Sync-based Replication:

Protocol and OpenLDAP Implementation

Jong Hyuk Choi  Kurt Zeilenga
IBM Research  OpenLDAP Project

OpenLDAP Developer’s Conference – Vienna – 18 July 2003
Agenda

- Directory Replication / Synchronization
  - Replication
  - Synchronization
  - OpenLDAP Slurpd
- LDAP Content Synchronization Protocol
  - Why not LCUP ?
  - Basic Protocol Description
  - Optimized Protocol for Traffic Reduction
- SyncRepl : A New Replication Engine
  - Sync-based Replication Engine Design
  - Client-based Replication Engine Design
- Target Applications
- Summary
Directory Replication

- Replication for High Availability, Performance, Security, Locality …

- When the directory is updated in a replicated setup, replicas need to be synchronized to each other to provide a single directory image

- Partition: unit of replication
  Replica: copy of a partition

- Master-slave vs. Multi-master
  - Distributed directories via referral: referral chasing or chaining
  - Separate masters for different roles

- Partial vs. Whole replication

- Replication topology
Directory Synchronization

- Keeping multiple directories up-to-date with each other

- Stateful vs. Stateless
  - Stateful: synchronization action is based on replica status
  - Stateless: provider assumes the replica status and synchronize accordingly

- State-based vs. History-based
  - State-based: synchronization action is determined based on the current replica status
  - History-based: history lookup is required for synchronization

- Incremental vs. Full Reload
  - Incremental: only changes made after last sync be transmitted
  - Full Reload: requires full reloads per every (or most) sync

- Push vs. Pull: provider-initiated or consumer-initiated
- Polling vs. Listening: periodic sync or event-driven sync
- Unit of synchronization: Entry-level vs. Attribute-level
**Slurpd**

- **Slurpd**: standalone LDAP Update Replication Daemon
  - Master-slave
  - Multi-master for one level replication only without predefined URP (Update Reconciliation Protocol)
  - Stateless, History (relog) based, Push, Incremental synchronization

**Example**

1. Initial replication (db copy or Idif load) with master read-only
2. Promote master to read-write
3. Incremental synchronization
LDAP Content Synchronization Protocol

- Stateful: cookie represents current replica status
- State-based: does not mandate history store
- Incremental: only changes after last sync are to be transmitted
- Pull: clients initiate synchronization sessions
- Polling & Listening: refreshOnly & refreshAndPersist
- Partial replication: supports arbitrary search specification
- Eventual consistency
- Unit of synchronization: entry
- Does not require predefined synchronization arrangement
  per-consumer information history
Sync-based Replication

LDAP Sync : Example

provider.company.com
ou=assets
ou=computers
com1
com2
2003071309:00:00Z0000:00:0000
2003071712:00:00Z0000:00:0000

ou=users
ou=computers
com1
com2

cn=foo

Sync search
base=assets
(objectClass=computers)
attrs=* 
refreshOnly sync control
cookie=2003071807:30:00Z0000:00:0000

Sync search entry
cn=com2,ou=computers,
dc=comany,dc=com
attributes, add state
Sync search entry

cn=com1
present state
Sync Info
Cookie=2003071809:00:00Z0000:00:0000

Sync search result
Sync done control

consumer.company.com
ou=assets
ou=computers
com1
com2

cn=foo

com1 < cookie
com2 > cookie
Why not LCUP?

- LCUP (LDAP Client Update Protocol)
  - Sends \{updates + deletes\}
  - Requires history information for reasonably efficient implementation
  - OpenLDAP doesn’t maintain history information (tombstone, changelog …)

- 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

- add/modify/moddn/delete
Basic Protocol: Refresh

Initial Content Request

Content Refresh Request

{ update + present / delete }

Send Entries

Done with Cookie

Send “Changed” Entries

Send “Present” Messages or “Delete” Messages

Done with “Cookie”

Repeat
Basic Protocol: Refresh & Persist

Content Request

Send Entries
Refresh Done
Send “Changed” Entries and/or “Deleted” messages
New “Cookie”

Cancel

Repeat

Done with Cookie / Acknowledge Cancel
Protocol Optimization (Present Phase + Delete Phase)

- **Delete mode**: requires full reload if replica state is out of history
- **Present mode**: requires present entry transmission even if replica is within history
- **Present + Delete**: sends deletes for the scope-outs covered by the history store
  - Sends presents for those not covered by the history

Initial Content Request

Content Refresh Request

Repeat

Send Entries
Done with Cookie

Send “Changed” Entries
Send “History Cookie”
Send “Present” Messages
Send “Delete” Messages

Done with “Replica Cookie”
Replication Engine Design

- Periodic execution of refresh tasks:
  - scheduling facility in `slapd_daemon_task()`
  - simple `runqueue` implementation

- Storage of sync cookie in consumer:
  - `Subentry`: `syncConsumerSubentry`

- Operational attributes:
  - Regenerated at replica as needed

- `structuralObjectClass`:
  - generate on-the-fly in `slapModsOpattrs()`

- UUID, CSN based implementation
  - UUID: stable entry identifier
  - CSN as cookies

- Configuration example
  - `test017`: `refreshOnly`, `test018`: `refreshAndPersist`, `test020`: cascading
Replication Engine Design: Glueing

- **Glues**: Naming entries for holes in DIT
  - Delivery can be out of hierarchy order after several rounds of updates
  - Partial replication

- **Glue construction**
  - `syncrepl_add_glue()`
  - Find first non-glue superior object
  - Create glues from down to the entry

- Schema checking bypass for glues (rdn attribute requirement)
- Glueing for LDAP Proxy Cache
Client-based Replication Engine

- Heterogeneous replication
  - SyncRepl engine needs to talk to generic LDAP servers

- Synchronization without LDAP Content Synchronization
  1. Search for (&(original filter)) asking only UUID and CSN attributes  - Present phase
  2. Delete replica entries not returned by (1)
  3. Search for (&(original filter)(entryCSN>cookie)(entryCSN=<maxCSN(1))
     asking replicated attributes + UUID + CSN  - Update phase
    - Replica is synchronized to the point maxCSN(1)

- Comparison with SyncRepl with LDAP Content Synchronization
  - Only supports polling
  - Extra requests / replies
  - Extra traffic (only present mode)
Target Applications

- Slurpd replacement
  - OpenLDAP to OpenLDAP replication based on LDAP content sync protocol
  - Heterogeneous replication by using client-based replication engine

- LDAP Proxy Cache synchronization
  - Replace current TTL based scheme
  - Replication and Caching

- IBM Directory Integrator Connector
  - Heterogeneous Directory Synchronization : Meta-directory
Summary

- LDAP Content Sync Protocol
  - draft-zeilenga-ldup-sync-xx.txt

- OpenLDAP SyncRepl Engine
  - servers/slapd/syncrepl.c
  - tests/data/slapd-syncrepl-master.conf
  - slapd-syncrepl-slave-xxxxx.conf

- Any Questions?