Implementing SMB semantics in a Linux cluster

2020 Linux Storage and Filesystems Conference
Santa Clara, CA

Volker Lendecke

Samba Team / SerNet

2020-02-25

Who am I?

- Co-Founder of SerNet in Göttingen, Germany
- First Samba patches in 1994
- Early Samba Team member
- Samba infrastructure (tdb, tevent, etc)
- File server
- Clustered Samba
- Winbind
- AD controller is my colleague Stefan Metzmacher's domain
  - Stefan implemented AD multi-master replication in Samba

What is Samba?

- www.samba.org: Samba is the standard Windows interoperability suite of programs for Linux and Unix
- Server- and Client-Implementation of the Server Message Block (SMB) protocol
  - SMB is **the** Windows protocol to share drives across the network
  - Comparable to NFS (NFSv4 RFCs feels very familiar)
- Print server for Windows clients
- Active Directory domain member
  - Make Active Directory users and groups available on Linux
- Active Directory domain controller
  - Provide user database for Windows and Unix clients

What is SMB?

- "Server Message Block"
  - Started in the 1980s, developed until today
  - Since EU verdict (2007?) well documented
- SMB semantics: single-tasking DOS "on the wire"
  - Every application by definition had exclusive file access
  - SHARE.EXE maintained illusion by blocking concurrent access
- Network-aware applications could explicitly permit sharing per open
- Posix opens only have to read metadata
  - Permissions, file location etc
- Inherent scalability problem through share modes
  - SMB opens need to examine all other opens

Samba architecture

- For every client Samba forks a new process
  - Distinct memory space for every process
- Spec (MS-SMB2/MS-FSA) suggests a lot of shared tables
  - Lists of clients, open files, lots more
- Samba can't use any of those data structures directly
- Samba shares data structures via shared key/value stores
  - TDB is a memory-mapped hash table
  - Protection via fcntl locks or shared mutexes
- TDB provides a clean separation layer
  - This made clustering initially possible
  - Process separation extended to nodes

SMB share modes and leases

- Share Modes (a.k.a Share Reservations)
  - Every open call requests access permissions
    - READ, WRITE or DELETE (among others)
  - Every open call allows other permissions
    - Concurrent READ, WRITE or DELETE permitted
  - First come, first serve
  - NFS4 does not have DELETE
- Oplocks / Leases (a.k.a. Delegations)
  - Cache coherency protocol, per-file granularity
- Interoperability with NFS highly welcome
  - Linux fcntl F_SETLEASE and flock don't match SMB semantics
  - Samsung's in-kernel SMB server needs this as well

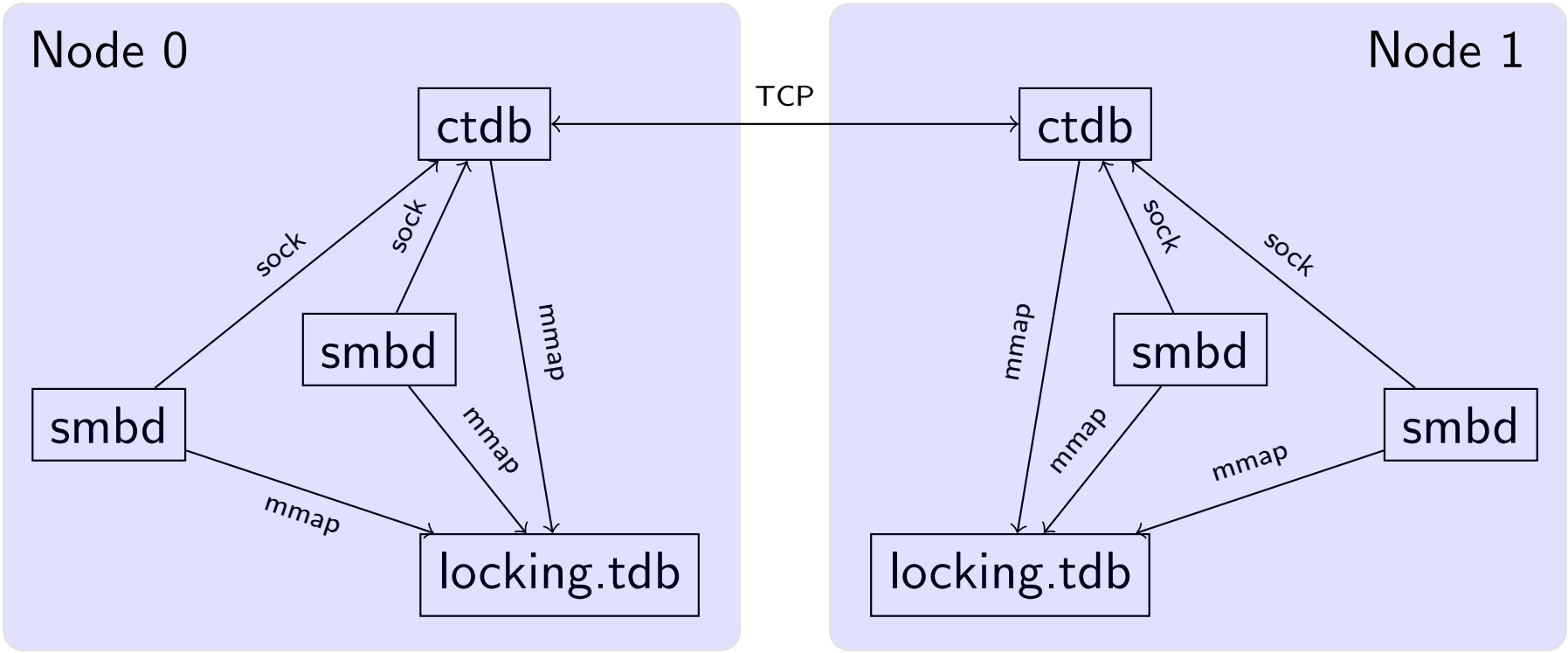
Implementation of SMB locking

- One locking.tdb record per inode
  - Metadata: File name, delete token, time stamps
  - One share_mode_entry per fd
  - One share_mode_lease per lease key (leases shared across fds)
- Open a file
  - Walk the share_mode_entry array, on conflict return NT_STATUS_SHARING_VIOLATION
  - Look at the share_mode_lease array
  - On conflict, send a message to lease holding process
  - Lease holder will "break the lease" with the client
- Close a file
  - Clean up, inform potential lease breakers
- Problem: There can be LOTS of open handles on an inode

Clustered TDB ctdb

- ctdb extends tdb files beyond a single machine
- ctdbd is a daemon to move records around
  - smbd requesting a record gets a local copy
  - ctdb maintains the most recent record location
- locking.tdb can be lossy
  - Share mode state valid only for open file handles
  - A crashed node's file handles are closed by definition
  - Samba deals with crashed processes since day one
- ctdb record access is like NUMA with extreme node distance
- More services by ctdbd:
  - Cluster membership
  - Remote messaging transport
  - Remote process_exists() API

ctdb Architecture



Scalability work on progress

- Avoid walking the share mode array
- Share mode conflict:
  - I want to write, but someone else did not grant FILE_SHARE_WRITE
  - I don't grant FILE_SHARE_WRITE, but someone already writes
  - Same for READ and DELETE, First come, first serve
- Central flags field to hold most restrictive share mode
  - Intersection of all share modes granted
  - Union of all granted access
- Opening a file just checks the per-file summary
- If there's a conflict, recalculate the truth
- Share mode array exists in a separate TDB file
  - Handling much more efficient than before
  - Roughly factor 100 for specific tests

Next steps

- Move share_entries.tdb back into locking.tdb
  - Non-contended file access got slower (3 instead of 2 records)
  - Now that the logic works, we can optimize data structures
- Base locking.tdb on g_lock.tdb technology
  - Avoid tdb locks while doing open/close/unlink/rename etc
  - Improve parallelism, reduce contention
  - Enable ctdb recovery while cluster file system is stuck
- Spread locking.tdb across per-node per-inode records
  - Parallel case (no share mode conflicts) only looks at one record
  - Conflicting case must take all records into account

Questions?

vl@samba.org / vl@sernet.de
http://www.sambaxp.org/