Lightweight Replication for OpenLDAP

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

Mar 21, 2003
OpenLDAP Replication

- **Slurpd**
  - producer-initiated, log-based replication
  - out-of-band replica management

- **A New Lightweight Replication**
  - consumer-initiated, state-based, pull replication
  - eventual consistency
  - replication by search operation - no prior agreement
  - minimal history information
  - minimal consumer information in the producer side
  - master-slave
  - support partial replication (fractional and sparse)
  - primary and secondary replication
  - based on the client synchronization protocols
    - draft-zeilenga-ldup-sync-01
    - draft-ietf-ldup-lcup-04
**Sync Protocol - LCUP**

- **LDAP Client Update Protocol** (Megginson et al.)
  - synchronize directory entries, assume history information
  - syncOnly, syncAndPersist, persistOnly

- **Diagram**

  - searchResultDone
    + syncDone(cookie)

  - searchRequest
    + syncRequest(syncOnly)
      (cookie)

  - searchResultEntry
    + syncUpdate(leftSet=0)

  - searchResultEntry
    + syncUpdate(leftSet=1)

  - searchResultDone
    + syncDone
      (cookie | reloadRequired)

  - searchRequest
    + syncRequest(syncAndPersist)
      (cookie | reload)

  - searchResultEntry
    + syncUpdate
      (persistPhase, leftSet=0 | 1)

  - add
    modify/moddn/delete

  - modify/moddn/delete

  - leftSet=1 case of syncOnly requires history information such as log or tombstone to determine whether the entry was within the search result before the operation

  - if consistency is too costly or impossible to achieve, producer issues reloadRequired

  - tradeoff in history info vs. chattiness

  - add/modify/moddn/delete
Sync Protocol - LDAP SYNC

LDAP Synchronization Protocol (Zeilenga and Choi)

✓ synchronizing shadowed information, no history information assumed
✓ refreshOnly, refreshAndPersist

- searchResultDone + syncDone(cookie)
- searchRequest + syncRequest(refreshOnly)
  (cookie)
- searchResultEntry + syncState
  (all present + add)
- searchResultDone
  (success or reloadReq)
  + syncDone (cookie)
- searchRequest
  + syncRequest(refreshAndPersist)
  (cookie | reload)
  (all present + add)
- intermediateResponse + syncInfo(refreshDone)
- searchResultEntry
  + syncUpdate
  (add | modify | delete)

✓ present: only send DN and entryUUID
✓ in the refresh phase,
if entryCSN <= cookieCSN, send present
else, send add
✓ no history information is required
✓ chatty
To reduce chattiness,

- maintain lightweight history information for the finite number of refreshOnly consumers

replicaSubentry contains

&n#x00D7; syncSearchSpec: {base, scope, filter, attrs, binddn}
&n#x00D7; presentRange: {startCSN, endCSN}

- if an update causes an entry to leave one of the subtree refinement in the replicaSubentries under admPoint, add e.CSN before change as a presentRange attribute value
- when a refreshOnly of the same spec arrives, send present entries within the sync search result set only if (startCSN <= e.CSN <= endCSN) for each values of the presentRange attribute
- presentRange values shall be delivered as the syncInfo cookie
- tradeoff of consumer info vs. chattiness
  - presentRange merge, open-ended presentRange value (0,inf)
  - when consumer info is dropped -> presentRange = [0..inf]
Information Model

- shadowed information: prefix info + area info + subordinate info
- objectClass
  - glue: only have naming information, objectClass attribute
  - replicaProducerSubentry: cn, replicaType, replicaStatus, replicationMode, syncSearchSpec, presentRange, secondaryConsumer, subtreeSpec
  - replicaConsumerSubentry: cn, replicaStatus, replicationMode, producer, binddn, bindmethod, credentials, interval, secondaryProducer, subtreeSpec
- replicaSubentries: rdns of all replicaSubentries subordinate to admPoint
- alias, reference: as in manageDsaIT
CSN and UUID

Change Sequence Number

- imposes total ordering of a sequence of updates
- time#change count#replica id#modification number
- 2003032111:21:31z#0x00A7#1#0x0000
- entryCSN, replicaCSN, startCSN, endCSN

CSN Generation

- OpenLDAP generates CSN before backend update operation
- max CSN of the committed update may not be the max generated CSN
  -> replicaCSN may be greater than the actual replace state
- replicaCSN of the producer should be set to the min outstanding CSN - δ

UUID : Universally Unique Identifier

- DNs can change and so are unreliable
- entryUUID
**Configuration**

- **consumer configuration**
  - slapd.conf

```plaintext
database           bdb
suffix             "o=ibm,c=us"
syncrepl id=1
    producer=ldap://producer.ibm.com:9009
    binddn="cn=repluser,o=ibm,c=us"
    bindmethod=simple
    credentials=secret
    replicationbase="ou=enterprise linux,o=ibm,c=us"
    filter="objectClass=organizationalPerson"
    attrs="cn sn description telephoneNumber l ou title"
    scope=sub
    type=refreshOnly
    interval=24h
```
Replication Thread

replication thread pool of consumer

✓ submit a replication thread
upon parsing a `syncrepl` line of slapd.conf

```c
void *do_syncrepl( void *ctx, void *arg )
{
    get consumer replica cookie from replicaSubentry
    init connection to the producer
    bind to the producer
    ldap_search_ext(ld, si->base, si->scope, si->filterstr,
                    si->attrs, 0, NULL, NULL, NULL, -1, &msgid)
    while (msg) {
        if (LDAP_RES_SEARCH_ENTRY) {
            entry = msg_to_entry( msg, modlist, &rctrls )
            parse control ( &rctrls ) to get syncState, entryUUID, entryCSN
            syncrepl_entry ( entry, modlist )
        } else if (LDAP_RES_SEARCH_RESULT) {
            update consumer replica cookie
            delete entries within presentRange but not in si->presentlist
            reschedule do_syncrepl
        } else if (LDAP_RES_INTERMEDIATE_RESP) {
            if (syncInfo == cookie) update consumer replica cookie
            else if (syncInfo == presentRangeCookie) update presentRange
        }
    }
}
```
Synchronization Update

- synchronization update
  ✓ internal modify and add with null callback functions

```c
static int syncrepl_entry ( entry, modlist ) {
    set null_callback for 4 callbacks
    switch ( syncState ) {
        case LDAP_SYNC_PRESENT:
            avl_insert ( &si->presentlist, syncUUID )
        case LDAP_SYNC_ADD :
        case LDAP_SYNC_MODIFY :
            rc = be->be_modify ( &entry->e_name, modlist )
            if ( rc == LDAP_NO_SUCH_OBJECT ) {
                rc = be->be_add ( entry )
                if ( rc == LDAP_REFERRAL ) {
                    syncrepl_add_glue ( entry )
                }
            }
        case LDAP_SYNC_DELETE :
            rc = be->be_delete ( &entry->e_name );
    }
}
```
Summary

- **SyncRepl**
  - a lightweight replication engine for OpenLDAP

- **Applications**
  - Lightweight Directory Replication
  - Synchronization for OpenLDAP Proxy Cache from IBM India Research
  - IBM Directory Integrator Connector for OpenLDAP

- **Plans**
  - Initial code available
  - Initial source code contribution soon
  - Solicit requirements from the community