Performance Improvement of OpenLDAP Transactional Backend

Jong Hyuk Choi
jongchoi@us.ibm.com
IBM Thomas J. Watson Research Center
Enterprise Linux Group

Mar 21, 2003
OpenLDAP Transactional Backend

Transactional Backend : back-bdb
- A New backend of recently released OpenLDAP 2.1
  - Use native Berkeley DB API - BDB transactional store
  - Transaction-protected updates via write-ahead logging
    - Rollback upon transient errors such as deadlock
    - Recovery from catastrophic failures
  - Page-level locking
    - Back-lbpm : backend giant lock
    - Improved concurrency
    - Concurrent execution of slapd and admin tools (slapcat, slapadd...)
  - Improvements
    - Store DB data in a binary format : reduce mallocs
    - Simple IDL management
  - Back-bdb performance
    - Overhead of BDB transaction env for non-transaction ops
- **Entry cache**
  - Cache for the id2entry database
  - Essential for high performance directory searches
  - Evolved from back-ldbm entry cache

- **Locking Issues in back-bdb**
  - Entry level locking interferes with BDB page-level locking
  - To avoid deadlock, entry cache locking was redesigned to use BDB locking primitives

---

**Latency**

![Graph showing latency vs. Entry Cache Size]

**Throughput**

![Graph showing throughput vs. Entry Cache Size]
Transaction Overhead

- Hash back-ldbm w/o transaction environment
  - DirMark (mesg) : 1956.8 ops/sec
- Hash back-ldbm w transaction environment
  - DirMark (mesg) : 1821.9 ops/sec (down 6.9%)
- Btree back-ldbm w/o transaction environment
  - DirMark (mesg) : 2006.6 ops/sec
- Btree back-ldbm w transaction environment
  - DirMark (mesg) : 1932.3 ops/sec (down 3.7%)
- Transaction overhead exists for non transaction-protected operations
Dissection of Search Operation

1. **get base entry**
   - **DN2ID, ID2ENTRY**
   - `IDL1`

2. **get search candidates**
   - Scope: SUBTREE
   - Filter: & (cn=Jong*)(l=New York)
   - Candidates = IDL1 & (IDL2 & IDL3)

3. **foreach ID in candidates**
   - `e = ID2ENTRY`
   - `test_filter (e)`
   - `if ( e matches filter and scope )`
     - `send_search_entry (e)`

4. **send_search_result (LDAP_SUCCESS)**
# Base System Profiling

## IBM Trace Facility for Linux (x86)
- Use Pentium performance counter
- INST RETIRED event with x10000 sampling

### Back-bdb

<table>
<thead>
<tr>
<th>Program</th>
<th>Time</th>
<th>Program</th>
<th>Time</th>
<th>Program</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>slapd</td>
<td>31.47%</td>
<td>libdb-4.0</td>
<td>26.99%</td>
<td>vmlinux</td>
<td>13.79%</td>
</tr>
<tr>
<td>ber_printf</td>
<td>2.01%</td>
<td>_ham_item</td>
<td>6.55%</td>
<td>do_zap_page_range</td>
<td>2.26%</td>
</tr>
<tr>
<td>send_search_entry</td>
<td>1.81%</td>
<td>_ham_lookup</td>
<td>6.54%</td>
<td>get_unmapped_area</td>
<td>0.72%</td>
</tr>
<tr>
<td>ber_write</td>
<td>1.51%</td>
<td>_ham_get_cpage</td>
<td>2.22%</td>
<td>save_i387</td>
<td>0.60%</td>
</tr>
<tr>
<td>is_ad_subtype</td>
<td>1.48%</td>
<td>_ham_item_next</td>
<td>1.81%</td>
<td>do_signal</td>
<td>0.39%</td>
</tr>
<tr>
<td>ad_inlist</td>
<td>1.29%</td>
<td>_lock_get_internal</td>
<td>1.16%</td>
<td>schedule</td>
<td>0.38%</td>
</tr>
<tr>
<td>avl_find</td>
<td>1.26%</td>
<td>_db_tas_mutex_lock</td>
<td>0.91%</td>
<td>tcp_sendmsg</td>
<td>0.37%</td>
</tr>
<tr>
<td>libc</td>
<td>15.57%</td>
<td>libpthread</td>
<td>9.09%</td>
<td>misc</td>
<td>3.09%</td>
</tr>
</tbody>
</table>

### Back-ldbm

<table>
<thead>
<tr>
<th>Program</th>
<th>Time</th>
<th>Program</th>
<th>Time</th>
<th>Program</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>slapd</td>
<td>42.72%</td>
<td>libdb-4.0</td>
<td>4.44%</td>
<td>vmlinux</td>
<td>11.44%</td>
</tr>
<tr>
<td>ber_printf</td>
<td>2.58%</td>
<td>_bam_cmp</td>
<td>0.87%</td>
<td>save_i387</td>
<td>0.77%</td>
</tr>
<tr>
<td>send_search_entry</td>
<td>2.46%</td>
<td>_bam_search</td>
<td>0.61%</td>
<td>schedule</td>
<td>0.70%</td>
</tr>
<tr>
<td>ber_write</td>
<td>2.05%</td>
<td>_memp_fget</td>
<td>0.40%</td>
<td>send_sig_info</td>
<td>0.56%</td>
</tr>
<tr>
<td>is_ad_subtype</td>
<td>1.99%</td>
<td>_db_c_get</td>
<td>0.29%</td>
<td>restore_i387</td>
<td>0.54%</td>
</tr>
<tr>
<td>ber_put_seqorset</td>
<td>1.74%</td>
<td>_memp_fput</td>
<td>0.27%</td>
<td>tcp_sendmsg</td>
<td>0.53%</td>
</tr>
<tr>
<td>ad_inlist</td>
<td>1.53%</td>
<td>_db_icursor</td>
<td>0.25%</td>
<td>do_signal</td>
<td>0.47%</td>
</tr>
<tr>
<td>libc</td>
<td>23.08%</td>
<td>libpthread</td>
<td>14.06%</td>
<td>misc</td>
<td>4.26%</td>
</tr>
</tbody>
</table>
Candidate Caching

1. get base entry
   DN2ID, ID2ENTRY

2. get search candidates
   scope: SUBTREE
   filter: \& (cn=Jong*) \& (1=New York)
   candidates = IDL1 \& (IDL2 & IDL3)

3. foreach ID in candidates
   e = ID2ENTRY test_filter (e)
   if (e matches filter and scope)
      send_search_entry (e)

4. send_search_result (LDAP_SUCCESS)
Candidate Caching

Caching IDL of Search Candidates

- Cache IDL of candidates of a search
  - Performs better than back-ldbm
  - DirMark (mesg) : 2378 ops/sec (13% higher than back-ldbm)

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>slapd</td>
<td>44.14%</td>
<td>libdb-4.0</td>
<td>3.45%</td>
<td>vmlinux</td>
<td>12.69%</td>
</tr>
<tr>
<td>send_search_entry</td>
<td>2.70%</td>
<td>_lock_get_internal</td>
<td>0.49%</td>
<td>save_i387</td>
<td>0.87%</td>
</tr>
<tr>
<td>ber_printf</td>
<td>2.69%</td>
<td>_ham_func5</td>
<td>0.36%</td>
<td>schedule</td>
<td>0.69%</td>
</tr>
<tr>
<td>avl_find</td>
<td>2.27%</td>
<td>_ham_lookup</td>
<td>0.36%</td>
<td>send_sig_info</td>
<td>0.64%</td>
</tr>
<tr>
<td>ber_write</td>
<td>2.24%</td>
<td>_ham_item</td>
<td>0.23%</td>
<td>restore_i387</td>
<td>0.61%</td>
</tr>
<tr>
<td>is_ad_subtype</td>
<td>2.18%</td>
<td>_lock_put_internal</td>
<td>0.21%</td>
<td>do_signal</td>
<td>0.54%</td>
</tr>
<tr>
<td>ber_put_seqset</td>
<td>1.93%</td>
<td>_lock_getobj</td>
<td>0.14%</td>
<td>tcp_sendmsg</td>
<td>0.51%</td>
</tr>
<tr>
<td>libc</td>
<td>21.54%</td>
<td>libpthread</td>
<td>13.66%</td>
<td>misc</td>
<td>4.52%</td>
</tr>
</tbody>
</table>

Consistency Problem

- Too costly to manage mappings of component IDLs in dn2id / index DB to candidates
IDL Stack Slab and IDL Cache

1. get base entry
   DN2ID, ID2ENTRY

2. get search candidates
   scope: SUBTREE
   filter: \&(cn=Jong\")\&l=New York
   candidates = IDL1 \& (IDL2 \& IDL3)

3. foreach ID in candidates
   e = ID2ENTRY
   test_filter (e)
   if (e matches filter and scope)
      send_search_entry (e)

4. send_search_result (LDAP_SUCCESS)
IDL Stack Slab and IDL Cache

### Caching Component IDLs
- Dn2id / index databases, efficient consistency maintenance

<table>
<thead>
<tr>
<th>Component</th>
<th>slapd</th>
<th>libdb-4.0</th>
<th>vmlinux</th>
</tr>
</thead>
<tbody>
<tr>
<td>ber_printf</td>
<td>2.28%</td>
<td>_lock_get_internal</td>
<td>0.69%</td>
</tr>
<tr>
<td>send_search_entry</td>
<td>2.24%</td>
<td>_ham_lookup</td>
<td>0.67%</td>
</tr>
<tr>
<td>is_ad_subtype</td>
<td>1.96%</td>
<td>_db_tas_mutex_lock</td>
<td>0.46%</td>
</tr>
<tr>
<td>ber_write</td>
<td>1.92%</td>
<td>_ham_item</td>
<td>0.45%</td>
</tr>
<tr>
<td>avl_find</td>
<td>1.86%</td>
<td>_ham_get_epage</td>
<td>0.39%</td>
</tr>
<tr>
<td>ber_put_seqorset</td>
<td>1.63%</td>
<td>_ham_item_next</td>
<td>0.35%</td>
</tr>
<tr>
<td>libc</td>
<td>19.33%</td>
<td>libpthread</td>
<td>12.78%</td>
</tr>
</tbody>
</table>

### Caching IDL Stack Slabs
- `search_candidates()` allocs `(depth+1)*IDL_UM_SIZE` stack
- Slab cache for IDL stack, depth is fully configurable

<table>
<thead>
<tr>
<th>Component</th>
<th>slapd</th>
<th>libdb-4.0</th>
<th>vmlinux</th>
</tr>
</thead>
<tbody>
<tr>
<td>send_search_entry</td>
<td>2.51%</td>
<td>_lock_get_internal</td>
<td>0.75%</td>
</tr>
<tr>
<td>ber_printf</td>
<td>2.39%</td>
<td>_ham_lookup</td>
<td>0.66%</td>
</tr>
<tr>
<td>is_ad_subtype</td>
<td>2.00%</td>
<td>_ham_item</td>
<td>0.47%</td>
</tr>
<tr>
<td>avl_find</td>
<td>1.98%</td>
<td>_ham_get_epage</td>
<td>0.41%</td>
</tr>
<tr>
<td>ber_write</td>
<td>1.91%</td>
<td>_db_tas_mutex_lock</td>
<td>0.38%</td>
</tr>
<tr>
<td>ber_put_seqorset</td>
<td>1.73%</td>
<td>_ham_func5</td>
<td>0.38%</td>
</tr>
<tr>
<td>libc</td>
<td>20.43%</td>
<td>libpthread</td>
<td>13.79%</td>
</tr>
</tbody>
</table>

**misc**
**IDL Stack Slab and IDL Cache**

- Performance of Combined IDL Stack Slab and IDL Caching

- Currently in HEAD
- Fully configurable in slapd.conf
  - `idlcachesize <cache size in # of IDLs>`
  - `searchstack <slab size in filter depth>`
BER Transfer Cache

1. get base entry
   DN2ID, ID2ENTRY

2. get search candidates
   scope: SUBTREE
   filter: &{(cn=Jong*)(l=New York)}
   candidates = IDL1 & (IDL2 & IDL3)

3. foreach ID in candidates {
   e = ID2ENTRY
   test_filter (e)
   if (e matches filter and scope)
      send_search_entry (e)
}

4. send_search_result (LDAP_SUCCESS)

DUA : Directory User Agent
DSA : Directory Service Agent

INDEX (CN)
Jong*->IDL
Jon*->IDL
Hub*->IDL

INDEX (OC)
ou->IDL
inetOrgPerson->IDL

INDEX (L)
New York->IDL
CA->IDL

ID2ENTRY
ID -> ENTRY
7 -> {ou=research,o=ibm,c=us},
    {ou=ou}, {ou=research}
100 -> {cn=Hubertus Franke,ou=research,o=ibm,c=us},
       {ou/inetOrgPerson},
       {cn=Hubertus Franke}, {l=New York}, ...
101 -> {cn=Jong Choi,ou=research,o=ibm,c=us},
       {ou/inetOrgPerson},
       {cn=Jong Choi}, {l=New York}, ...

SearchRequest
PDU in BER

ASN1 Decode
base, nbase
scope, derefAlias, limit, attr
index, filter

Build Attr Desc
attr

search specification: base, scope, derefAlias, limit, attr

Berkeley Database
(BDB)

DUA (client)

slapd frontend

slapd backend

back-sql
back-perl
back-ldap
back-ldb
entry cache
ldbm API

RDBMS (DB2)
LDAP DSA (server)

BDB
ndbm
gdbm

LDAP DSA (server)

BER cache
search response
matched entries

BER Transfer Cache
Summary

- **Back-bdb Performance Improvements (mesg)**
  - Entry cache: 1006 ops/sec -> 1520 ops/sec (50.5%)
  - IDL / IDL stack cache:
    - 1520 ops/sec -> 1993 ops/sec (31.1%)
  - BER cache: 1993 ops/sec -> 2258 ops/sec (13.3%)

- **Further Performance Studies**
  - BER cache
  - DN cache
  - ...