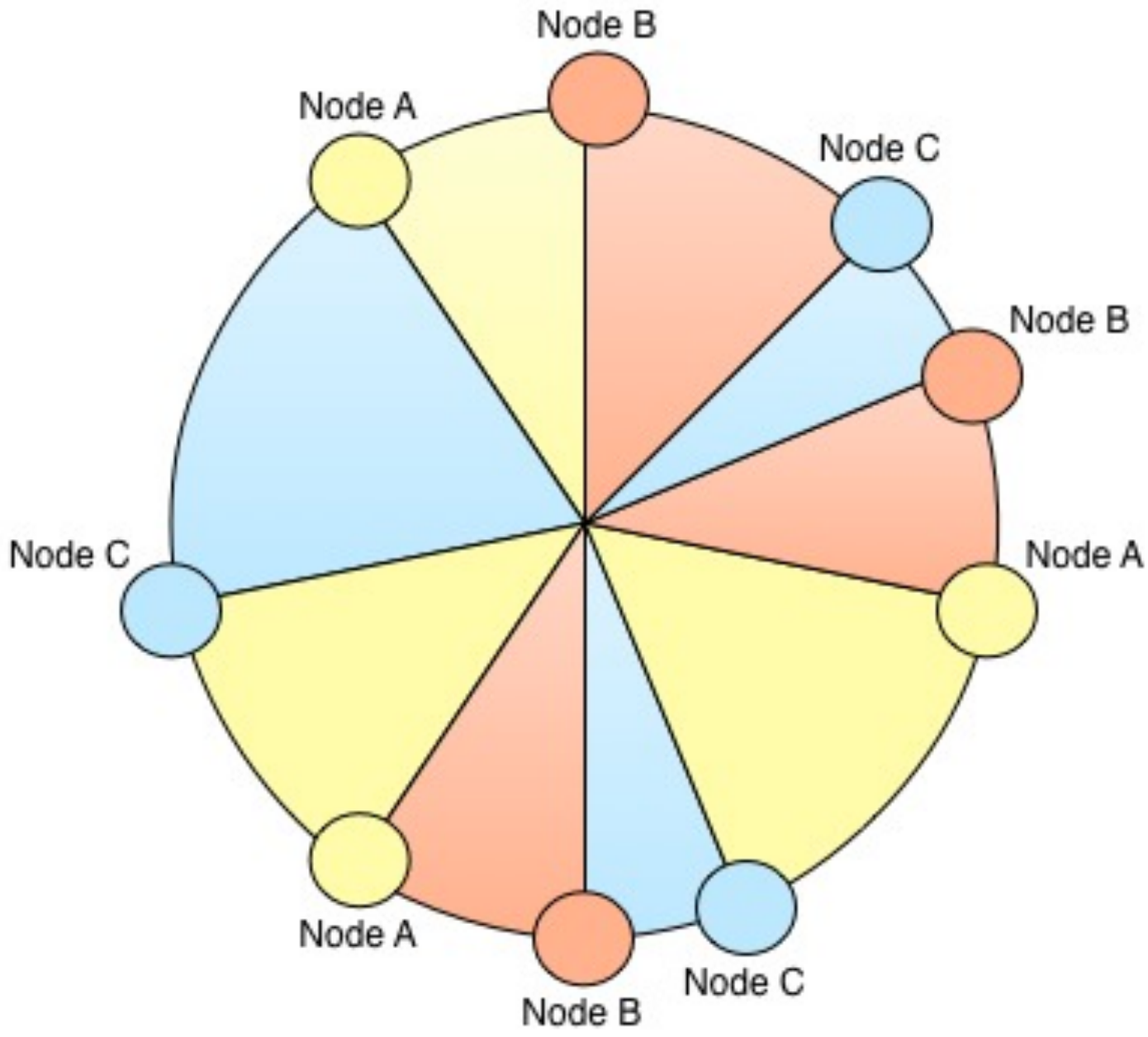
- DIST
  - Sync/Async
- REPL
  - Sync/Async
- LOCAL
  - Doesn't have async
- INV
  - Sync/Async

DIST again



@plmuir

DIST + VNodes

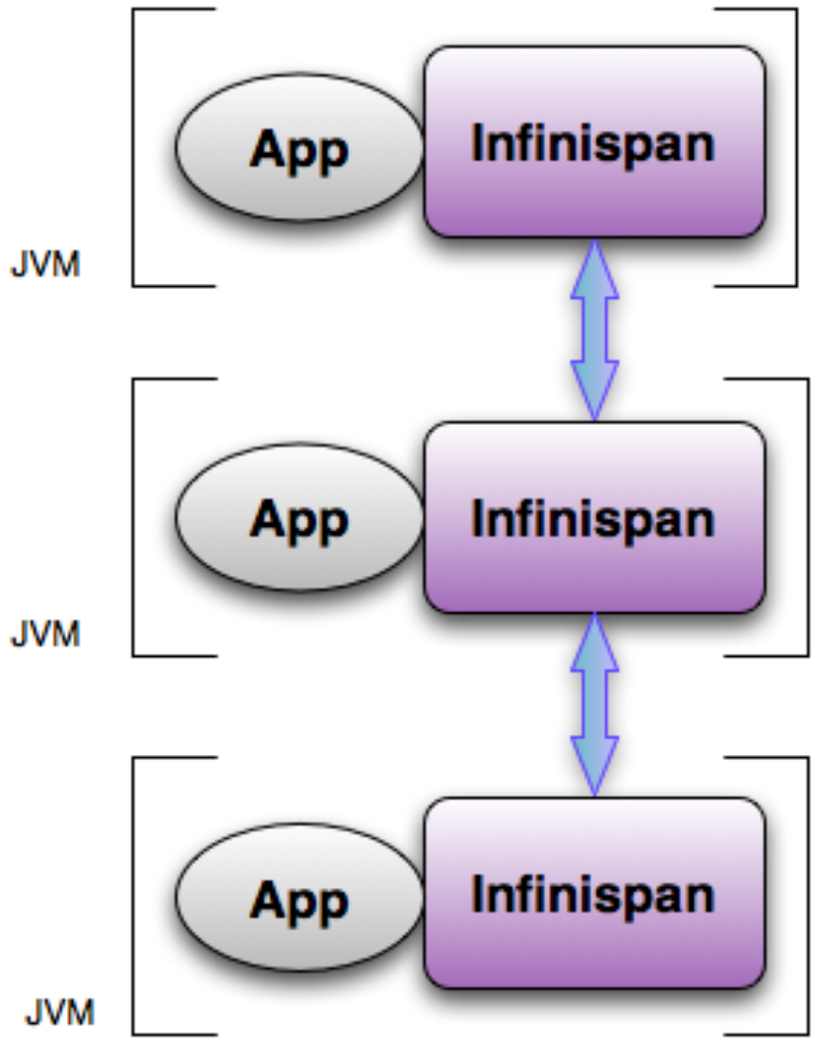


@plmuir

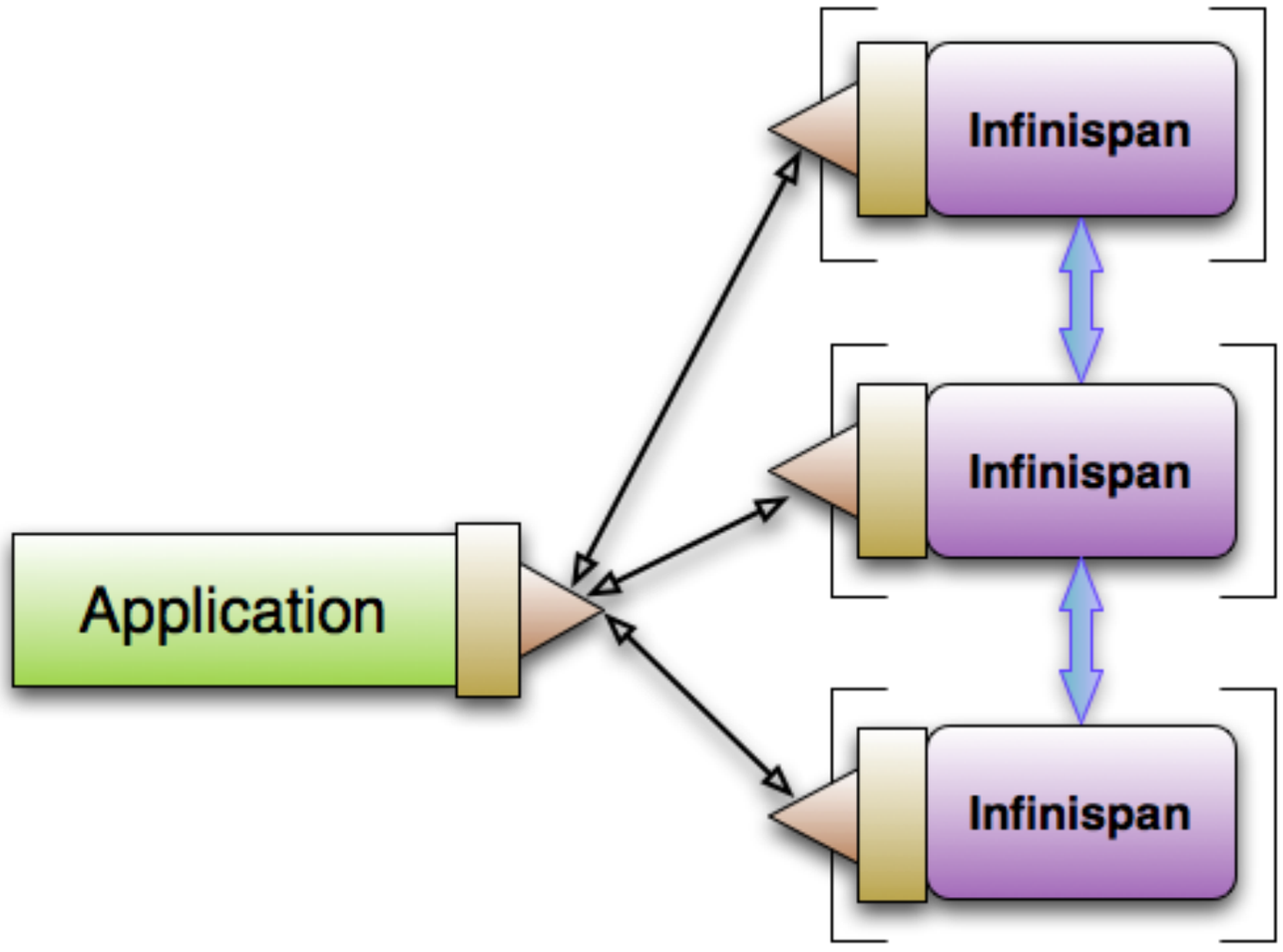
Client Server

@plmuir

Peer to peer



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

Cache Stores

@plmuir

Why use cache stores?

- Durability
- More caching capacity
- Warm caches
    - preload

Features

- Chaining
  - more than one per cache
- Passivation
  - with eviction
- Async
  - write behind
- Shared

Types of cache stores

- File system
  - FileCacheStore
  - BdbjeCacheStore
- JDBC
- Cloud cache store (JCouds)

More cache stores

- RemoteCacheStore
  - use Hotrod
- Cassandra
- ClusterCacheStore
  - alternative to state transfer
- Custom!

Extras

@plmuir

Querying

@plmuir

To query a Grid

- What's in C7 ?

```
Object p =
    cache.get("c7");
```

- Where is the white King?

@plmuir

Infinispan and Queries

- How to query the grid
  - Key access
  - Statistics
  - Map/Reduce
  - Indexing of stored objects
- Integrate with existing search engines
  - Scale
  - Highly available

Indexing of stored objects

- Maven module: infinispan-query
- Configuration: indexing=true
    - Will trigger on annotated objects
- Integrates hibernate-search-engine
- Based on Apache Lucene

Enable indexing

```
Configuration c = new Configuration()
        .fluent()
        .indexing()
        .addProperty(
  "hibernate.search.option", "value" )
        .build();
CacheManager manager = new DefaultCacheManager(c);
```

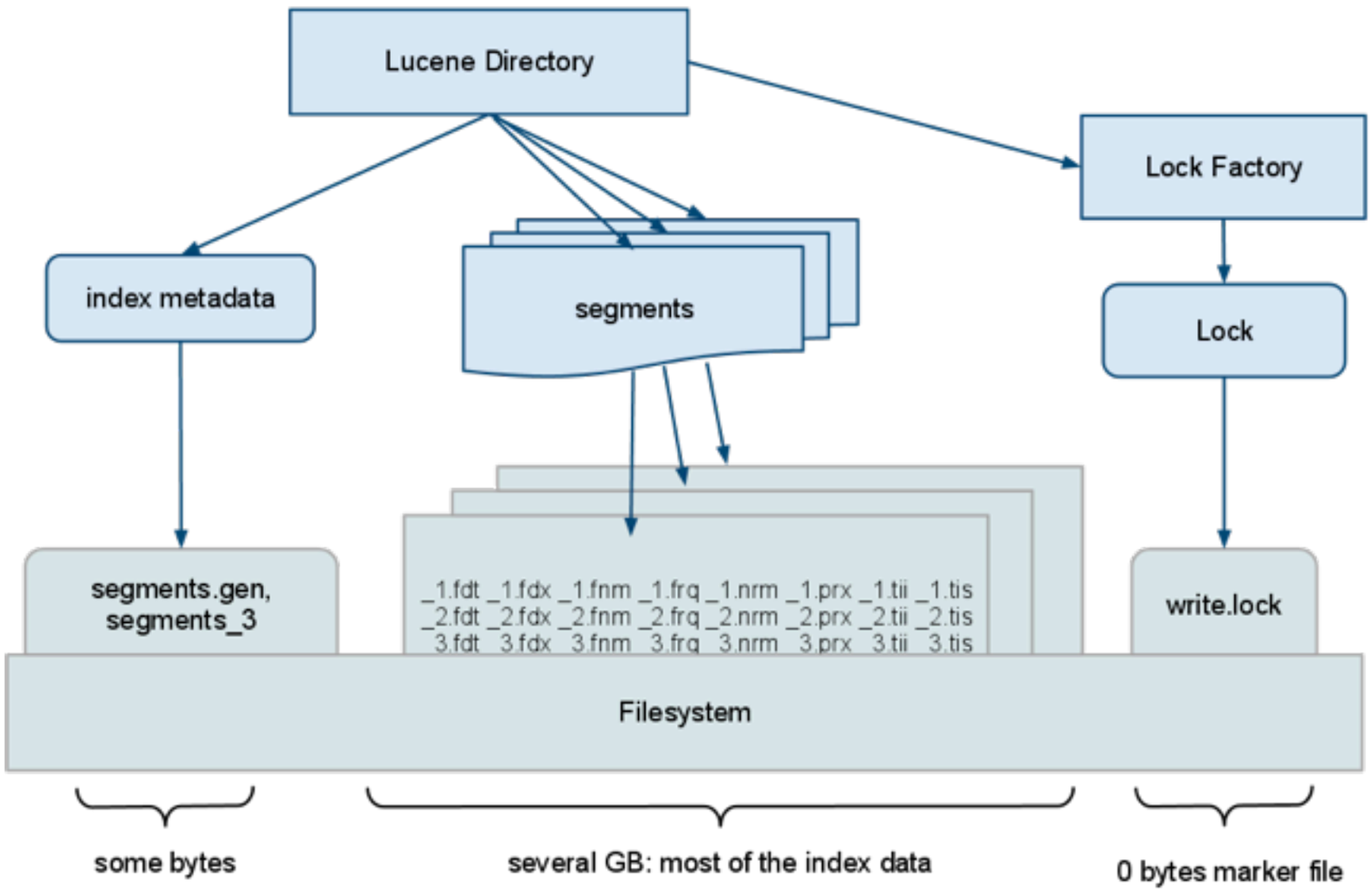

Annotate your objects

```
@ProvidedId @Indexed
public class Book implements Serializable {

    @Field String title;
    @Field String author;
    @Field String editor;
    ...

}
```

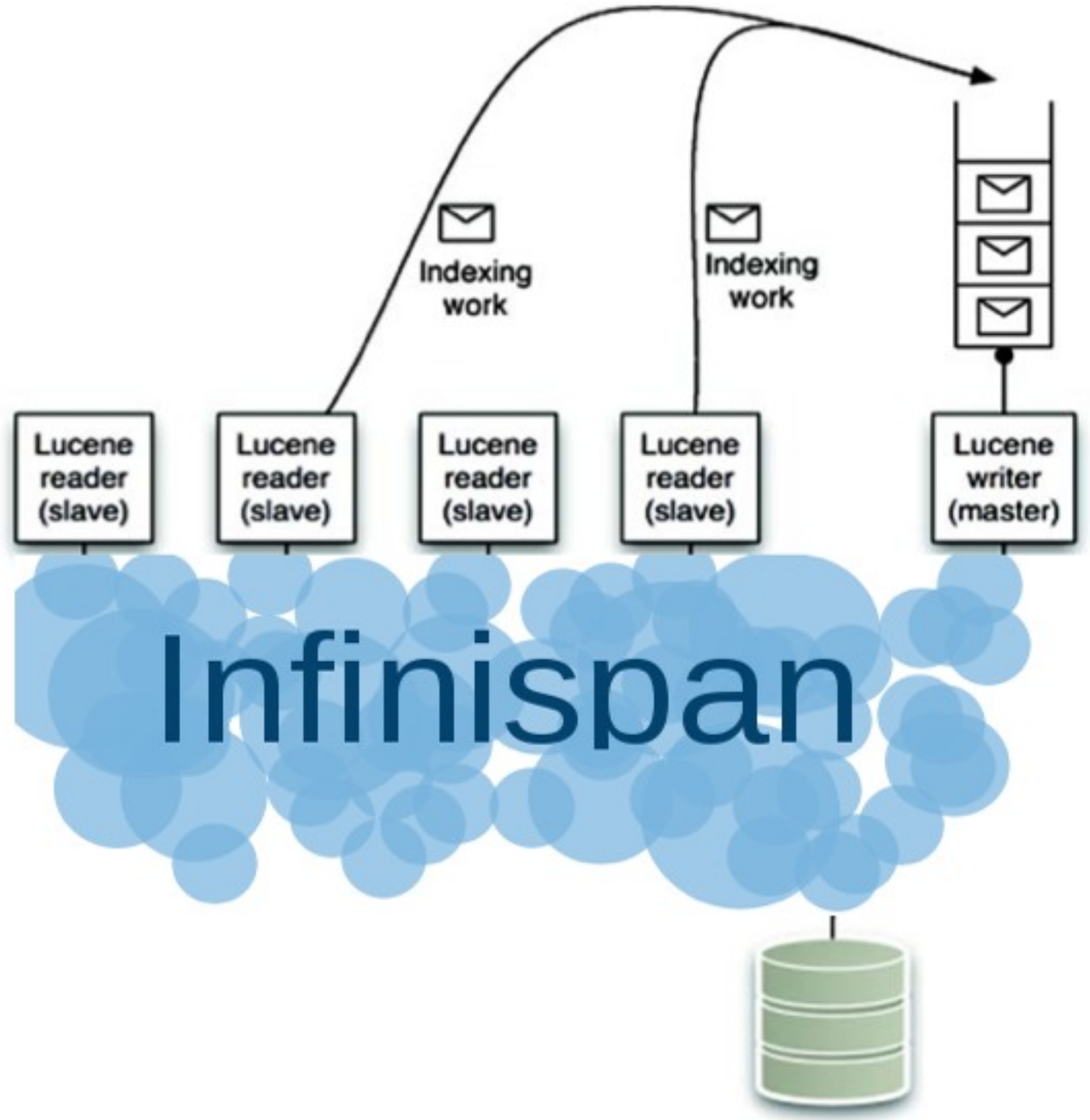

Search them!

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

Query query = sm.buildQueryBuilderForClass(Book.class)
    .get()
        .phrase()
            .onField("title")
            .sentence("in action")
    .createQuery();

List<Object> list = sm.getQuery(query).list();
```

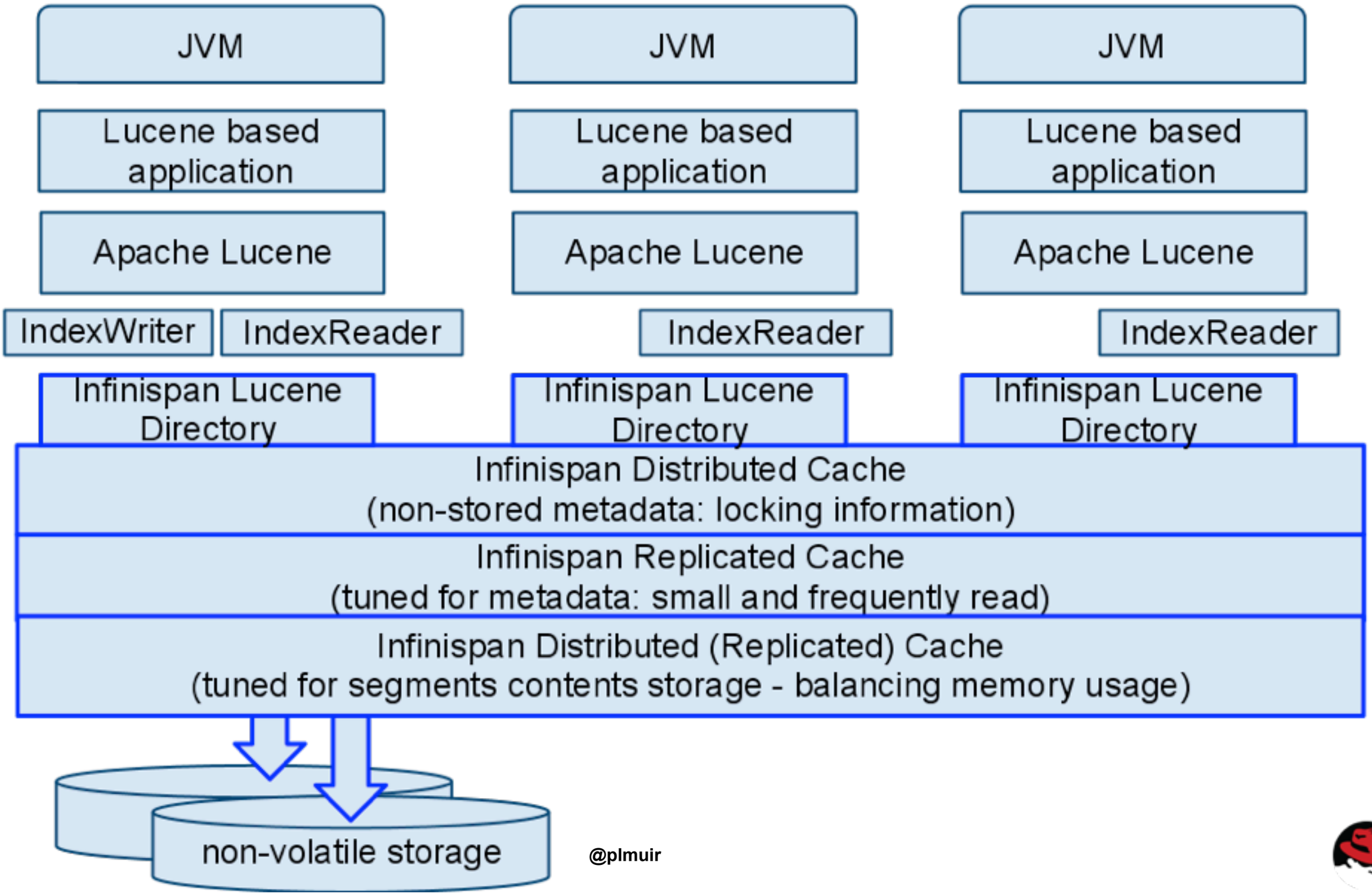

Lucene API, storing in Infinispan



Limited write concurrency



Example of multi-cache app



@plmuir

@plmuir

- OGM: Object/Grid Mapper
- Implements JPA for NoSQL engines
  - Infinispan as first supported “engine”
  - More coming
- Simplified migration across different NoSQL, SQL databases
  - With transactions, or whatever is possible.
  - Fast? Contribute tests and use cases!

- JPA on NoSQL: an approach with Hibernate OGM
  - Devoxx 2011
    - November 17th (conf Day 4) - 14:00 – 15:00
    - Emmanuel Bernard

@plmuir

Conclusion

@plmuir

Use Cases

- Local Cache
- Distributed Cache
- Data Grid

Access Modes

- Embedded
- Remote
    - Hot Rod
    - REST
    - Memcache

Control

- Eviction
- Expiration
- Management

Transaction & Locking

- XA
- Local

Persistence

- Cache Stores

Q&A

@plmuir