JBoss Community

manik@jboss.org | http://twitter.com/maniksurtani | http://blog.infinispan.org
Data-as-a-Service with Infinispan

Manik Surtani
Founder and project lead, Infinispan
Principal Engineer, Red Hat, Inc.
“Could data storage be the one thing that hampers true cloud scalability and elasticity?”
Who is Manik?

- R&D Engineer, Red Hat Inc.
- Founder and project lead, Infinispan
- Project lead, JBoss Cache
- Frequent speaker on cloud computing and cloud data storage

http://twitter.com/maniksurtani
http://blog.infinispan.org
Agenda

• What is Data-as-a-Service?
• Introducing Infinispan
• Implementing Data-as-a-Service with Infinispan
Traditional 3-tier App
Typical IaaS App
Typical PaaS App
Where’s your data stored??
Clouds are ephemeral!
Virtualizing Data

• Some public services do exist
  • Amazon RDS and SimpleDB
  • FathomDB
  • Cloundant
  • MongoHQ
  • etc.
What about private clouds?

• Not all cloud deployments are public!
• Private cloud is very important
• How can you build DaaS yourself?
Characteristics of DaaS

- Elastic data
- Needs to scale with other tiers
- Response times should be linear
- Needs to be highly available
- Nodes will die! The service shouldn’t.
SQL and NoSQL DB

• Lack of distribution hampers elasticity and HA
• These limitations can be worked around
  • ... but this isn’t trivial
  • ... or cheap
SQL and NoSQL DB

• Lack of distribution hampers elasticity and HA
• These limitations can be worked around
• ... but this isn’t foolproof
• ... or cheap
Distributed Data Grid

• Far better suited to elastic data
• Distributed by nature
• Highly available by nature
• A good building block for your data service
API is king

• Apps should use their native data storage APIs
  • E.g., JPA (Java EE), ActiveRecord (Ruby), etc.
• Key/value stores too low level
• Akin to direct JDBC calls!
Introducing Infinispan
What is Infinispan?

- Open source (LGPL) in-memory Data Grid
- Some concepts from Amazon Dynamo

2 usage modes

- Embedded
- Client-server
  - memcached
  - Hot Rod
  - REST
Consistent hash based distribution
• Self healing
• No single point of failure

Highly concurrent
• MVCC locking
XA Transactions

• 2-phase commit based
• Deadlock detection algorithms
• Coming soon: Atomic Broadcast
Map/Reduce
• In a pre-release state right now

Querying
• Using Lucene and Hibernate Search to index
Persistence
• Not just in memory!
• Write through and write behind
• Pluggable “drivers”

Eviction and expiry
• Efficient, adaptive algorithms
• Addresses shortcomings of LRU & FIFO
API

- Map-like key/value store
- Upcoming JPA-like layer
- Other high-level APIs being discussed in the community e.g., ActiveRecord
So is Infinispan a data grid?

- In-memory
- P2P, distributed
- Low-latency, fast key/value store
So is Infinispan a data grid?

- In-memory
- P2P, distributed
- Low-latency, fast key/value store

... or is it a NoSQL database?

- Persistence
- Map/Reduce
So is Infinispan a data grid?

- In-memory
- P2P, distributed
- Low-latency, fast key/value store

... or is it a NoSQL database?

- Persistence
- Map/Reduce

... or something else?

- Querying support
- Transactional
So why is Infinispan sexy?
Why is Infinispan sexy?

- Transparent horizontal scalability
  - Elastic in both directions
- Fast, low latency data access
- Ability to address a very large heap
- Cloud-ready datastore
- Not just for Java
- Free and doesn't suck!
Peer To Peer Architecture
Client/Server Architecture

Supported Protocols
- REST
- Memcached
- Hot Rod
WTF is Hot Rod?

- Wire protocol for client server communications
- Open
- Language independent
- Built-in failover and load balancing
- Smart routing
DaaS with Infinispan
## Server Endpoint Comparison

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Client Libraries</th>
<th>Clustered</th>
<th>Smart Routing</th>
<th>Load Balancing/Failover</th>
</tr>
</thead>
<tbody>
<tr>
<td>REST</td>
<td>Text</td>
<td>N/A</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Memcached</td>
<td>Text</td>
<td>Plenty</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Hot Rod</td>
<td>Binary</td>
<td>Currently only Java</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>
APIs

- REST
  - REST-* Caching API, OData/GData
- Memcached pseudo-standard (key/value)
  - Use any existing memcached client library
- JSR-107 (JCACHE)
  - Use the Hot Rod Java client
- JPA-like API
  - being developed
- More platform-specific API bindings
  - We welcome community participation!
Starting an Infinispan Server

• Hot Rod or memcached server endpoint

$ bin/startServer.sh -r hotrod \ 
   -c infinispan.xml
$ bin/startServer.sh -r memcached \ 
   -c infinispan.xml

• REST endpoint
  • Deploy `infinispan-server-rest.war` in your favorite servlet container.
So what have we learned?

• Elastic data is hard
• Public data services not always suitable
• Data grids make elastic storage easy
• Infinispan server endpoints help build elastic data tiers
Roadmap

4.0.0 Starobrno

- Map-like API
- Async API
- Consistent Hash based distribution
- Write-through, write-behind
- Eviction, expiration
- Management tooling
- REST API
- Hibernate 2nd Level Cache
- Released Feb 2010
Roadmap

4.1.0 Radegast

- Deadlock detection
- Client/Server protocols
  - Memcached
  - Hot Rod
  - Smart clients using Hot Rod
- Lucene Directory implementation
- LIRS: adaptive, recency-based eviction policies
- Released August 2010
Roadmap

5.0.0 Pagoa

- JPA-like API
- Fine-grained replication
- Distributed code execution
  - Map/reduce
- In active development

5.1.0 and beyond

- Dynamic provisioning based on SLAs
- Complex event processing features
Questions?

http://www.infinispan.org
http://blog.infinispan.org
http://twitter.com/infinispan
  #infinispan
IRC: #infinispan on FreeNode
http://github.com/infinispan
Embedded API
Standalone (LOCAL) mode

```java
Configuration cfg = new Configuration();
Cache<String, String> c = new DefaultCacheManager(cfg).getCache();

// Basic Map API
    c.put("hello", "Devoxx!");
    assert 1 == c.size();
    assert "Devoxx!".equals(c.get("hello"));

// ConcurrentHashMap API
    c.putIfAbsent("hello", "again, Devoxx!");
    c.replace("hello", "again, Devoxx!");

// Expiration
    c.put("expirable", "value", 10, TimeUnit.MINUTES);

// Async API
    Future<String> f = c.putAsync("key", "value");
```
```java
// On server 1
Configuration cfg = new Configuration();
cfg.setCacheMode(Configuration.CacheMode.DIST_SYNC);

Cache<String, String> c1 = new DefaultCacheManager(
    GlobalConfiguration.getClusteredDefault(),
    cfg).getCache();

c1.put("hello", "world");

// On server 2
Configuration cfg = new Configuration();
cfg.setCacheMode(Configuration.CacheMode.DIST_SYNC);

Cache<String, String> c2 = new DefaultCacheManager(
    GlobalConfiguration.getClusteredDefault(),
    cfg).getCache();

assert "world".equals(c2.get("hello"));
```
Embedded API
Config via XML

Stand-alone config:
```xml
<infinispan />
```

Clustered config:
```xml
<infinispan>
  <global>
    <transport />
  </global>

  <default>
    <clustering mode="distributed" />
  </default>

</infinispan>
```
Questions?

http://www.infinispan.org
http://blog.infinispan.org
http://twitter.com/infinispan

#infinispan

IRC: #infinispan on FreeNode
http://github.com/infinispan
http://bit.ly/rate_me