Hibernate Search - Shard Query Optimization 12 Jun 2009

Hibernate Search shards allow you to break down your index data into separate Lucene directories. Typically, indexes would be broken down either into N equals chunks (using a hashing algorithm), or by some logical criteria (customer, location, etc). The former was done for performance; smaller indexes mean faster indexing. The later was typically done to make customers feel better.

In my current project, we have another reason. We're breaking down indexes by customer, but for purely technical reasons. Separate indexes are more robust; if you have a fatal corruption of an index, now you only have to re-index a fraction of your data. The other is speed. Since there is no reason for searches to be cross-customer, why not take advantage of smaller indexes for query performance?

Unfortunately, Hibernate Search defaults to searching ALL the shards, and then merging the result sets. While some of this can be done in parallel, in the end the search is much slower than before. While this strategy is necessary in the hashing case, it's needlessly wasteful in the customer case.

Granted, the customer case was definitely not the initial shard use case. But it did get enough demand to warrant a new JIRA issue, HSEARCH-251.

I actually got to work with the Hibernate Search maintainers to provide this functionality.

```
http://anonsvn.jboss.org/repos/hibernate/search/trunk
Revision: 16755
Author: epbernard
Date: 8:43:09 PM, Wednesday, June 10, 2009
```

```
Message:
HSEARCH-251 Query on a shard subset based on a filter activation
----
Modified : /search/trunk/src/main/docbook/en-US/modules/configuration.xml
Modified : /search/trunk/src/main/docbook/en-US/modules/query.xml
Modified : /search/trunk/src/main/java/org/hibernate/search/filter/ChainedFilte
r.java
Added : /search/trunk/src/main/java/org/hibernate/search/filter/FullTextFilterI
mplementor.java
Added : /search/trunk/src/main/java/org/hibernate/search/filter/ShardSensitiveO
nlyFilter.java
Modified : /search/trunk/src/main/java/org/hibernate/search/query/FullTextFilte
rImpl.java
Modified : /search/trunk/src/main/java/org/hibernate/search/query/FullTextQuery
Impl.java
Modified : /search/trunk/src/main/java/org/hibernate/search/store/IdHashShardin
gStrategy.java
Modified : /search/trunk/src/main/java/org/hibernate/search/store/IndexSharding
Strategy.java
Modified : /search/trunk/src/main/java/org/hibernate/search/store/NotShardedStr
ategy.java
Modified : /search/trunk/src/test/java/org/hibernate/search/test/configuration/
UselessShardingStrategy.java
Added : /search/trunk/src/test/java/org/hibernate/search/test/shards/CustomerSh
ardingStrategy.java
Added : /search/trunk/src/test/java/org/hibernate/search/test/shards/CustomerSh
ardingStrategyTest.java
```

You can download the latest build now, and give it a shot. Here is an example of how to use the new feature.

Here is your entity, with the filter defined.

```
view plain
```

```
@Indexed(index="Email")
// this "impl" is only a flag, not the actual filter class
@FullTextFilterDef(name="shard", impl=ShardSensitiveOnlyFilter.class)
public class Email {
...
```

Here is the filter.

```
view plain
```

```
public class ShardFilter {

    private Integer index;

    public void setIndex(Integer setIndex) {
        this.index = setIndex;
    }

    @Key
    public FilterKey getKey() {
        StandardFilterKey key = new StandardFilterKey();
        key.addParameter(index);
        return key;
    }

    @Factory
    public Filter getFilter() {
  Query query = new TermQuery(new Term("index", index.toString()));
        return new CachingWrapperFilter(new QueryWrapperFilter(query));
    }

}
```

Here is your indexing strategy, which implements the new method getDirectoryProvidersForQuer(). From here, you can define which shards a given Filter could possibly return data from.

```
view plain

public class SpecificShardingStrategy extends IdHashShardingStrategy {

 @Override
 public DirectoryProvider<?>
[] getDirectoryProvidersForQuery(FullTextFilterImplementor[] filters) {

  FullTextFilter filter = getFilter(filters, "shard");
  if (filter == null) {
   return getDirectoryProvidersForAllShards();
  }
  else {
   return new DirectoryProvider[] { getDirectoryProvidersForAllShards()
[Integer.parseInt(filter.getParameter("index").toString())] };
  }
 }

 private FullTextFilter getFilter(FullTextFilterImplementor[] filters, String name) {
  for (FullTextFilterImplementor filter: filters) {
   if (filter.getName().equals(name)) return filter;
  }
  return null;
 }

}
```

Finally, here is the actual search code.

```
view plain

FullTextSession fts = Search.getFullTextSession( s );
QueryParser parser = new QueryParser("id", new StopAnalyzer() );
FullTextQuery fullTextQuery = fts.createFullTextQuery( parser.parse( "body:message" ) );
fullTextQuery.enableFullTextFilter("shard").setParameter("index", 0);
```

Of course, there are many more ways to shard the cat. For example, the filter could be on customerID, region, etc. Thanks to the Hibernate Search team for incorporating my code!

Follow @chase_seibert on Twitter

AROUND THE WEB WHAT'S THIS?

WURA
Huge Celebrity Implants You've Got To See

Naturalon
16 Cancer Causing Foods

Moneynews
The $152k Social Security Mistake That 70% of Seniors Make

BeenVerified
Stop 'Googling' Names. New Site's Users Find Much More!

ALSO ON CHASE SEIBERT | BLOG

Development on a Mac versus Linux
1 comment

Joining a software startup right out of school 1 comment

Python script to delete merged git branches
2 comments

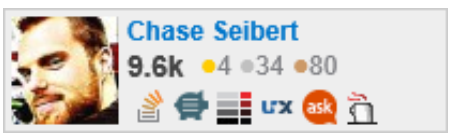
1 Comment Chase Seibert | blog Login

Sort by Best Share Favorite

Join the discussion

Chase Seibert

Facts, hacks and attacks from my life as a web application developer



Tags

- hibernate
- lucene
- hibernate-search

Related Posts

- Thoughts on Object Oriented Class Design
- Creating a Budget and Sticking to It
- Sync dotfiles with GitHub

Social Links

- LinkedIn
- Facebook
- Twitter
- Resume