JBoss Cache & A Use Case:
Implementing JBoss/Tomcat Fine-Grained Http Session Replication

Ben Wang & Bela Ban
JBoss Inc.

March 8, 2005

Topic

- JBoss Cache
  - ✓ Feature
  - ✓ Architecture
  - ✓ Configuration
- Http session clustering with Tomcat
  - ✓ Goal
  - ✓ Implementation detail
  - ✓ Configuration

What is JBoss Cache?

- Two modules
  - ✓ TreeCache
  - ✓ TreeCacheAOP
- TreeCache
  - ✓ Stores and replicates values from a tree structure (hence the name)
  - ✓ Each value is associated with a path and key
- TreeCacheAOP
  - ✓ Can manage caching and replication on full Java objects (POJOs)
  - ✓ Object oriented cache

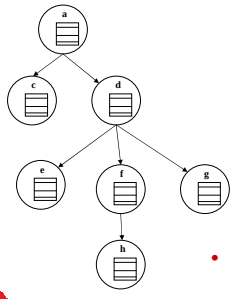
What is JBoss Cache? (continue)

- Uses JGroups as underlying transport stack
- Available as stand-alone or embeddable (MBean)
  - ✓ TreeCacheAOP requires JBossAOP runtime
  - ✓ Can be used with other app servers as well

JBoss Cache Key Feature

- Local and Replication mode
  - ✓ Synchronous (blocking until replication completes)
  - ✓ Asynchronous
- Fine-grained field-level replication using TreeCacheAop component
  - ✓ No need to declare POJO Serializable

TreeCache Architecture



- Tree structure
  - ✓ Each node has a name and zero or more children
  - ✓ Can be navigated recursively from node to node or using a fully qualified name (/a/d/g)
  - ✓ Multiple roots per cache allowed
  - ✓ Each node is a map with keys and values
- Locking
  - ✓ Isolation level per node

JBoss Aop Features

- Aop is a modular way of defining and applying cross-cutting concerns
- Current release 1.1
- Definition: Advice, Aspect, Jointpoint, Interceptor, and Pointcut
- Define a rich set of metadata language as pointcut
- Supports dynamic Aop
- Supports annotation (under JDK1.4 as well)

7

Dynamic Aop

- Adding cache interceptor at runtime

TreeCacheAop.java

```
InstanceAdvisor advisor = ((Advised) obj)._getInstanceAdvisor();
advisor.appendInterceptor(new CacheInterceptor(this, fqn, type));
```

- Declare POJO to be instrumented

jboss-aop.xml

```
<aop>
 <prepare expr="field(* $instanceof{org.jboss.test.cluster.web.aop.Address}->*)" />
 <prepare expr="field(* $instanceof{org.jboss.test.cluster.web.aop.Person}->*)" />
</aop>
```

8

Use of Annotation

- Can also use annotation instead of jboss-aop.xml. Need to run annotation compiler under 1.4.

Person.java

```
import org.jboss.aop.Prepare;
/**
 * @@org.jboss.aop.Prepare ("field(* this->*)" ")
*/
Public class Address {
  String city;
  String street;
  int    zip;
...
}
```

9

TreeCacheAop

- Use JBossAop's dynamic aop feature
- A subclass of TreeCache
  - ✓ all the features of TreeCache
- Eviction policy and replication pojo style as well
  - ✓ Requires a separate aop eviction policy

10

TreeCacheAop

- Fine-grained field-level caching with POJO style
  - ✓ Object graph
    - Sub-object reference
    - Cross and multiple reference
  - ✓ Polymorphism
  - ✓ Inheritance
- Automatic Collections class support
  - ✓ List, Map, and Set
  - ✓ A proxy is generated for the Collections

11

TreeCacheAop API

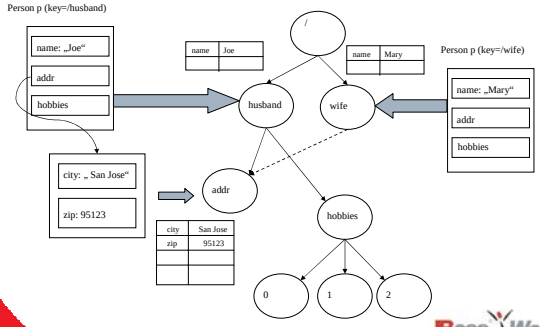
- Plain cache API
  - ✓ put(FQN name, Object key, Object value)
    - Insert an object into cache under name fqn with key in the hashmap
  - ✓ Object get(FQN name, Object key)
    - Retrieve object from name fqn
  - ✓ remove(FQN name, Object key)
    - Remove object from name fqn with key

12

TreeCacheAop API

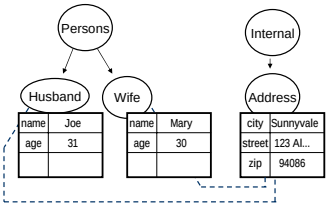
- CacheAop API
  - ✓ putObject(FQN name, Object pojo)
    - Insert an „aspectized“ pojo into cache. Will map the object graph recursively. Need to do it only once.
  - ✓ Object getObject(FQN name)
    - Retrieve pojo from cache. Call this from the failover node, e.g., only once.
  - ✓ removeObject(FQN name)
    - Remove it from cache

TreeCacheAop Mapping



TreeCacheAop Mapping

- Object relationship management has reference counting
  - ✓ Object instance that is referenced more than once is moved to an internal area




```
Person joe = new Person();
joe.setName("Joe Black");
joe.setAge(31);

Person mary = new Person();
mary.setName("Mary White");
mary.setAge(30);

Address addr = new Address();
addr.setCity("Sunnyvale");
addr.setStreet("123 Albert Ave");
addr.setZip(94086);

joe.setAddress(addr);
mary.setAddress(addr);
```


JBoss Cache Configuration

replySync-service.xml

```
<server>
<mbean code="org.jboss.cache.aop.TreeCacheAop"
      name="jboss.cache:service=TreeCacheAop">
...
<attribute
      name="TransactionManagerLookupClass">org.jboss.cache.transaction.JBossTrans
      actionManagerLookup</attribute>
<attribute name="IsolationLevel">REPEATABLE_READ</attribute>
<!-- Valid modes are LOCAL, REPL_ASYNC and REPL_SYNC -->
<attribute name="CacheMode">REPL_SYNC</attribute>

...
```

- Note that this file can be used both inside JBoss and as standalone

Tomcat Http Clustering/High Availability

- Front-end load balancer
  - ✓ Software based. Apache mod_jk
  - ✓ Hardware based. Cisco, etc.
- Session replication
  - ✓ Replication mode
  - ✓ Sticky session during failover

JBoss' Previous Session Replication Solution

- Release 3.2.5 and before
- Distributed State Management on top of JGroups channel
- EJB for in-memory store (and persistence)
- Too complicated and difficult to maintain
- Only supports a single replication granularity (session)

New Http Session Replication Goal

- Leverage existing state replication software stack (e.g., JBossCache)
  - ✓ Replication mode
    - Synchronous or Asynchronous
    - Per server instance
  - ✓ Provide persistency and memory usage control for future releases
- Provide different levels of replication granularity
  - ✓ Session
  - ✓ Attribute
  - ✓ Field
- Replication frequency
  - ✓ Snapshot manager: instant and interval

Replication Granularity Level: Session

- Replication is on per http session object. It is a blob of hash map.
- User can define whether a getAttribute is dirty or not. Such that object updates can be replicated (albeit blindly).
- Implemented using TreeCache API

```
Pojo pojo = (Pojo)session.getAttribute("pojo");
pojo.setName("Ben");
Session.setAttribute("pojo", pojo); // Need to do this manually and replicates the session
```


Replication Granularity Level: Attribute

- Replication is on session attribute only. It is more efficient. Nonetheless, user still needs to manage object relationship.
- Implemented using TreeCache API

```
Pojo pojo = (Pojo)session.getAttribute("pojo");
pojo.setName("Ben");
Session.setAttribute("pojo", pojo); // Only attribute "pojo" gets replicated!
```


Replication Granularity Level: Field

- Replication is on attribute pojo field level where pojo is a user-specified object that needs to be instrumented with aop. Use TreeCacheAop API.

Multiple reference

```
Person joe = (Person)session.getAttribute("joe");
joe.setName("joe"); // Only this field gets replicated.
Address addr = new Address("San Jose", 95123);
Person mary = (Person)session.getAttribute("mary");
mary.setName("mary"); // Only this field gets replicated.
joe.setAddress(addr);
mary.setAddress(addr);
addr.setZip(94086); // Only this field gets replicated.
```

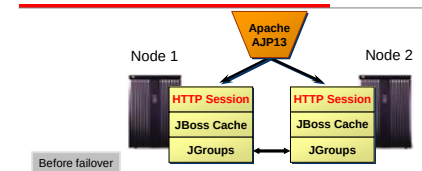

Replication Granularity Level: Field

- When a pojo object size is huge, e.g., a long List of size 100K, field level replication is efficient!

Big pojo object

```
Subsription subs =
(Subscription)session.getAttribute("subscription");
// What if list is size of 100K?
List mailingList = (List)subs.getMailingList();
Person joe = findSubsriber("joe");
joe.setZip(94086); // Only replicates this field!
```


Replication Granularity Level: Field



Before failover

```
Person joe = new Person("joe");
Person mary = new Person("mary");
Address addr = new Address("San Jose", 95123);
joe.setAddress(addr);
mary.setAddress(addr);
session.setAttribute("joe", joe);
session.setAttribute("mary", mary);
```

2nd request

```
addr.setZip(94086); // Will get replicated
```

After failover

```
Person joe = (Person)session.getAttribute("joe");
Person mary = (Person)session.getAttribute("mary");
out.print(joe.getAddress().getZip()); // is 94086
out.print(mary.getAddress.getZip()); // is 94086
```

Http Clustering Configuration

- Run "-c all" configuration
- tc5-cluster-service.xml
  - ✓ Essentially a TreeCacheAop configuration xml that defines the MBean service binding.
- jboss-aop.xml (or annotation in the Pojo)
  - ✓ Needed if using Field level replication granularity
  - ✓ Per web application
- Jboss-service.xml
  - ✓ UseJK. If using mod_jk, can activate this to append the node id to the session for failover.
  - ✓ Per server instance

25

Http Clustering Configuration

- Jboss-web.xml
  - ✓ Replication granularity level
    - SESSION, ATTRIBUTE, FIELD
  - ✓ Replication field batch mode
    - If true, will do replication per request as a batch mode
    - Applies when granularity is FIELD
  - ✓ Replication trigger
    - SET_AND_GET, SET, SET_AND_NON_PRIMITIVE_GET
  - ✓ Per web application

26

Restriction In Current Release

- If using field level granularity, can't mixed "aspectized" pojo with Serializable sub-objects
- Collections class uses proxy so you will need to use the proxy reference instead of the original one

27

Clustering Roadmap

- HA-JNDI
- Distributed State
- EJB
  - ✓ SFSB and Entity
- cluster-service.xml vs. tc-cluster-service.xml
- Unified JGroups management interface

28

JBossCache Roadmap

- JBossCache roadmap
  - ✓ Current at 1.2.1 release
  - ✓ Buddy replication
  - ✓ Optimistic locking
- http://jira.jboss.com/jira/browse/JBCACHE?report=com.atlassian.jira.plugin.system.project:roadmap-panel

29

Release

- Two different releases
  - ✓ Replication granularity of SESSION and ATTRIBUTE
    - 3.2.6 and 4.0.x
  - ✓ Additional granularity of FIELD (use of Aop)
    - > 4.0.2 and head (5.0)

30