The CTDB Report

Martin Schwenke

<martin@meltin.net>

Samba Team · DDN

SambaXP 2022

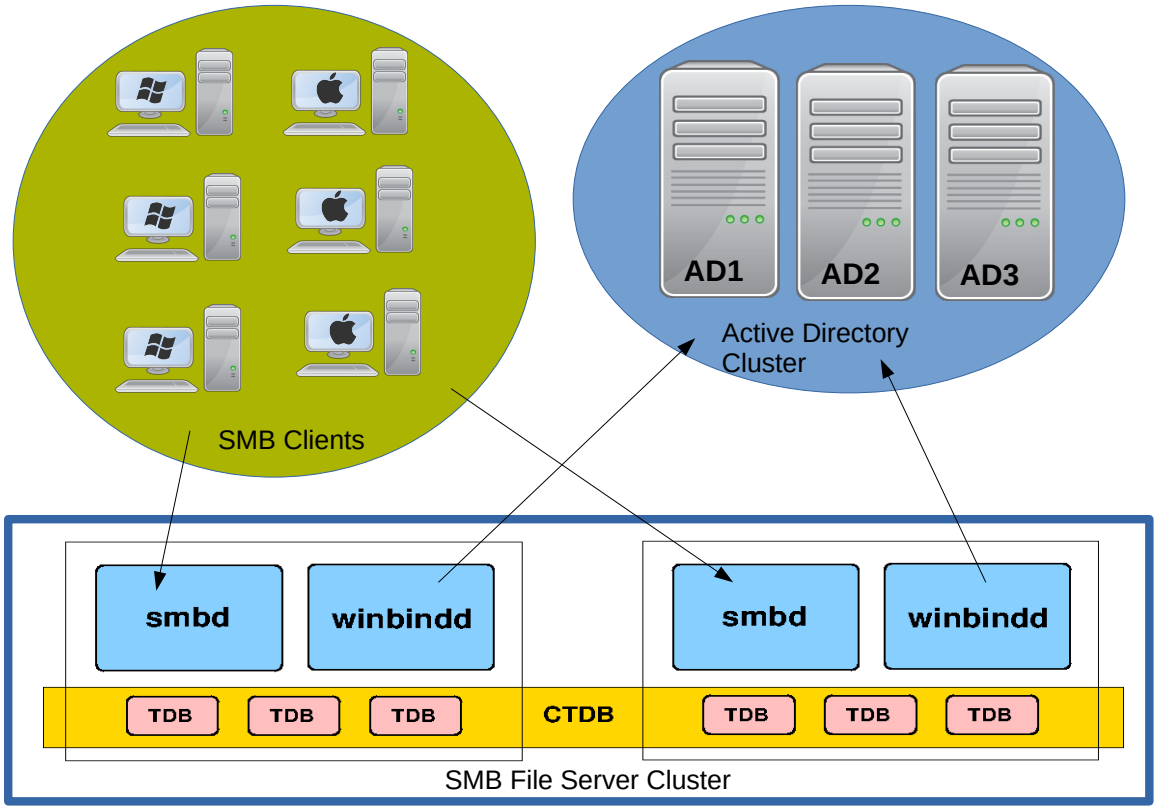
Overview

Audience

Everyone!

. . . not just but developers

Clustered Samba



What is CTDB?

- Clustered database for Samba metadata
- Cluster-wide messaging transport
- Cluster management — leadership, membership
- Dynamic IP address failover
- Service management (`smbd`, `winbindd`, NFS, ...)

Progress

Authors

Martin Schwenke	~~255~~	217
Volker Lendecke		14
Amitay Isaacs		5
Ralph Boehme		3
Pavel Filipenský		3
David Disseldorp		3
Stefan Metzmacher		2
Archana		2
Vinit Agnihotri		1
David Mulder		1
Andreas Schneider		1
	~~290~~	252

Reviewers

Amitay Isaacs	203
Martin Schwenke	11
Jeremy Allison	10
David Disseldorp	8
Jose A. Rivera	8
Ralph Boehme	8
Volker Lendecke	5
Andrew Bartlett	3
Douglas Bagnall	2
Samuel Cabrero	2

Commits by area

Leader broadcast, election, cluster lock, ...	60
Bulk test code quality/portability improvements	19
Fix `ctdb disable` race (BZ14784, 2021-08) (really!)	18
Load tunables in ctdbd, remove `ctdbd_wrapper`	15
Log reopening on `SIGHUP`	12
Fix `ctdb disable` race (BZ14466, August 2020-08) (not really...)	12
`debug_locks.sh` support for POSIX robust mutexes	11
Drop unused serial recovery code	10
Clean up `ctdb_etcd_lock`	9
Fix `run_event` crash	6
Allow rfc5952 "[2001:db8::1]:80" ipv6 notation	5
`onnode` improvements	4
Fix `ctdb disable` race (BZ14513, 2020-09) (still not...)	3
Other	68
Total	252

Elections, cluster lock and banning

- Cluster recoveries use timeout heuristic
- What if cluster filesystem recovers slower than CTDB?
- 1 second, 5 seconds, 2 minutes?
- New ‘recovery master’ elected
- Attempts to take ‘recovery lock’
- Fails
- Bans itself
- Recovery extended by ≤ ban time (i.e. 5 minutes)
- Up to 5 minutes, by default

Elected before connected

- Cluster election occurs before connection to other nodes
- Node elects itself leader of 1-node cluster
- Can't take cluster lock, bans itself, ...
- Sleep to give other nodes time to join?
- Cluster recoveries use timeout heuristic!
- Add leader broadcast
- **Leader**: `ctdb leader`, ...
- At startup, wait for leader broadcast timeout
- Timeout? Start an election!
- A lot less elections...

Cluster lock

- Can now safely take the ‘recovery lock’ after election win
- **Cluster lock**:
  - `[cluster] -> cluster lock`
  - `ctdb reclock` (‘really excellent’!)

A tale of 2 elections

- 2 elections?
  - Traditional election: ‘best’ is winner
  - Leader must hold cluster lock
- New leader can’t take cluster lock?
- Banning time!
- Or some weak heuristic. . .
- What if only 1 node can take cluster lock?
- Potentially ban all nodes except 1. . .
- **Election via race for cluster lock**

What's that? We have a leader broadcast?

- When to trigger election?
- Lots of *ad hoc* leader sanity-checks...
- We have a leader broadcast!
- Leader broadcast only used at startup?
- Use leader broadcast to trigger all elections
- Carefully design leader broadcast and timeouts...
- Throw away *ad hoc* leader sanity-checks
- Cluster recoveries use timeout heuristic!!!

Election loose-ends

- Traditional election: don't trigger if can't be leader (patch✓)
- Traditional election: sequence numbers to avoid out-of-order
- Still setting recovery mode at start of election
  - Restricts database attach
  - Necessary?
  - Probably not...
  - Need to ensure correctness of recovery trigger code
- Leader broadcast timeout (5s) is much less than inter-node timeout (25s)
  - Hold leader to higher standard
  - No problem, yet...
  - If problematic then avoid flagging recovery

ctdbd_wrapper is gone

- CTDB_STARTUP_TIMEOUT not handled by config_migrate.sh
- CTDB_STARTUP_TIMEOUT used to avoid failure on unknown tunables
- Tunables will go away in future...
- ...but we really hate ctdbd_wrapper!
- Load tunables in daemon, directly from file, instead of in event script
- Drop ctdbd_wrapper

Fixing the `ctdb disable` race

- Occasional test failures for many years...
- ...due to `ctdb disable` and `ctdb enable` failures
- Commands used generic flag update controls
- Amitay: 'This can't be fixed without dedicated controls'
- Several attempts to fix (without dedicated controls) (BZ14466, BZ14513)
- Test failures continued...
- This time for sure!
- Oh! I see! You can't fix this without dedicated controls!
- Fixed by adding dedicated controls (BZ14784)
- Mental note: listen to Amitay

Log reopening on `SIGHUP`

- BZ6595, reported 2009-07-31, need `max log size`
- syslog support added, so no big deal?
- How about supporting log file rotation?
- Use Samba's file logging instead of custom CTDB callback?
- Samba's file logging not atomic:
  - 1 write for header
  - 1 write for message
- Make it atomic
- Add a syslog-like logging format
  - We merge logs from cluster by sorting
  - So, RFC5424 timestamps are preferred
- Plumb into `ctdbd` and chain into other processes
- BZ6595 still open, really wanted `max log size`, maybe... (procrastination: PoC patch✓)

Queue

Avoid reporting remote nodes UNHEALTHY at startup

- `ctdbd` marks remote nodes as UNHEALTHY at startup
- Avoids flapping if node is actually unhealthy...
- ...but causes momentary flapping if node is healthy
- UNKNOWN state was considered long ago
- Implementing UNKNOWN easier with recent improvements
- Amitay:
  - 'The code is better, but you'll still break something'
  - 'What about ...?'
  - 'Fake it in the tool, based on run-state...'
- Vinit Agnihotri and I have implemented support for fake CONNECTED state

Other branches

ctdb-ban Clean up banning code in recovery daemon, avoid use of `struct ctdb_context`, important detangling

ctdb-cluster-lock-io Cluster lock tests health of cluster filesystem using blocking I/O, attempts to release if blocked

ctdb-nfs Impossible to properly disable `rquotad` with kernel NFS

ctdb-prio Use elevated nice setting instead of real-time scheduling

ctdb-recoverd Capabilities clean-up, similar to banning clean-up; VNN map validation clean-up (unfinished)

ctdb-tunnel Tunnel generalisation, planning to use this to write duplicate transport client library...

Plans

Separate daemons

- event daemon
- service daemon
- failover daemon + connection tracking daemon
- cluster daemon
- database daemon
- transport
- smbd proxy
- ...

Incremental progress?

- Retro-fitting current `ctdbd` to use new transport API is not a sane option
- Should we implement the transport API against existing `ctdbd` (i.e. use existing `ctdbd` as transport)?
- This is churn but potentially lets us get some of our work into a release before everything is finished
- Maybe this is worth doing... ~~Maybe this is worth doing...~~
- I plan on doing this

Questions?