Flexible LDAP load balancing

Ondřej Kuzník, Symas Corp.

October 2018

Why do this?

- Some applications cannot do failover nor load balancing on their own
- Most load balancers stay at session level
- Backends going unavailable temporarily might not receive enough traffic after they rejoin

You can do better if you decouple the client and backend connections and distribute on a per request basis.

Where we left off

At the time of LDAPCon 2017, the Load Balancer was:

- A fully asynchronous LDAP-aware PDU router buildable as a standalone daemon
- A prototype gone through initial load testing at a client
- Round-robin load distribution only
- Just started planning development of the missing features
- Expected to be feature-complete mid-2018

Where we are now

- Nadya joined the project
- Online monitoring and reconfiguration is supported through a slapd bridge
- Support for full RFC4511, (Start)TLS, SASL binds
- Fragile and error-prone memory subsystem replaced by EBR
- Still only round-robin
- Through rigorous testing at a client integrating it into their systems

Closer to OpenLDAP

- Support for building as an OpenLDAP module presenting itself as a backend implementation
- Shared task queue and thread pool, but otherwise two systems coexisting in the same process, each managing its own sockets, I/O, ...
- Can expose monitoring data through `cn=monitor`
- Can hook into `cn=config` for online reconfiguration
- In the future even tighter integration might be possible

Protocol support

Full LDAP support for clients (almost)

- vanilla - Add, Search, Modify, ...
- Unbind, Abandon
- Bind - proxied, multi-stage SASL binds as well (with caveats)
- Extended ops - StartTLS, any that don't interact with the session
- LDAPS, client certificates and SASL EXTERNAL

Can add ProxyAuthz control to forwarded operations.

Can use `libsasl2` to do SASL to bind its own connections

Epoch based memory reclamation (EBR) (Fraser 2004)

Pure reference counting with circular dependencies is crushing

Epoch based memory reclamation (EBR) (Fraser 2004)

Pure reference counting with circular dependencies is crushing - maintaining guarantees that a pointer can be traversed
Maybe delay freeing an object until no tasks that saw it are active anymore:

Epoch based memory reclamation (EBR) (Fraser 2004)

Pure reference counting with circular dependencies is crushing - maintaining guarantees that a pointer can be traversed
Maybe delay freeing an object until no tasks that saw it are active anymore:

1. Tasks enter into the '`current`' epoch
2. Items to be disposed are registered with the '`current`' epoch
3. Once epoch '`current-1`' is empty, items in '`current-2`' can be freed; '`current++`'

Epoch based memory reclamation (EBR) (Fraser 2004)

Pure reference counting with circular dependencies is crushing - maintaining guarantees that a pointer can be traversed
Maybe delay freeing an object until no tasks that saw it are active anymore:

1. Tasks enter into the '`current`' epoch
2. Items to be disposed are registered with the '`current`' epoch
3. Once epoch '`current-1`' is empty, items in '`current-2`' can be freed; '`current++`'

- Make sure tasks always leave their epoch in a timely manner.
- Load balancer is non-blocking (libsasl2 isn't! Plugins)

Performance

We can easily achieve sub-millisecond response times if the OS can

- Most of the heavy lifting is done by the OS
- Load balancer latencies suffer badly when OS overloads
- LDAP is not I/O and interrupt friendly

Further work on performance and being more predictable during overload is under consideration.

Bigger picture

Server is tiny and has little memory or understanding of the environment

- Are all backends really up-to-date?
- What is the replication latency?
- Is one of the backends misconfigured/misbehaving?

Monitoring, scaling and management tools needed to orchestrate the environment and react to changes.

Where next?

- Weighted load balancing, high availability, connection affinity
- Bigger picture tooling
- LDAP Transactions (RFC 5805)
- Portability
- Optimise for performance (more intelligent resource management, zero-copy, . . .)

Find this useful?

Join us and with your help, we can make it sooner, better.