



JBoss Clustering

The Good, the Bad and the Ugly



Andreas Schaefer, Senior Software Engineer

andy@madplanet.com

madplanet.com/weblog

Founder of Madplanet.com Inc.

Independent Consultant

OnJava.com: Maven Tips & Tricks

Co-author of JMX book



A Word for my Sponsor

- **Madplanet.com Inc.** provides:
 - ✿ In-depth JBoss Consulting including
 - ◆ Code Analysis
 - ◆ Patches
 - ◆ 24 / 7 Technical Support
 - ✿ Java / J2EE Project Consulting
 - ◆ Project Management
 - ◆ Build Systems / Continuous Integration
 - ◆ Product Evaluation
 - ◆ Java / J2EE Development
 - ✿ Java / J2EE Contracting

Table of Content:



● JBoss Clustering

✱ **Introduction**

✱ Concepts

✱ HA-JNDI

✱ HA-JMS

✱ EJB Clustered

✱ Basics

✱ JGroups

✱ JBoss Cache

✱ Troubleshooting

✱ Q & A

Setup



- Extract JBoss (I use 4.0.5.GA)
- Install it:
 - ✱ On two separate boxes
 - ✱ Or duplicate the all environment and use the Service Binding to keep the ports apart
- Fire up the first box with the all environment
- After this is up and running start the second
 - ✱ The console of the first box should print out:

```
INFO [DefaultPartition] New Members : 1 ([192.170.1.6:1199])  
INFO [DefaultPartition] All Members : 2 ([192.170.1.6:1099, 192.170.1.6:1199])
```



- An Application needs to be deployed on each Node
- Farm Service automates this
 - ✱ Deploy it under farm deployment directory: <server env.>/farm
- Application is then:
 - ✱ First deployed on the local Node
 - ✱ Then copied to each other Node where it is deployed subsequently
 - ✱ Each Node has now a copy of the file in the farm deployment directory
- Application can be undeployed by deleting the file on any Node in the farm deployment directory



● EJBs

- ✿ Using HA-JNDI to lookup remote objects
 - ◆ Client provides a list of HA-JNDI server
 - ✗ `java.naming.provider.url=localhost:1100,localhost:1200`
 - ◆ Client uses automatic node detection
 - ✗ Either leave the `java.naming.provider.url` empty
 - ✗ Or when all the servers in list fails it will try it to find new nodes
- ✿ The Home / Remote interface Stub knows how to fail over
- ✿ The LocalHome / Local interface don't need to know it

● Web

- ✿ Load Balancer
- ✿ Apache Web Server with `mod_jk` or `mod_jk2`



● JMS

- ✿ Access remote JMS Objects through HA-JNDI
 - ◆ No fail-over on JMS Connection Factories and the objects it creates
 - ◆ Client must:
 - ✗ detect failure
 - ✗ lookup the Connection Factory from HA-JNDI
 - ✗ recreate all the necessary objects
 - ✗ reissue the calls
- ✿ Only use **java:/JmsXA** inside the application server
- ✿ Destinations are immune to node failures
- ✿ MDBs do a crude failover by failing and reconnect



● JMX

- ✿ Use **JRMPProxyFactory** to bind MBean to JNDI
- ✿ To add Failover take the approach from **ProxyFactoryHA** and adjust the **JRMPProxyFactory** with it
 - ◆ Get **HAPartition** from JNDI
 - ◆ Obtain **DistributedReplicantManager** from HA Partition
 - ◆ Obtain **InvokerHA** from JBoss Registry
 - ◆ Create **HATarget** and register HA Target with Invoker HA
 - ◆ Register a Listener with DistributedReplicantManager
 - ✗ Listener will be notified when the cluster setup has changed
 - ◆ Create **Proxy** with Invoker HA and bind it to JNDI

Table of Content:



● JBoss Clustering

- ✱ Introduction
- ✱ **Concepts**
- ✱ HA-JNDI
- ✱ HA-JMS
- ✱ EJB Clustered
- ✱ Basics
- ✱ JGroups
- ✱ JBoss Cache
- ✱ Troubleshooting
- ✱ Q & A

Clustering Concepts



- Not mentioned in the Java EE specification
- Means that any configuration must work as specified
- A Java EE cluster means:
 - ✱ Several Application Server Instance work like one
 - ✱ The provide:
 - ◆ High Availability
 - ◆ Load Balancing
 - ✱ Vendor specific implementation
 - ✱ Should conform to the Java EE specification like a single server

High Availability



- Increase the availability of a Java EE application
- In case of a failure to connect to a server it transparently redirects to another node -> failover
- Nodes can be added or removed to the cluster without any disruption of the application's behavior
- Nodes must detect a crash / stall of another node
 - ✱ Because the failing node is not able to send a message
- HA is not available through JBoss on:
 - ✱ Database and therefore also JMS Data
 - ✱ Web Server

Load Balancing



- A load can be distributed over the nodes of a cluster to
 - ✱ Shorten the execution to avoid timeouts and deadlocks
 - ✱ Better utilization of the available resources
 - ✱ Go beyond limitation of the JDK
- Improve the Scalability of the Application(s)
- Can be achieved even without clustering

Table of Content:



● JBoss Clustering

- ✱ Introduction
- ✱ Concepts
- ✱ **HA-JNDI**
- ✱ HA-JMS
- ✱ EJB Clustered
- ✱ Basics
- ✱ JGroups
- ✱ JBoss Cache
- ✱ Troubleshooting
- ✱ Q & A



- JNDI with the ability to find an object on another node on the cluster
- The connection is not HA enabled:
 - ✱ Use list of HA-JNDI server in the provider url
 - ✱ Or let the system try to auto detect
- Only lookup() is working properly
 - ✱ list() and other methods are not working and so it is not possible:
 - ◆ list HA-JNDI on the JNDIView Mbean
 - ◆ Use an External Context on HA-JNDI
- Entries are looked up first locally



- Look Ups are not load balanced but executed in the order of the list of nodes
- The list of nodes must not contain all **nodes** of the cluster
- Pitfalls:
 - ✱ Do not bind different EJBs with the same JNDI on different nodes
 - ✱ Make sure that at least one node is up and running that is listed in the provider url list
 - ✱ Make sure that the HA-JNDI port is opened on a firewall

Table of Content:



● JBoss Clustering

- ✱ Introduction
- ✱ Concepts
- ✱ HA-JNDI
- ✱ **HA-JMS**
- ✱ EJB Clustered
- ✱ Basics
- ✱ JGroups
- ✱ JBoss Cache
- ✱ Troubleshooting
- ✱ Q & A



- HA but no Load Balancing
- Uses a Master – Slave(s) Model
 - ✱ Only the Master is the active JMS Server
 - ✱ Slaves have no JMS Server running
 - ✱ When Master dies a Slave is promoted to become the new Master and then starts the JMS Server
 - ✱ JMS Messages provided by a Database (default) are only HA when the DB is HA (clustered) as well
 - ✱ Temporary Separation may lead to two Masters
- Each node must be configured identical
- Failover changes Topology permanently



- Connection Factories and Destinations must be looked up by HA-JNDI
 - ✱ Client does not know the current Master node
 - ✱ Only the HA-JNDI can find the current Master
 - ✱ EJB JMS Resource Refs must point to local HA-JNDI server
 - ◆ **jnp://\${jboss.bind.address}:1100/queue/A**
 - ✱ Not necessary for “**java:/JmsXA**” because it is **local**
 - ◆ **Attention**: this prefix does only work with HA-JNDI
- JMS Objects do not know how to fail over
 - ✱ Client must detect node failures
 - ✱ Client must know how to reconnect



● JMS Client on the Cluster

- Queue: only one node receives a given message -> load balancing
- Topic: every node is treated as a client and therefore receives a copy of a sent in message
- Durable Subscription: second node would fail because there is already a registered client with that name
 - Such a client must be deployed in its own application within the HA-Singleton directory
 - This way only one copy of the application is running on the cluster

● Server Connection Factory “java:/JmsXA” is HA

Table of Content:



● JBoss Clustering

- ✱ Introduction
- ✱ Concepts
- ✱ HA-JNDI
- ✱ HA-JMS
- ✱ **EJB Clustered**
- ✱ Basics
- ✱ JGroups
- ✱ JBoss Cache
- ✱ Troubleshooting
- ✱ Q & A



- Clusterable EJBs
 - ✱ Stateless Session Bean
 - ✱ Stateful Session Bean
 - ✱ Entity Beans
- Enabling Clustering in **jboss.xml**

```
<jboss>
  <enterprise-beans>
    <session>
      <ejb-name>Sample_EJB</ejb-name>
      <clustered>True</clustered>
    </session>
  </enterprise-beans>
</jboss>
```



EJB: Stateless Session Bean

- Transparent failover
 - ✱ by default only one failure (SingleRetryInterceptor)
 - ◆ Node A in the list, Node B comes alive and A dies afterwards
 - ◆ This leads to two exceptions
 - ✕ Could not connect to EJB
 - ✕ Failed to lookup HA-JNDI on A (first in the list)
- Use `org.jboss.proxy.ejb.RetryIntercept`
 - ✱ This class leads to unlimited number of retries
 - ✱ Create custom class extending `RetryInterceptor` for a limited number of retries

EJB: Stateless Session Bean II



- Retry Interceptor is set in **invoker-proxy-bindings**

```
<invoker-proxy-binding>
  <proxy-factory-config>
    <client-interceptors>
      <home>
        <interceptor>
          org.jboss.proxy.ejb.RetryInterceptor
        </interceptor>
      </home>
    </client-interceptors>
  </proxy-factory-config>
</invoker-proxy-binding>
```

Either adjust **standardjboss.xml** or add it to your **jboss.xml**

- Customer Interceptor is set in **jboss.xml**:

```
<session>
  <invoker-bindings>
    <invoker>
      <invoker-proxy-binding-name>
        mpg-clustered-stateless-rmi-invoker
      </invoker-proxy-binding-name>
    </invoker>
  </invoker-bindings>
</session>
```



EJB: Stateless Session Bean III

- Simple Load Balancing
- Available Policies for Home / Remote Stub
 - Round Robin: each calls goes to another node
 - First Available: Each Interceptor elects a main target and calls this each and every time until it fails, then it elects a new one
 - First Available Identical All Proxies: each member of a proxy family will use the same, elected main target
- LB Policy is set in the jboss.xml Session Bean:

```
<session>
  <cluster-config>
    <home-load-balance-policy>
      org.jboss.ha.framework.interfaces.RoundRobin
    </home-load-balance-policy>
  </cluster-config>
```



● State Replication

- ✱ State is replication throughout the Cluster
- ✱ JBoss replicates the entire SFSB instance
 - ◆ Passivation mandates that the SFSB is fully serializable
 - ◆ JBoss uses this feature to replicate the instance
- ✱ Bean can control replication by implementing
 - ◆ *public boolean isModified()*
- ✱ Replication is done by this MBean
 - ◆ `org.jboss.ha.hsessionstate.server.HASessionStateService`



EJB: Message Driven Bean

- Not Clusterable

- ✱ Because the caller (JCA) is a local client

- ✱ Cluster has an impact on the MDB design

- ◆ Queue -> Load Balancing

- ◆ Topic -> multiplying Messages by the number of nodes can be avoided by deploying on the HA Singleton

- ◆ Durable Subscription -> deployment errors can be fixed by deploying on the HA Singleton



Web Clustering

- JBoss Clustering Support for the Web
 - ✱ Replicates the HttpSession
 - ✱ If a client can failover to another node it will not lose its HttpSession
- JBoss does not support load balancing
 - ✱ because the client calls the server directly (no proxy in between)
 - ✱ Call must be directed to a load balancer
 - ◆ hardware
 - ◆ software
 - ✕ Apache Web Server using **mod_jk** to connect to the cluster

Table of Content:



● JBoss Clustering

- ✱ Introduction
- ✱ Concepts
- ✱ HA-JNDI
- ✱ HA-JMS
- ✱ EJB Clustered
- ✱ **Basics**
- ✱ JGroups
- ✱ JBoss Cache
- ✱ Troubleshooting
- ✱ Q & A



• JBoss Clustering Files and Directories

- `deploy/`: is only deployed locally
 - `cluster-service.xml`: Cluster configuration
 - `deploy-hasingleton-service.xml`: HA Singleton configuration
 - `deploy.last/farm-service.xml`: Farm Deployer Configuration
 - `httpha-invoker.sar`: replaces `http-invoker.sar`
 - `tc5-cluster.sar`: TreeCache for Tomcat 5 Cluster
- `deploy-hasingleton/`: like `deploy` directory but is only deployed if that node become the HA Singleton (master)
- `farm/`: its content is deployed on all nodes of the cluster or contains the files that were deployed through the farm service



JBoss Clustering Basics II

- JBoss uses JGroups to build the Cluster Network
- JBoss uses JBossCache to distribute Data
- JGroups used by default **guaranteed UDP calls**
 - ✱ but also supports TCP and Tunnel (messages are sent to an external router to bridge network boundaries or firewalls)
- JBoss Cluster is defined by
 - ✱ Partition Name (default: DefaultPartition)
 - ✱ Network Boundaries
-

Table of Content:



● JBoss Clustering

- ✱ Introduction
- ✱ Concepts
- ✱ HA-JNDI
- ✱ HA-JMS
- ✱ EJB Clustered
- ✱ Basics
- ✱ **JGroups**
- ✱ JBoss Cache
- ✱ Troubleshooting
- ✱ Q & A



● JGroups Architecture

- ✿ Transport (Default: UDP)
- ✿ Failure Detection
- ✿ Acknowledgment: Unicast / Nakack
- ✿ Group Member Ship (GMS): handles a group of nodes (cluster)
- ✿ Channel: a way a client talks to a Group by a Group Name
 - ◆ Cluster: Partition Name
- ✿ Building Blocks: glue between Application and Channel

- JGroups is defined in cluster-service.xml file for the cluster



```
<mbean code="org.jboss.ha.framework.server.ClusterPartition"
  name="jboss:service=${jboss.partition.name:DefaultPartition}"
>
  <attribute name="PartitionName">
    ${jboss.partition.name:DefaultPartition}
  </attribute>
  <attribute name="NodeAddress">
    ${jboss.bind.address}</attribute>
  <attribute name="DeadlockDetection">False</attribute>
  <attribute name="StateTransferTimeout">30000</attribute>
  <attribute name="PartitionConfig">
    <Config>
      <UDP mcast_addr="${jboss.partition.udpGroup:228.1.2.3}"
        mcast_port="45566"
        ip_ttl="${jgroups.mcast.ip_ttl:8}"
        ip_mcast="true"
        mcast_rcv_buf_size="2000000"
        mcast_send_buf_size="640000"
        ucast_rcv_buf_size="2000000"
        ucast_send_buf_size="640000"
        loopback="false"/>
    </Config>
  </attribute>
</mbean>
```



```
<PING timeout="2000" num_initial_members="3"
  up_thread="true" down_thread="true"/>
<MERGE2 min_interval="10000" max_interval="20000"/>
<FD SOCK down_thread="false" up_thread="false"/>
<FD shun="true" up_thread="true" down_thread="true"
  timeout="10000" max_tries="5"/>
<VERIFY_SUSPECT timeout="3000" num_msgs="3"
  up_thread="true" down_thread="true"/>
<pbcast.NAKACK
  gc_lag="50" retransmit_timeout="300,600,1200,2400,4800"
  max_xmit_size="8192" up_thread="true" down_thread="true"/>
<UNICAST timeout="300,600,1200,2400,4800" window_size="100"
  min_threshold="10" down_thread="true"/>
<pbcast.STABLE desired_avg_gossip="20000"
  max_bytes="400000" up_thread="true" down_thread="true"/>
<FRAG frag_size="8192" down_thread="true" up_thread="true"/>
<pbcast.GMS join_timeout="5000" join_retry_timeout="2000"
  shun="true" print_local_addr="true"/>
<pbcast.STATE_TRANSFER up_thread="true" down_thread="true"/>
</Config></attribute>
<depends>jboss:service=Naming</depends></mbean>
```

Table of Content:



● JBoss Clustering

- ✱ Introduction
- ✱ Concepts
- ✱ HA-JNDI
- ✱ HA-JMS
- ✱ EJB Clustered
- ✱ Basics
- ✱ JGroups
- ✱ **JBoss Cache**
- ✱ Troubleshooting
- ✱ Q & A



JBoss Cache's Tree Cache

- TreeCache is the data structure JBossCache manages and replicates
- TreeCache is a Data Structure with these features
 - ✱ in memory
 - ✱ replicated
 - ✱ transacted
 - ✱ persistent
- JBoss Cluster uses Tree Cache for
 - ✱ HTTP Session Replication
 - ✱ SFSB State Replication
 - ✱ Cached Entity Beans



Tree Cache II

- Custom Tree Caches can be added
 - preferable as MBeans
- Standalone TreeCache still should only contain serializable entires
- One TreeCache can have multiple Roots
 - but require the use of the TC API -> `org.jboss.cache.Fqn`
- Beware of Fqn:
 - `new Fqn(“/test/one”) -> ROOT.”/test/one”`
 - this String is NOT parsed
 - `Fqn.fromString(“/test/one”) -> ROOT.”test”.”one”`
 - this String is parsed



Tree Cache III

● Replication

- ✱ Data are replicated to other nodes to distribute it
 - ◆ Best suited when there are more reads than writes
- ✱ Buddy List / Pools reduces traffic
- ✱ For Failover TC uses Data Gravitation to locate Data
 - ◆ On another node if not found locally
 - ◆ If enabled on another node's backup data

● Invalidation

- ✱ Entries on other nodes are invalidated on an update
 - ◆ Best suited when there are more writes than reads

● Synchronous or Asynchronous Message Delivery



Tree Cache IV

● Pessimistic Locking

✱ Isolation Level

- ◆ None
- ◆ Read Uncommitted -> Dirty Reads
- ◆ Read Committed -> Non Repeatable Reads
- ◆ Repeatable Reads -> Phantom Reads
- ◆ Serializable



Tree Cache V

● Optimistic Locking

- ✿ Uses a Workspace to hold the changed data
 - ◆ Each has a Version
 - ◆ At the end of a Tx it is merged back into the Tree
 - ✕ If older then the transaction is rolled back
 - ✕ Otherwise the version of the Tree is updated
- ✿ Only locks while
 - ◆ building a Workspace at the start of the Tx
 - ◆ committing the Tx
- ✿ Isolation Level does not apply



Tree Cache VI

● Eviction

- ✱ When Resources gets tight TC must evict entries
 - ◆ Eviction Policy defines which entries

● Eviction Policies available

- ✱ LRU: last recently used
- ✱ FIFO: first in first out
- ✱ MRU: most recently used
- ✱ LFU: least frequently used

● Cache Loader

- ✱ Stores and Retrieves Nodes from a Data Store
- ✱ Can be used during Node Passivation / Activation on Eviction

Tree Cache VII



```
<mbean code="org.jboss.cache.TreeCache"
  name="jboss.cache:service=MyTreeCache">
  <depends>jboss:service=Naming</depends>
  <depends>jboss:service=TransactionManager</depends>
  <attribute name="TransactionManagerLookupClass">
    org.jboss.cache.BatchModeTransactionManagerLookup
  </attribute>
  <attribute name="IsolationLevel">REPEATABLE_READ</attribute>
  <attribute name="CacheMode">REPL_SYNC</attribute>
  <!-- Every Node in the Cluster must use this name when
    deploying this MBean to share data -->
  <attribute name="ClusterName">my-cache</attribute>
  <attribute name="ClusterConfig">
    <config> ... JGroups Configuration ... </config>
  </attribute>
  <attribute name="InitialStateRetrievalTimeout">
    5000</attribute>
  <attribute name="SyncReplTimeout">10000</attribute>
  <attribute name="LockAcquisitionTimeout">15000</attribute>
</mbean>
```



Tree Cache VIII

- Lookup the TreeCache interface on HA-JNDI

```
<mbean
  code=
  "org.jboss.invocation.jrmp.server.JRMPProxyFactory"
  name=
  "mpg.cache:service=proxyFactory,type=jrmp,target=mytreecache"
>
  <attributename="InvokerName">
    jboss:service=invoker,type=jrmp
  </attribute>
  <attribute name="TargetName">
    mpg.cache:service=MyOwnTreeCache
  </attribute>
  <attribute name="JndiName">MyCache</attribute>
  <attribute name="InvokeTargetMethod">>true</attribute>
  ...
```

Tree Cache IX



```
...
<attribute name="ExportedInterface">
  org.jboss.cache.TreeCacheMBean
</attribute>
<attribute name="ClientInterceptors">
  <interceptors>
    <interceptor>
      org.jboss.proxy.ClientMethodInterceptor
    </interceptor>
    <interceptor>
      org.jboss.proxy.SecurityInterceptor
    </interceptor>
    <interceptor>
      org.jboss.invocation.InvokerInterceptor
    </interceptor>
  </interceptors>
</attribute>
<depends>jboss:service=invoker,type=jrmp</depends>
<depends>mpg.cache:service=MyOwnTreeCache</depends>
</mbean>
```



- Same as Tree Cache except
 - ✱ it is not relying on serialization
 - ✱ it used AOP to replicate the Data
 - ◆ can replicate on field level
 - ◆ can avoid the duplication of shared objects
- Supports all the other features of the Tree Cache
- Objects must be (AOP) Prepared and Instrumented
- Java 5+ one can use Annotations
-

Table of Content:



- JBoss Clustering
 - ✱ Introduction
 - ✱ Concepts
 - ✱ HA-JNDI
 - ✱ HA-JMS
 - ✱ EJB Clustered
 - ✱ Basics
 - ✱ JGroups
 - ✱ JBoss Cache
 - ✱ **Troubleshooting**
 - ✱ Q & A



Troubleshooting Tips

- Write a simple Test Case and use the JBoss default cluster environment (all) to deploy and test it
- Use “Port Shifts” with Service Binding to stack JBoss servers vertically
- Use Xen or any other Virtualization to simulate a real cluster setup
- Avoid Hot Deployment especially in production
- To make a clean deployment through the farm service
 - ✱ Delete the application
 - ✱ Restart the complete cluster
 - ✱ Deploy it the updated application



Troubleshooting Tips II

- Remove any unnecessary services
 - ✱ CORBA not needed for a pure Java application
- Change the JBoss default Configurations
 - ✱ But do it step by step
 - ✱ Use your own configuration files
- Create Application / EJB specific configuration
 - ✱ Invocation-proxy-bindings
 - ✱ Container-configurations
- Keep other single point of failures in mind
 - ✱ DB, Web Server, Network, JCA sources



Troubleshooting Tips III

- Test various Scenarios
 - ✱ Different Number of Nodes
 - ✱ Various restart scenarios
 - ✱ Kill a JVM
 - ✱ Disrupt Network
 - ◆ Isolate node(s)
 - ◆ Reconnect them after a while
- Ensure a proper start of the cluster
 - ✱ Especially when using the HA-Singleton

Table of Content:



● JBoss Clustering

- ✱ Introduction
- ✱ Concepts
- ✱ HA-JNDI
- ✱ HA-JMS
- ✱ EJB Clustered
- ✱ Basics
- ✱ JGroups
- ✱ JBoss Cache
- ✱ Troubleshooting
- ✱ **Q & A**

Q & A



Andreas Schaefer

andy@madplanet.com

Web Log

madplanet.com/weblog/

Presentation can be found on

madplanet.com/mpg/profile/presentations.html