Session Clustering with Terracotta

Session Clustering with Terracotta

Jetty has been integrated with Terracotta, providing a great clustering solution.
Since Jetty 6.1.12, the Jetty-Terracotta integration has been rewritten to provide better performance.
The Jetty-Terracotta integration is not bundled by default; it must be built from sources following the instructions below.

Requirements

1. You need J2SE 1.5 or greater to build and run the Jetty-Terracotta integration.
2. You need Jetty 6.1.12 or greater.
3. You need to install Terracotta, we have tested Terracotta 2.6.4 or greater. We will refer to the installation directory as `$TC_HOME`.
4. You need to download this Jetty TIM (Terracotta Integration Module).
   We are working on an updated version of the official Jetty TIM and update the instructions when the new version of the official Jetty TIM is compatible with the current Jetty-Terracotta integration. Don't be fooled by the '1.1.1' version on the official TIM, you really need this 1.0.4. The official Jetty TIM will be downloadable from the Terracotta Forge.
   Copy the downloaded Jetty TIM file `tim-jetty-6.1-1.0.4.jar` to `$TC_HOME/modules/`.

Build instructions

1. Download and unzip a source bundle of Jetty 6.1.12 or greater from here. We will refer to the installation directory as `$JETTY_HOME`.
2. Build it:

```
$ cd $JETTY_HOME
$ mvn clean install
```

3. Copy the target/jetty-terracotta-sessions-6.1.12.jar you just built into `$JETTY_HOME>/lib/ext`:

```
$ cp contrib/terracotta/target/jetty-terracotta-sessions-6.1.12.jar
$JETTY_HOME>/lib/ext/
```

Jetty Configuration Files

Configuring Jetty to use Terracotta consists of creating a single TerracottaSessionIdManager per Jetty instance to generate unique session ids, and then setting up a special TerracottaSessionManager per each webapp that you want to be clustered.

The TerracottaSessionIdManager

One TerracottaSessionIdManager is configured per Jetty instance to generate unique session ids. These are the relevant lines to add to add to a separate `$JETTY_HOME/etc/jetty-terracotta.xml`:

```
<Configure id="Server"
class="org.mortbay.jetty.Server">
    <New id="tcIdManager"
class="org.mortbay.terracotta.servlet.Terrac
ottaSessionIdManager">
        <Arg>
            <Ref id="Server" />
        </Arg>
        <Set name="workerName">
            <SystemProperty
name="jetty.node" default="node1" />
        </Set>
    </New>
    <Call name="setAttribute">
        <Arg>tcIdManager</Arg>
        <Arg>
            <Ref id="tcIdManager" />
        </Arg>
    </Call>
</Configure>
```

The TerracottaSessionIdManager is stored as an attribute on the Server instance for later retrieval under the name `tcIdManager`.
The `workerName` is a unique name for the Jetty node. In the example above it is "node1" but you can use any naming scheme you'd like. This is useful when hardware components such as load balancers can "stick" the requests to the same node to improve performances by limiting the session migrations among nodes.

The TerracottaSessionManager

Each web application whose sessions you want to cluster must use a TerracottaSessionManager instead of the default HashSessionManager.
The easiest way to do this is to create individual context deployer config files for each web application, and include these lines:

```
<Configure
class="org.mortbay.jetty.webapp.WebAppContex
t">
    ...
    <Property name="Server" id="Server">
        <Call id="tcIdManager"
name="getAttribute">
            <Arg>tcIdManager</Arg>
        </Call>
    </Property>
    <Set name="sessionHandler">
        <New
class="org.mortbay.terracotta.servlet.Terrac
ottaSessionHandler">
            <Arg>
                <New
class="org.mortbay.terracotta.servlet.Terrac
ottaSessionManager">
                    <Set name="idManager">
                        <Ref
id="tcIdManager" />
                    </Set>
                </New>
            </Arg>
        </New>
    </Set>
    ...
</Configure>
```

These lines ensure that a TerracottaSessionManager is established for each web application, and has access to the

Jetty instance's unique TerracottaSessionIdManager we configured above.

Terracotta Configuration File

You need one Terracotta configuration file for each Jetty instance.
In the Terracotta configuration file, part of the configuration is needed to setup correctly the Jetty-Terracotta integration, and the rest of the configuration is needed to setup the Terracotta server and the web applications.

The base Terracotta Configuration File

The base Terracotta configuration file that sets up the Jetty-Terracotta integration is the following:

```
<?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.or
g/config
http://www.terracotta.org/schema/terracotta-
4.xsd">
    <servers>
        <server host="localhost">

<data>%(user.dir)/terracotta/server-data</da
ta>

<logs>%(user.dir)/terracotta/server-logs</lo
gs>
        </server>
    </servers>

    <clients>
```

```
        <modules>
            <module name="tim-jetty-6.1"
version="1.0.1" />
        </modules>

<logs>%(user.dir)/terracotta/client-logs</lo
gs>
    </clients>

    <application>
        <dso>
            <instrumented-classes>
                <include>

<class-expression>...</class-expression>
                </include>
            </instrumented-classes>
        </dso>
    </application>
</tc:tc-config>
```

You can create a Terracotta configuration file wherever you prefer in the file system. The file will be referenced by path in a system property to be passed on the command line (see below).
Copy and paste the example file content above into, for example, `$JETTY_HOME/tc-config.xml`.

Modifications to the Terracotta Configuration File

There are few places that needs to be modified in order for the Terracotta configuration file to correctly cluster your web application.

1. Modify the location of the Terracotta server.
   The server element has the `host` attribute that needs to be modified to point to the host name or host address of the Terracotta server. In most simple configurations this can be the local host, but in more advanced configurations the Terracotta server is deployed in a separate host.
2. Instrumented classes includes
   If you web application puts in the HTTP session only instances of JDK available classes (such as `java.lang.Integer` or `java.lang.String`), then you do not need the instrumented-classes element.
   In case your web application puts in the HTTP session instances of classes belonging to the web application (such as a domain `com.acme.domain.User` class), then you need to specify also those classes as instrumented, for example:

```
<tc:tc-config ...>
    ...
    <application>
        <dso>
            <instrumented-classes>
                <include>

<class-expression>com.acme.domain.User</class-expression>
                </include>
            </instrumented-classes>
        </dso>
    </application>
    ...
</tc:tc-config>
```

   You can also specify class expressions to match multiple classes, as specified here.
3. Web application contexts
   You do not need to specify the context path at which the web application is configured, by defining the `web-application` elements in the Terracotta configuration file, since the intention of clustering a context is already specified in the Jetty context configuration file.

Starting Terracotta and Jetty

The final steps require to start the Terracotta server and the Jetty servers.

1. Start the Terracotta server (in one console)

   ```
   $ cd $TC_HOME/bin/
   $ ./start-tc-server.sh
   ```

2. Start the Jetty instance (in another console)

   ```
   $ cd $JETTY_HOME/
   $ $TC_HOME/bin/dso-java.sh -Dtc.config=tc-config.xml -jar start.jar
   etc/jetty.xml etc/jetty-terracotta.xml
   ```

   The command line above assumes that the Terracotta configuration file has been saved to `$JETTY_HOME/tc-config.xml`, and that the Jetty configuration file `jetty-terracotta.xml` is in its usual location under `$JETTY_HOME/etc/`.
   Repeat this step for all Jetty nodes and you are done.

Do not forget to change the `workerName` of the TerracottaSessionIdManager in each node you deploy (this is to help other hardware devices such as load balancers).

You can inspect that the clustering is working by starting the Terracotta administration console:

```
$ cd $TC_HOME/bin
$ ./admin.sh
```

You should see one root named "sessionIds" and 2 roots for each web application named "sessionData:<context>:<vhost>" and "sessionExpirations:<context>:<vhost>".