OpenLDAP, syncrepl and multimaster replication

Ondřej Kuzník, Symas Corp.

November 2019

Hello Earth

The why

- There is little documentation and a lot of confusion
- A lot of work has gone into improving OpenLDAP's replication functionality with more to come
- Many distributed systems exist, but not many that openly discuss guarantees/concessions

Pull Out the Pin

A step down from ACID in distributed systems

CAP theorem

- (Eventual) Convergence
- Availability
- Persistence of successful updates

And Dream Of Sheep

A step down from ACID in distributed systems

CAP theorem

- (Eventual) Convergence
- Availability
- Persistence of successful updates
- Pick two

In Search Of Peter Pan

Content synchronization is defined in RFC 4533 (a.k.a. Syncrepl)

- Like every self-respecting replication protocol piggy-backs on a search request
- Start with no knowledge - get entries just like a regular search
- Further searches can be more efficient
- Utilises entryUUID attribute (RFC4530) - stable across renames
- A session maintained with an opaque cookie

OpenLDAP has a client implementation in `syncrepl.c`, as well as other protocols (master only).

Reaching Out

Search every time

- Search request + Sync Request Control (no cookie)
  - `refreshOnly`
- Entry + Sync State Control: entryUUID and state:
  - add (1)
- Search response + Sync done control:
  - cookie
  - refreshDeletes = FALSE

Jig of Life

Search every time

- Search request + Sync Request Control (w. cookie)
  - `refreshOnly`
- Entry + Sync State Control: entryUUID and state:
  - present (0)
  - add (1)
- Search response + Sync done control:
  - cookie
  - refreshDeletes = FALSE

Running Up That Hill

Search every time

- Search request + Sync Request Control (w. cookie)
  - `refreshOnly`
- Entry + Sync State Control: entryUUID and state:
  - present (0)
  - add (1)
- Search response + Sync done control:
  - cookie
  - refreshDeletes = FALSE / TRUE

symas

Breathing

Search every time

- Search request + Sync Request Control (w. cookie)
  - `refreshOnly`
- Entry + Sync State Control: entryUUID and state:
  - present (0)
  - add (1)
- Sync Info Intermediate Response of type
  - `refreshPresent` w. refreshDone = FALSE
- Entry + Sync State Control: entryUUID and state:
  - delete (3)
- Search response + Sync done control:
  - cookie
  - refreshDeletes = FALSE / TRUE

King of the Mountain

Search every time

- Search request + Sync Request Control (w. cookie)
  - `refreshOnly`
- Entry + Sync State Control: entryUUID and state:
  - present (0)
  - add (1)
- Sync Info Intermediate Response of type either
  - `refreshPresent` w. refreshDone = FALSE
  - `syncIdSet`
    - refreshDeletes = FALSE / TRUE
    - set of UUIDs
- Entry + Sync State Control: entryUUID and state:
  - delete (3)
- Search response + Sync done control:
  - cookie
  - refreshDeletes = FALSE / TRUE

Leave It Open

When client wants to be aware of changes as they happen.

- Search request + Sync Request Control
  - `refreshAndPersist`
- [optional present/delete phase messages]
- Sync Info Intermediate Response of type either
  - `refreshPresent`: cookie and refreshDone = TRUE
  - `refreshDelete`: cookie and refreshDone = TRUE
- Entry + Sync State Control: entryUUID and state:
  - add (1) / modify (2) / delete (3)
  - optional cookie
- interspersed with Sync Info Intermediate Responses of type
  - `newcookie`: cookie

Feel It

Cookies are opaque to clients but server uses it to identify:

- entries added/changed since
- entries deleted since
- or nothing changed

If all else fails, either:

- return Search Done Response with result e-syncRefreshRequired (4096)
- act as if no cookie was received

In OpenLDAP, you'll find this in overlay `syncprov` with an ephemeral sessionlog to track deletes

Experiment IV

Replicating changes: Delta-sync

- Initial load off target DB, then from its log (append-only, specific to each replica)
- Overlay `accesslog`, separate DB+`syncprov`
- Log DB is a representation of changes and that's what we replicate **from**
  - We only ever delete (expire) the oldest entry
  - `syncprov` configured never to propagate deletes
  - If oldest entry not new enough to resume a session, tell to refresh
  - Client falls back to replicating the target DB to catch up

Big Stripey Lie

Case not catered for by RFC 4533

Somewhere in Between

Case not catered for by RFC 4533 - what if the client is also an LDAP server?

- Maybe it needs to accept and send writes back - both replicate off each other

Walk Straight Down the Middle

Case not catered for by RFC 4533 - what if the client is also an LDAP server?

- Maybe it needs to accept and send writes back - both replicate off each other
- Cookie can't stay opaque - rid, serverID, CSN / contextCSN set (a vector clock keyed on serverID)
- Makes serverID 0 special - single master

Deeper Understanding

Case not catered for by RFC 4533 - what if the client is also an LDAP server?

- Maybe it needs to accept and send writes back - both replicate off each other
- Cookie can't stay opaque - rid, serverID, CSN / contextCSN set (a vector clock keyed on serverID)
- Makes serverID 0 special - single master or pure client
- In refreshAndPersist even clients need to interpret the cookie
- Limits on entry broadcasts - based on rid/sid/csn combo (do not transmit to originator, do not transmit to sender)

Even more fun with delta-MMR.

Love and Anger

Conflicts are inevitable

- Add/Add
- Add/Delete of parent
- Rename/Modify/Delete
- etc.

Not This Time

Conflicts are inevitable

- Add/Add
- Add/Delete of parent
- Rename/Modify/Delete
- etc.

"Last version" wins (maintains convergence).
We always try to preserve C, plus give A (in OpenLDAP we don't have the tools to provide P).

Don't Give Up

Conflicts are inevitable

- Add/Add
- Add/Delete of parent
- Rename/Modify/Delete
- etc.

"Last version" wins (maintains convergence).
We always try to preserve C, plus give A (in OpenLDAP we don't have the tools to provide P).

Alternative approaches exist - see LDAPCon 2017 presentation by Ludwig Krispenz

How to be Invisible

Operational concerns

- Clocks - OpenLDAP uses timestamps for conflict resolution
- Chattiness
  - Plain sends full entries
  - Delta only changes with low (constant) overhead
  - In MMR, messages still get duplicated
    - Prune the graph while maintaining reachability
    - Needs extra communication between replicas - new protocol
    - Not done in OpenLDAP

Be Kind to my Mistakes

Issues in current implementation

- sessionlog is ephemeral (too many costly present phase refreshes)
- ITS#8768: delete phase can't be interrupted safely
- ITS#8125: present phase refresh not always MMR ready (2.5 item)
- contextCSN is cached by `syncprov` (can break delta-MMR during DR)
- overlays break guarantees and need to be aware of too much

The Big Sky

Plans

- Persistent sessionlog
- Merge stuff into a single overlay
- Transactions (RFC 5805)

Aerial

Plans

- Persistent sessionlog
- Merge stuff into a single overlay
- Transactions (RFC 5805)

Wishlist

- Testbed (even a chaos-monkey one)
- Help finding/implementing an protocol to maintain a more efficient MMR

π

Thank you