Google Summer of Code 2009

Mentoring organization:
Fedora Project & JBoss.org (Red Hat)

Project:
Hibernate Search

Mentor:
Emmanuel Bernard

Student:
Lukasz Moren

Hibernate Search brings the power of full text search engines to the persistence domain model. It provides easy way to enable users to search through database data and find what they need. It works in clustered and non clustered environments. Hibernate Search uses clustering based on Java Messaging Service - JMS (to send changes, if works according to master/slave design pattern). The first part of my work during Google Summer of Code 2009 was to implement alternative for JMS solution, based on JGroups library. That allows to run Hibernate Search in master/slave mode outsider J2EE container and where JNDI service is not available e.g. Java standalone applications. The second part included implementation of custom Lucene directory, based on Infinispan data grid and applying it into Hibernate Search context.

1. JGroups backend for Hibernate Search

JGroups approach is similar to JMS. Application working in the cluster contains number of nodes – slaves which not change the Lucene index but send changes to node configured as a master. Master node is responsible for receiving messages from other slaves and applying it into index.

To use JGroups backend the only necessary Hibernate configuration comes down to:

```
<property name="hibernate.search.worker.backend" value="jgroupsMaster" />

or

<property name="hibernate.search.worker.backend" value="jgroupsSlave" />
```

depends if it is master or slave node.

Optional configuration properties:

- optional name for JGroups cluster

```
<property name="hibernate.search.worker.backend.jgroups.clusterName"
value="HSCluster" />
```

- JGroups network stack configuration

  - configuration provided in configuration file

```
<property name="hibernate.search.worker.backend.jgroups.configurationFile"
value="udp.xml" />
```

udp.xml file must be located in classpath

  - configuration provided in XML

```
<property name="hibernate.search.worker.backend.jgroups.configurationXml"
value="{configurationInXML}" />
```

where {configurationInXML} is JGroups configuration in XML format e.g. for UDP protocol:

```
<config xmlns="urn:org:jgroups"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="urn:org:jgroups file:schema/JGroups-2.8.xsd">
   <UDP
         mcast_addr="${jgroups.udp.mcast_addr:228.10.10.10}"
         mcast_port="${jgroups.udp.mcast_port:45588}"
         tos="8"
         thread_naming_pattern="pl"
```

```
         thread_pool.enabled="true"
         thread_pool.min_threads="2"
         thread_pool.max_threads="8"
         thread_pool.keep_alive_time="5000"
         thread_pool.queue_enabled="false"
         thread_pool.queue_max_size="100"
         thread_pool.rejection_policy="Run"/>

   <PING timeout="1000" num_initial_members="3"/>
   <MERGE2 max_interval="30000" min_interval="10000"/>
   <FD_SOCK/>
   <FD timeout="3000" max_tries="3"/>
   <VERIFY_SUSPECT timeout="1500"/>

   <pbcast.STREAMING_STATE_TRANSFER/>
   <!-- <pbcast.STATE_TRANSFER/> -->
   <pbcast.FLUSH timeout="0"/>
</config>
```

- configuration provided in plain text

```
<property
name="hibernate.search.worker.backend.jgroups.configurationString"
value="{stringConfiguration}" />
```

where {stringConfiguration} is JGroups configuration provided in string format e.g.

```
UDP(mcast_addr=228.1.2.3;mcast_port=45566;ip_ttl=32):PING(timeout=3000;
num_initial_members=6):FD(timeout=5000):VERIFY_SUSPECT(timeout=1500):
pbcast.NAKACK(gc_lag=10;retransmit_timeout=3000):UNICAST(timeout=5000):FRAG
:pbcast.GMS(join_timeout=3000;shun=false;print_local_addr=true)
```

If there is no JGroups configuration provided, flush-udp.xml is default. That and some other stack configurations are part of JGroups.jar which is in Maven dependencies of Hibernate Search

2. Infinispan based Lucene directory

Extension of Lucene Directory that allows to store indexes in Infinispan data grid. New *InfinispanDirectoryProvider* applies it into Hibernate Search.

To select Infinispan grid as index storage to Hibernate configuration file should be added:

```
<property name="hibernate.search.default.directory_provider"
value="org.hibernate.search.store.infinispan.InfinispanDirectoryProvider"/>
```

with this setting all indexes will be stored in Infinispan data grid.
However storage place can be scoped per indexed entity:

```
<property name="hibernate.search.Movie.directory_provider"
value="org.hibernate.search.store.infinispan.InfinispanDirectoryProvider"/>

<property name="hibernate.search.default.directory_provider"
value="org.hibernate.search.store.FSDirectoryProvider"/>
```

With that configuration only *Movie's* indexes will be stored in Infinispan grid.

Infinispan data grid is highly customizable. Here are examples how to provide custom configuration:

```
<property name="hibernate.search.default.infinispan.cache_config_xml"
                      value="ispn-cache-default-conf.xml"/>
```

or

```
<property name="hibernate.search.default.infinispan.cache_config_class"
value="org.hibernate.search.store.infinispan.InfinispanCacheManagerConfigur
ationImpl"/>
```

Notice that all configuration classes passed as value for parameter cache_config_class have to implement *org.hibernate.search.store.infinispan.InfinispanCacheManagerConfiguration*

Implementation of Infinispan Lucene directory is flexible. It is possible to have one shared directory between all indexes, one directory for each index, and provide different configuration to each of them.

With below configuration indexes for *Movies* will be stored in grid called *It's a movie cache.* Another use default grid.

```
<property name="hibernate.search.Movie.infinispan.cache_name"
                      value="It's a movie cache"/>
```

Configuration for cache named *It's a movie cache* have to be defined in configuration file (either *cache_config_xml* or *cache_config_class*).

For better performance data stored in Infinispan grid is splitted into parts. Size of each part can be configured ant it is also scoped per index:

```
<property name="hibernate.search.Movie.infinispan.buffer_size"
                      value="512"/>
```