Clustered Samba Scalability Improvements

SNIA SDC 2019
Santa Clara

Volker Lendecke

Samba Team / SerNet

2019-09-23

Samba architecture

- For every client Samba forks a new process
- Distinct memory spaces in every process
- MS-SMB2 and MS-FSA suggest a lot of shared tables
  - Lists of clients, tree connects, open files
- 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

SMB history

- SMB semantics date back to DOS single-user OS
  - Every application by definition had exclusive file access
- SHARE.EXE maintained illusion by blocking concurrent access
- Network-aware applications could explicitly permit sharing
  - Different modes of access permitted on a per-open basis
- 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

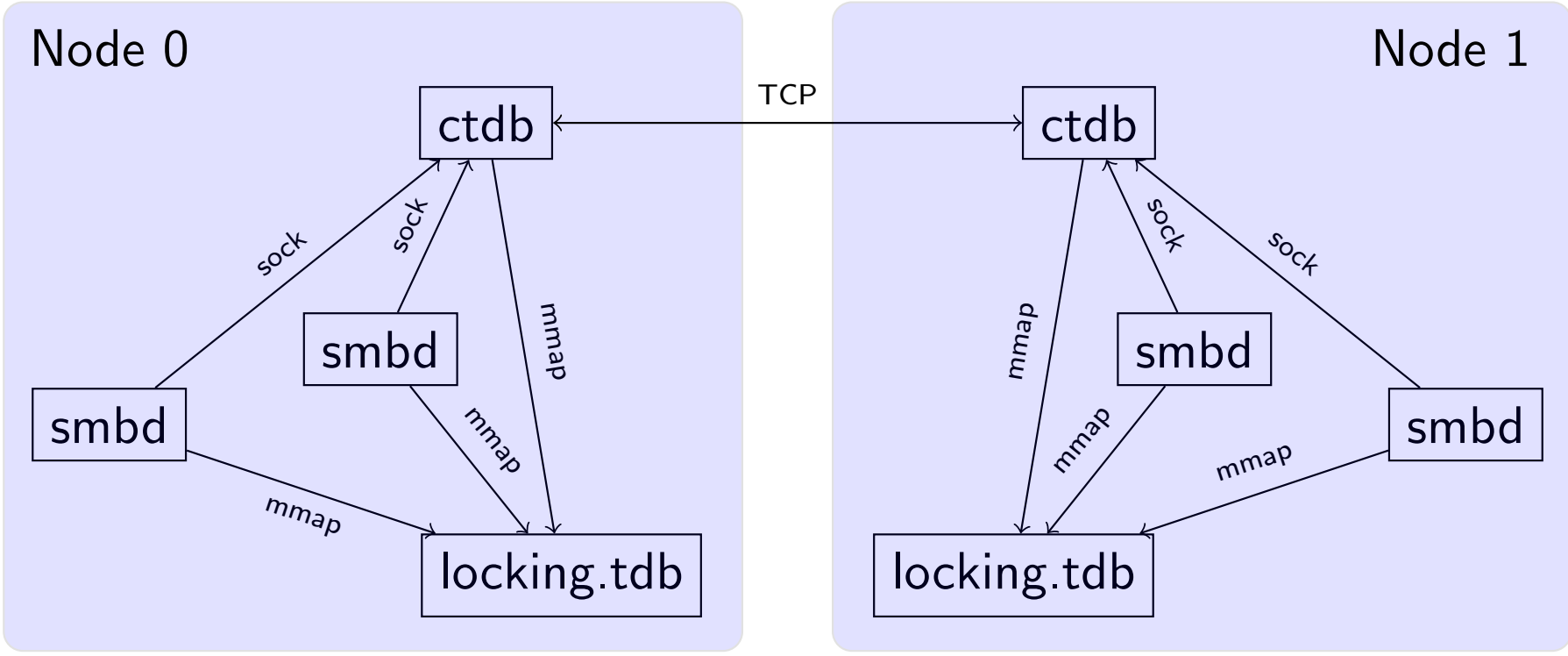
SMB share modes

- 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
- Samba stores an array of shareing information per inode in locking.tdb
- That array is costly: NDR marshalling can be top CPU consumer
  - OEMs have written their own marshalling, our NDR is not super-optimized
- Restructure data structures to eliminate array NDR marshalling

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
- ctdb record access is like NUMA with extreme node distance

ctdb Architecture



Cleanup in share_mode_data

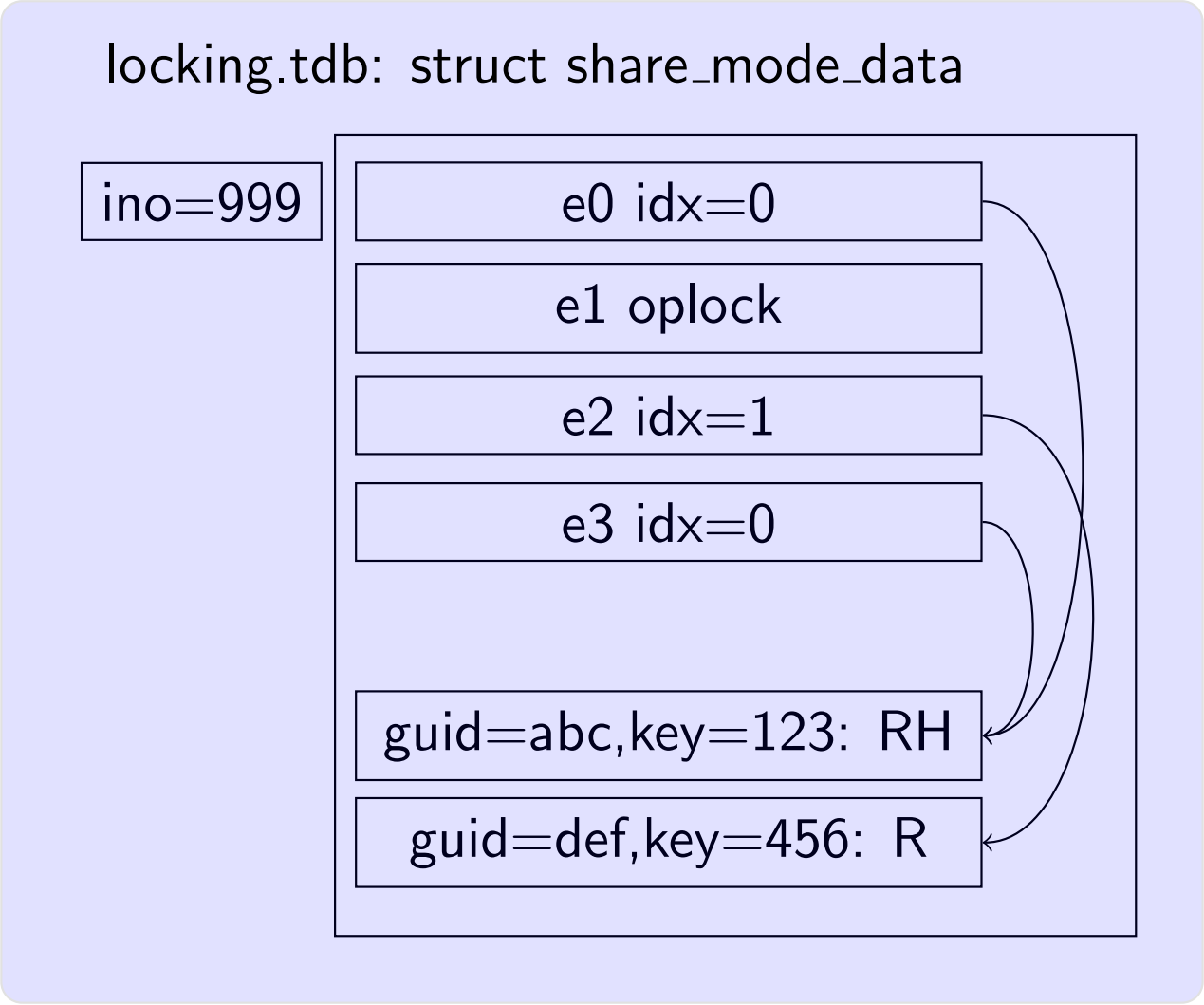
- One share_mode_data record in locking.tdb per inode

```
share_mode_entry share_modes[];
share_mode_lease leases[];
```

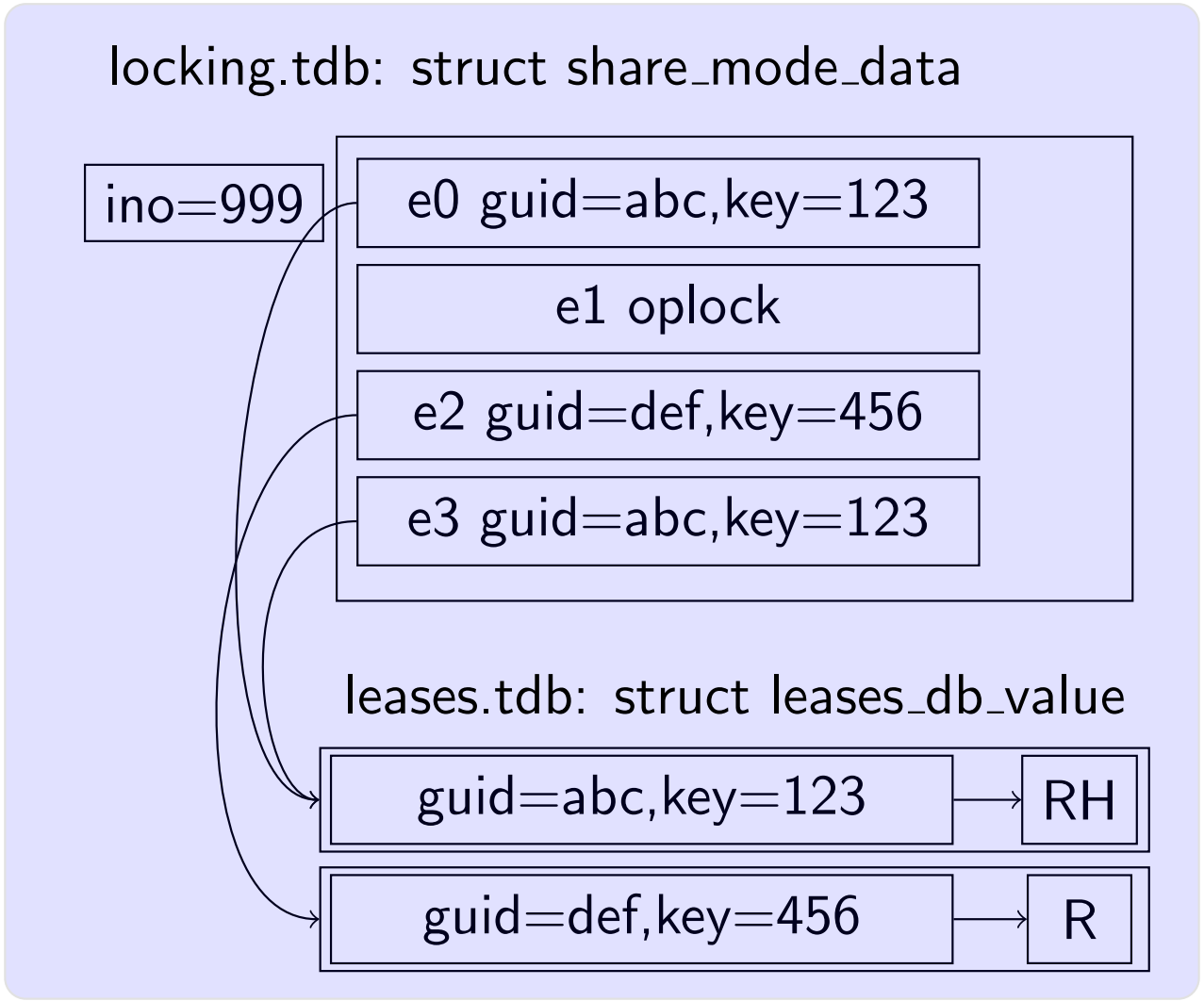
- Every share_mode_entry represents an open handle on a file
- share_mode_entry→lease_idx references the lease array
- struct share_mode_lease:

```
GUID client_guid;
smb2_lease_key lease_key;
smb2_lease_state current_state;
```

4.10 locking.tdb



4.11 locking.tdb



Byte Range Locking cleanup in 4.11

- Write requests break read leases
  - Every write request must query read leases
  - Reading locking.tdb entries is expensive
  - For mandatory locking, brlock.tdb is required per write
- brlock.tdb holds number of read leases
  - Locking order violation locking.tdb is master, everything else locked after it
  - breaking read leases was done with clumsy async callback
- 4.11: Small abstraction violation instead of locking order violation
  - locking.tdb records start with a flags field about read lease presence
  - Lazy update: Adding R lease sets the flag, walking for lease break resets
  - Test run for larger lazy update share mode calculation
- While there, locking&x handling moved from core to SMB1 code

Separating the share_modes array

- Roughly 30 places in Samba 4.11 reference share_mode_data→share_modes[]
- Most of the references walk the array
  - Lease breaks, durable file handling, file rename notification, etc
- Introduce share_mode_forall_entries() with a callback
- Introduce share_entries.tdb with sorted share_mode_entry arrays
  - share_mode_entry is fixed size, no variable components
  - Finding a record with binary search
  - Closing a file down from $O(N)$ to $O(log(N))$
  - Opening still $O(N)$

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
- Byte range locking cleanup introduced a 1-bit flags field
  - SHARE_MODE_HAS_READ_LEASE
- Extend that 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

Demo Time

DEMO

Questions?

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