New asynchronous meta back-end for OpenLDAP

Nadezhda Ivanova
Software Engineer @ Symas Corp
Slapd-meta

- Ldap-proxy backend, capable of proxying an operation to multiple targets, aggregate the results and present them to the client as if coming from a single server

- Uses one thread per operation to:
  - Find or create a new connection
  - Bind all targets (synchronously)
  - Send to all targets
  - Get entries/results from each target (synchronously) and return them to the client
The problem
The solution

Separate the sending of the proxied operation from the receiving of results:

- Threads will exit instead of waiting, thus freeing the threadpool for other operations.
- A new thread will be used to process the result, only when the result is actually available.
- Instead of connections being cached based on client's credentials, implement a configurable number of persistent connections, and distribute load between them.
Multi-threaded programming

Theory

Practice
Relationship with meta

- Reuses some code
- Data structures very similar
- Shares most configuration options

- A separate module, does not link to meta
- Similarly, requires ldap and monitor
- Options named the same, but in a separate tree
Data

a_metainfo_t – represents a database.
a_metatarget_t – represents a configured target.
a_metaconn_t – contains an array of LDAP connections to each of the configured targets (a_metasingleconn_t), and stores temporary operation data. As many as configured by max-target-conn parameter.
a_metasingleconn_t – represents a single LDAP connection.
bm_context_t – a holder for the temporary operation data.
n - number of configured targets

m - number of connections, as configured by max-target-conns
Threads

- Sender – one for each operation being processed
- Reader – one per metaconn doing actual work
- Timeout – one per database
Sender

1. Select a connection
2. Create a context
3. get a target
   - If No, go back to Select a connection
   - If Yes, check if it's a target
     - If No, go back to get a target
     - If Yes, check if it's bound
       - If No, go back to Bind
       - If Yes, check if this is the last target
         - If No, go back to Send
         - If Yes, exit
Timeout

• Triggered every second
• Walks every metaconn_t in the array
• Checks in the message queue if there are timed out operations, or operations for which result has been sent
• Removes them from the queue if they are not being handled by other threads
• Checks if any of the LDAP connections need reset or have timed out, and resets them if they are not in use.
Configuration

- --enable-asyncmeta

New options
- database asyncmeta – define an asyncmeta database
- max-timeout-ops <number> - the number of consecutive requests that have timed out, after which the connection will be reset. Not configured by default.
- max-pending-ops <number> - the maximum number of operations stored in a connection's message queue. If this limit is reached, the server will return LDAP_BUSY. The default is 128.
- max-target-conns <number> - the number of persistent connections to each target – the default is 255.

Removed options
- single-conn
- use-temporary-conn
Configuration Cont.

database asyncmeta

suffix "dc=missy,dc=com"

network-timeout 3

max-timeout-ops 100

max-pending-ops 100

max-target-conns 10

idle-timeout 3

rootdn "cn=admin,dc=missy, dc=com"

rootpw apassword

uri
"ldap://10.0.2.19/dc=isreally,dc=themaster,dc=com"

rewriteEngine on

rewriteContext bindDN

rewriteRule "(.*)dc=missy,dc=com"
"%1dc=themaster,dc=com" ":" 

rewriteContext searchBase alias bindDN

rewriteContext searchResult

rewriteRule "(.*)dc=themaster,dc=com"
"%1dc=missy,dc=com" ":" 

idassert-bind bindmethod=simple
binddn="cn=admin,dc=isreally,dc=themaster,dc=com"
credentials="somethingsecret"
mode=none

idassert-authzFrom "dn:*"
## Performance – average number of operations per second

<table>
<thead>
<tr>
<th>Number of server threads</th>
<th>Default (15)</th>
<th>3</th>
<th>100</th>
<th>backend</th>
</tr>
</thead>
<tbody>
<tr>
<td>Search spanning multiple targets</td>
<td>205.17</td>
<td>221.9</td>
<td>230.5</td>
<td>asyncmeta</td>
</tr>
<tr>
<td></td>
<td>244.9</td>
<td>210.9</td>
<td>260</td>
<td>meta</td>
</tr>
<tr>
<td>Search spanning one target</td>
<td>301.10</td>
<td>308.2</td>
<td>310.5</td>
<td>asyncmeta</td>
</tr>
<tr>
<td></td>
<td>285.5</td>
<td>284.0</td>
<td>304.0</td>
<td>meta</td>
</tr>
</tbody>
</table>
Performance

- Slower than meta – involves more processing and some additional locking for data protection
- But – throughput is not noticeably affected by number of configured threads
Caveats

- No guarantee it will work with any existing overlays
- In custom modules, do not use o_tmpmem allocator for o_extra data
- Make sure the callbacks are dynamically allocated.