



Anthony Gelibert – LIG Grenoble

OSGi and Terracotta: State replication of clustered services

Agenda

- Context
- Terracotta
- OSGi and Terracotta
- Future work

Agenda

● Context

- Terracotta
- OSGi and Terracotta
- Future work

About

- Anthony Gelibert
 - *MSc at Master of Science in Informatics at Grenoble (2010)*
 - *Engineer Degree at École Nationale Supérieure d'Informatique et de Mathématiques Appliquées de Grenoble (2010)*

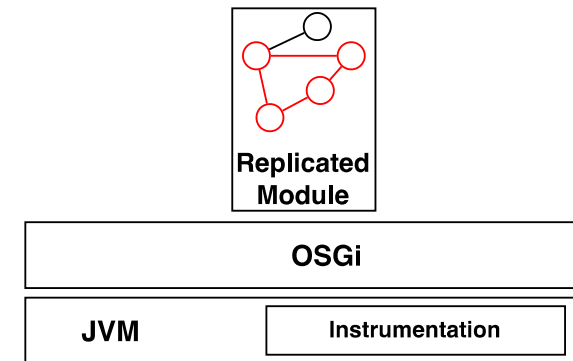
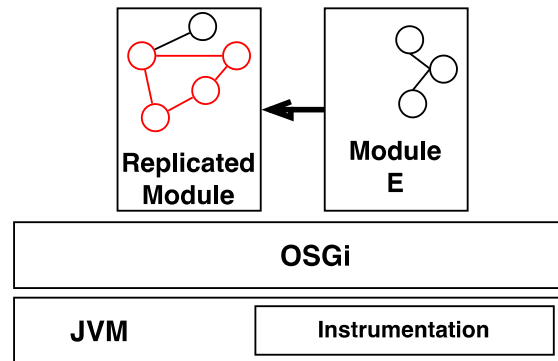
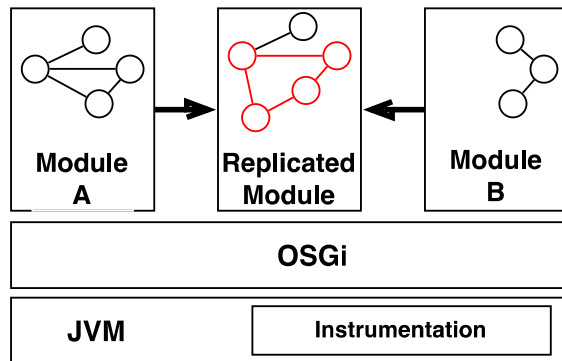
OSGi and distribution

- Distributed execution:
 - R-OSGi (Research)
 - OSGi Remote Services (Specification)
- Distributed deployment:
 - OSGi4C (Research)
- **Shared Memory : our approach**

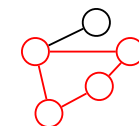
OSGi and distribution (cont.)

- R-OSGi:
 - Transparently invoke distant services
 - RMI-like
- OSGi4C:
 - Distributed bundle deployment
 - Runs locally
- Terracotta:
 - Shared memory

Objective



Distributed Shared
Memory
- Terracotta Server -



Graph of classes
Red = replicated

Use cases

- Traditional targets of distributed shared memory:
 - State share, Message bus, ...
- Problems requiring the use of distributed methods.
- **Re-engineering of existing solutions.**

Agenda

- Context

• Terracotta

- OSGi and Terracotta
- Future work

Terracotta: N.A.M.

- Network-Attached Memory
- No API
- No Serialization
- Cross-JVM coordination
- Distributed Method Invocations
- Runtime monitoring and control

Terracotta: configuration

- No API: declarative configuration

```
<?xml version="1.0" encoding="UTF-8"?>
<tc:tc-config xmlns:tc="http://www.terracotta.org/config"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.terracotta.org/schema/terracotta-5.xsd">
  <servers>
    <server host="%i" name="sample" />
  </servers>
  <system>
    <configuration-model>development</configuration-model>
  </system>
  <clients>
    <logs>terracotta/client-logs/pojo/chatter/%D</logs>
  </clients>
  <application>
    <dso>
      <roots>
        <root>
          <field-name>org.ow2.chameleon.tc.container.annotation.test.RootTestAlt.m_dummy</field-name>
        </root>
      </roots>
      <locks>
        <autolock>
          <method-expression>* org.ow2.chameleon.tc.container.annotation.test.RootTestAlt.*(.)</method-expression>
          <lock-level>write</lock-level>
        </autolock>
      </locks>
      <instrumented-classes>
        <include>
          <class-expression>org.ow2.chameleon.tc.container.annotation.test.RootTestAlt</class-expression>
          <honor-transient>>true</honor-transient>
        </include>
      </instrumented-classes>
      <distributed-methods>
        <method-expression>void org.ow2.chameleon.tc.container.annotation.test.RootTestAlt.testSetDummy(..)</method-expression>
      </distributed-methods>
      <injected-instances>
        <injected-field>
          <field-name>org.ow2.chameleon.tc.container.annotation.test.RootTestAlt.m_dummy</field-name>
        </injected-field>
      </injected-instances>
    </dso>
  </application>
</tc:tc-config>
```

Teracotta: no serialization

- Plain POJO clustering
- Dynamic instrumentation through a Java Agent
- Fine-grained replication

Terracotta: D.M.I

- Distributed Method Invocations.
- When a node calls a method, all nodes replicate it (locally).
- Development close to MPI:
 - The same code is executed at the same time on each node.
 - Requires distinguishing each element.

Agenda

- Context
- Terracotta

•OSGi and Terracotta

- Future work

OSGi and Terracotta: Toolchain

- Delegate class loading from OSGi to Terracotta.
- A set of Java 5 Annotations.
- An APT processor to generate the Terracotta config file.

Toolchain (cont.)

```
/* This static bloc permits to use Terracotta in Felix. */  
static  
{  
    final NamedClassLoader ncl = (NamedClassLoader) MyEventHandler.class.getClassLoader();  
    ncl.__tc_setClassLoaderName(MyEventHandler.class.getCanonicalName());  
    ClassProcessorHelper.registerGlobalLoader(ncl);  
}
```

Enabling the delegation of class loading



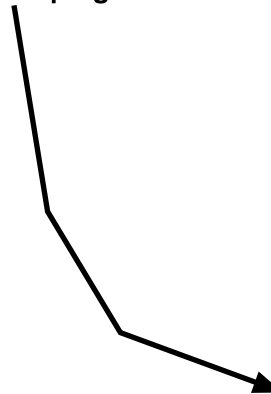
```
/* Annotations for the Terracotta container. */  
@ClientsConfiguration(logs = "terracotta/%D")  
@ServerConfiguration(host = "%i", name = "sample")  
@InstrumentedClass  
@AutoLock  
public class MyEventHandler implements EventHandler, LocalEventSenderInterface  
@InjectedField(fieldClass = MyEventHandler.class)  
private final DsoCluster m_cluster = null;  
  
@SharedField(fieldClass = MyEventHandler.class)  
private final DistributedPart m_sender = new DistributedPart();
```

Annotations for Terracotta

Toolchain (cont.)

```
[INFO] -----  
[INFO] Building OW2 :: Chameleon :: TerraCotta :: EventAdmin Bridge 1.0.0  
[INFO] -----  
[INFO] --- apt-maven-plugin:1.0-alpha-3:process (default) @ tc.eabridge ---  
[INFO] Processing 8 source files to /Users/tony/IDEA/Chameleon/sandboxes/tony3107/TC.EABridge/target/generated-sources/apt
```

APT Maven plugin



```
<?xml version="1.0" encoding="UTF-8"?>  
<tc:tc-config xmlns:tc="http://www.terracotta.org/config"  
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
  xsi:schemaLocation="http://www.terracotta.org/schema/terracotta-5.xsd">  
  <servers>  
    <server host="%i" name="sample" />  
  </servers>  
  <clients>  
    <logs>terracotta/%D</logs>  
  </clients>  
  <application>  
    <dso>  
      <roots>  
        <root>  
          <field-name>org.ow2.chameleon.tc.eabridge.impl.MyEventHandler.m_sender</field-name>  
        </root>  
      </roots>  
      <locks>  
        <auto lock>  
          <method-expression>* org.ow2.chameleon.tc.eabridge.impl.MyEventHandler.*(..)</method-expression>  
          <lock-level>write</lock-level>  
        </auto lock>  
      </locks>  
      <instrumented-classes>  
        <include>  
          <class-expression>org.ow2.chameleon.tc.eabridge.impl.DistributedPart</class-expression>  
          <honor-transient>true</honor-transient>  
        </include>  
        <include>  
          <class-expression>org.ow2.chameleon.tc.eabridge.impl.DistributedPart$NullEventSender</class-expression>  
        </include>  
        <include>  
          <class-expression>org.ow2.chameleon.tc.eabridge.impl.OSGiEncapsulatedEvent</class-expression>  
        </include>  
        <include>  
          <class-expression>org.ow2.chameleon.tc.eabridge.impl.MyEventHandler</class-expression>  
        </include>  
        <include>  
          <class-expression>org.ow2.chameleon.tc.eabridge.impl.OSGiEventFilter</class-expression>  
        </include>  
      </instrumented-classes>  
      <distributed-methods>  
        <method-expression>void org.ow2.chameleon.tc.eabridge.impl.DistributedPart.sendEvent(..)</method-expression>  
      </distributed-methods>  
      <injected-instances>  
        <injected-field>  
          <field-name>org.ow2.chameleon.tc.eabridge.impl.MyEventHandler.m_cluster</field-name>  
        </injected-field>  
      </injected-instances>  
    </dso>  
  </application>  
</tc:tc-config>
```

Terracotta Configuration File

Validation

- EventAdmin: propagation of events across multiple platforms.
- Cilia: replication/persistence in pervasive applications.
- H-Omega: Replacement of message passing by a shared memory.
- uGASP: State share for a DTN application

Agenda

- Context
- Terracotta
- OSGi and Terracotta

• Future work

Future work

- Extension of the container:
 - Enhancement Terracotta integration
 - Add persistence, cache, ...
- Dynamically update clustered services at run-time

For more information

- **Terracotta**

Documentation, download, ...

<http://www.terracotta.org>

Open Source

<http://www.terracotta.org/open-source>

- **Felix**

<http://felix.apache.org>

- **Chameleon**

<http://wiki.chameleon.ow2.org>

THANK YOU FOR YOUR ATTENTION!

Contact: anthony.gelibert@me.com