

JBossCache: an In-Memory Replicated Transactional Cache

Bela Ban (bela@jboss.org) - Lead JGroups / JBossCache

Ben Wang (bwang@jboss.org) - Lead JBossCacheAOP

JBoss Inc

<http://www.jboss.org>



Overview of the architecture and API of JBossCache

What is JBossCache ?

Architecture

How do I use it ? - API

JBossCache/AOP and POJOs

Demo

JBossCache consists of

- TreeCache

and

- TreeCacheAop

Those are the names of the actual implementation classes

TreeCacheAop extends TreeCache

Tree structured cache

Runs inside JBoss (as MBean) or stand-alone

Local or replicated

- Synchronous or asynchronous replication (using JGroups)

Transactional or non-transactional

- Transactional
 - Replication at TX commit
 - DB isolation levels supported
 - Support for pluggable TxManagers
- Non-Transactional
 - Replication after each modification

Pluggable eviction policies

- Ships with time-based and size-based (LRU) policies

Cache loader

- Persistent backend store (load - store)

Subclass of TreeCache

Supports POJOs (Plain Old Java Objects)

- Uses a combination of reflection and AOP to manage entire objects
- POJOs can be replicated across processes
- POJOs can be persisted using a CacheLoader

POJO-specific eviction policies

What is JBossCache ?

Architecture

How do I use it ? - API

JBossCache/AOP and POJOs

Demo

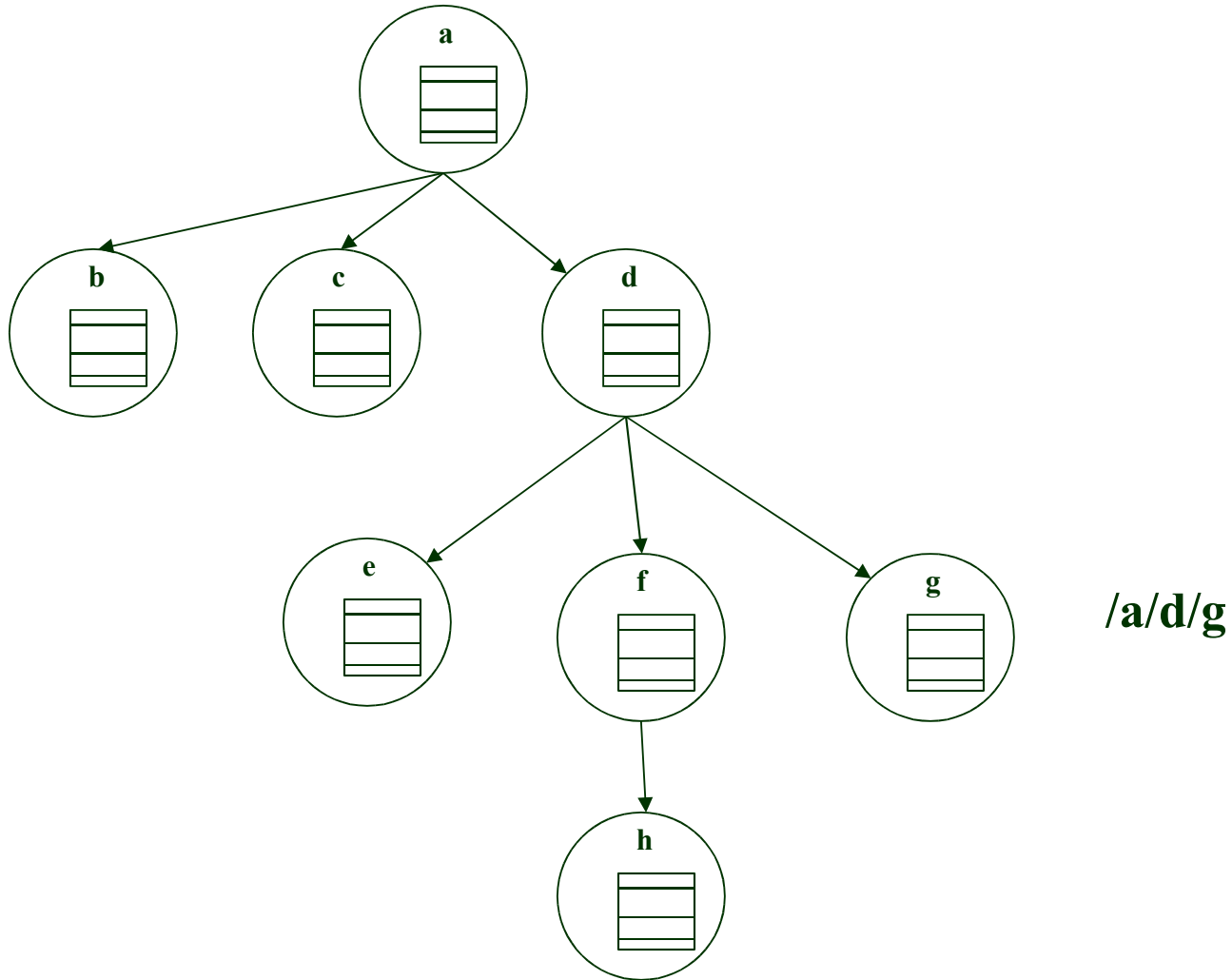
A Tree with nodes

Each Node has

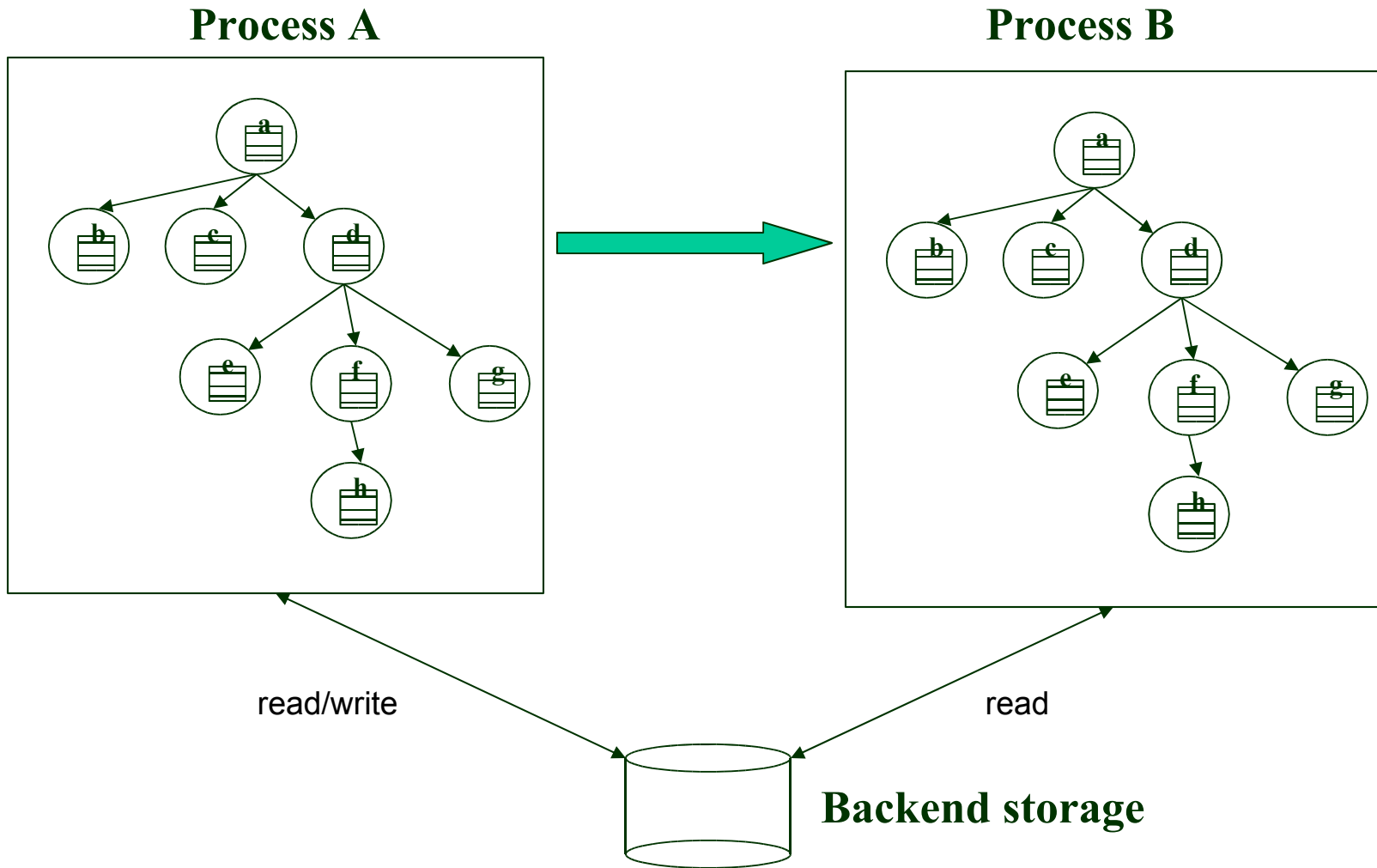
- a name (Object) ("address")
- a fully qualified name (FQN) ("/322649/address")
- child node(s)
- a hashmap (attributes)

Navigation

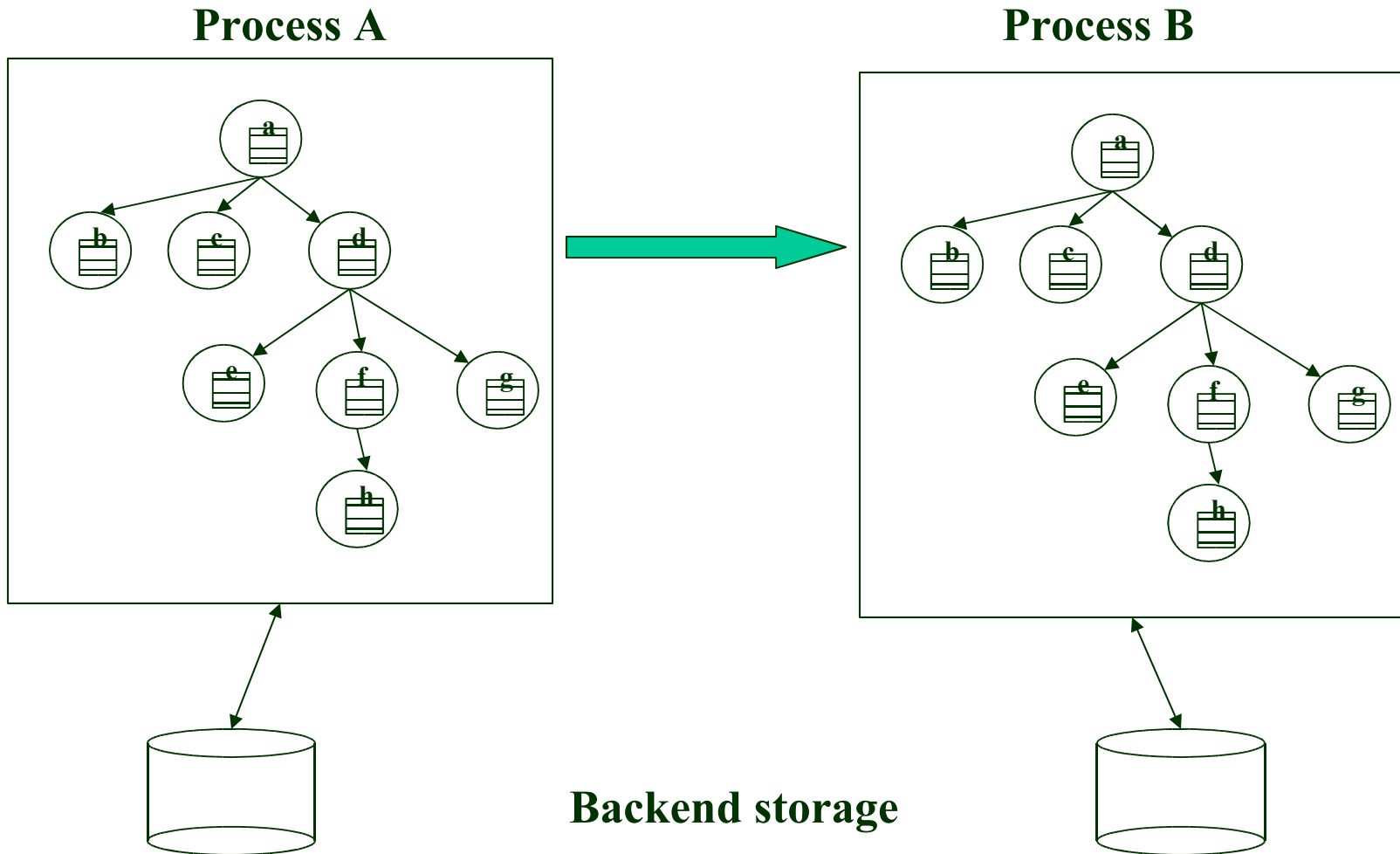
- Get root node and then navigate yourself (Node.getChildren())
- External naming for nodes
 - Strings: "/322649/address"
 - FQNs: `FQN.create(new Object[]{new Integer(322649), "address"})`



Architecture - Replication (shared datastore)



Architecture - Replication (local datastores)



Backend storage

What is JBossCache ?

Architecture

How do I use it ? - API

JBossCache/AOP and POJOs

Demo

- Lifecycle: `create()`, `start()`, `stop()`, `destroy()`
- Configuration (getting / setting mode, locking, timeouts etc)
- `put(FQN name, Object key, Object val)`
- `put(FQN name, Map data)`
- `Object get(FQN name, Object key)`
- `Node get(FQN name)`
- `remove(FQN name)`
- `remove(FQN name, Object key)`
- `boolean exists(FQN name)`

```
try {
    tx=(UserTransaction)new
        InitialContext(p).lookup("UserTransaction");
    TreeCache c=new TreeCache("test", null, 10000);
    c.setMode(TreeCache.REPL_ASYNC);
    c.start();
    tx.begin();
    c.put("/a/b/c", "age", new Integer(38));
    c.put("/a/b/c", "age", new Integer(39));
    tx.commit();
    assertEquals(new Integer(39), c.get("/a/b/c",
        "age"));
}
catch(Throwable t) {
    if(tx != null) tx.rollback();
    fail(t.toString());
}
```

What is JBossCache ?

Architecture

How do I use it ? - API

JBossCache/AOP and POJOs

Demo

Subclass of TreeCache

Enables managing of entire POJOs (= Plain Old Java Objects), vs
managing keys/values in TreeCache

POJOs with links/graphs supported

Adds 3 methods:

- putObject(FQN name, Object pojo)
- Object getObject(FQN name)
- removeObject(FQN name)

Objects are added once, AOP keeps track of state changes and replicates on TX commit. No need to call putObject() again

Only fields that are modified are replicated

POJOs do not need to implement Serializable

POJO - specific eviction policies available

Short detour: JBossAop

Byte code instrumentation (runtime, offline) to intercept

- Constructors
- Method invocations
- Field access (read/write)

Similar to around-methods (CLOS)

Run-time instrumentation

- Classloader overrides loadClass(), instruments classes listed in XML, then calls defineClass()

Offline instrumentation

- AOP compiler (aopc) takes classes/JAR plus XML, generates class/JAR

Pointcuts define what needs to be intercepted

Interceptors define code to be executed when pointcut is reached

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<aop>
```

```
<bind pointcut="execution(* org.apache.jasper.runtime.HttpJspBase->service(..)">
```

```
<interceptor class="org.jboss.aop.JspDemoInterceptor"/>
```

```
</pointcut>
```

```
</aop>
```

```
public class JspDemoInterceptor implements Interceptor {
    public Object invoke(Invocation invocation) throws Throwable {
        MethodInvocation methodInvocation = (MethodInvocation)invocation;
        Object[] args = methodInvocation.arguments;
        HttpServletRequest request = (HttpServletRequest)args[0];
        String uri = request.getRequestURI();
        Integer count = (Integer)urls.get(uri);
        if (count == null) {
            count = new Integer(0);
        }
        count = new Integer(1 + count.intValue());
        urls.put(uri, count);

        return invocation.invokeNext();
    }
}
```

```
public Object invoke(Invocation invocation) throws Throwable {
    if (invocation instanceof FieldWriteInvocation) {
        FieldWriteInvocation fieldInvocation = (FieldWriteInvocation) invocation;
        Field field = fieldInvocation.getField();
        CachedType fieldType = cache.getCachedType(field.getType());
        if (fieldType.isImmediate())
            cache.put(fqn, field.getName(), fieldInvocation.getValue());
        else
            cache.putObject(new Fqn(fqn, field.getName()), value);
    } else if (invocation instanceof FieldReadInvocation) {
        FieldReadInvocation fieldInvocation = (FieldReadInvocation) invocation;
        Field field = fieldInvocation.getField();
        CachedType fieldType = cache.getCachedType(field.getType());
        Object result;
        if (fieldType.isImmediate())
            result = cache.get(fqn, field.getName());
        else
            result = cache.getObject(new Fqn(fqn, field.getName()));
        ...
    }
}
```

Interceptors can be arranged in sequence

Pointcut is handled by all interceptors

Any interceptor can return the call (normal termination or exception)

At the end of the interception --> call next interceptor in chain

- return invocation.invokeNext();

Interceptor list can be accessed and modified:

```
import org.jboss.aop.Advised;

public static void main(String[] args) {
    POJO p = new POJO();
    Advised obj = (Advised)p; //Typecast
    obj._getInstanceAdvisor().insertInterceptor(new MetricsInterceptor());
    ...
}
```

Back to TreeCacheAop now

Uses reflection to discover the structure of a POJO

Uses AOP to keep track of changes to a POJO

putObject() breaks an object apart and maps it to the TreeCache

- Primitive fields are mapped to entries in a node's attributes
- Complex fields are mapped to child nodes (recursively)
- We dynamically add a field interceptor to each complex object to keep track of state changes

Each interceptor remembers the node to which it maps

On field read: interceptor returns the value from the TreeCache

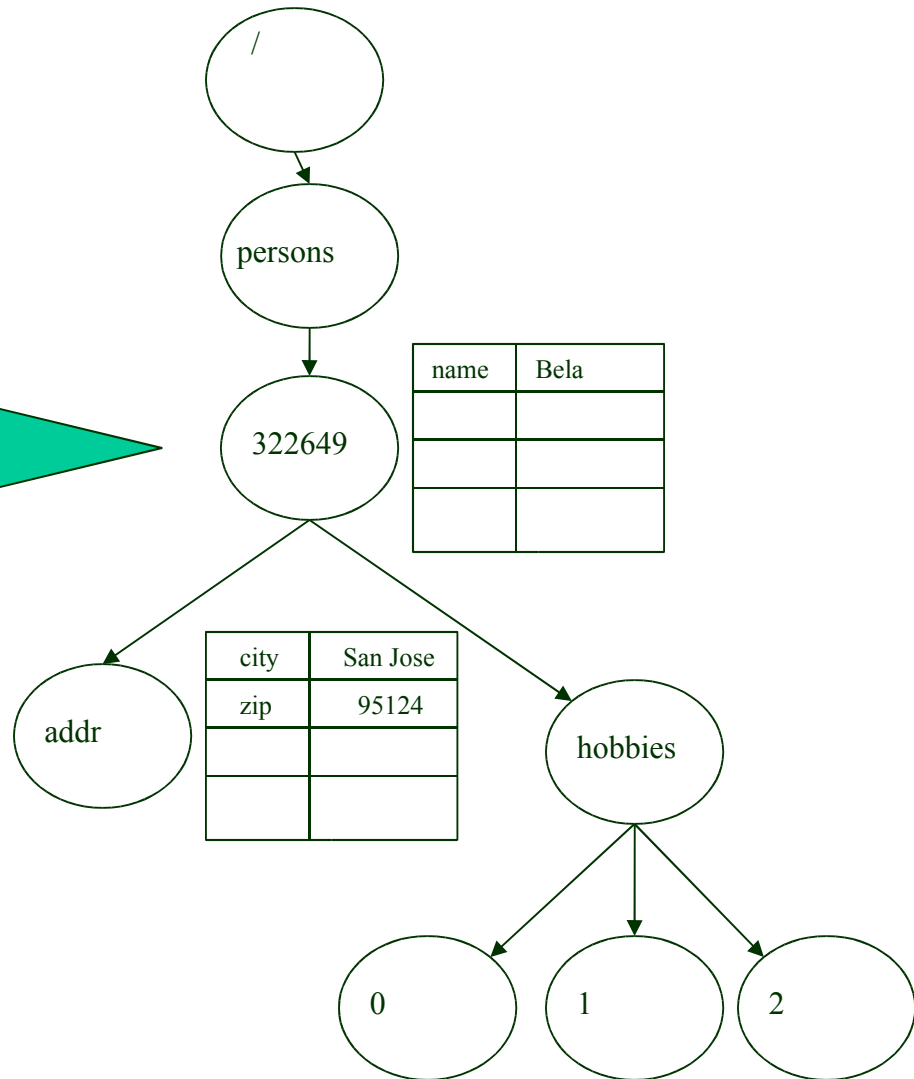
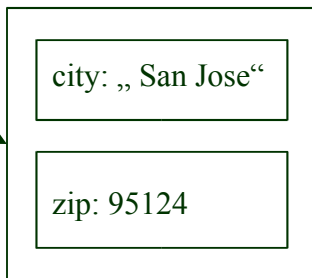
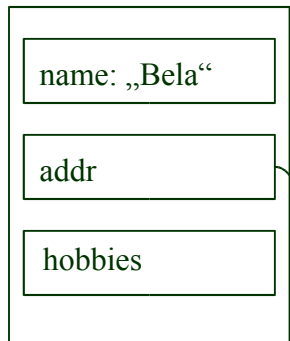
On field write: interceptor updates the associated node's attributes,
e.g.

- `person.getAddress().setCity("San Jose")` generates a `put("/322649/addr", "city", "San Jose")`

On TX commit: modified fields are replicated and written back to the POJO

TreeCacheAop - Mapping

Person p (key=/persons/322649)



How to instrument system classes ?

- Collection support implemented via dynamic proxies

`jboss-aop.xml` has to include the classes to be added to `TreeCacheAop`

- Fall-back to serialization if not

Instrumentation

- Inside JBoss: classloader (load-time) or aopc (offline)
- Standalone: system classloader
(`org.jboss.aop.standalone.SystemClassLoader`) or aopc (offline)

Objects already loaded need to be redeployed to be instrumented at runtime (redeployment work only inside JBoss)

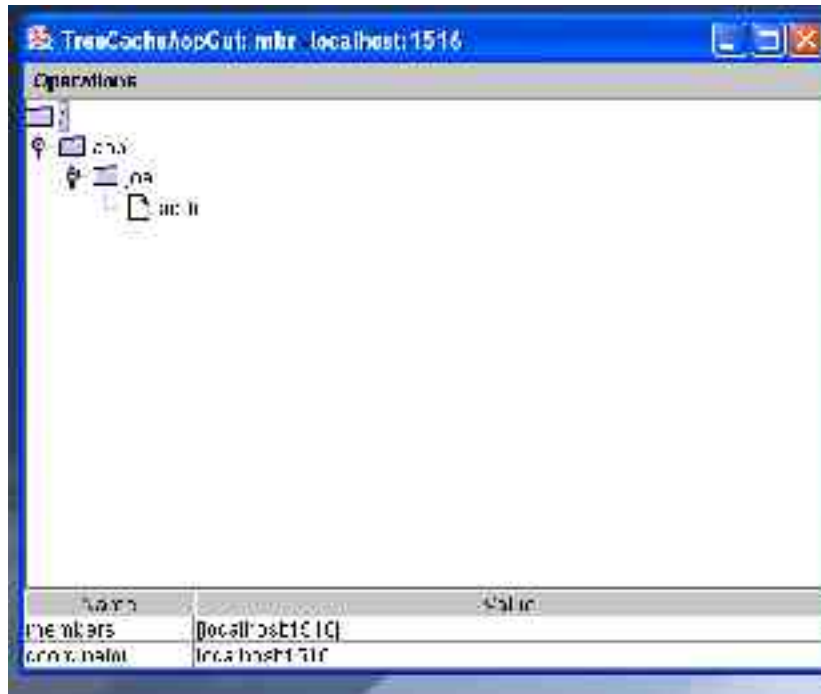
What is JBossCache ?

Architecture

How do I use it ? - API

JBossCache/AOP and POJOs

Demo

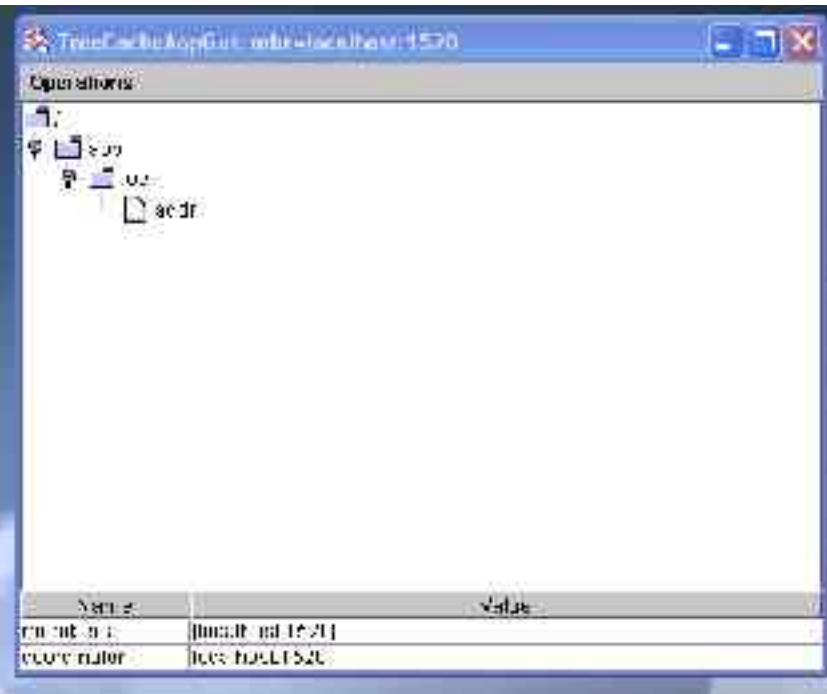


TreeCacheAppGui: mbr: localhost:1516

Operations

- +
- +
- +
- +
- +

| name | value |
|-----------|------------------|
| members | [localhost:1516] |
| conn.name | localhost:1516 |



TreeCacheAppGui: mbr: localhost:1520

Operations

- +
- +
- +
- +
- +

| name | value |
|-----------|------------------|
| members | [localhost:1520] |
| conn.name | localhost:1520 |

JBossCache is a tree-structured cache

Can be 'aspectized' (a.k.a. configured) with

- replication
- transactions / locking (isolation levels)
- eviction policies
- persistence
- roll your own (cache hit/miss ratio metrics, auditing)

Can be used for plain data, or POJOs

Can be plugged into other appservers

Use

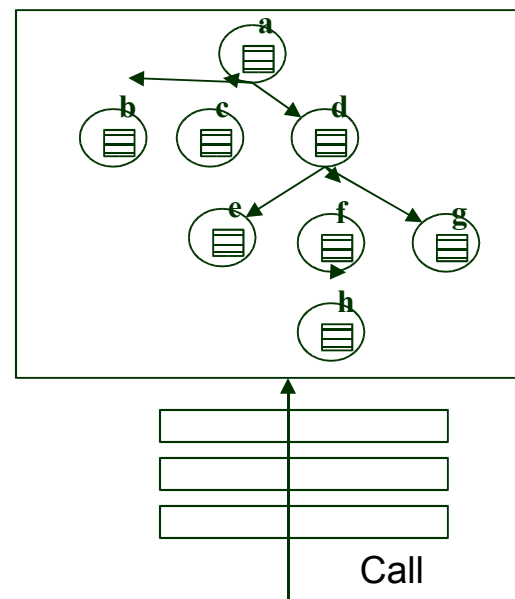
- JBoss/TC5 HTTP session replication, HA-JNDI, SFSB Repl, Hibernate 2nd level cache

JBoss/Tomcat 5 fine-grained session replication

JDBC CacheLoaders / common state transfer format

Aspectizing of JBossCache

- JBossCache not only aspectizes POJOs, but also uses aspects for its own implementation
- Possible aspects:
 - Replication (repl-to-all, buddy-replication)
 - Locking
 - Eviction
 - Persistence (CacheLoaders)
 - Security



Buddy Replication

Optimistic locking

Links

- www.jboss.org (documentation, tutorial, forum)
- JGroups: www.jgroups.org