CTDB

Andrew Tridgell

(and Volker, Ronnie, Jim, Sven, Peter ....)

Clustering? What's that?

- Want to use N compute nodes
  - each node has its own local storage and memory
  - a 'fast' network is available (of the order of microsecond latency)
- Shared filesystem
  - all nodes also have access to a shared filesystem
  - shared filesystem is assumed data coherent, but may be slow at some operations (like locking)

Scaling

- Positive Scaling
  - we want N+1 nodes to perform better than N nodes, for some range of N
  - we want 2 nodes to perform better than best non-clustered approch for a single node
- Cluster size
  - in practice, we are aiming for clusters up to approximately 100 nodes

Protocol Coherence

- SMB protocol has strong coherence constraints
  - all read/write calls use mandatory locking
  - file operations are strongly ordered
- Not like NFS
  - NFS servers and clients commonly assume that if meta data is recent it is still valid

Current Architecture

- Multi-process server
    - multiple smbd daemons, one per client
    - each daemon attached to a number of databases
    - databases store all shared meta-data
- Clustering this should be easy!
    - why not just use a cluster database?
    - each smbd talks to cluster database instead of local database
    - obvious solutions can be wrong :)

Samba Databases

- Lots of small databases
  - Total of about 20 in normal install
- Most performance sensitive:
  - byte range locking
  - open files
  - messaging

Current TDB

- key-value database
  - similar in concept to berkley db
  - records have a single binary key
  - records are binary blobs
- very fast
  - uses shared memory (mmap)
  - fcntl byte range locking for coherence
  - often achieves 100k to 500k operations/second

Precious data?

- What data does a normal clustered database preserve when a node dies?
    - all of it!
- How is this achieved?
    - all data must either be on all nodes, or on stable shared storage
    - this means that all write operations must be VERY SLOW
- What about clustered filesystems?
    - same constraints, same problem

Losing Data Safely

- Can clustered Samba survive data loss?
  - yes!
  - but only the right data ....
- Safe to lose
  - If node N goes down, we can lose data associated with open connections on node N
  - open files, locks, messages to node N
- Data Recovery
  - data stored on node N but not associated with node N can be recovered from other nodes

Remote Locking

- Normal pattern in a cluster
  - get lock on data
  - perform operation
  - possibly update
  - release lock on data
- Remote data
  - when data is remote, this makes for an inherant bottleneck
  - Remote locking is evil!
- Solution?
  - send the function to the data
  - never hold a lock during a network operation

CTDB API

- RPC-like API
  - 'calls' are like database stored procedures
  - all calls are associated with a data record
  - a call receives call data and record data
  - can return arbitrary data, plus update record
- fetch_lock API
  - fetches a locked record

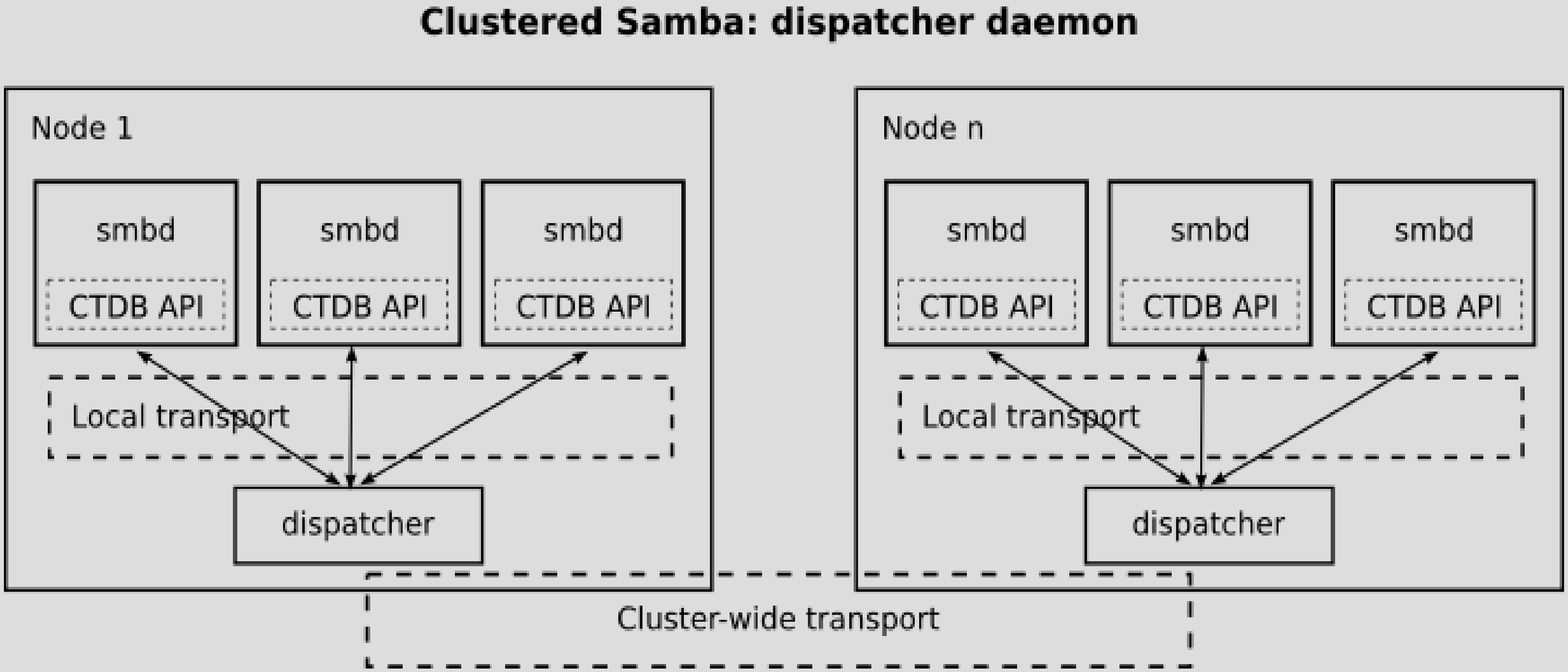
```
struct ctdb_context *ctdb_init(struct event_context *ev);
int ctdb_set_transport(struct ctdb_context *ctdb, const char *transport);
int ctdb_set_call(struct ctdb_context *ctdb, ctdb_fn_t fn, int id);
int ctdb_call(struct ctdb_db_context *ctdb_db, struct ctdb_call *call);
                void *ctdb_fetch_lock(struct ctdb_db_context *ctdb_db,
                                        TALLOC_CTX *mem_ctx, TDB_DATA key,
                  TDB_DATA *data);

(API has been simplified for this slide)
```

CTDB architecture

- Clustered TDB
    - each node uses a local tdb (ltdb) for storage
    - ltdb is in memory, or local storage
- LMASTER
    - LMASTER == location master
    - location master knows where a record is stored
- DMASTER
    - DMASTER == data master
    - data master holds data for a record
- Backends
    - TCP and Infiniband backends
    - async, event driven API

Dispatcher Daemon



Record Migration

- LMASTER fixed
  - LMASTER is based on record key only
  - LMASTER knows where the record is stored
  - new records are stored on LMASTER
- DMASTER moves
  - DMASTER owns data for a record
  - remote call can trigger a DMASTER move
  - N consecutive requests by the same node causes DMASTER move to that node

fetch_lock

- Wanted to avoid this, but couldn't :-(
  - fetches a locked record
  - store/unlock operations to complete
  - built on top of ctdb_call, with special migration flag
- Needed for
  - fitting with Samba3 clustering model
  - used in open database in Samba4

ltdb shortcut

- shortcut for direct tdb access
  - 1) get record chainlock
  - 2) check if we are the dmaster
  - 3) if dmaster, then operate locally, with lock held
  - 4) if not dmaster, then need to talk to ctdb daemon via unix domain socket
- local-equivalent speed
  - result is that non-contended access runs at same speed as non-clustered operation

Scaling Results

- NBENCH test
  - 16 clients
  - 1 to 4 nodes

OLD (pre-CTDB) approach

1 node	30.0 Mbytes/sec
2 nodes	2.1 MBytes/sec
3 nodes	1.8 MBytes/sec
4 nodes	1.8 MBytes/sec

NEW (CTDB) approach

1 node	42 Mbytes/sec
2 nodes	168 MBytes/sec
3 nodes	211 MBytes/sec
4 nodes	243 MBytes/sec

Demo!

- early days, but it does work!
  - 4 nodes
  - ctdb used for byte range locking, messaging and open files database
  - works with both Samba3 and Samba4
  - testing with smbtorture tests

Questions?

- For more information on CTDB see

  http://wiki.samba.org/index.php/Samba_%26_Clustering