Distributed XNAT

The text here is mostly taken from an email chain between Rick, Dan, and Tim.

Load balancing for multiple XNAT servers would be configured using Apache HTTP 2.2's **mod_proxy** and **mod_proxy_balancer**modules. This provides the quickest implementation for load balancing by supporting back-end stickiness, which obviates the requirement for back-end session pooling. Using weighted traffic counting as the balancing criteria will allow both for heterogeneous hardware environments and for unbalanced client requests.

The caching mechanism methods within XNAT will be refactored to handle consistent session state across multiple XNAT instances on separate tomcat servers. Existing caching structures will be moved to use a common externalizable API, Ehcache.

The caching mechanisms, when refactored from XNAT's existing home-grown solution, will let us use more advanced load balancing strategies as well. Back-end stickiness doesn't really require the caching changes, since each user session is tied to a particular Tomcat instance, so you don't have to worry about the session and state pooling problems. The problem is that, if two users are assigned to the same Tomcat and one starts to increase the load on the server, the load balancer won't dynamically shift the second user to another Tomcat server to balance that load. You can only move users in subsequent sessions. So it'll be better to have the pooled session and state caching overall, because then you're doing *real* dynamic load balancing.

The complication raised by caching is when servers are on physically separate infrastructure, e.g. servers within network firewalls and on EC2 or perhaps in different institutions in the case of a multi-site study. This indicates support for a number of advanced features as well:

- Distributed Caching Using Ehcache on Amazon EC2
- Creating Terracotta Server Arrays with EC2 CloudFormation for use by Ehcache
- Distributed cache pools that can be accessible from multiple locations. There are lots of security concerns here, especially around authentication and possible breech of PHI data.
- Geo-location or time-to-response selection of back-end servers.

Configuring Shared Sessions and Automatic failover

It is possible to configure XNAT to perform shared (sticky) HTTP sessions and load balancing across several tomcat servers. This can be accomplished by using the memcached-session-manager library, located here. The process involves downloading and installing several libraries into your tomcat installation and into the XNAT install libraries directory.

Required Libraries

First, download and install the following libraries into your tomcat/lib directory:

- memcached-session-manager-${version}.jar and either memcached-session-manager-tc6-${version}.jar for tomcat6 or memcached-session-manager-tc7-${version}.jar for tomcat7.
- In this example, we are going to use memcached, so we also need the spymemcached-2.8.12.jar and couchbase-client-1.1.4.jar libraries.

Now, since we are going to be using the kryo serializer (a very efficient binary serializer for java), download and install the following libraries into your xnat /WEB-INF/lib directory:

- kryo-serializer: msm-kryo-serializer, kryo-serializers-0.10, kryo, minlog, reflectasm, asm-3.2

Install memcached

Install Memcached, the high-performance shared caching solution. Installation methods vary by Unix flavor, but for ubuntu, you can install it using the apt-get method. As root, run the command:

```
$ apt-get install memcached
```

Or if you're using CentOS, use:

```
$ yum install memcached
```

These commands should install and then launch memcached on the current machine, with a default port configuration of 11211. No matter how many tomcats you intend to use, you only need memcached installed on one node.

Tomcat Configuration changes

Now that we have memcached installed and all of the important libraries installed in their proper places, we are ready to modify our tomcat installation. First, locate your tomcat installation directory. Common locations include /var/lib/tomcat7 or /usr/share/tomcat7. There should be a directory called "conf" in there. Inside that conf directory, you should see a directory called Catalina, and inside there, a directory called "localhost". Inside of the localhost directory, create a file called "xnat.xml" and make it look like the following:

```
<?xml version='1.0' encoding='utf-8'?>
<Context
 path="/xnat"
 docBase="/var/lib/tomcat7/webapps/xnat"
 workDir="work/Catalina/localhost/xnat"
 debug="0"
 reloadable="true">

<Logger className="org.apache.catalina.logger.FileLogger"
 prefix="localhost_bzzagent_log." suffix=".txt" timestamp="true"/>

<Manager className="de.javakaffee.web.msm.MemcachedBackupSessionManager"
 memcachedNodes="n1:localhost:11211,n2:localhost:11212"
 failoverNodes="n2"
 requestUriIgnorePattern=".*\.(ico|png|gif|jpg|css|js)$"
 transcoderFactoryClass="de.javakaffee.web.msm.JavaSerializationTranscoderFactory"/>
</Context>
```

NOTE- We originally tried to use the Kryo transcoder factory for its superior performance, but it kept running into problems trying to serialize a ConncurrentHashMap. Please use the JavaSerializationTranscoderFactory instead.

The important parts here are the docBase setting, it should point to wherever your XNAT webapp gets deployed to. Workdir can generally be left as is. The important part we are adding is the Manager section, which defines MemcachedBackupSessionManager as our tomcat's session manager. In the memcachedNodes setting, place the server settings for memcached nodes in the format "node_identifier:hostname:port". In this example, we only have one node, but if you have multiple memcached nodes (which generally is a good idea, in case one of them crashes), you can list them here separated by commas. For example, lets say we had two memcacheds, we could use: "memcachedNodes="n1:localhost:11211, n2:otherserver:11211, n3:otherserver:11211". The "failoverNodes" configuration parameter is optional but can be useful for specifying a backup server, for even more robustness. **IMPORTANT: this configuration needs to be identical for every tomcat in your cluster.**

Apache Configuration changes

To make use of our multiple tomcats, we can use Apache2 to act as a load-balancer to balance the user sessions across several tomcats. For more information, see this link: http://tomcat.apache.org/connectors-doc/generic_howto/loadbalancers.html To enable load balancing on your configuration, install Apache2 on your machine, and then enable the following modules: proxy, proxy_http, proxy_balancer and headers. Then, add the following to your apache2.conf file:

```
Header add Set-Cookie "ROUTEID=.%{BALANCER_WORKER_ROUTE}e; path=/" env=BALANCER_ROUTE_CHANGED
<Proxy balancer://mycluster>
BalancerMember http://host1:8080/xnat route=1
BalancerMember http://host2:8080/xnat route=2
ProxySet stickysession=ROUTEID
</Proxy>
ProxyPass /xnat balancer://mycluster
ProxyPassReverse /xnat balancer://mycluster

<Location /balancer-manager>
 SetHandler balancer-manager
</Location>

#ProxyPass /xnat http://localxnat:8080/xnat
#ProxyPassReverse /xnat http://localxnat:8080/xnat
```

This will tell Apache where to find your tomcats, and how it should behave when balancing sessions between them.

Helpful Links

- http://tomcat.apache.org/connectors-doc/generic_howto/loadbalancers.html
- http://httpd.apache.org/docs/2.2/mod/mod_proxy_balancer.html
- https://code.google.com/p/memcached-session-manager/wiki/SetupAndConfiguration