Performance Tuning

Balancing Cache w/ Heap Memory

Indexing Attributes

The default partition implementation is based on JDBM. By default it may already index some common attributes like objectClass and ou for example. However you really want to index attributes that your applications use frequently in canned queries.

Adding an index on an attribute is pretty simple. The configuration in the server.xml file needs to be altered before bulk loading data into the server. Otherwise your index will not work properly.

> ⚠ Indices must be configured before loading data into the server. Indices configured after loading entries into the server will NOT work properly unless they are built using the index builder command supplied with the ApacheDS tools command line program. More information on this in the Building Indices section below.

Indices are configured in the partition configuration section of your server.xml. Each partition will have it's own set of indexed attributes. These index configurations reside under the indexedAttributes property of the org.apache.directory.server.core.partition.impl.btree.MutableBTreePartitionConfiguration spring bean. Here's that section for the stock dc=example,dc=com partition that is configured out of the box with ApacheDS.

```
<property name="indexedAttributes">
      <set>
        <bean class="org.apache.directory.server.core.partition.impl.btree.MutableIndexConfiguration">
          <property name="attributeId"><value>1.2.6.1.4.1.18060.1.1.1.3.1</value></property>
          <property name="cacheSize"><value>100</value></property>
        </bean>
        <bean class="org.apache.directory.server.core.partition.impl.btree.MutableIndexConfiguration">
          <property name="attributeId"><value>1.2.6.1.4.1.18060.1.1.1.3.2</value></property>
          <property name="cacheSize"><value>100</value></property>
        </bean>
        <bean class="org.apache.directory.server.core.partition.impl.btree.MutableIndexConfiguration">
          <property name="attributeId"><value>1.2.6.1.4.1.18060.1.1.1.3.3</value></property>
          <property name="cacheSize"><value>100</value></property>
        </bean>
        <bean class="org.apache.directory.server.core.partition.impl.btree.MutableIndexConfiguration">
          <property name="attributeId"><value>1.2.6.1.4.1.18060.1.1.1.3.4</value></property>
          <property name="cacheSize"><value>100</value></property>
        </bean>
        <bean class="org.apache.directory.server.core.partition.impl.btree.MutableIndexConfiguration">
          <property name="attributeId"><value>1.2.6.1.4.1.18060.1.1.1.3.5</value></property>
          <property name="cacheSize"><value>10</value></property>
        </bean>
        <bean class="org.apache.directory.server.core.partition.impl.btree.MutableIndexConfiguration">
          <property name="attributeId"><value>1.2.6.1.4.1.18060.1.1.1.3.6</value></property>
          <property name="cacheSize"><value>10</value></property>
        </bean>
        <bean class="org.apache.directory.server.core.partition.impl.btree.MutableIndexConfiguration">
          <property name="attributeId"><value>1.2.6.1.4.1.18060.1.1.1.3.7</value></property>
          <property name="cacheSize"><value>10</value></property>
        </bean>

        <bean class="org.apache.directory.server.core.partition.impl.btree.MutableIndexConfiguration">
          <property name="attributeId"><value>dc</value></property>
          <property name="cacheSize"><value>100</value></property>
        </bean>
        <bean class="org.apache.directory.server.core.partition.impl.btree.MutableIndexConfiguration">
          <property name="attributeId"><value>ou</value></property>
          <property name="cacheSize"><value>100</value></property>
        </bean>
        <bean class="org.apache.directory.server.core.partition.impl.btree.MutableIndexConfiguration">
          <property name="attributeId"><value>krb5PrincipalName</value></property>
          <property name="cacheSize"><value>100</value></property>
        </bean>
        <bean class="org.apache.directory.server.core.partition.impl.btree.MutableIndexConfiguration">
          <property name="attributeId"><value>uid</value></property>
          <property name="cacheSize"><value>100</value></property>
        </bean>
        <bean class="org.apache.directory.server.core.partition.impl.btree.MutableIndexConfiguration">
          <property name="attributeId"><value>objectClass</value></property>
          <property name="cacheSize"><value>100</value></property>
        </bean>
      </set>
    </property>
```

As you can see indices are specified using a MutableIndexConfiguration spring bean. Just add one of these to your existing configuration setting the attributeId to the OID or name of the attribute you want to index. There is cacheSize parameter used to set the amount of cache on your index as well. Most of the time 100 will suffice no matter how big in capacity your server is.

> *This number (100) is the number of entries stored in the cache, regardless to their size. Be carefull when dealing with huge entries - those which contains jpeg images -*

So if I wanted to index the attribute **initials** all I have to do is append the following xml fragment to this set of indexed attributes:

```
<bean class="org.apache.directory.server.core.partition.impl.btree.MutableIndexConfiguration">
          <property name="attributeId"><value>initials</value></property>
          <property name="cacheSize"><value>100</value></property>
        </bean>
```

That's it. Now queries on initials alone should perform about 50X faster.