JBoss Community

galder@jboss.org | twitter.com/galderz | zamarreno.com

Beyond Peer-to-Peer Data Grids with Infinispan Servers

Galder Zamarreño
Senior Engineer, Red Hat
4th October 2010

JBoss Community

galder@jboss.org | twitter.com/galderz | zamarreno.com

Who is Galder?

- R&D engineer (Red Hat Inc):
  - Infinispan developer
  - JBoss Cache developer
- Contributor and committer:
  - JBoss AS, Hibernate, JGroups, JBoss Portal,...etc
- Blog: zamarreno.com
- Twitter: @galderz

Agenda

- Introduction to Infinispan
- Peer-to-peer (P2P) data grids vs client-server
- Infinispan server comparison
- The path ahead for Infinispan servers
- Demo

galder@jboss.org | twitter.com/galderz | zamarreno.com

What is Infinispan?

- A data grid platform which is:
  - Open Source (LGPL)
  - In-memory
  - Distributed
  - Elastic
  - Highly Available

Infinispan

JBoss Community

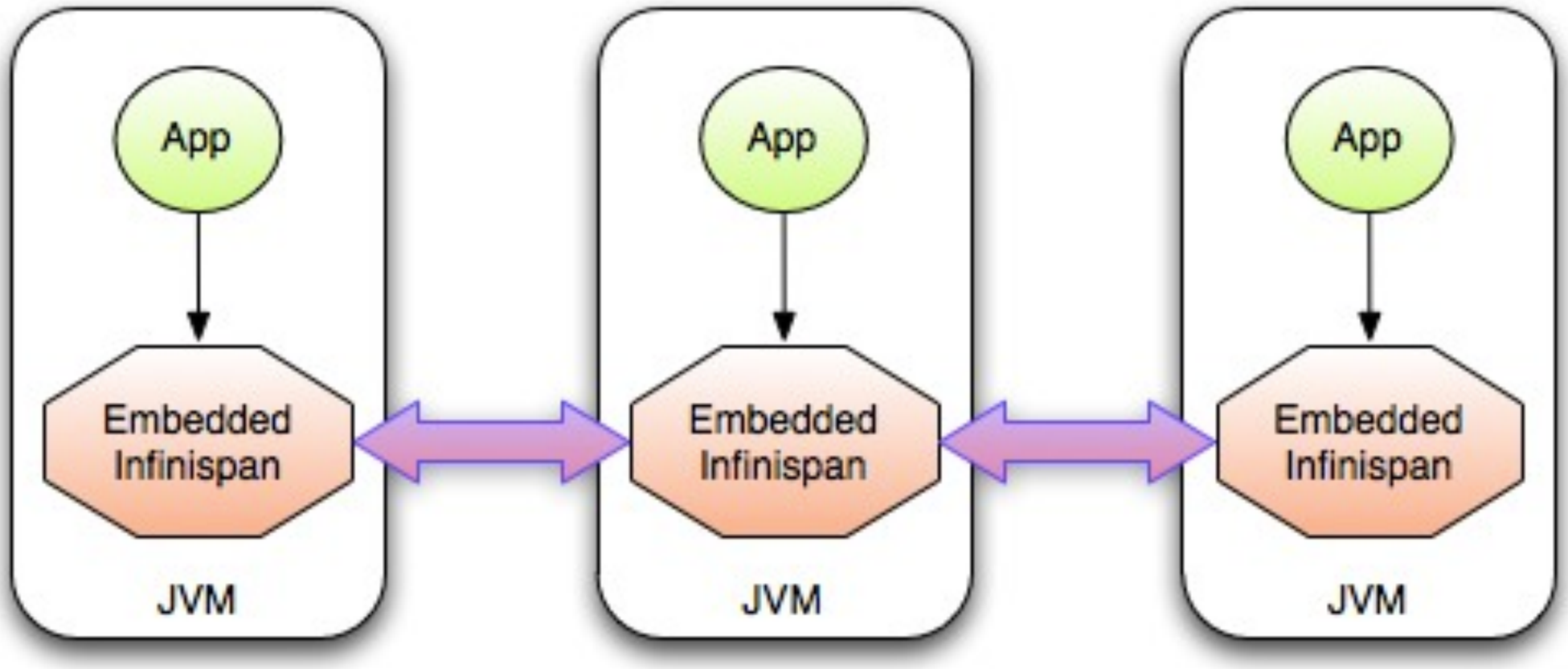
galder@jboss.org | twitter.com/galderz | zamarreno.com

Brief feature overview

- Seamless data distribution, linearly scalable data structures
- Implicit eviction
- Write through and write behind to persistent storage
- JTA and XA transactions
- Listeners and notifications
- Querying and indexing
- Alternative JPA-like API
- JMX management and GUI console
- Multiple endpoints for remote invocations...

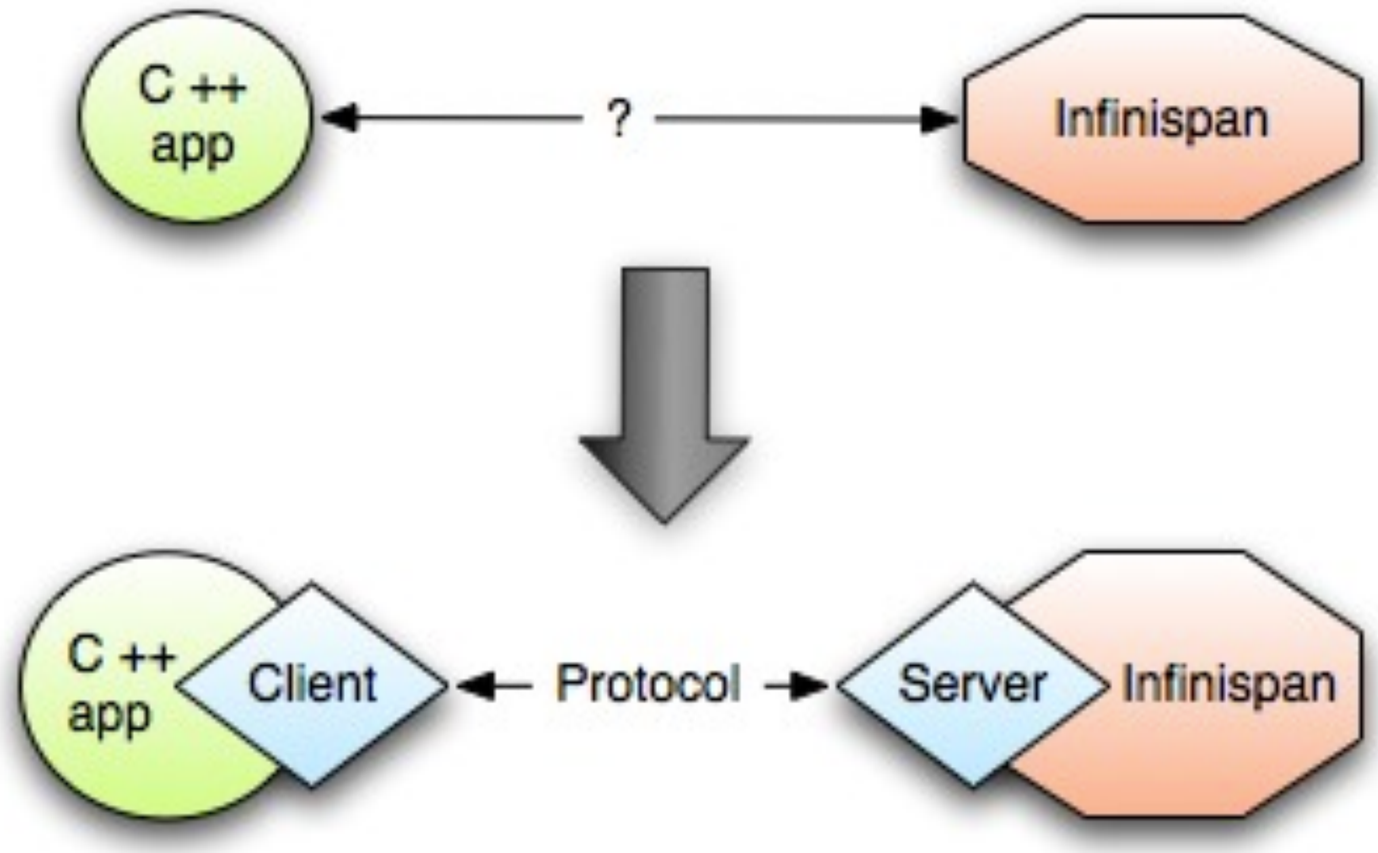
galder@jboss.org | twitter.com/galderz | zamarreno.com

Peer-to-Peer Setup



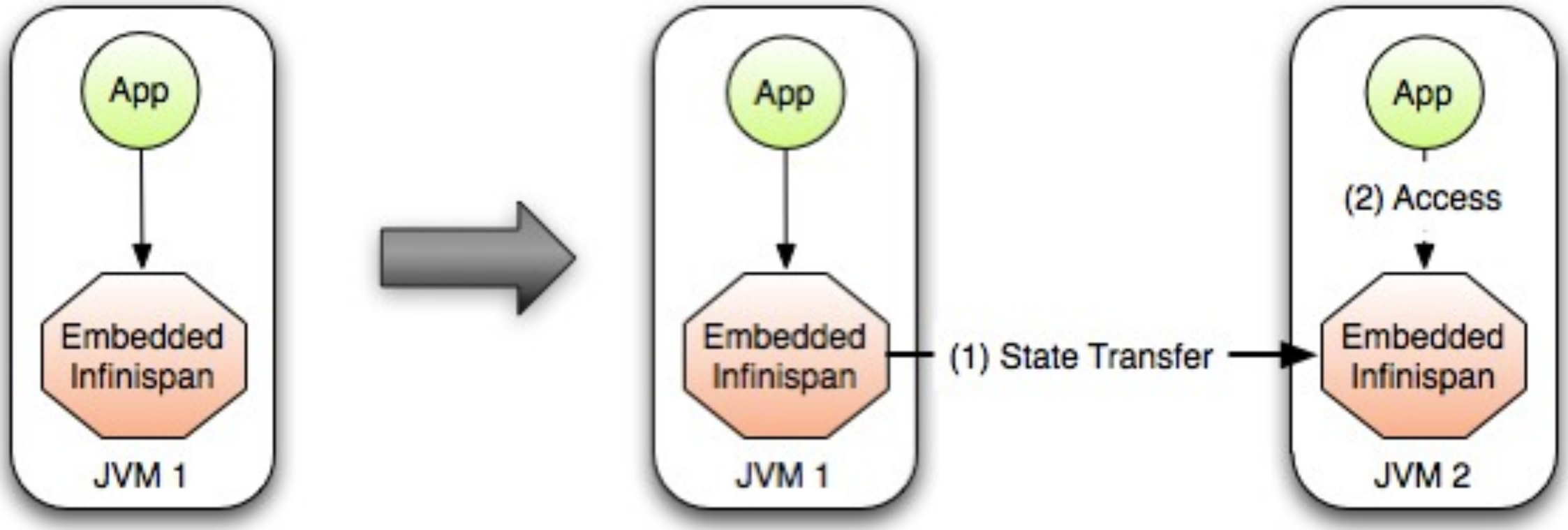
galder@jboss.org | twitter.com/galderz | zamarreno.com

Non-JVM access



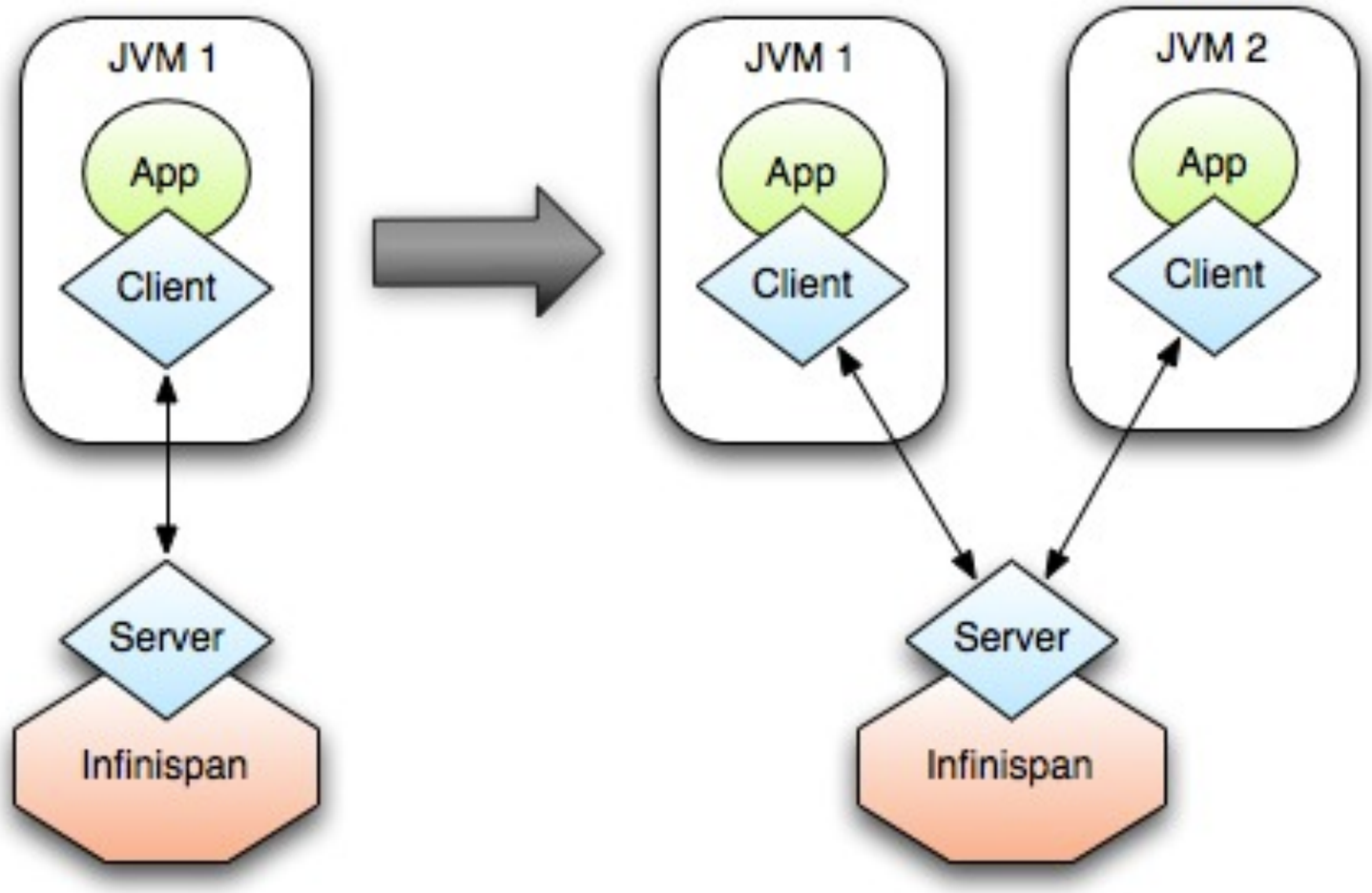
galder@jboss.org | twitter.com/galderz | zamarreno.com

Elasticity problems with P2P



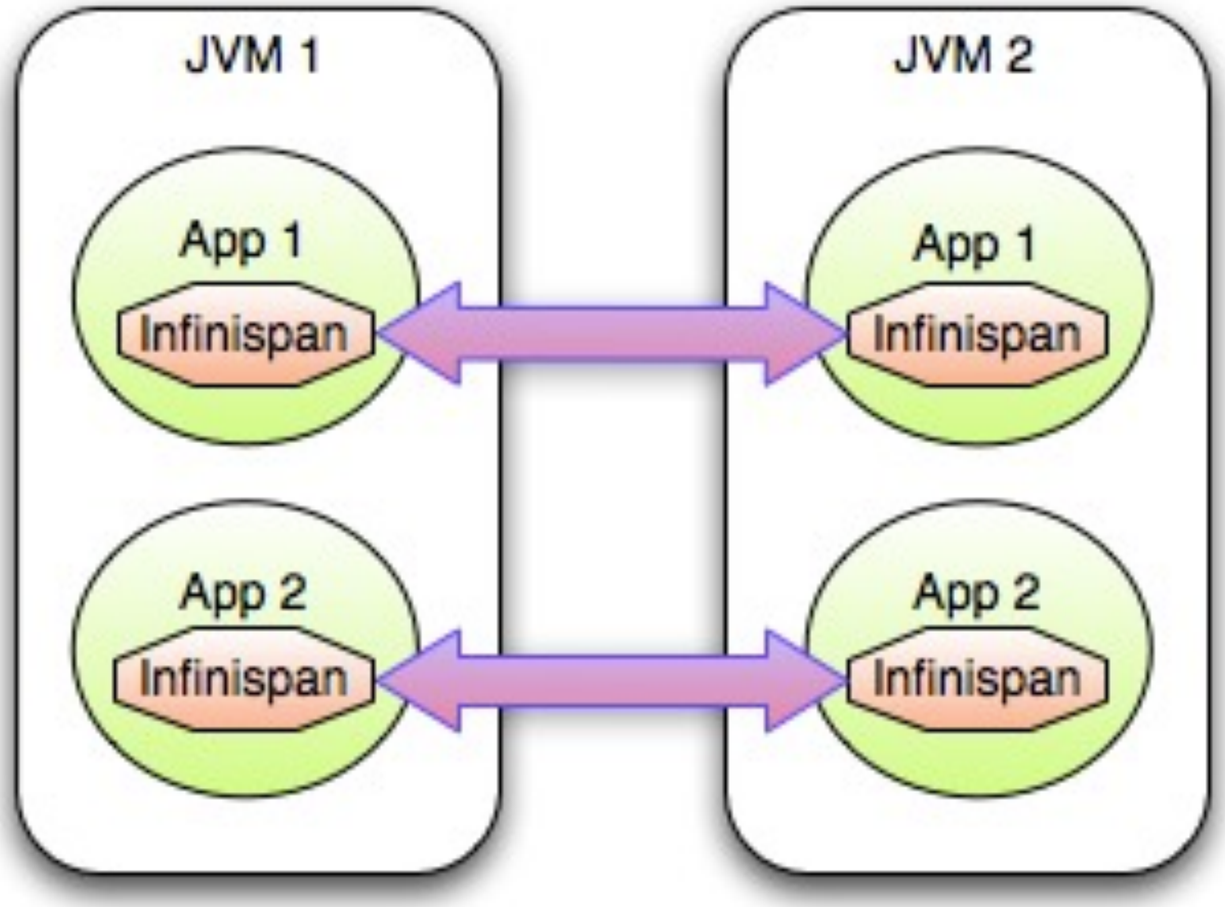
galder@jboss.org | twitter.com/galderz | zamarreno.com

Client-Server brings Elasticity



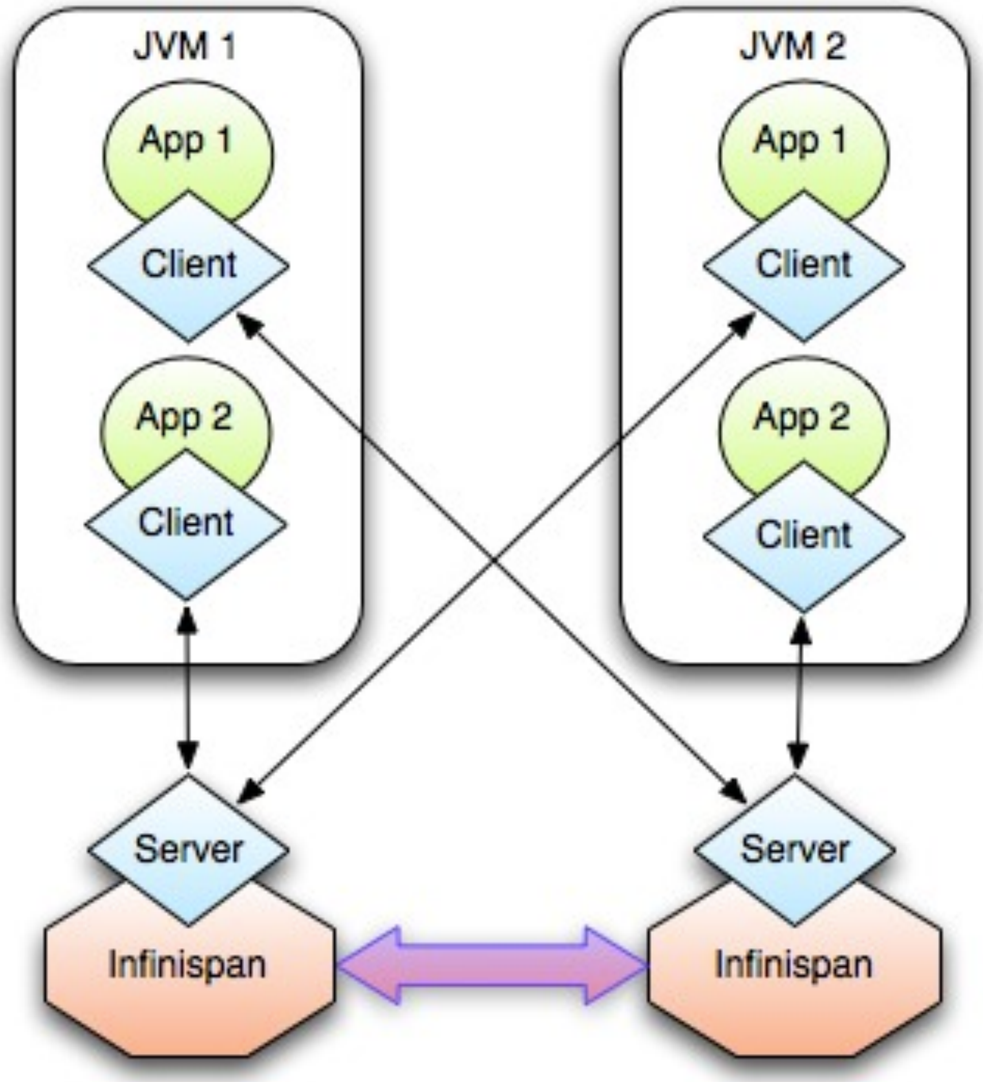
galder@jboss.org | twitter.com/galderz | zamarreno.com

Data Grid per Application?



galder@jboss.org | twitter.com/galderz | zamarreno.com

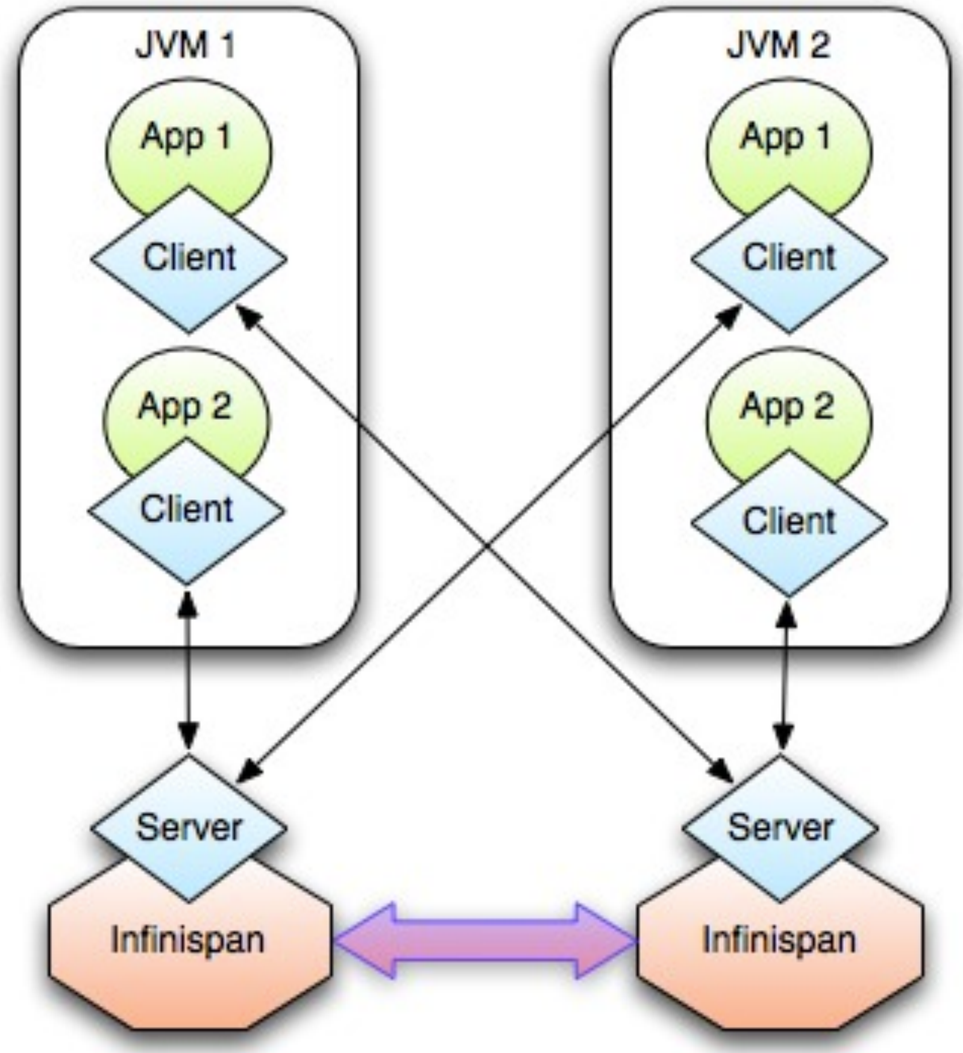
Shared Data Grid



galder@jboss.org | twitter.com/galderz | zamarreno.com

Independent Tier Management

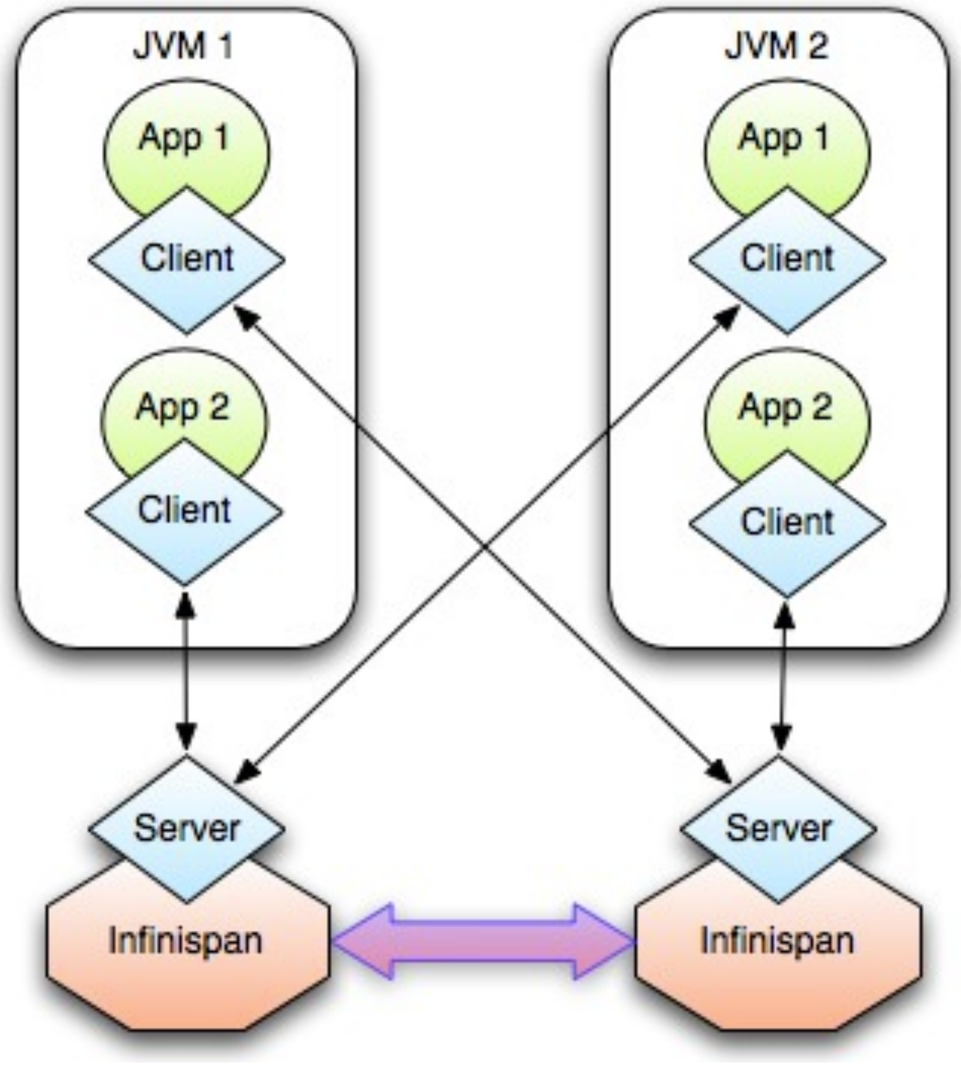
- Independently deploy new app version
- Security
- Incompatible JVM tuning requirements



galder@jboss.org | twitter.com/galderz | zamarreno.com

Infinispan Server Modules

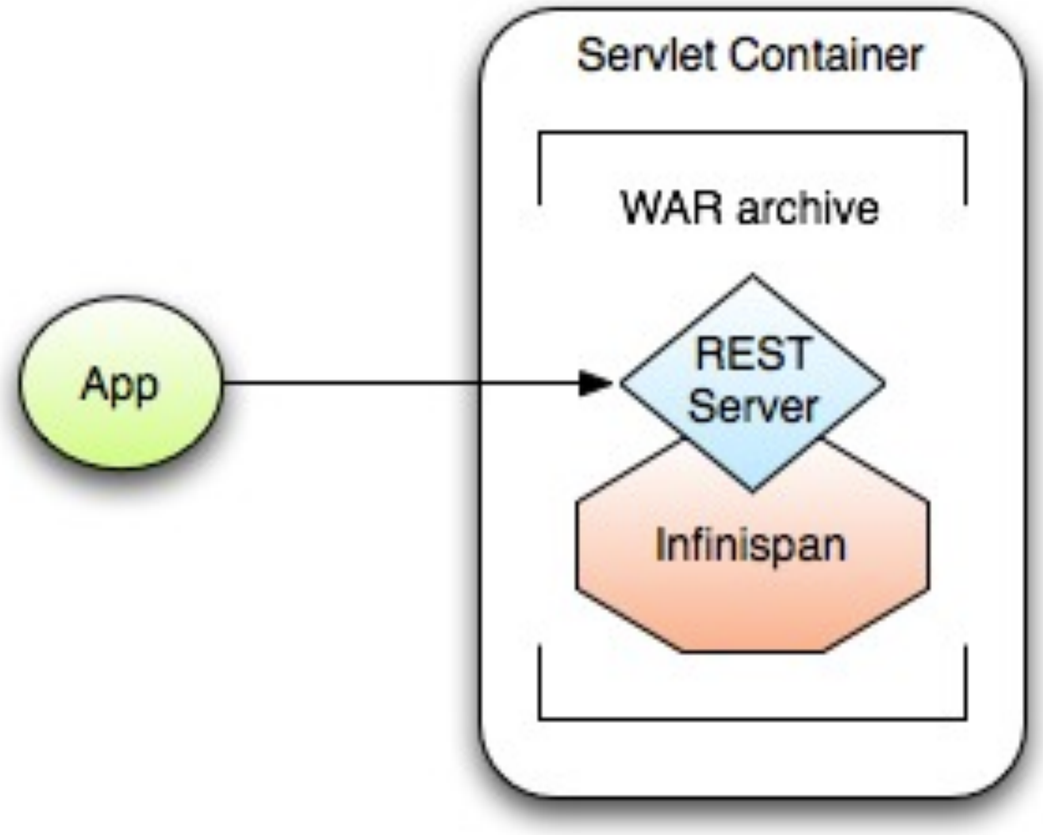
- Protocols supported in 4.1 :
  - REST
  - Memcached
  - Hot Rod
  - Websocket



galder@jboss.org | twitter.com/galderz | zamarreno.com

REST Server

- RESTful HTTP access
  - HTTP PUT/POST to store
  - HTTP GET to retrieve
  - Available since 4.0
    - In ‘all’ distribution

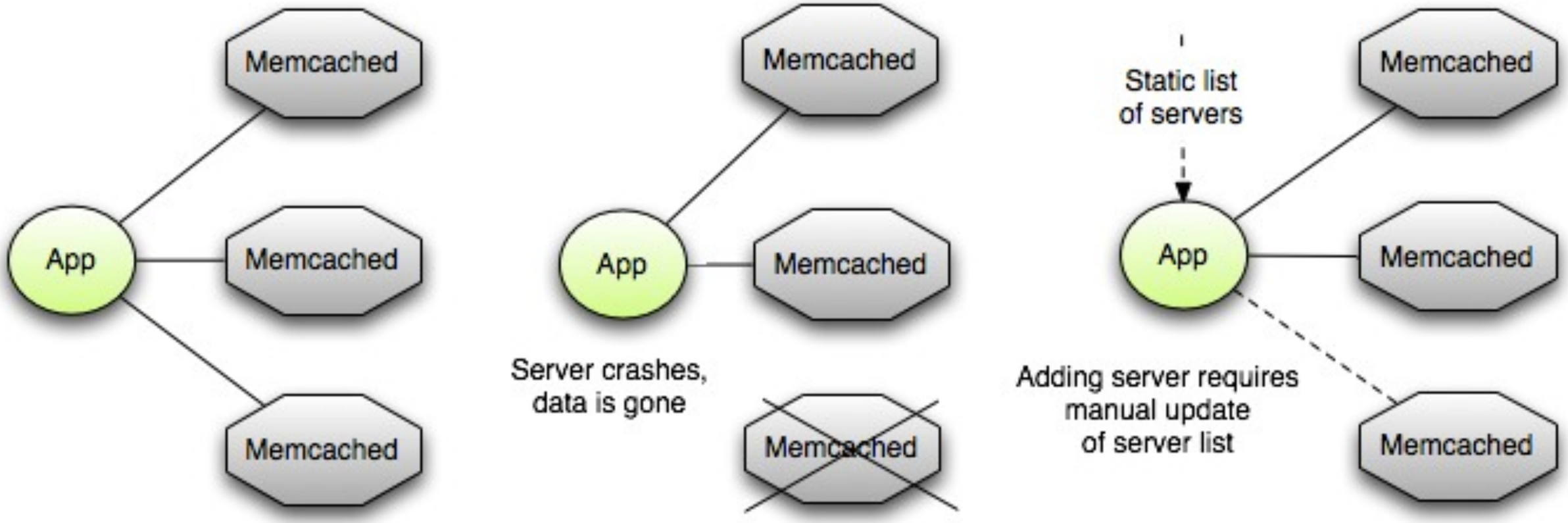


galder@jboss.org | twitter.com/galderz | zamarreno.com

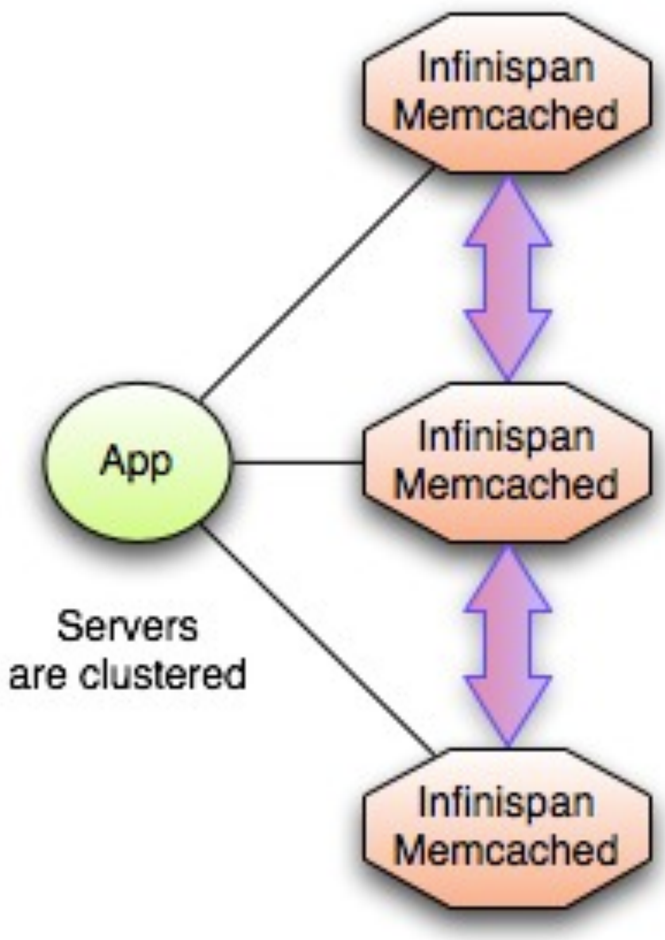
REST Server via Python

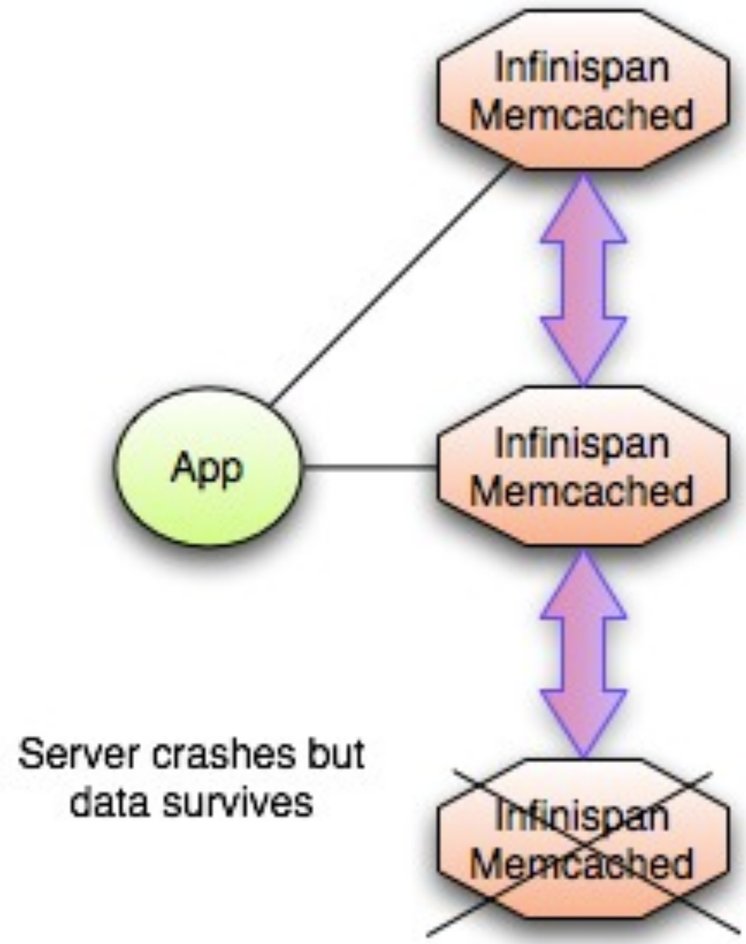
```python
#
# Sample python code using the standard http lib only
#

import httplib


#putting data in
conn = httplib.HTTPConnection("localhost:8080")
data = "SOME DATA HERE !" #could be string, or a file...
conn.request("POST", "/infinispan/rest/Bucket/0", data,
   {"Content-Type": "text/plain"})
response = conn.getresponse()
print response.status

#getting data out
import httplib
conn = httplib.HTTPConnection("localhost:8080")
conn.request("GET", "/infinispan/rest/Bucket/0")
response = conn.getresponse()
print response.status
print response.read()
```

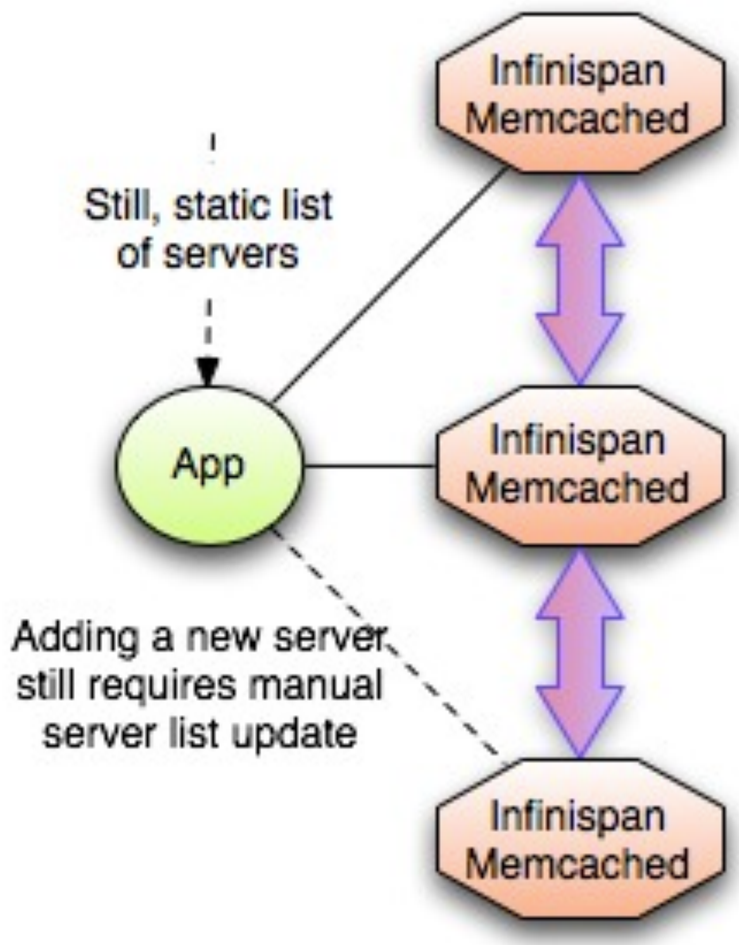
Memcached not good enough



Infinispan Memcached

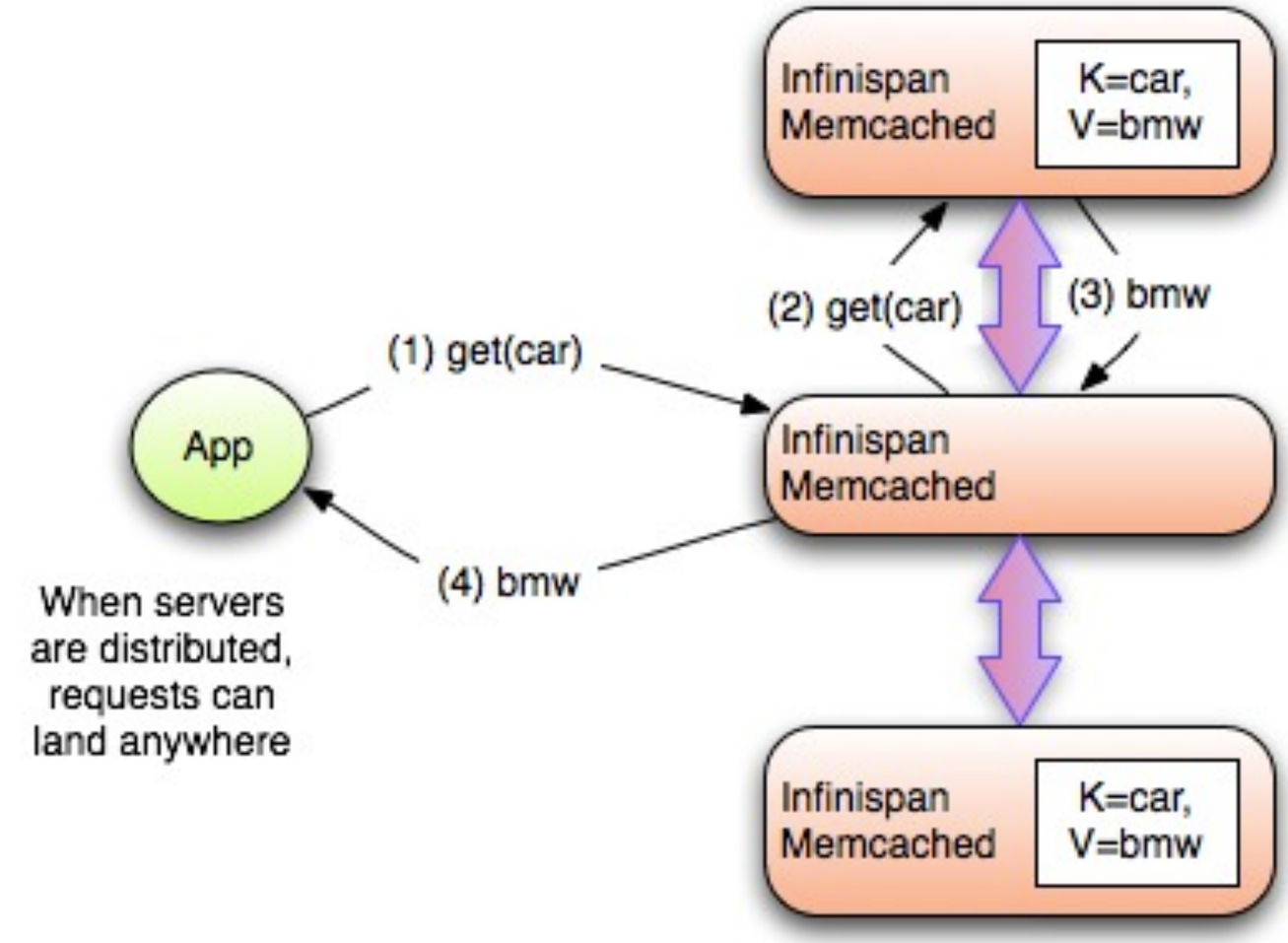






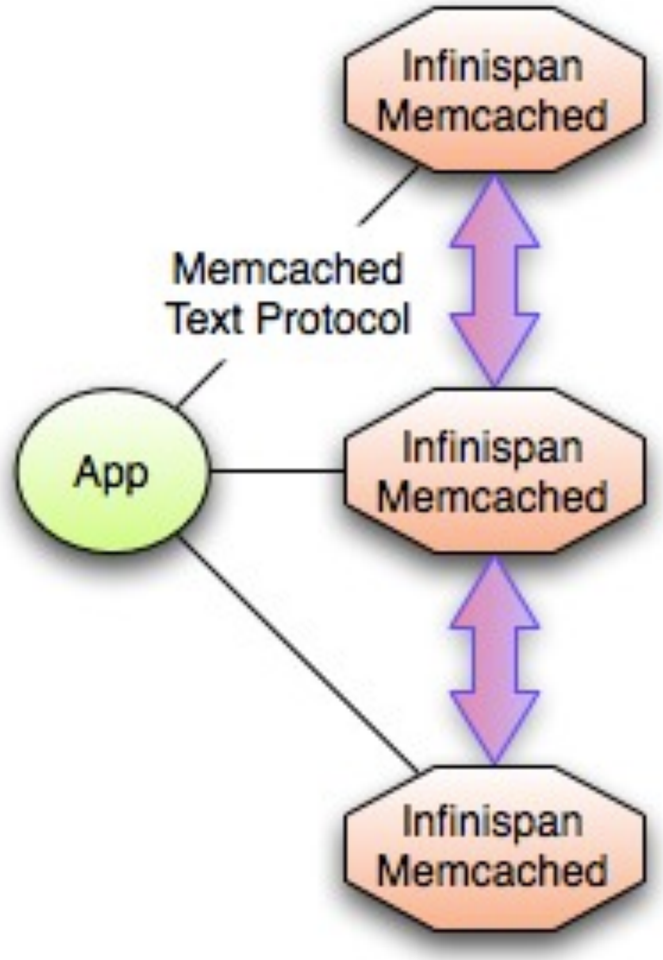
galder@jboss.org | twitter.com/galderz | zamarreno.com

Routing not so smart



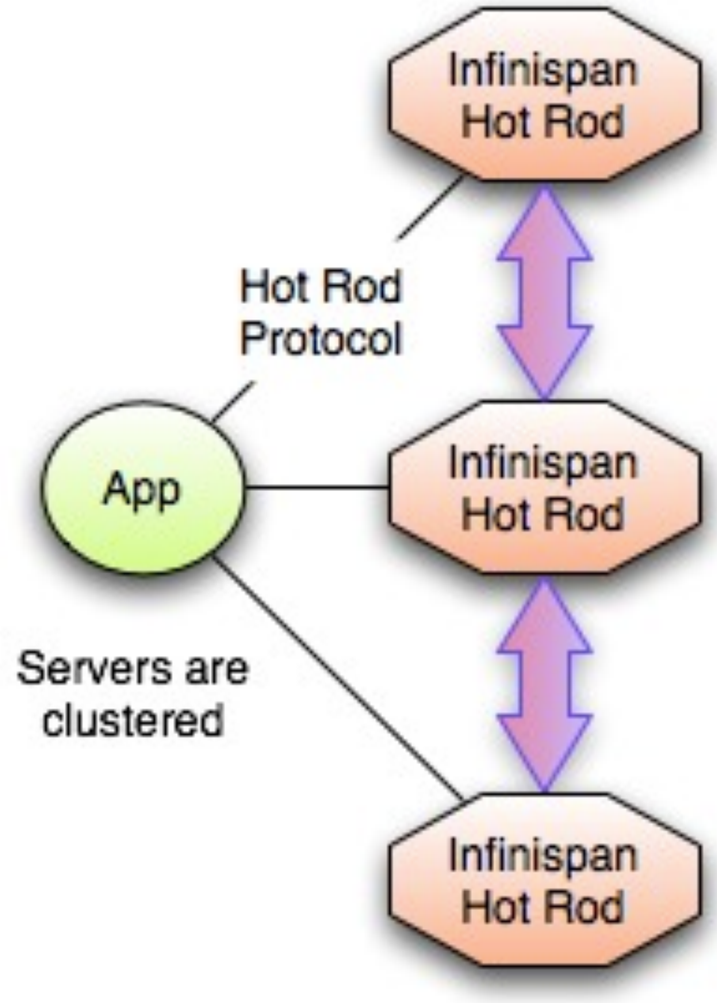
Essential Information

- To run, execute:
  - `startServer.sh -r memcached`
- New in 4.1
- Only text protocol supported
- Works with any Memcached client



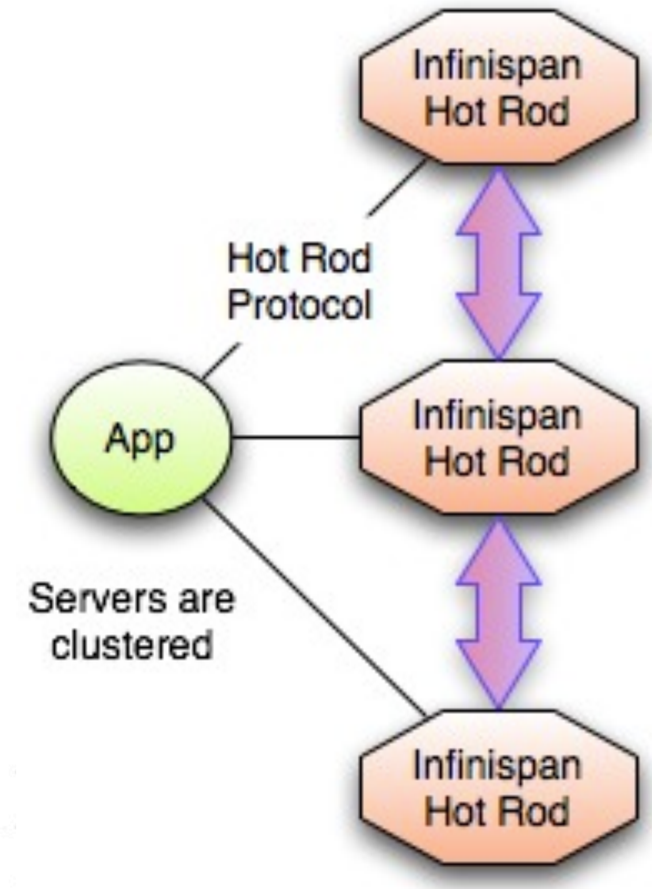
Hot Rod to the rescue!

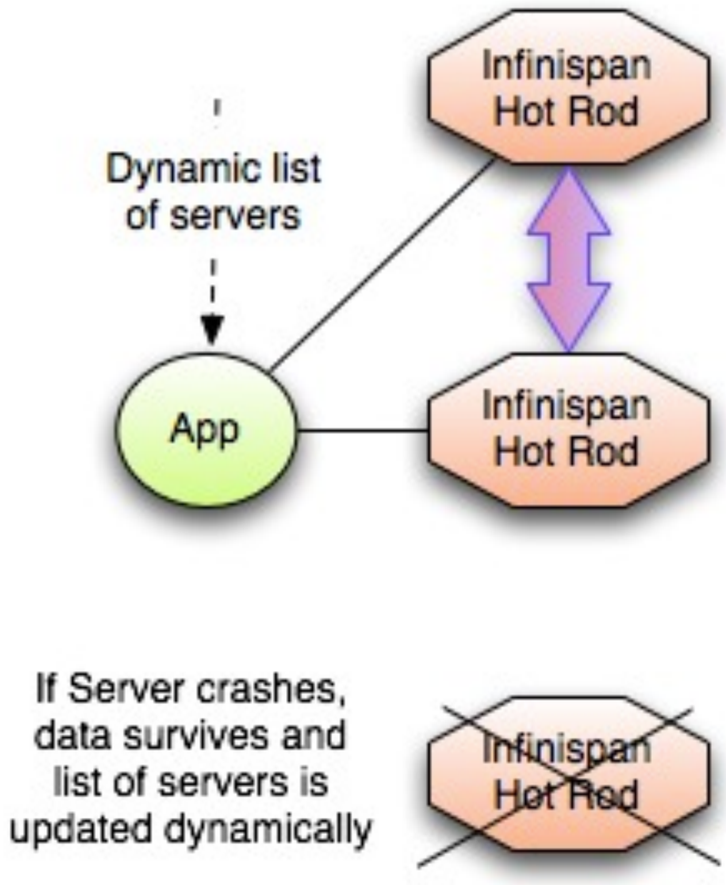
- What is Hot Rod?
  - Wire protocol for client-server communications
  - Open and language independent
  - Built-in dynamic failover and load balancing
  - Smart routing

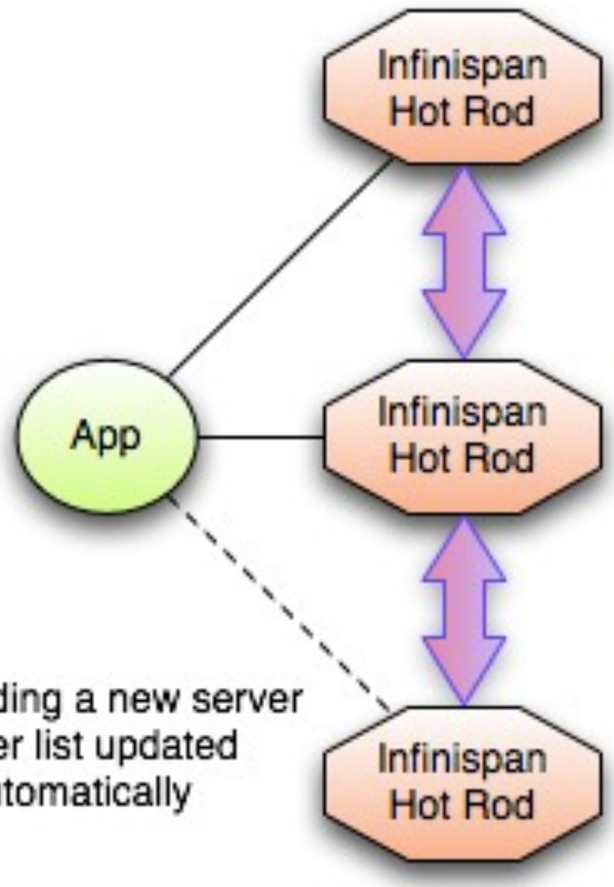


galder@jboss.org | twitter.com/galderz | zamarreno.com

Dynamic routing with Hot Rod

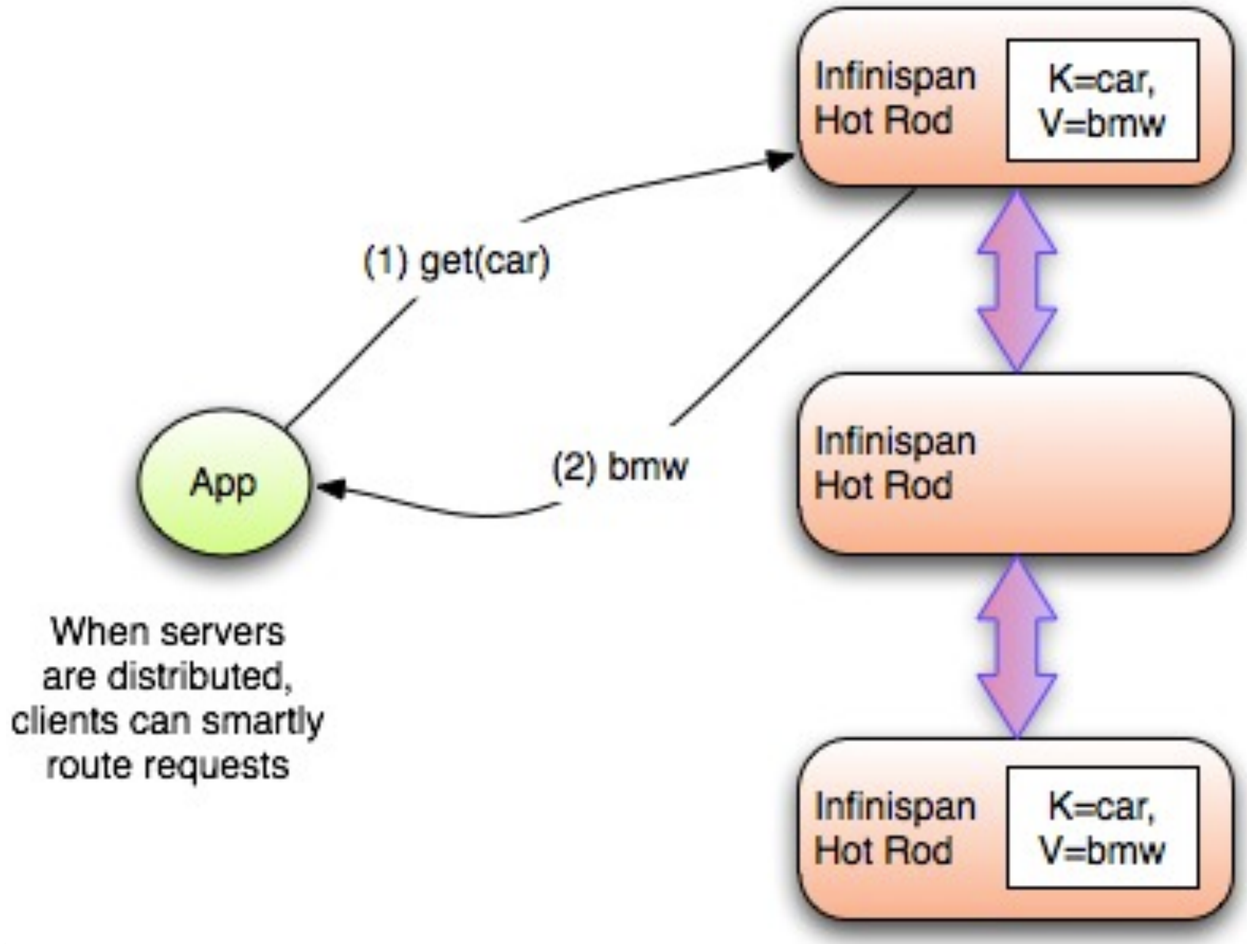






galder@jboss.org | twitter.com/galderz | zamarreno.com

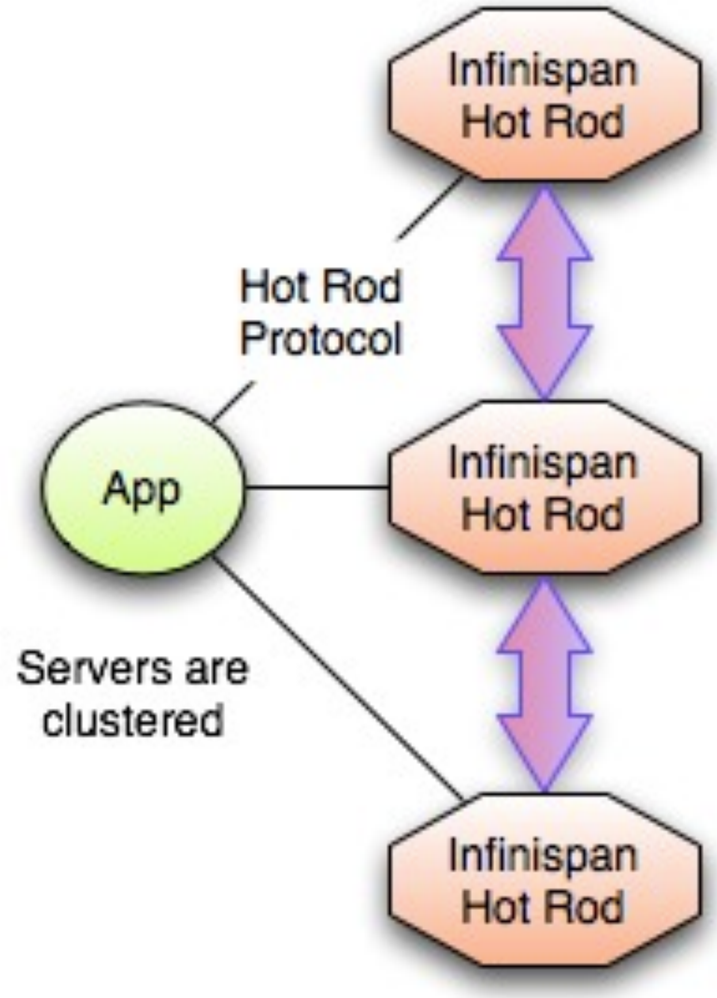
Smart routing with Hot Rod



galder@jboss.org | twitter.com/galderz | zamarreno.com

Essential Information

- To run:
  - `startServer.sh -r hotrod`
- New in 4.1
- Only a Java client available:
  - Supports smart routing and dynamic load balancing



galder@jboss.org | twitter.com/galderz | zamarreno.com

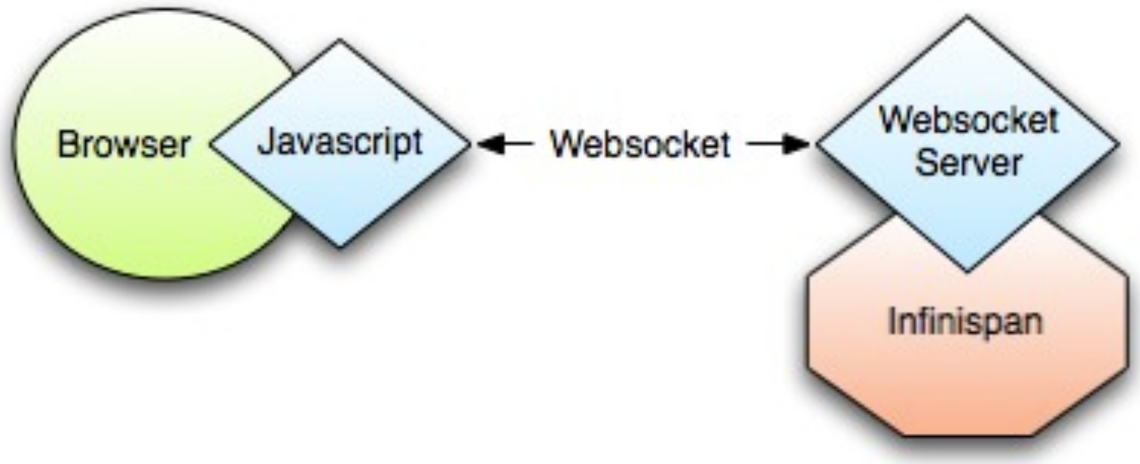
Java Hot Rod Client

```
//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", "bmw");
assert cache.get("car").equals("bmw");

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


galder@jboss.org | twitter.com/galderz | zamarreno.com

Websocket Server

- Exposes Infinispan Cache instance over Websocket
- To run it:
  - `startServer.sh -r websocket`
- Accessible via Javascript API



galder@jboss.org | twitter.com/galderz | zamarreno.com

Server Comparison

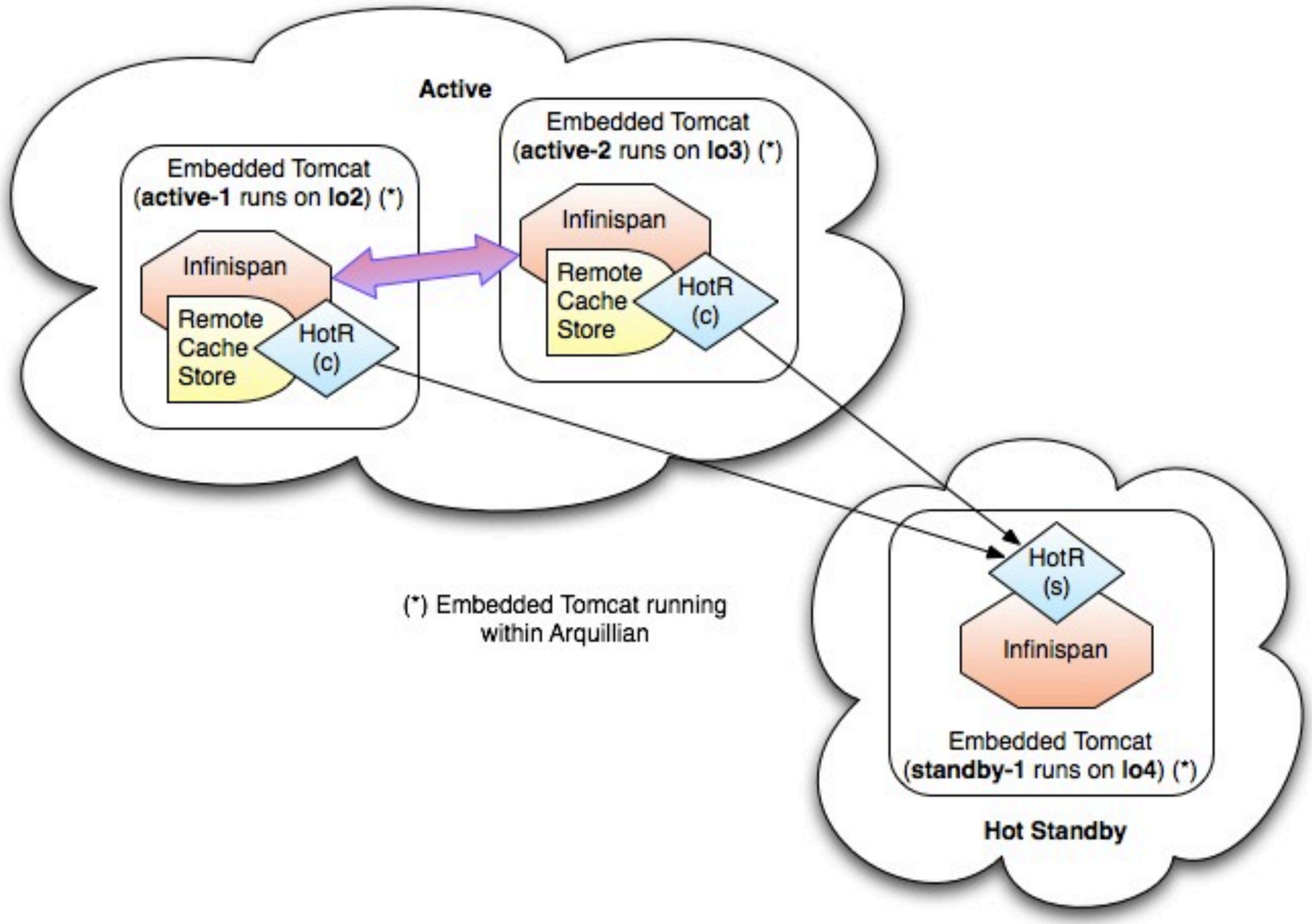
	Protocol	Client Libraries	Clustered ?	Smart Routing	Load balancing / Failover
REST	Text	N/A	Yes	No	Any HTTP load balancer
Memcached	Text	Plenty	Yes	No	Only with predefined server list
Hot Rod	Binary	Currently only Java	Yes	Yes	Dynamic
Websocket	Text	Javascript	Yes	No	Any HTTP load balancer

The path ahead

- Hot Rod improvements:
  - Remote querying
  - Event handling
- Submit Hot Rod protocol to standards body (maybe)
- Others:
  - Memcached binary protocol won’t be implemented

galder@jboss.org | twitter.com/galderz | zamarreno.com

Prototype Hot Standby Demo



galder@jboss.org | twitter.com/galderz | zamarreno.com

Summary

- Accessing data grids in client-server mode makes sense
- Infinispan 4.1 comes with a range of server modules
- Each server fits one type of use case
- We need your help to build more Hot Rod clients!

galder@jboss.org | twitter.com/galderz | zamarreno.com

Questions?

infinispan.org
blog.infinispan.org
twitter.com/infinispan
#infinispan

JBoss Community

galder@jboss.org | twitter.com/galderz | zamarreno.com