Cluster-Enabling Sakai with Terracotta

Cris J. Holdorph
Software Architect
Unicon, Inc.

JASIG Conference
Dallas, TX
March 3, 2009

UNICON

Agenda

1. Why Terracotta?
2. Current Status
3. Design and Development
4. Running Terracotta-Enabled Sakai
5. Terracotta-Cluster Enabling a Tool
6. Demo
7. Resources

Why Terracotta?

Why Terracotta

- Clustering / Session Failover problem
- What options were explored?
  - Container Managed Sessions
  - Shared Session Server
    - Ehcache
    - Database
  - REST style (no session data) web architecture
- What problems were uncovered?
  - Custom Classloaders
  - Performance
  - Serializable Objects

Enter Terracotta

- Semi-shared Session Server
  - Don't share everything
- Reduced Communication Between Nodes
  - “Partitioned Data” feature of Terracotta
- Less Invasive Sakai Changes
  - Not every object in Terracotta has to be Serializable
  - Many changes can be done in a Terracotta configuration file (tc-config.xml)
- Provides a Custom Classloader Solution

Current Status

Current Status

- John Wiley & Sons sponsored Unicon to develop Terracotta integration with Sakai
- Documentation is in Sakai Confluence
- Original Work done as a feature branch against Sakai 2.5.x
- Migrated into Sakai Trunk / Kernel Trunk
  - This will be Sakai 2.7.x / Kernel 1.1.x if released
- Feature branch (only) contains 2 cluster enabled tools
  - Custom Worksite Setup
  - Resources / Citations

Design and Development

Component Manager

- Create Custom Classloader
- Detecting “sakai.cluster.terracotta” System Property
- Using Custom Classloader if property is set

Code sample

https://source.sakaiproject.org/svn/kernel/trunk/component-manager/src/main/java/org/sakaiproject/util/TerracottaClassLoader.java

```
...
class TerracottaClassLoader extends URLClassLoader {
   private String classLoaderName;
   public TerracottaClassLoader(URL[] urls, ClassLoader
parent, String classLoaderName) { ... }
   public String __tc_getClassLoaderName() {
        return classLoaderName;
   }
...
}
```

SessionManager

- Make m_sessions a Terracotta Root
- Make MySession and MyLittleSession top level classes
  - Did not want *Outer* class, SessionManager, brought into the Terracotta cluster

Partial Session Sharing

- Not all data will be shared
- Original Idea, catch NonPortableException
  - Did not work
- New Idea, Tool whitelist
  - Only tools in the whitelist will be *shared*
  - Tools *not* in the whitelist will *not* be shared
- Final solution, enables gradually enabling clustering on a tool-by-tool basis
- Even if a tool supports clustering, a system administrator can turn clustering off for that tool

Session Maintenance

- SessionManager class has an inner class/thread called Maintenance
  - Responsible for removing expired sessions
- Need to make sure this behavior still works
- Need to make sure this thread/class does not violate the Terracotta “Partitioned Data” principle
- Solution: Create a parallel data structure to loosely track session timeout times along with session identifiers

UsageSession

- UsageSession tracks with user is on which server
- UsageSession is both a JVM object and database record
- Need to detect when a user has *failed over* to a new server and update the users UsageSession object and database record

RequestFilter

- Terracotta requires a lock (transaction) be obtained to change any data in the cluster
- Original design tried to automatically lock any clustered object before changing data on that object
- New design uses a *Course Grained* lock on the *Session* object, obtained and released in the Sakai RequstFilter

Unit Tests

- Unit Tests for Component Manager before Component Manager changes were made
- Unit Tests for Session Manager before Session Manager changes were made

Session Logging

- Code was added to Session Manager to enable logging of which tools put which objects into a Sakai Session
- This helps identify
  - Heavy users of the Session
  - Relative size of a Session
  - Which tools are more self referencing
  - Which tools tend to reference many other tools/components

Worksite Setup

- Customized Worksite Setup Tool for John Wiley & Sons
- Multi-page wizard
- Old Sakai / Velocity based tool
- Lots of inner classes
- Feature branch – Terracotta cluster enabled

Resources / Citations

- Provide a vanilla Sakai tool to demonstrate Terracotta clusterability
- Citation List creation can be a multi-page wizard style interaction
- Feature branch – Terracotta cluster enabled

Running a Terracotta-Enabled Sakai

Building

- Must compile and build the terracotta-config module
- Must run maven with a special flag

```
mvn -Dterracotta.enable=true clean install
```

Running

- Starting Terracotta Server
  - tc-config.xml file
  - Creating the TIM repository
- JAVA_OPTS for Sakai / Tomcat
  - -Dsakai.cluster.terracotta=true -Dtc.install-root=${TC_HOME} -Xbootclasspath/p:${TC_HOME}/lib/dso-boot/dso-boot-hotspot_linux_150_14.jar -Dtc.config=127.0.0.1:9510
  - tc-config.xml file

Administration

- Terracotta Administration Console
- Shutting down Sakai / Tomcat
- Shutting down Terracotta Server
- Clearing out the Terracotta Object "repository"

Terracotta-Cluster Enabling a Sakai Tool

Terracotta Enabling a Tool

- Add the tool to the clusterableTools property
- Create a new TIM (Terracotta Integration Module) for the tool
- Modify the terracotta-config module to know about the new TIM that was created

Creating a TIM

- Classes for instrumentation
  - Any class that will be stored in Terracotta
  - Any class that will modify a class stored in Terracotta
- Determine transient fields for any classes that will be stored in Terracotta
- Create a mechanism for resolving transient fields
  - Beanshell
  - Java method
- Promote inner classes to top level classes

Code sample

https://source.sakaiproject.org/svn/msub/unicon.net/content/branches/session-clustering-2-5-x/content-tim/src/main/resources/terracotta.xml

```
<?xml version="1.0" encoding="UTF-8" ?>
<xml-fragment>
      <instrumented-classes>
       <include>
            <class-
expression>org.sakaiproject.content.types.FolderType$*</c
lass-expression>
          <honor-transient>true</honor-transient>
       </include>
       <include>
...
```

Gotchas

- Direct field access
  - Prefer setters
- Inner classes
  - If the outer class is not going to be stored in Terracotta, the inner class must be static, however this does not prevent the outer class from direct field access (see above)
- Additional Java classes
- Reference classes from another component / service

Demo

Resources

Resources (1)

- *Introduction to Terracotta*
  - Monday, 11:30 am
- Websites
  - Open Source http://www.terracotta.org/
  - Commercial http://www.terracottatech.com/
- Website Resources
  - Downloads
  - Documentation
  - Forums
- Book
  - *The Definitive Guide to Terracotta*, ISBN-13: 978-1590599860

Resources (2)

- Sakai Confluence Terracotta link

http://confluence.sakaiproject.org/confluence/display/TERRA/Home

- Feature Branch subversion link

https://source.sakaiproject.org/svn/msub/unicon.net/distros/branches/session-clustering-2-5-›

Questions & Answers

Cris J. Holdorph
Software Architect
Unicon, Inc.

holdorph@unicon.net
www.unicon.net