Clustering

Contents

Motivation

As more and more institutions adopt DSpace as a repository platform, it will become increasingly important for DSpace to be able to scale well. Exactly what it means for DSpace to "scale" is up for debate, and should be eventually defined here: ScalabilityMetrics.

Why Tomcat?

Because it's the only servlet container I understand 😉 If others offer better clustering capabilities, I'd gladly learn about them, but for now, I'm sticking with Tomcat.

Why PostgreSQL?

I have decided to initially only include clustering information for PostgreSQL for two reasons. First, one of the core principles of DSpace is that it should be possible to run it using purely open-source software, and even though Oracle may offer performance improvements over PostgreSQL, it isn't an open source product. Secondly, it looks like the database-layer clustering will be done through JDBC, and so the underlying database implementation ought to be irrelevant.

Tomcat

As of Tomcat 5, there is native support for clustering, load balancing, and session replication.
Here are directions for serializing DSpace (tested with DSpace 1.4, Tomcat 5.5.17 and Tomcat 6).

1. In the tomcat/server.xml add this line under Engine, for example:
<Engine defaultHost="localhost" name="Catalina">

<Cluster className="org.apache.catalina.cluster.tcp.SimpleTcpCluster"/>

2. We also need to modify the objects that DSpace sessions use, to make them serializable. DSpace's sessions only use one object (as of 1.4), RequestInfo.java. To make this class serializable, add the following code in RequestInfo.java:

java class RequestInfo implements java.io.Serializable{....

3. After this you have to recompile the DSpace code, stop Tomcat, copy the
war in /tomcat-directory/webapps/ and restart Tomcat.

Caveats

Any "known issues" with clustering Tomcat should be described in this section.

References

- http://tomcat.apache.org/tomcat-5.0-doc/cluster-howto.html

PostgreSQL

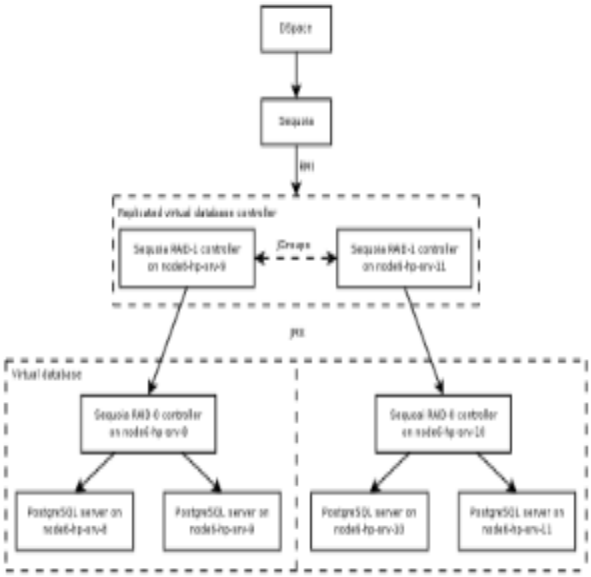
Slony-I

Slony-I is a "master to multiple slaves" replication system. Unfortunately, it lacks support for may key features I would consider necessary for clustering:

- No automatic failover or node promotion
- Trigger-based update propogation means that (eg) schema changes cannot be automatically propogated across nodes, and it is unable to replicate large objects
- It is unable to detect node failure
- No support for a multi-master replication topology (ie: there will always be a single point of failure)
- No support for connection brokering
  All of these things combined mean that I will not be looking into using Slony-I. If anyone has any positive experiences with using it, please update this section.

Sequoia

Sequoia (formerly C-JDBC) is a drop-in replacement for JDBC. No code changes are required for DSpace to use Sequoia in place of the current Postgres or Oracle JDBC drivers. It is released under an Apache license.
Sequoia claims to support lots of really useful features, such as RAIDb (think RAID for databases). A real "killer" feature (that, incidentally, could prove useful even without any clustering) is the support for a heterogeneous db environment by performing on-the-fly query translation. This means we could (in theory) drop support for anything other than postgres, and stick Sequoia in between DSpace and the database, and let it do the query translation for us. Naturally, this will introduce a performance penalty, but I'm not sure how great that will be. I suspect it would be comparable to using something like Hibernate, except that we wouldn't need to change any code to support Sequoia.



For testing, I am going to start by trying to construct a RAID 10-style replication architecture (see diagram). This basically means that the DSpace database will be distributed across two nodes, and those two nodes will be replicated on another two nodes. Once I have a working configuration, I will post configuration files, and write a step-by-step guide to re-creating this setup.

Update (2007-02-13): I think this configuration might actually be impossible. Though it is alluded to in the documentation, the sequoia driver complains when you try to connect it to another sequoia driver in a nested configuration. For this reason, I am currently looking into a RAIDb-2 setup which achieves many of the same characteristics as RAIDb-1+0.

Limitations of Sequoia:

- No full JDBC-2 or JDBC-3 support
- Failed nodes must be manually reactivated
- The extra network hop (driver -> controller -> database as opposed to driver -> database) introduces a performance penalty

HA_JDBC

HA-JDBC seems to be similar in scope to the Sequoia project. Extensive documentation is available on their website, and I will look into it further once I have finished my Sequoia testing.

Limitations of HA-JDBC:

- No result set caching
- Only supports a "mirror"-style topology (no striping)
- Failed node recovery is automatic and reliable, but inefficient

Caveats

Any "known issues" with clustering PostgreSQL should be described in this section.

References

- Sequoia: http://sequoia.continuent.org
- C-JDBC: http://c-jdbc.objectweb.org
- HA-JDBC: http://ha-jdbc.sourceforge.net
- Slony-I: http://slony1.projects.postgresql.org

Terracotta

Terracotta has just been released as an open source product. There are a few limitations, but this could allow for easier and faster Session replication than Tomcat's built in support. (For one thing, it doesn't require that every class is Serializable).

References

- http://www.terracotta.org/confluence/display/docs22/Terracotta+Sessions+Quick+Start+Guide