LotsOfCores

<!> Solr4.2

⚠️ NOT verified in SolrCloud! This code has not been tested in SolrCloud setups! ⚠️

- Overview
- Configuration
  - old-style
  - new-style (core discovery)
  - From the original discussion
- Further work
- Issues for reference, the work is done

Overview

Considerable work has been done to move Solr towards being suitable for a large number of homogeneous cores where you require fast/frequent loading /unloading of cores. This page describes the current state of the code as of Solr 4.4/5.0

The requirements of such a system are:

1. Very efficient loading of cores - Solr would be more efficient if it did not have to read and parse and create Schema, SolrConfig objects for each core every time the core has to be loaded (deferred).
2. Lazy load cores - Provide a way to START/STOP core.
3. Automatic loading of cores - Start a core automatically if a request comes in for a "stopped" core.
4. LRU Core Loading/Unloading - As there are a large number of cores, all the cores cannot be kept loaded always. There has to be an upper limit beyond which we need to unload a few cores.
5. Allowing a cores to be defined in a tree structure - If the number of cores is too high, all the cores' dataDirs cannot live in the same directory. There is an upper limit on the number of directories you can create in a directory w/o affecting performance.

Configuration

We are going to "core discovery" mode for defining cores, see the page Solr.xml 4.4 and beyond. The basic idea is that we're removing the <cores> and individual <core> tags from solr.xml, those tags will be unsupported as of 5.0. Instead, we'll start from <solr_home>, walk the directory tree looking for "core.properties" files which will define the location of each core. The parameters that define how this feature works will therefore be in different places depending on the mode.

There are two new attributes of a core (defaults in bold) and one new attribute for controlling how many transient cores are loaded at once.

- transientCacheSize=[NNN]. If this limit is crossed, old cores marked 'transient="true"' are removed to make room on an LRU basis. Default size **Integer.MAX_VALUE**. *Only* cores with "transient=true" are put in this cache, so specifying this attribute without having any cores marked as "transient" has no effect except wasting a bit of memory.
  - Having this size be less than the number of cores marked 'transient="true"' AND 'loadOnStartup="true"' *should* work, but it's wasteful since a bunch of cores will be loaded on startup then immediately unloaded after the cache fills up.
  - NOTE: All transient core information is read at startup and the "list of cores" is unbounded.
- old-style: this is an attribute of the <cores> tag
  - new-style: this is a child node of the <solr> tag, and looks like <int name="transientCacheSize">12</int>
- loadOnStartup=["**true**"|"false"]. Whether the core should be completely loaded upon startup.
  - old-style: this is an attribute of each individual <core> tag
  - new-style: this is an entry in the "core.properties" file for each core.
- transient=["true"|"**false**"]. Whether the core should be put in the LRU list of cores that may be unloaded. NOTE: When a core is unloaded, any outstanding operations (indexing or query) will be completed before the core is closed.
  - old-style: this is an attribute of each individual <core> tag
  - new-style: this is an entry in the "core.properties" file for each core.

So the idea is that there's really no reason to tie in "lazy loading" with whether the core can be swapped out or not, so by splitting up the two options we give the user control over how these are handled. Use cases below:

- loadOnStartup=**true** transient=**false**: Current case. Spend all the time necessary to fully load the cores on startup.
- loadOnStartup=**true** transient=**true**: There are some cores you want loaded when the server first starts up, but that you'll allow to be swapped out. It's wasteful to specify more cores like this than your transientCacheSize value.
- loadOnStartup=**false** transient=**false**: You'd specify this combination if starting Solr up quickly was more important than the inconvenience of having to wait for cores to be loaded the first time a request was made. One could imagine starting Solr this way and having a background thread fire queries at each core to load it.
- loadOnStartup=**false** transient=**true**: There are a large number of cores in your system that are short-duration use. You want Solr to load them as necessary, but unload them when the cache gets full on an LRU basis.

old-style

The following configuration applies to the patch given in SOLR-1293.

```
<?xml version='1.0' encoding='UTF-8'?>
<solr persistent='true'>
  <cores adminPath="/admin/cores"
         transientCacheSize="4"
         adminHandler="org.apache.solr.handler.admin.LotsOfCoresAdminHandler"
         shareSchema="true">
    <core name="core0" instanceDir="/opt/solr" loadOnStartup="false" transient="true"/>
  </cores>
</solr>
```

new-style (core discovery)

```
<?xml version='1.0' encoding='UTF-8'?>
<solr persistent='true'>
  much omitted
  <int name="transientCacheSize">64</int>
</solr>
```

Then in an individual core.properties file

```
loadOnStartup=true|false
transient=true|false
```

From the original discussion

- START/STOP commands. This is unnecessary with the transient cache definitions. Besides, the current functionality (admittedly confusing) is that UNLOAD implements STOP and CREATE implements START.
- **shareSchema** - Ensures that only one instance of IndexSchema is created in the Solr. Hasn't been rigorously tested in the new configuration, and with named config sets the support will be automatic if supported.
- **shareConfig** - deferred, not currently supported.
- **cleanOnUnload** - Clean up (delete) the index when a core is unloaded. Not implemented yet, for my particular use-case it probably won't be. I can see the utility though. Doesn't seem very hard code-wise.

Hmmm, haven't thought about the various status commands very deeply. There is an update to the 'status' command. Adding a parameter 'verbose=false' will return a minimal status report of the cores. The default status command uses Luke on the core's index to get very detailed information which is expensive if the status is queried very frequently.

Further work

- Alias/Unalias commands are not fully tested currently. In particular, aliases are not persisted for cores.
- We highly recommend that the 'alias' feature in Solr not be used due to the high synchronization overhead it brings.
- Alternatively, we should work towards reducing the synchronization involved

Issues for reference, the work is done

- SOLR-1293 - Support for large number of cores and faster loading/unloading of cores. This issue has many child issues focusing on individual changes:
  - Deferred
    - SOLR-919 - Cache and reuse SolrConfig. We've investigated this and while it would certainly lead to improved responsiveness, currently the use-cases do not require this to be implemented. It's not a trivial change, with potential to introduce too many bugs.
  - Fixed/Closed issues
    - SOLR-880 - SolrCore should have a STOP option and a lazy startup option (part of SOLR-1028)
    - SOLR-920 - Cache and reuse IndexSchema
    - SOLR-921 - SolrResourceLoader must cache short name vs fully qualified name
    - SOLR-943 - Make it possible to specify dataDir in solr.xml
    - SOLR-1028 - Automatic core loading unloading for multicore
    - SOLR-1106 - Pluggable CoreAdminHandler (Action) architecture that allows for custom handler access to CoreContainer / request-response
    - SOLR-1108 - Remove synchronization in SolrCore constructor

- SOLR-1306 - Support pluggable persistence/loading of solr.xml details. Wound up discovering cores instead
- SOLR-1416 - Reduce contention in CoreContainer#getCore(). Closed, haven't seen evidence we need to do this.
- SOLR-1530 - Open IndexSearcher lazily
- SOLR-1531 - Provide an option to remove the data directory on core unload. Already done in [[https://issues.apache.org/jira/browse/SOLR-2610|SOLR-2610]
- SOLR-1533 - Partition data directories into multiple "bucket" directories. Will be handled by SOLR-1306.
- SOLR-3980 - list not loaded (lazily loaded) cores for clients. See SOLR-4196
- SOLR-4083 - Move to a directory-based configuration. Part of other JIRAs
- SOLR-4196 - While it looks unrelated, "untangling solr.xml" turns out to contain, very probably, a huge amount of the actual work.
- SOLR-4401 - Add stress test to JUnit tests
- SOLR-4478 - Specify configuration sets. This is actually under development, but currently doesn't pertain to this issue. The reason it was originally included in this page was I was thinking of incorporating SOLR-919 in this JIRA so it's listed for reference.

Other features which may be needed for such a system include:

- Changes to SolrJ for new start/stop commands and better error codes/messages.