JUDCon

JBoss Users & Developers Conference

Boston : 2010

Infinispan's Hot Rod Protocol

Galder Zamarreño
Senior Software Engineer, Red Hat
21st June 2010

Who is Galder?

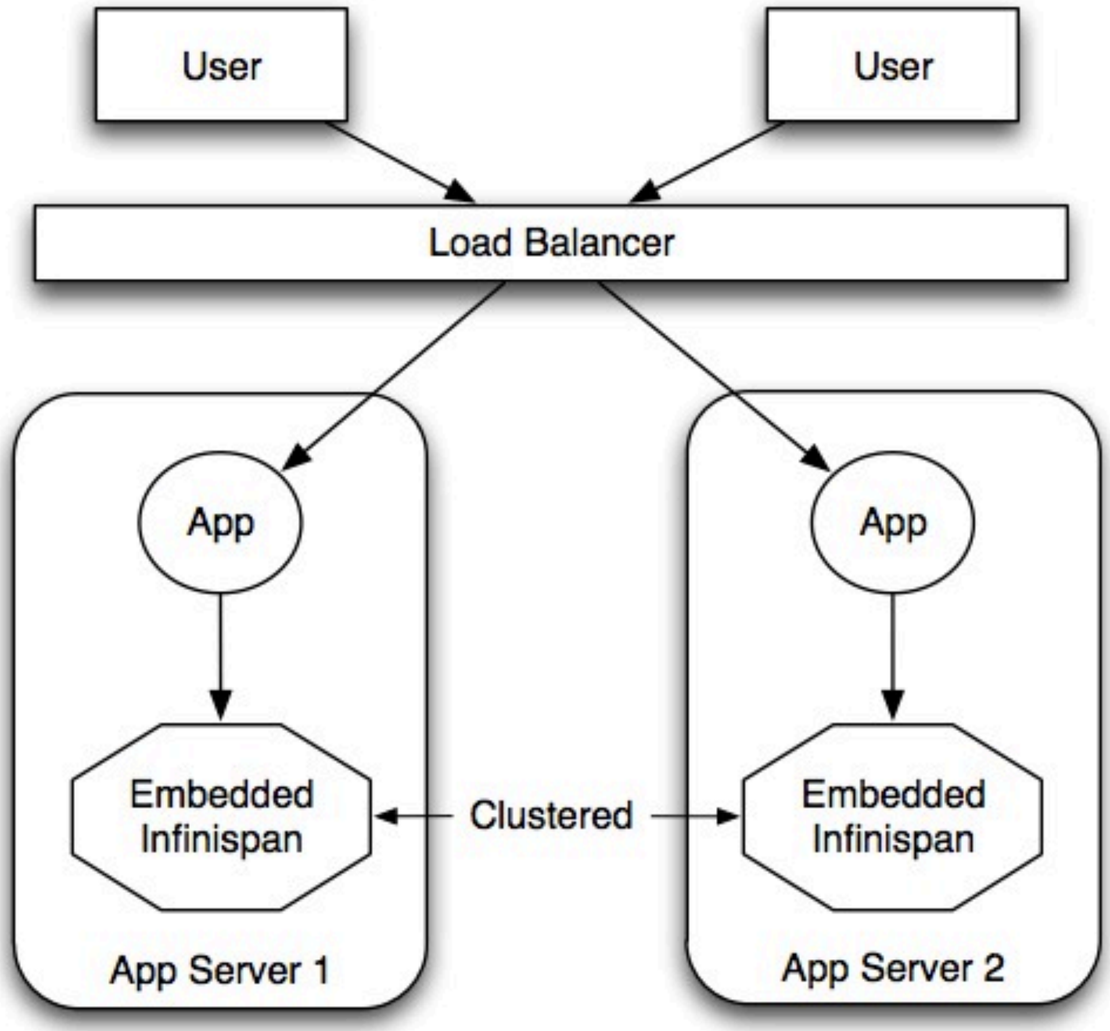
- Core R&D engineer on Infinispan and JBoss Cache
- Contributor and committer on JBoss AS, Hibernate, JGroups, JBoss Portal,...etc

Agenda

- Infinispan peer-to-peer vs Infinispan client-server mode
- What is Hot Rod
- The motivations behind Hot Rod
- Hot Rod implementations and sample code
- Infinispan server comparison
- The path ahead for Hot Rod
- Demo

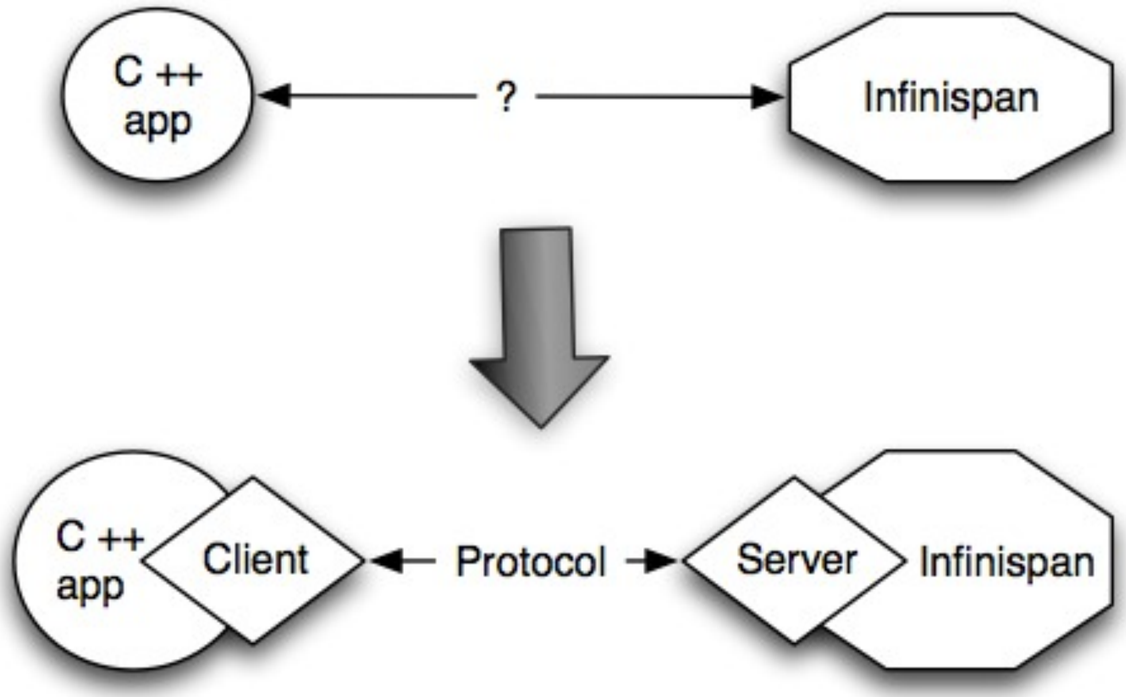
Infinispan Peer-To-Peer

- Infinispan is an in-memory distributed data grid
- Traditionally, deployed in peer-to-peer (p2p) mode



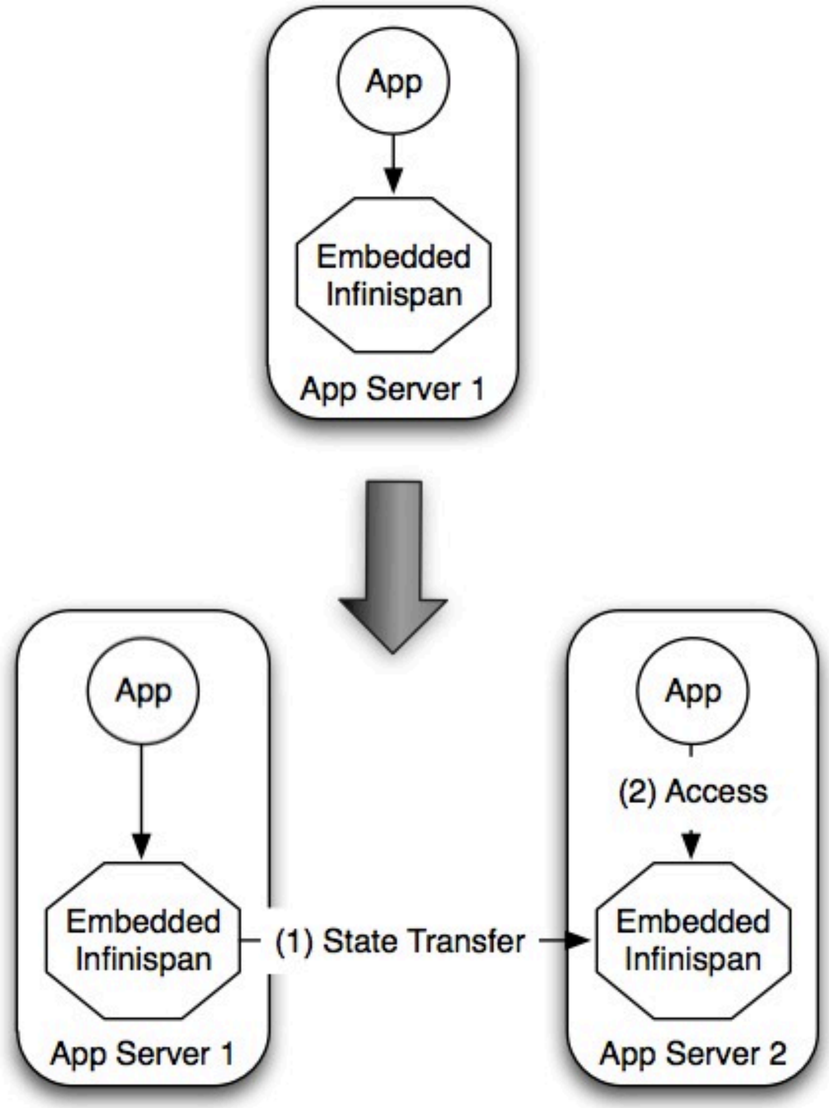
Infinispan Client-Server

- Sometimes client-server makes more sense
- E.g., access from non-JVM environment
- No Infinispan running on client



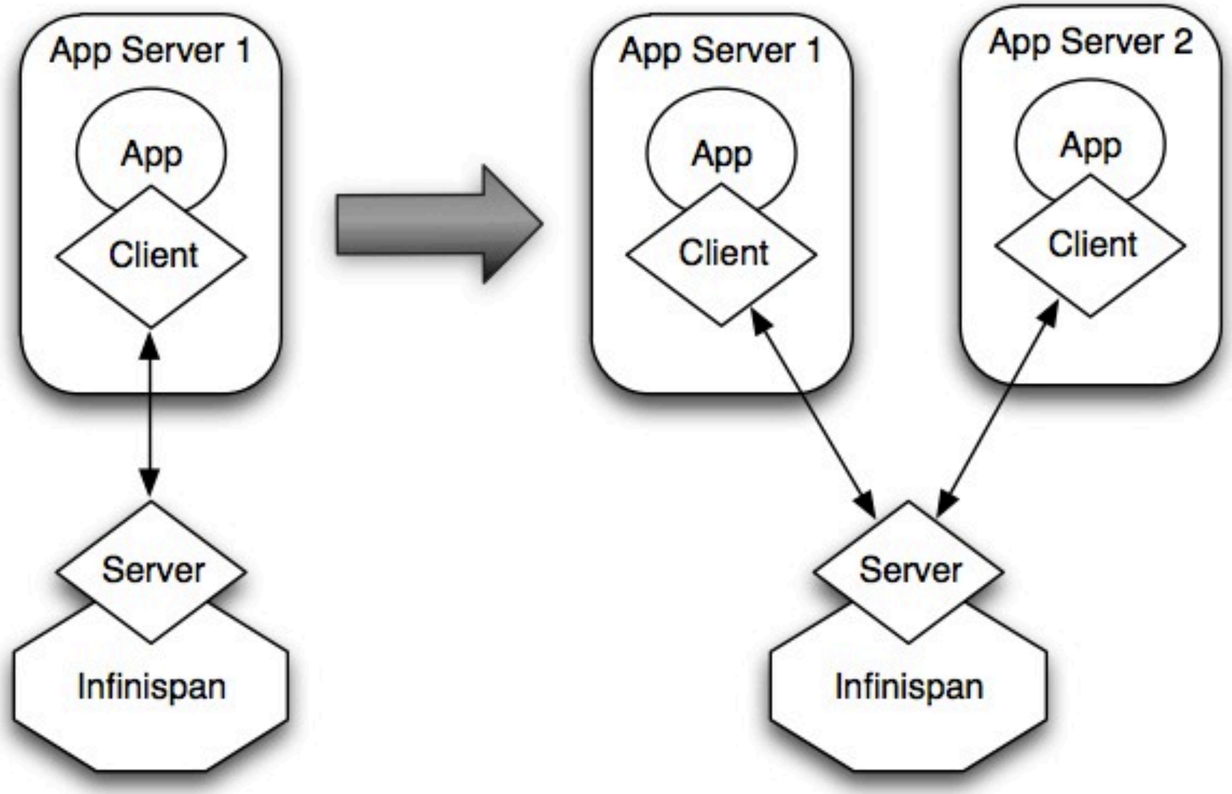
Infinispan Client-Server

- P2P data grids do not get along with elastic application tiers



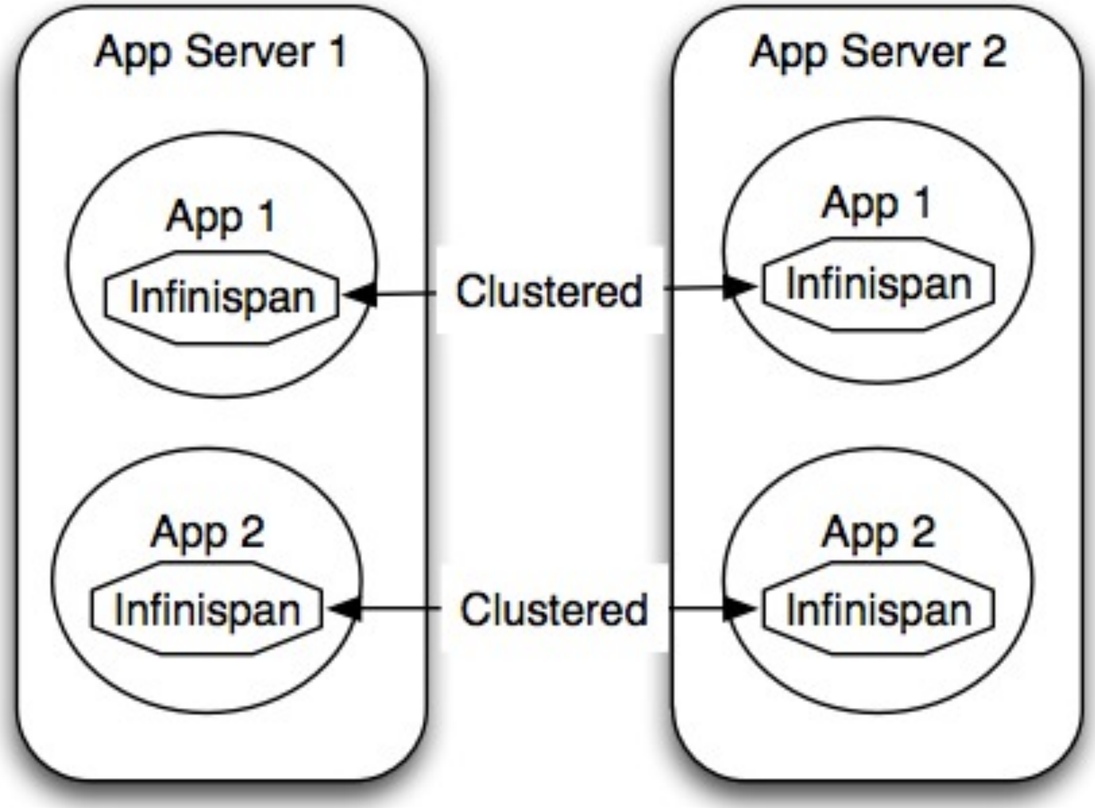
Infinispan Client-Server

- Elastic application tiers work better with client-server



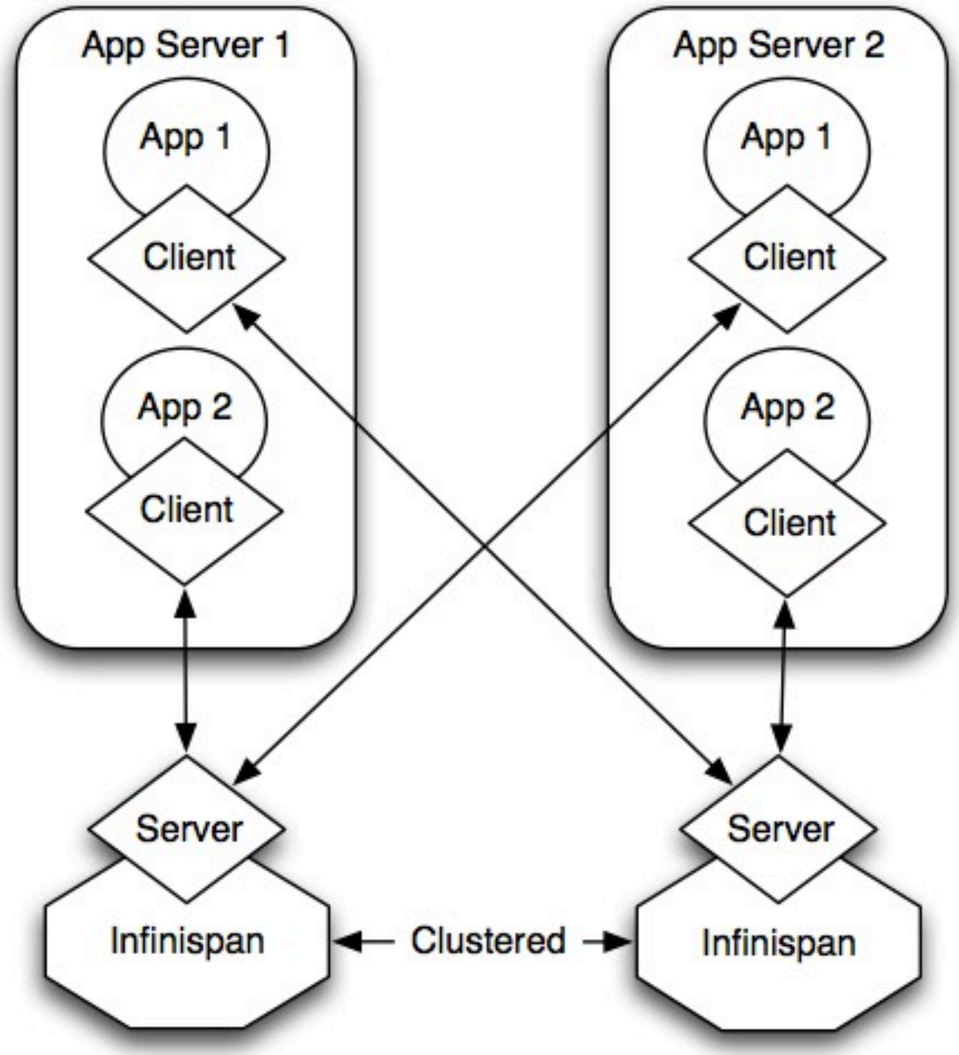
Infinispan Client-Server

- Multiple applications with data storage needs
  - Starting a data grid per app is wasteful



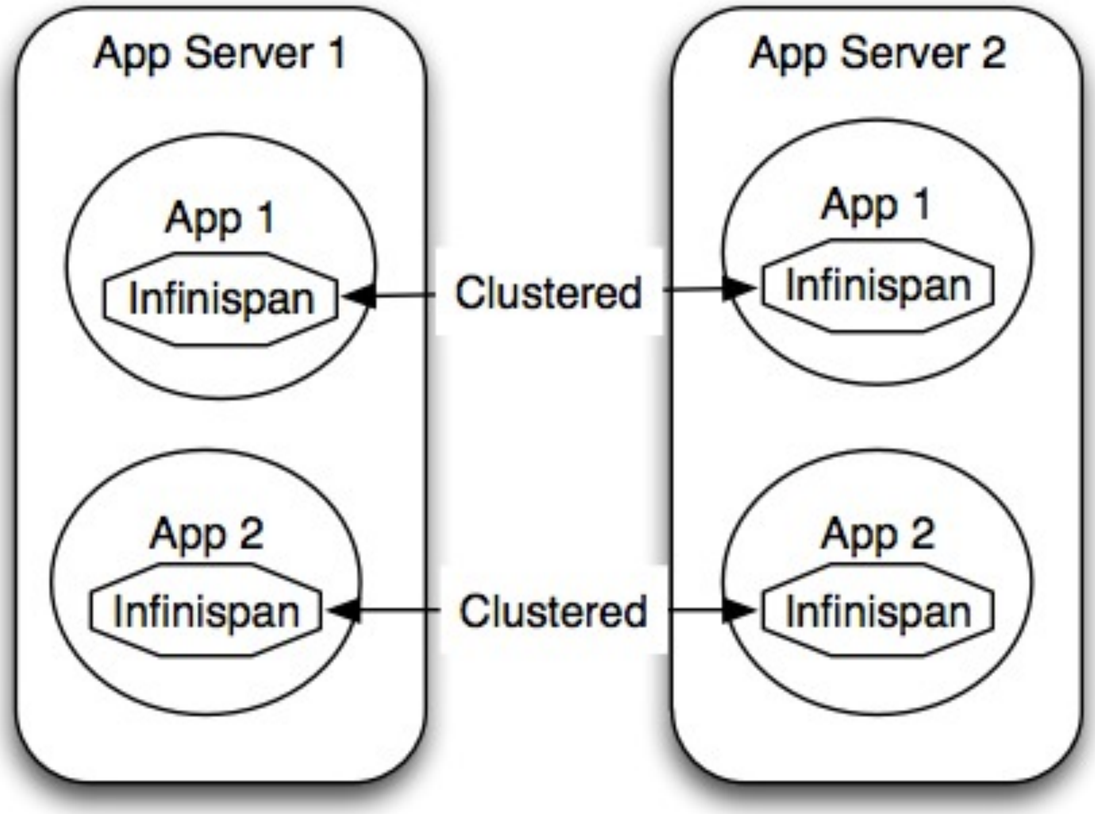
Infinispan Client-Server

- Data service tier
  - Keep a pool of data grid nodes as shared storage tier

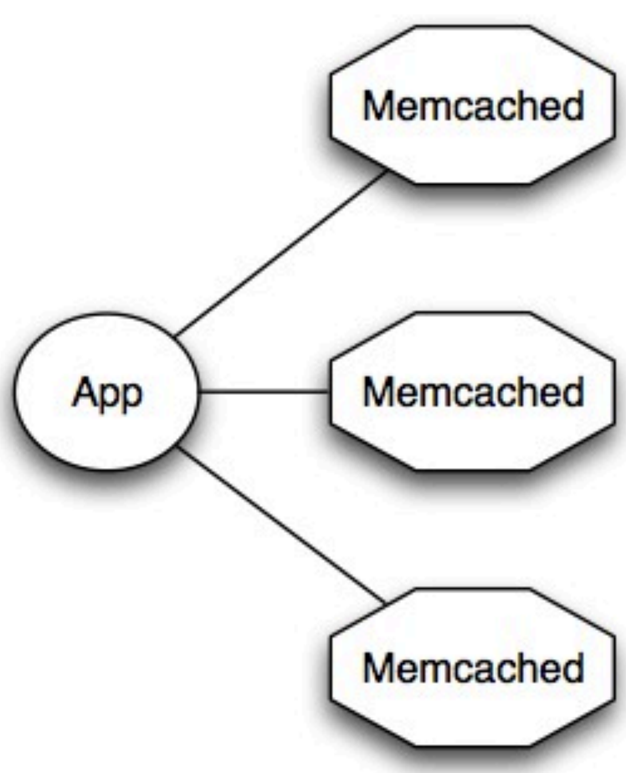


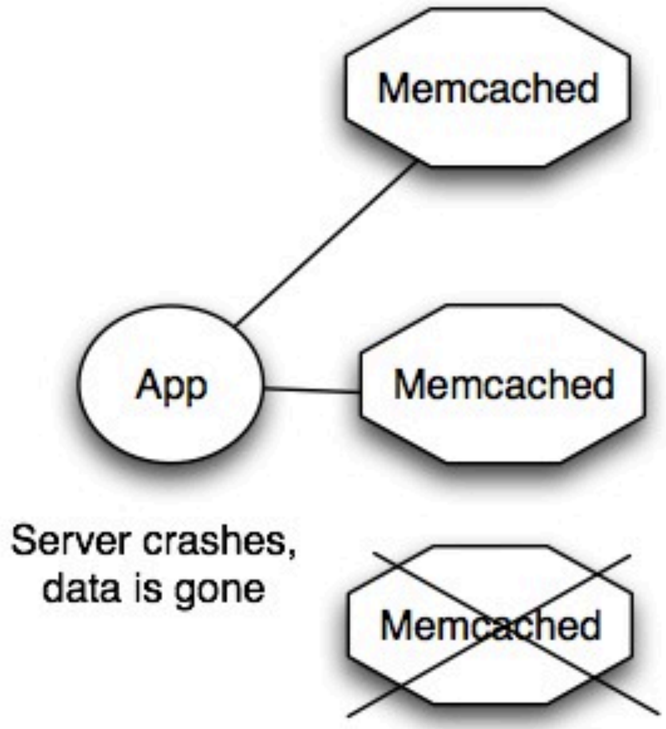
Infinispan Client-Server

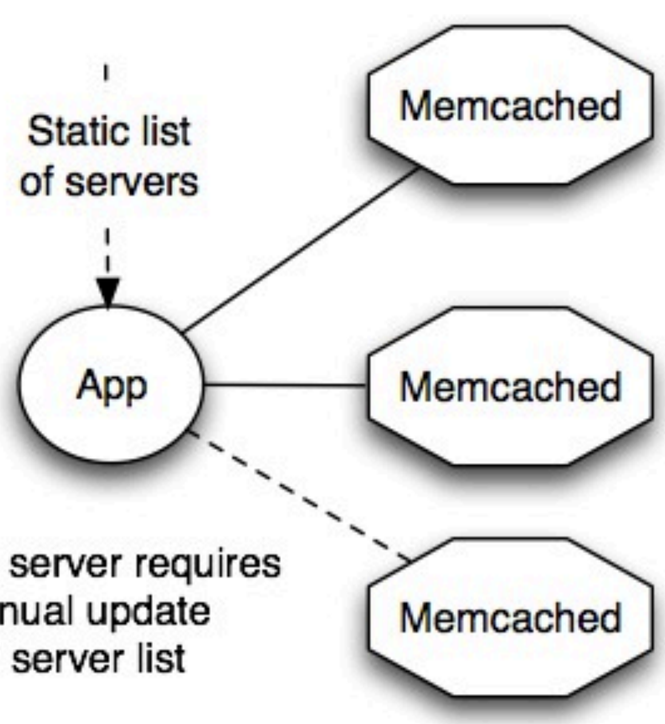
- More examples:
  - Independent tier management
    - E.g., upgrading AS without bringing down DB
  - Contrasting JVM tuning needs - CPU vs Memory
  - Security



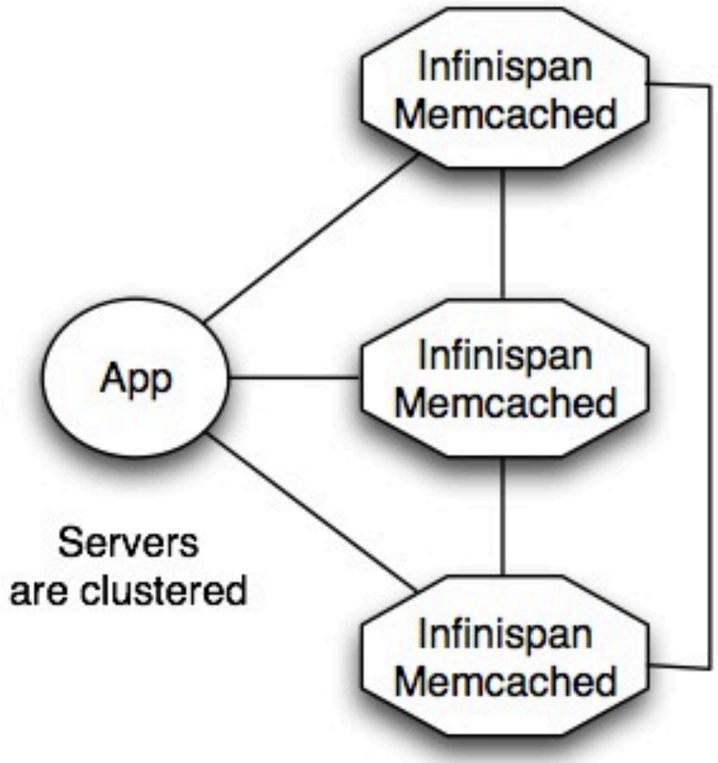
Client-Server with Memcached

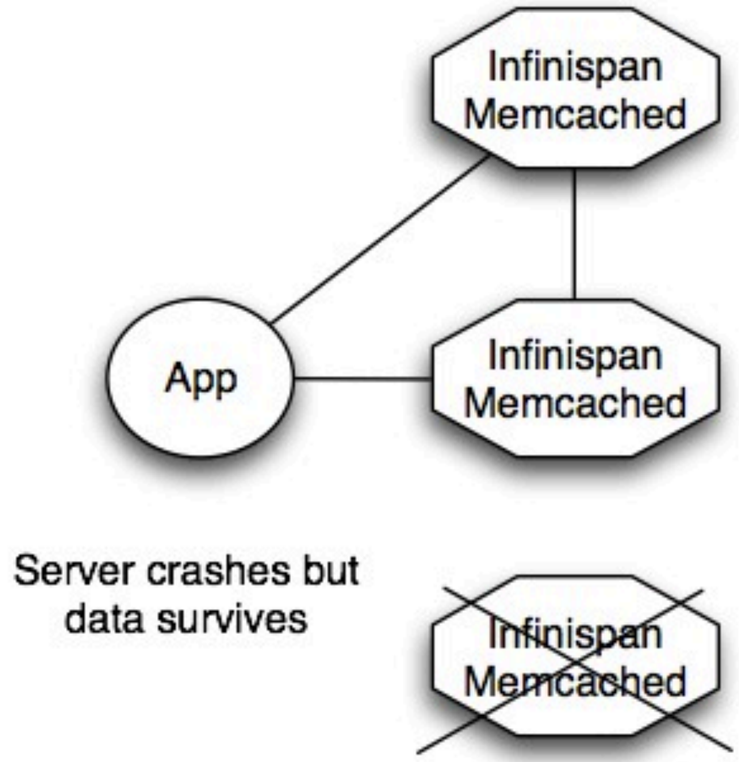


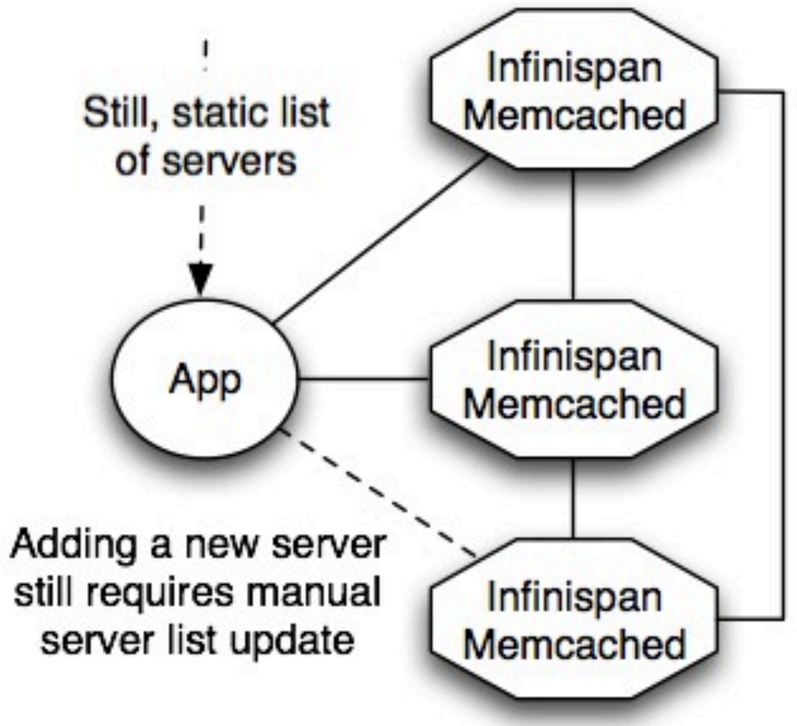




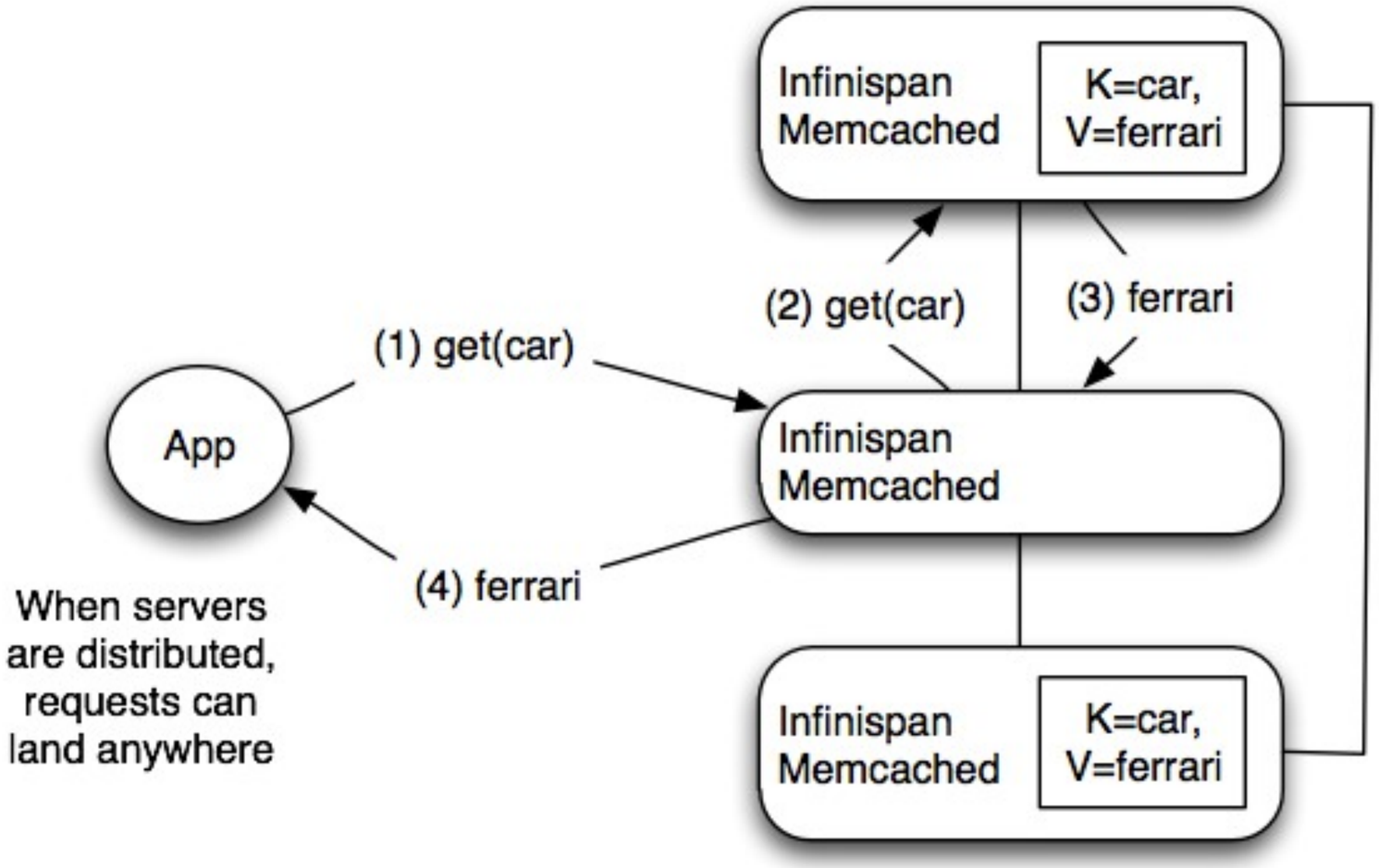
Client-Server with Infinispan Memcached







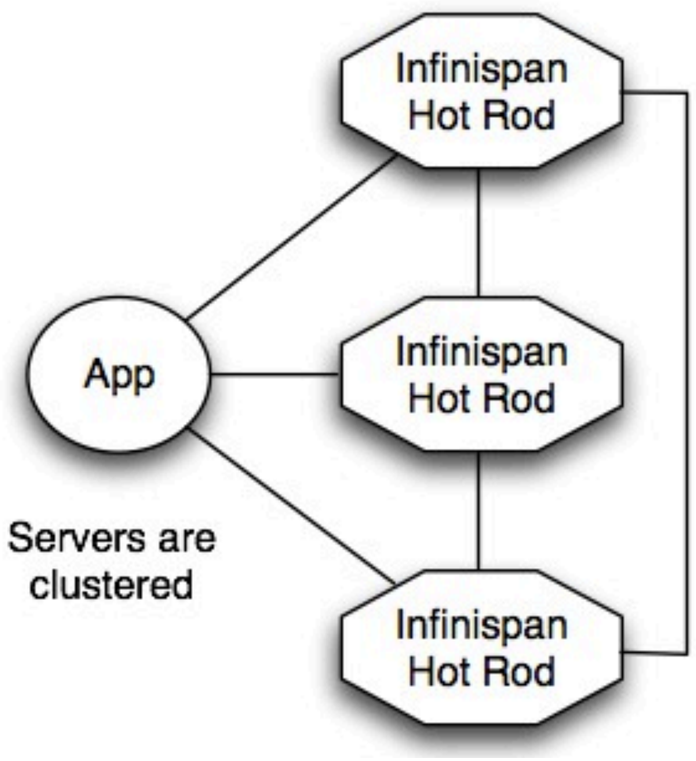
Client-Server with Infinispan Memcached

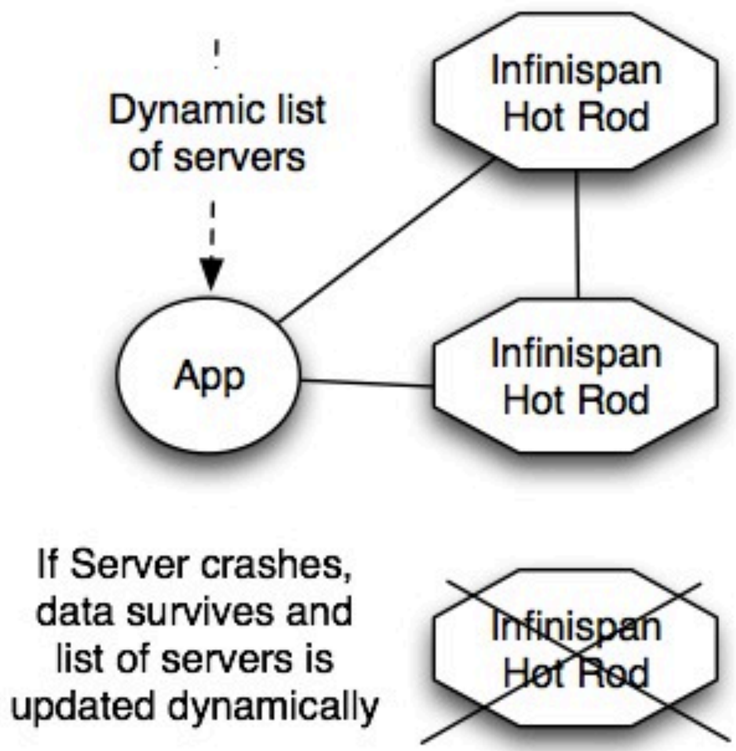


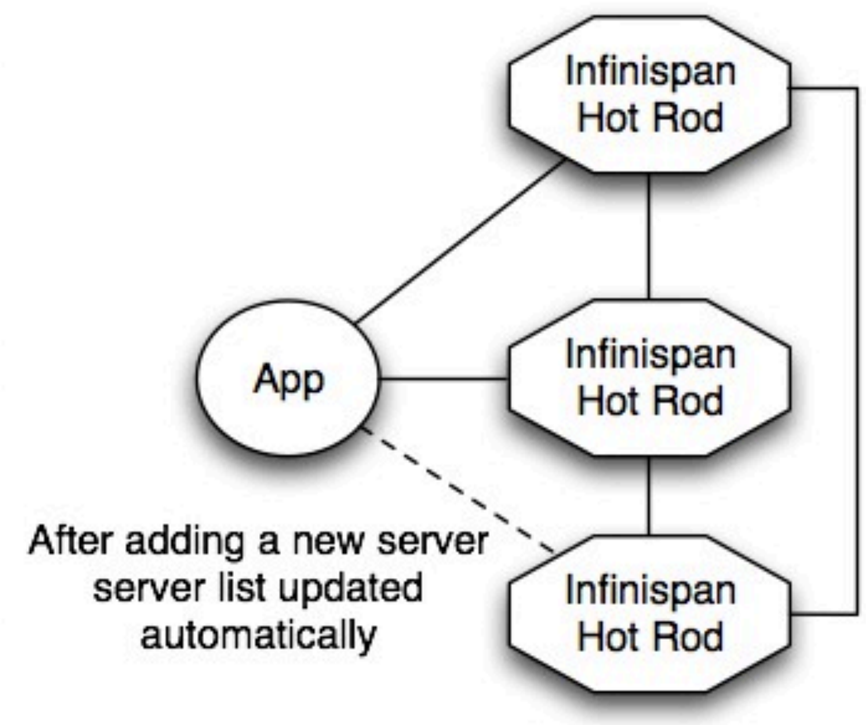
What is Hot Rod?

- Hot Rod is Infinispan's binary client-server protocol
- Protocol designed for smart clients, which have the ability to:
  - Load balance and failover dynamically
  - Smartly route requests

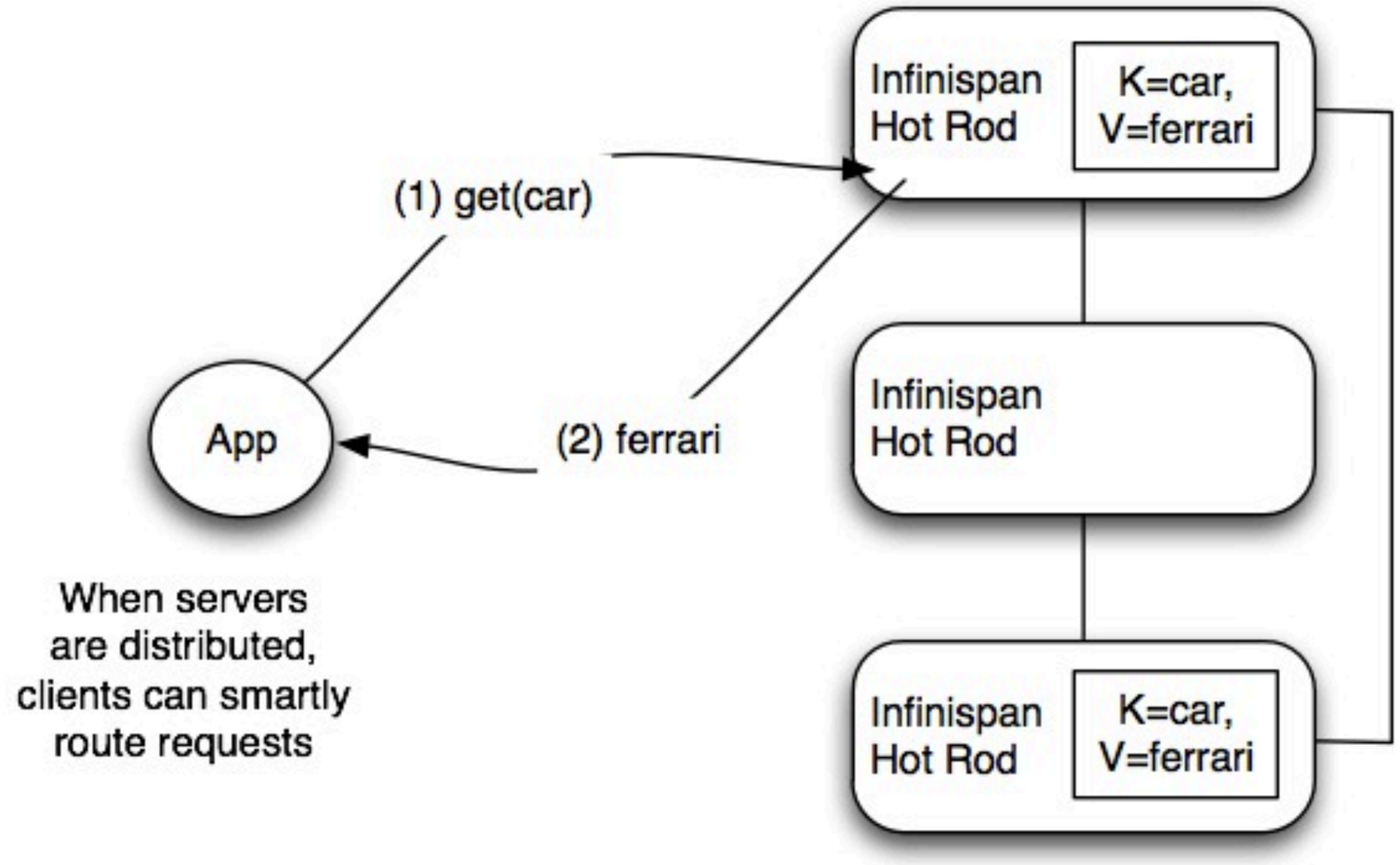
Client Server with Hot Rod







Client Server with Hot Rod



Galder Zamarreño | galder@jboss.org

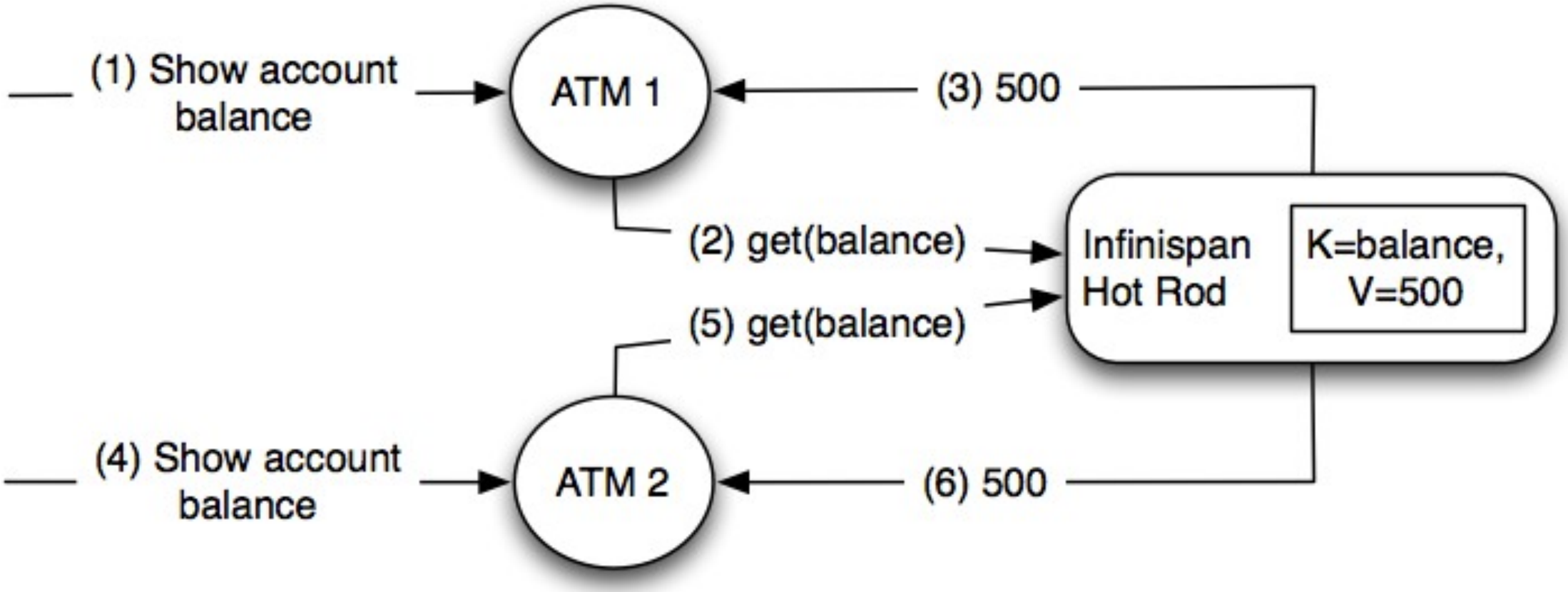
The Hot Rod Protocol

- Transmitted keys and values treated as byte[]
  - To ensure platform neutral behaviour
- Each operation prepended with cache name
- Basic operations:
  - `put, get, remove, containsKey, putIfAbsent, replace, clear`
  - `stats, ping`

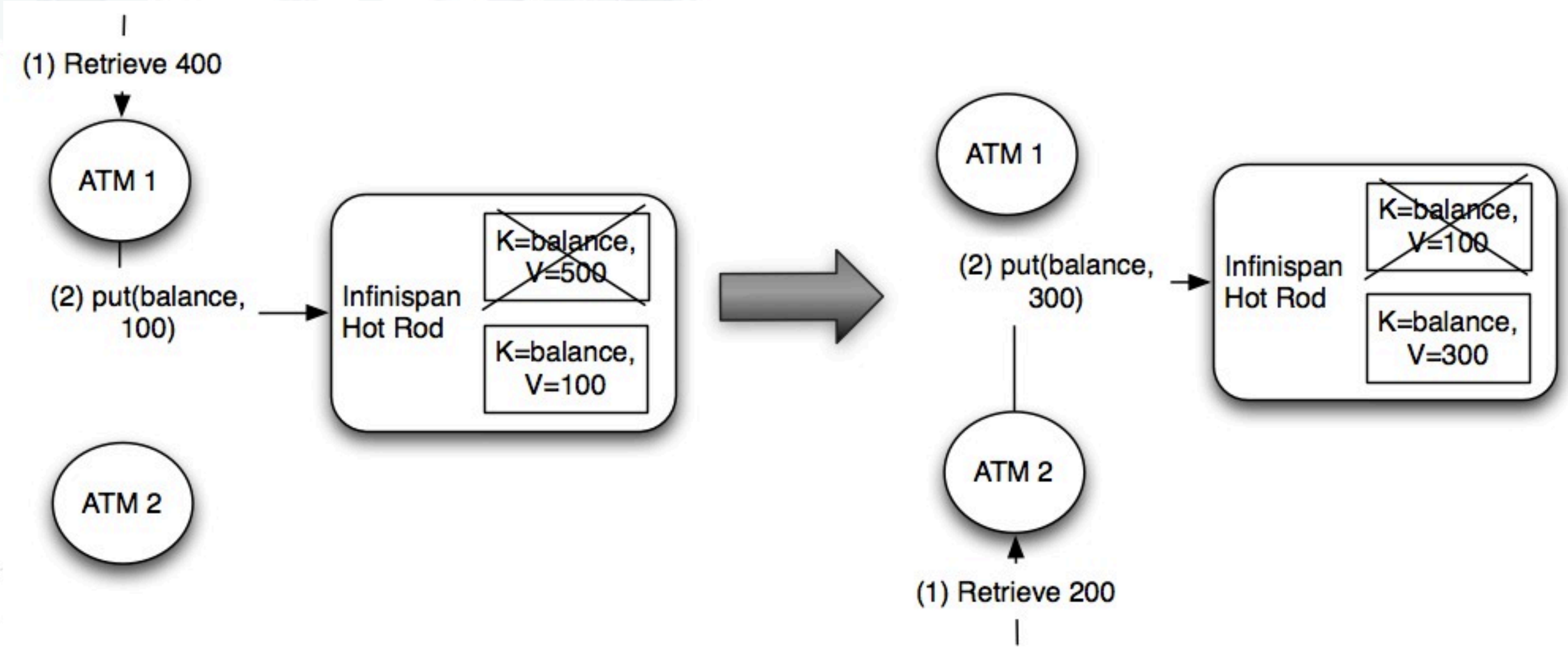
Data Consistency

- Concurrently accessed structures can suffer data consistency issue
- Normally solved with JTA
- No JTA in Hot Rod (yet)
- Versioned API as solution

Data Consistency Problem

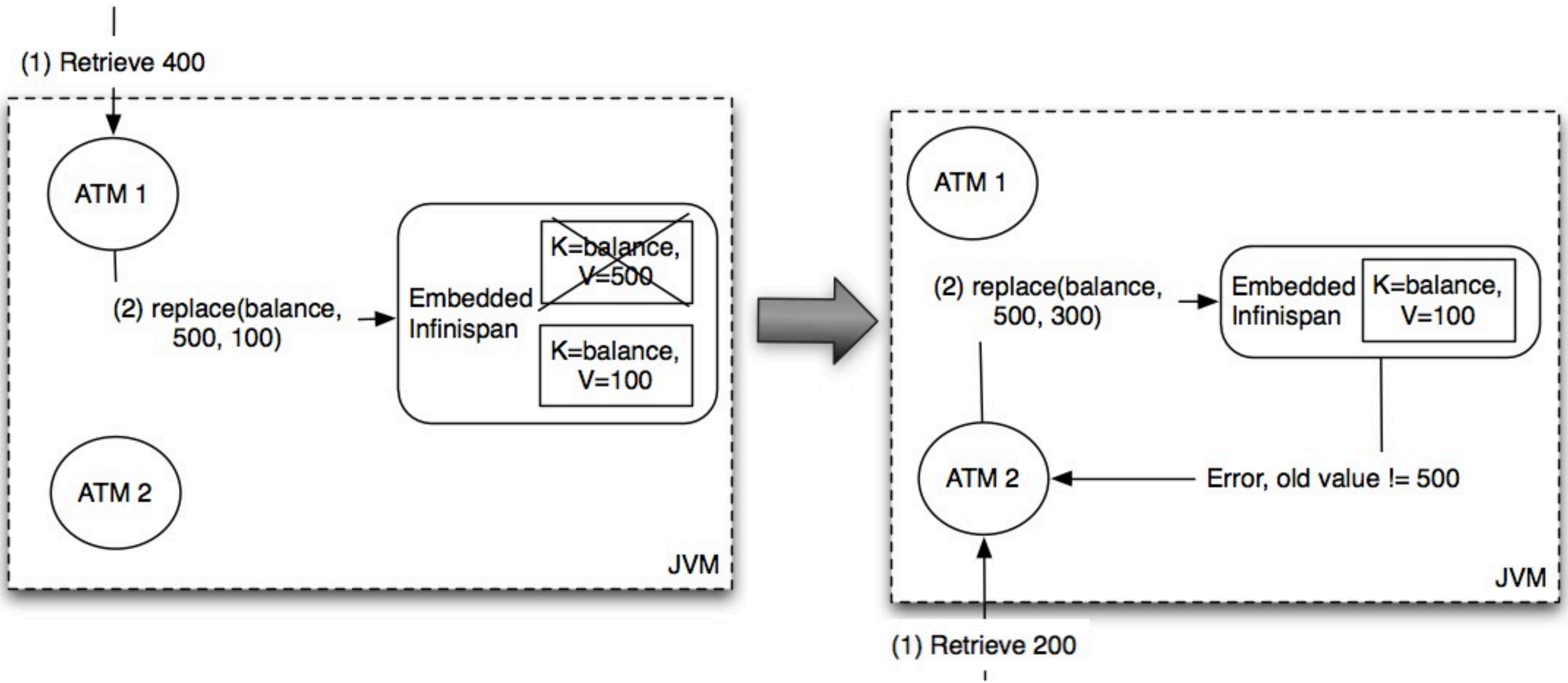


Data Consistency Problem

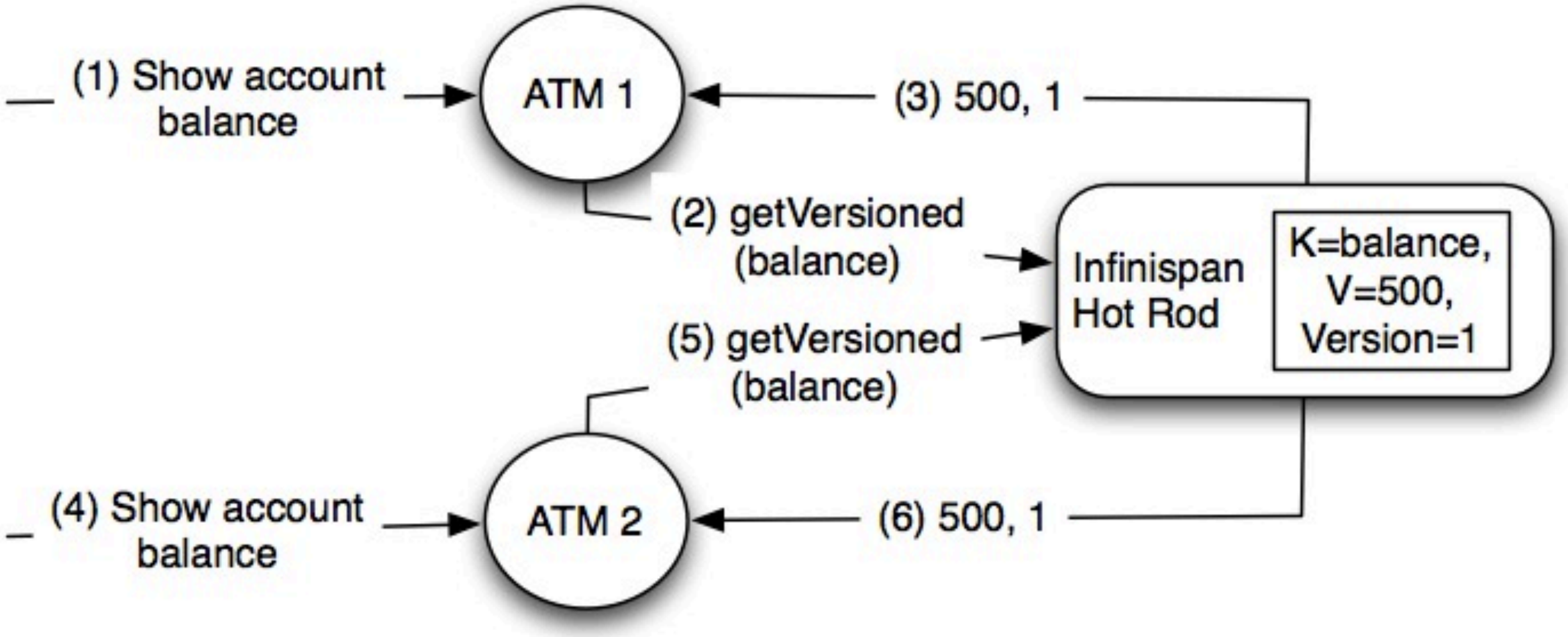


Galder Zamarreño | galder@jboss.org

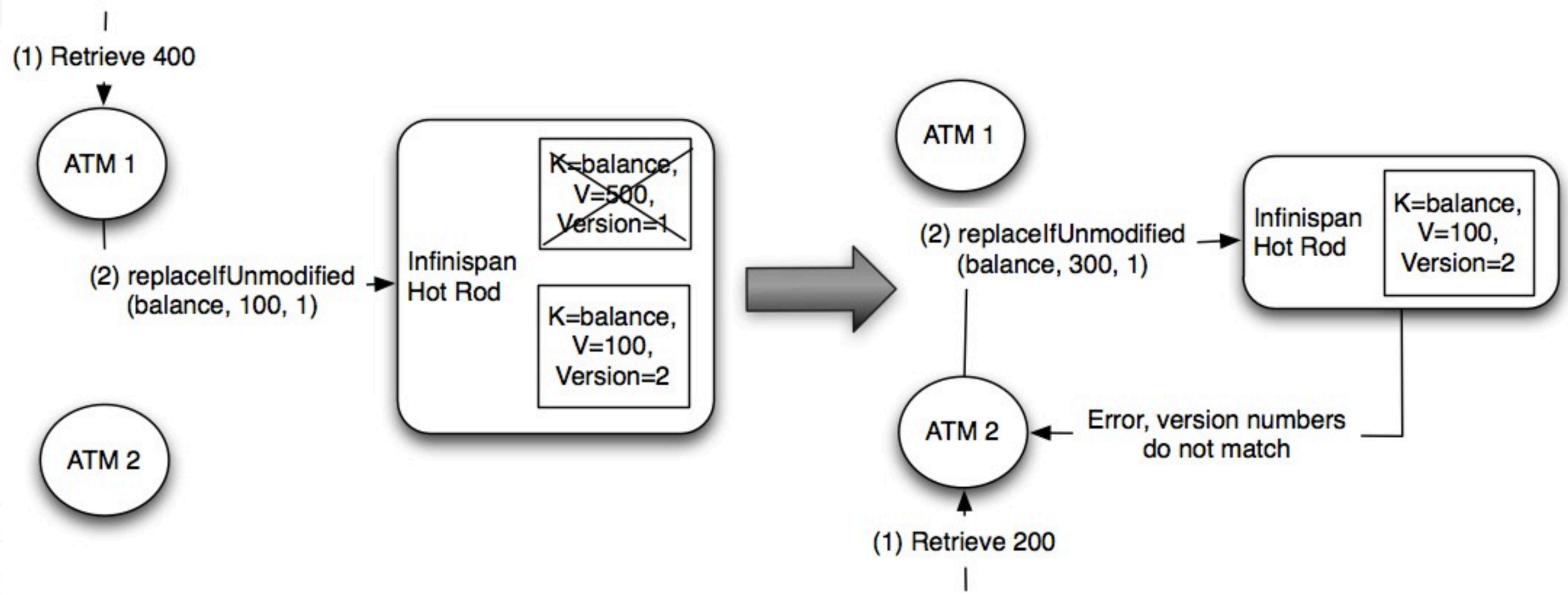
Data Consistency in P2P



Hot Rod Versioned API



Hot Rod Versioned API



Hot Rod Client Intelligence

- Different client intelligence levels supported:
  - Basic clients
  - Topology-aware clients
  - Hash-distribution-aware clients

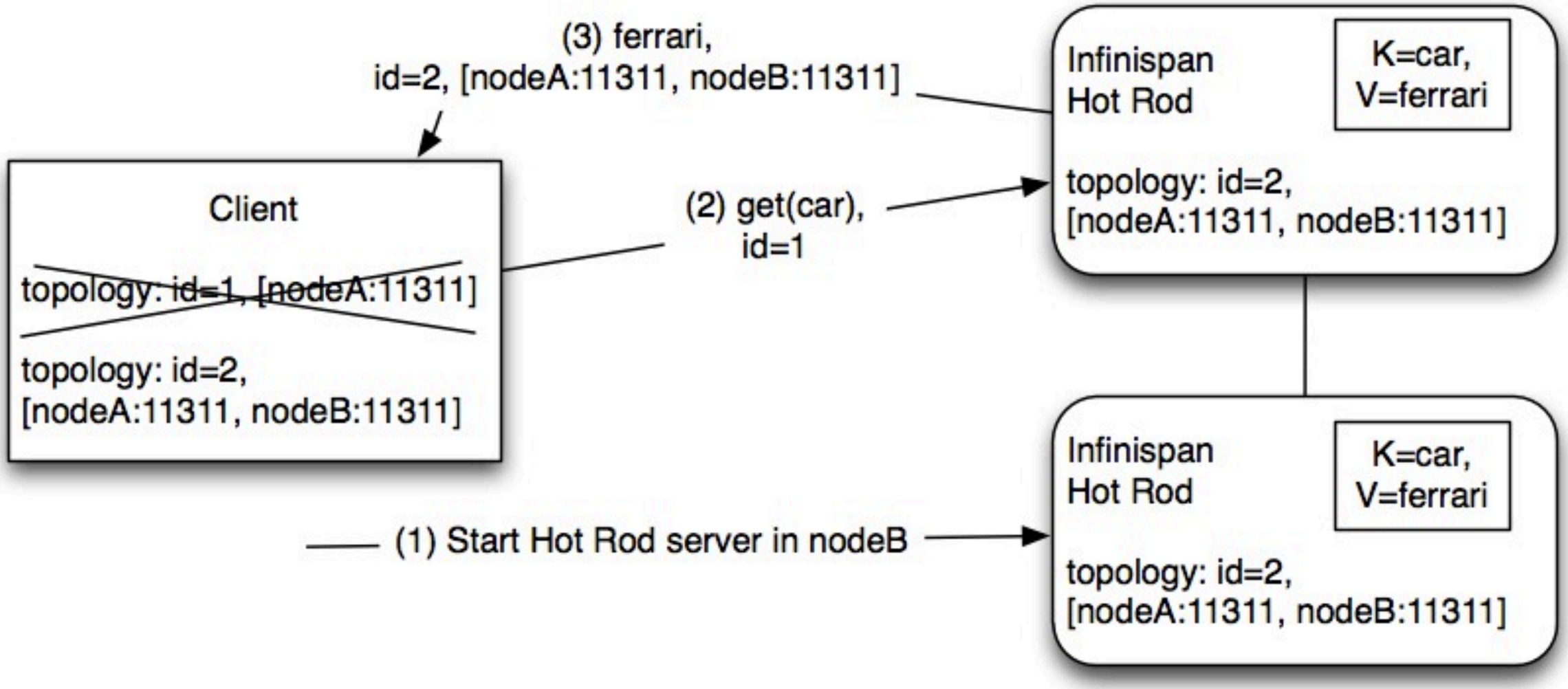
Infinispan Hash Functions

- Infinispan uses language independent hash functions
  - Used for smart routing
- Enables smart client implementations in any language
- So far, MurmurHash 2.0 implemented

Galder Zamarreño | galder@jboss.org

Topology Information Delivery

Topology Information Delivery



Hot Rod Implementations

- Server implementation included in 4.1.0.Beta2
  - Uses high performance Netty socket framework
  - Start via script: `startServer.sh -r hotrod`
- Java client reference implementation also available
  - Supports all client intelligence levels
- Volunteers for writing clients in other languages welcomed :)
  - If interested, join us at the Cloud Hackfest!

Hot Rod Client Basic API

```
//API entry point, by default it connects to localhost:11311
CacheContainer cacheContainer = new RemoteCacheManager();

//obtain a handle to the remote default cache
Cache<String, String> cache = cacheContainer.getCache();

//now add something to the cache and make sure it is there
cache.put("car", "ferrari");
assert cache.get("car").equals("ferrari");

//remove the data
cache.remove("car");
assert !cache.containsKey("car") : "Value must have been removed!";
```

Hot Rod Client Versioned API

```
//API entry point, by default it connects to localhost:11311
CacheContainer cacheContainer = new RemoteCacheManager();

//obtain a handle to the remote default cache
RemoteCache<String, String> remoteCache = cacheContainer.getCache();

//put something in the cache
remoteCache.put("car", "ferrari");

//retrieve the value and the version
RemoteCache.VersionedValue value = remoteCache.getVersioned("car");

//replace it with a new value passing the version read
assert remoteCache.replace("car", "mclaren", value.getVersion());
```

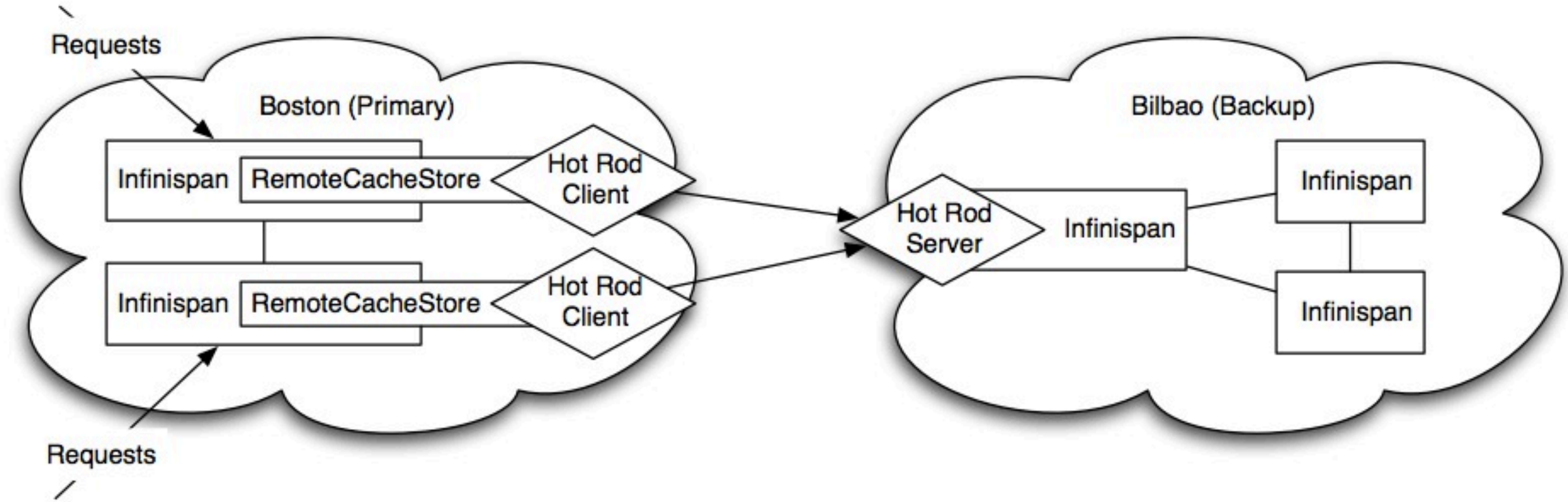
Infinispan Servers Comparison

	Protocol	Client Availability	Clustered	Smart Routing	Load Balancing / Failover
Hot Rod	Binary	Right now, only Java	Yes	Yes	Yes, dynamic via Hot Rod client
Infinispan Memcached	Text	Tons	Yes	No	Only with predefined list of servers
Infinispan REST	Text	Tons	Yes	No	Any Http Load Balancer

The path ahead for Hot Rod

- Within Hot Rod:
  - Clients in other languages
  - Querying
  - Event handling...
- Submit protocol to a standards body (maybe)

Hot Rod as base for new functionality



Demo

Summary

- Infinispan client-server architectures are needed
- Hot Rod is Infinispan's binary client-server protocol
- Designed for load balancing, failover and smart routing
- Server and java client available now
- We need your help to build more clients!
- Hot Rod as foundation for interesting new functionality

Questions?

- Project: www.infinispan.org
- Blog: blog.infinispan.org
- Twitter:
  - @infinispan, @galderz
  - #infinispan #judcon
- Join us at the Cloud Hackfest!!!
- JBoss Asylum Podcast recording - panel discussion
  - Tonight, 8.30pm community room

Learn more about Infinispan!

- *Storing Data on Cloud Infrastructure in a Scalable, Durable Manner* - Wed 23rd
- *Using Infinispan for High Availability, Load Balancing, & Extreme Performance* - Thu, 24th
- *How to Stop Worrying & Start Caching in Java* - Thu 24th
- *Why RESTful Design for Cloud is Best* - Fri 25th