CTDB Project

**Motivation:** Support for clustered Samba
- Multiple nodes active simultaneously
- Communication between nodes (heartbeat, failover)
- Share databases between nodes

**Features:**
- Volatile and Persistent databases
- IP failover and load balancing
- Service monitoring

**Community:**
- http://ctdb.samba.org
- git://git.samba.org/ctdb.git,
  git://git.samba.org/samba.git
Merging CTDB tree in Samba tree

Development Stories
- High hopcount bug
- Getting lock scheduling right
- All nodes banned on single node failure

Regression Stories
- Real time or not
- Fixing compiler warnings
Story of the Merge

SambaXP 2013

Merge CTDB in Samba tree?
Replace duplication of talloc, tdb, tevent, replace libraries
Autobuild testing of clustered Samba
Leverage off Samba release process
Attract more developers

Nov 2013
CTDB tree merged with Samba

SambaXP 2014

To Do
Create waf build for CTDB, Clustered Samba
Setting up clustered samba instance for autobuild
Split monolithic code

Amitay Isaacs
CTDB Stories
Story of the Merge

SambaXP 2013

- Merge CTDB in Samba tree?
  - Remove duplication of talloc, tdb, tevent, replace libraries
  - Autobuild testing of clustered Samba
  - Leverage off Samba release process

CTDB Stories

Nov 2013

CTDB tree merged with Samba
Story of the Merge

SambaXP 2013

- Merge CTDB in Samba tree?
  - Remove duplication of malloc, tdb, tevent, replace libraries
  - Autobuild testing of clustered Samba
  - Leverage off Samba release process
  - Attract more developers

Nov 2013

CTDB tree merged with Samba

SambaXP 2014

To Do

- Create waf build for CTDB, Clustered Samba
- Setting up clustered Samba instance for autobuild
- Split monolithic code
SambaXP 2013

- Merge CTDB in Samba tree?
  - Remove duplication of talloc, tdb, tevent, replace libraries
  - Autobuild testing of clustered Samba
  - Leverage off Samba release process
  - Attract more developers

Nov 2013

CTDB tree merged with Samba
Story of the Merge

**SambaXP 2013**
- Merge CTDB in Samba tree?
  - Remove duplication of talloc, tdb, tevent, replace libraries
  - Autobuild testing of clustered Samba
  - Leverage off Samba release process
  - Attract more developers

**Nov 2013**
CTDB tree merged with Samba

**SambaXP 2014**
- To Do
  - Create waf build for CTDB, Clustered Samba
  - Setting up clustered samba instance for autobuild
  - Split monolithic code
Story of the Merge

Step 1
Convert CTDB autoconf build to waf build
Finished implementation before reaching Australia

Step 2
Integrate CTDB build into toplevel build
lib/util has diverged
Can't get rid of ctdb/lib/util
Start hacking lib/util
Gave up! Too long for a plane trip.

June 2014
CTDB standalone waf build committed.

Amitay Isaacs
CTDB Stories
Step 1

- Convert CTDB autoconf build to waf build
Story of the Merge

Step 1
- Convert CTDB autoconf build to waf build
- Finished implementation before reaching Australia

June 2014
CTDB standalone waf build committed.

Amitay Isaacs
Story of the Merge

Step 1
- Convert CTDB autoconf build to waf build
  - Finished implementation before reaching Australia

Step 2
- Integrate CTDB build into toplevel build
Story of the Merge

Step 1
- Convert CTDB autoconf build to waf build
  - Finished implementation before reaching Australia

Step 2
- Integrate CTDB build into toplevel build
  - lib/util has diverged
Step 1
- Convert CTDB autoconf build to waf build
  - Finished implementation before reaching Australia

Step 2
- Integrate CTDB build into toplevel build
  - lib/util has diverged
  - Can’t get rid of ctdb/lib/util

June 2014
CTDB standalone waf build commited.

Amitay Isaacs
CTDB Stories
Step 1

- Convert CTDB autoconf build to waf build
  - Finished implementation before reaching Australia

Step 2

- Integrate CTDB build into toplevel build
  - lib/util has diverged
  - Can’t get rid of ctdb/lib/util
  - Start hacking lib/util

CTDB Stories
Story of the Merge

Step 1
- Convert CTDB autoconf build to waf build
  - Finished implementation before reaching Australia

Step 2
- Integrate CTDB build into toplevel build
  - lib/util has diverged
  - Can’t get rid of ctdb/lib/util
  - Start hacking lib/util
  - Gave up! Too long for a plane trip.

June 2014
CTDB standalone waf build committed.

Amitay Isaacs
CTDB Stories
Story of the Merge

Step 1
- Convert CTDB autoconf build to waf build
  - Finished implementation before reaching Australia

Step 2
- Integrate CTDB build into toplevel build
  - lib/util has diverged
  - Can’t get rid of ctdb/lib/util
  - Start hacking lib/util
  - Gave up! Too long for a plane trip.

June 2014
CTDB standalone waf build committed.
Story of the Merge

Martin takes over

Remove dependency on includes.h

Untangle functions & dependencies . . .

idtree.c depends on lib/crypto

util.c depends on charset

Factor out samba-util-core from samba-util to avoid pulling in non-library code.

Clean up ctdb/lib/util

Clean up CTDB logging

Create new subsystem ctdb-util

Drop CTDB log ringbuffer, adopt lib/util/debug.[ch]

Replace dependency on ctdb-util with samba-util

Hook CTDB into top level using --with-cluster-support
Story of the Merge

Martin takes over
Story of the Merge

Martin takes over

- Remove dependency on includes.h
Story of the Merge

Martin takes over

- Remove dependency on includes.h
- Untangle functions & dependencies ...
  - idtrea.c depends on lib/crypto
  - util.c depends on charset
Martin takes over

- Remove dependency on includes.h
- Untangle functions & dependencies ...
  - idtree.c depends on lib/crypto
  - util.c depends on charset
- Factor out samba-util-core from samba-util to avoid pulling in non-library code.
Martin takes over

- Remove dependency on includes.h
- Untangle functions & dependencies ...
  - idtree.c depends on lib/crypto
  - util.c depends on charset
- Factor out samba-util-core from samba-util to avoid pulling in non-library code.
- Clean up ctdb/lib/util
**Martin takes over**

- Remove dependency on includes.h
- Untangle functions & dependencies ...  
  - idtree.c depends on lib/crypto  
  - util.c depends on charset
- Factor out samba-util-core from samba-util to avoid pulling in non-library code.
- Clean up ctdb/lib/util
- Clean up CTDB logging
Story of the Merge

Martin takes over

- Remove dependency on includes.h
- Untangle functions & dependencies ...
  - idtree.c depends on lib/crypto
  - util.c depends on charset
- Factor out samba-util-core from samba-util to avoid pulling in non-library code.
- Clean up ctdb/lib/util
- Clean up CTDB logging
- Create new subsystem ctdb-util
Story of the Merge

Martin takes over

- Remove dependency on includes.h
- Untangle functions & dependencies ...
  - idtree.c depends on lib/crypto
  - util.c depends on charset
- Factor out samba-util-core from samba-util to avoid pulling in non-library code.
- Clean up ctdb/lib/util
- Clean up CTDB logging
- Create new subsystem ctdb-util
- Drop CTDB log ringbuffer, adopt lib/util/debug.[ch]
Story of the Merge

Martin takes over

- Remove dependency on includes.h
- Untangle functions & dependencies...
  - idtree.c depends on lib/crypto
  - util.c depends on charset
- Factor out samba-util-core from samba-util to avoid pulling in non-library code.
- Clean up ctdb/lib/util
- Clean up CTDB logging
- Create new subsystem ctdb-util
- Drop CTDB log ringbuffer, adopt lib/util/debug.[ch]
- Replace dependency on ctdb-util with samba-util
Martin takes over

- Remove dependency on `includes.h`
- Untangle functions & dependencies …
  - `idtree.c` depends on `lib/crypto`
  - `util.c` depends on `charset`
- Factor out `samba-util-core` from `samba-util` to avoid pulling in non-library code.
- Clean up `ctdb/lib/util`
- Clean up CTDB logging
- Create new subsystem `ctdb-util`
- Drop CTDB log ringbuffer, adopt `lib/util/debug.[ch]`
- Replace dependency on `ctdb-util` with `samba-util`
- Hook CTDB into top level using `--with-cluster-support`
November 2014

CTDB build integrated into toplevel build.
November 2014
CTDB build integrated into toplevel build.
CTDB Releases

- **2.5.4 (September 2014) - 156 patches**
  - Support for TDB robust mutexes
  - Add ctdb detach
  - Avoid running ctdb helpers at real-time priority
  - Improved vacuuming performance

- **2.5.5 (April 2015) - 119 patches**
  - Fix handling of IPv6 addresses
  - Fix regression in socket handling code
  - Make statd-callout scalable
Developers

Contributions in 2014

196  Martin Schwenke
184  Amitay Isaacs
 55  Michael Adam
 10  Volker Lendecke
   3  Srikrishan Malik
   3  Andrew Bartlett
   2  Stefan Metzmacher
   2  Gregor Beck
   2  Bjorn Baumbach
   1  Matthias Dieter Wallnofer
   1  Jeremy Allison
   1  Ira Cooper
   1  David Disseldorp
Developers

Contributions since Jan 2015

118 Martin Schwenke
15 Amitay Isaacs
12 Volker Lendecke
 3 Rajesh Joseph
 1 Michael Adam
 1 Led
 1 Jelmer Vernooij
 1 David Disseldorp
 1 Christof Schmitt
High hopcount bug

Problem
Logs filled with entries like:
ctdbd: High hopcount 2823099 dbid:0x7a19d84d key:0x6f9f65c4

static void ctdb_call_send_redirect(ctdb, ctdb_db, key, c, header)
{
    uint32_t lmaster = ctdb_lmaster(ctdb, &key);
    c->hdr.destnode = lmaster;
    if (ctdb->pnn == lmaster) {
        c->hdr.destnode = header->dmaster;
    }
    c->hopcount++;
    if (c->hopcount%100 > 95) {
        DEBUG(DEBUG_WARNING,("High hopcount ...",));
    }
    ctdb_queue_packet(ctdb, &c->hdr);
}

Amitay Isaacs
CTDB Stories
High hopcount bug

Problem

Logs filled with entries like:
ctdbd: High hopcount 2823099 dbid:0x7a19d84d key:0x6f9f65c4
High hopcount bug

**Problem**

Logs filled with entries like:

```
ctdbd: High hopcount 2823099 dbid:0x7a19d84d key:0x6f9f65c4
```

```c
static void ctdb_call_send_redirect(ctdb, ctdb_db, key, c, header)
{
    uint32_t lmaster = ctdb_lmaster(ctdb, &key);

    c->hdr.destnode = lmaster;
    if (ctdb->pnn == lmaster) {
        c->hdr.destnode = header->dmaster;
    }
    c->hopcount++;

    if (c->hopcount%100 > 95) {
        DEBUG(DEBUG_WARNING,("High hopcount ..."));
    }

    ctdb_queue_packet(ctdb, &c->hdr);
}
```
High hopcount bug

Record Migration

- Node 0
- Node 1 (LMASTER)
- Node 2 (DMASTER)

- Record: Node 1 is LMASTER, Node 2 is DMASTER
High hopcount bug

Record Migration

- Node 0
- Node 1: LMASTER
- Node 2: DMASTER

- Record: Node 1 is LMASTER, Node 2 is DMASTER
- Request for record received on Node 0 (REQ_CALL)
Record Migration

- Record: Node 1 is LMASTER, Node 2 is DMASTER
- Request for record received on Node 0 (REQ_CALL)
- Request redirected to Node 1 (REQ_CALL)
High hopcount bug

Record Migration

- Record: Node 1 is LMASTER, Node 2 is DMASER
- Request for record received on Node 0 (REQ_CALL)
- Request redirected to Node 1 (REQ_CALL)
- Request redirected to Node 2 (REQ_CALL)
### Record Migration

- **Record:** Node 1 is LMASTER, Node 2 is DMASKER
- **Request for record received on Node 0** (REQ_CALL)
- **Request redirected to Node 1** (REQ_CALL)
- **Request redirected to Node 2** (REQ_CALL)
- **Reply to Node 1** (DMASKER_REQ)
High hopcount bug

Record Migration

- Record: Node 1 is LMASTER, Node 2 is DMASTER
- Request for record received on Node 0 (REQ_CALL)
- Request redirected to Node 1 (REQ_CALL)
- Request redirected to Node 2 (REQ_CALL)
- Reply to Node 1 (DMASTER_REQ)
- Reply to Node 0 (DMASTER_REPLY)
Record Migration

- Record: Node 1 is LMASTER, Node 2 is DMASKER
- Request for record received on Node 0 (REQ_CALL)
- Request redirected to Node 1 (REQ_CALL)
- Request redirected to Node 2 (REQ_CALL)
- Reply to Node 1 (DMASKER_REQ)
- Reply to Node 0 (DMASKER_REPLY)
- Reply to Client (REPLY_CALL)
High hopcount bug

Debugging

noticed after fixes for vacuuming/recovery interaction bug. The problem was hard to reproduce. Many times the problem resolved itself. Suspects: Two requests chasing each other. Record header corruption. Fixes for vacuuming/recovery interaction bug. Did identify few issues in the fixes. However, the problem did not go away. Locking code was being modified.
High hopcount bug

Debugging

- Noticed after fixes for vacuuming/recovery interaction bug

The problem was hard to reproduce
Many times the problem resolved itself
Suspects
- Two requests chasing each-other
- Record header corruption
- Fixes for vacuuming/recovery interaction bug

Did identify few issues in the fixes
However, the problem did not go away
Locking code was being modified
Debugging

- Noticed after fixes for vacuuming/recovery interaction bug
- The problem was hard to reproduce
High hopcount bug

Debugging

- Noticed after fixes for vacuuming/recovery interaction bug
- The problem was hard to reproduce
- Many times the problem resolved itself
# High hopcount bug

## Debugging
- Noticed after fixes for vacuuming/recovery interaction bug
- The problem was hard to reproduce
- Many times the problem resolved itself

## Suspects
- Two requests chasing each-other
Debugging
- Noticed after fixes for vacuuming/recovery interaction bug
- The problem was hard to reproduce
- Many times the problem resolved itself

Suspects
- Two requests chasing each-other
- Record header corruption
High hopcount bug

Debugging
- Noticed after fixes for vacuuming/recovery interaction bug
- The problem was hard to reproduce
- Many times the problem resolved itself

Suspects
- Two requests chasing each-other
- Record header corruption
- Fixes for vaccuming/recovery interaction bug
High hopcount bug

Debugging
- Noticed after fixes for vacuuming/recovery interaction bug
- The problem was hard to reproduce
- Many times the problem resolved itself

Suspects
- Two requests chasing each-other
- Record header corruption
- Fixes for vacuuming/recovery interaction bug
  - Did identify few issues in the fixes
High hopcount bug

**Debugging**
- Noticed after fixes for vacuuming/recovery interaction bug
- The problem was hard to reproduce
- Many times the problem resolved itself

**Suspects**
- Two requests chasing each-other
- Record header corruption
- Fixes for vacuuming/recovery interaction bug
  - Did identify few issues in the fixes
  - However, the problem did not go away
High hopcount bug

Debugging

- Noticed after fixes for vacuuming/recovery interaction bug
- The problem was hard to reproduce
- Many times the problem resolved itself

Suspects

- Two requests chasing each-other
- Record header corruption
- Fixes for vacuuming/recovery interaction bug
  - Did identify few issues in the fixes
  - However, the problem did not go away
- Locking code was being modified
High hopcount bug

Instrument record request processing code

Node 1 is the DMASTER for a record (hash 0x0aa13d47)

Record is getting updated regularly on Node 1

UPDATE db[notify_index.tdb]: store: hash[0x0aa13d47] rsn[9620] dmaster[1]

UPDATE db[notify_index.tdb]: store: hash[0x0aa13d47] rsn[9621] dmaster[1]

UPDATE db[notify_index.tdb]: store: hash[0x0aa13d47] rsn[9622] dmaster[1]

UPDATE db[notify_index.tdb]: store: hash[0x0aa13d47] rsn[9623] dmaster[1]

Node 0 requests the record. Node 1 updates DMASTER.

UPDATE db[notify_index.tdb]: store: hash[0x0aa13d47] rsn[9640] dmaster[1]

UPDATE db[notify_index.tdb]: store: hash[0x0aa13d47] rsn[9641] dmaster[1]

UPDATE db[notify_index.tdb]: store: hash[0x0aa13d47] rsn[9641] dmaster[0]

And Node 1 migrates the record to Node 0

On Node 0 CTDB tries to grab the record lock

Cannot get a lock in non-blocking mode

Creates a lock request
Instrument record request processing code
High hopcount bug

- Instrument record request processing code
- Node 1 is the DMASTER for a record (hash 0x0aa13d47)
High hopcount bug

- Instrument record request processing code
- Node 1 is the DMASTER for a record (hash 0x0aa13d47)
- Record is getting updated regularly on Node 1

```
UPDATE db[notify_index.tdb]: store: hash[0x0aa13d47] rsn[9620] dmaster[1]
UPDATE db[notify_index.tdb]: store: hash[0x0aa13d47] rsn[9621] dmaster[1]
UPDATE db[notify_index.tdb]: store: hash[0x0aa13d47] rsn[9622] dmaster[1]
UPDATE db[notify_index.tdb]: store: hash[0x0aa13d47] rsn[9623] dmaster[1]
```
High hopcount bug

- Instrument record request processing code
- Node 1 is the DMASTER for a record (hash 0x0aa13d47)
- Record is getting updated regularly on Node 1

UPDATE db[notify_index.tdb]: store: hash[0x0aa13d47] rsn[9620] dmaster[1]
UPDATE db[notify_index.tdb]: store: hash[0x0aa13d47] rsn[9621] dmaster[1]
UPDATE db[notify_index.tdb]: store: hash[0x0aa13d47] rsn[9622] dmaster[1]
UPDATE db[notify_index.tdb]: store: hash[0x0aa13d47] rsn[9623] dmaster[1]

- Node 0 requests the record. Node 1 updates DMASTER.

UPDATE db[notify_index.tdb]: store: hash[0x0aa13d47] rsn[9640] dmaster[1]
UPDATE db[notify_index.tdb]: store: hash[0x0aa13d47] rsn[9641] dmaster[1]
UPDATE db[notify_index.tdb]: store: hash[0x0aa13d47] rsn[9641] dmaster[0]
High hopcount bug

- Instrument record request processing code
- Node 1 is the DMASTER for a record (hash 0x0aa13d47)
- Record is getting updated regularly on Node 1

UPDATE db[notify_index.tdb]: store: hash[0x0aa13d47] rsn[9620] dmaster[1]
UPDATE db[notify_index.tdb]: store: hash[0x0aa13d47] rsn[9621] dmaster[1]
UPDATE db[notify_index.tdb]: store: hash[0x0aa13d47] rsn[9622] dmaster[1]
UPDATE db[notify_index.tdb]: store: hash[0x0aa13d47] rsn[9623] dmaster[1]

Node 0 requests the record. Node 1 updates DMASTER.

UPDATE db[notify_index.tdb]: store: hash[0x0aa13d47] rsn[9640] dmaster[1]
UPDATE db[notify_index.tdb]: store: hash[0x0aa13d47] rsn[9641] dmaster[1]
UPDATE db[notify_index.tdb]: store: hash[0x0aa13d47] rsn[9641] dmaster[0]

And Node 1 migrates the record to Node 0
High hopcount bug

- Instrument record request processing code
- Node 1 is the DMASTER for a record (hash 0x0aa13d47)
- Record is getting updated regularly on Node 1

```
UPDATE db[notify_index.tdb]: store: hash[0x0aa13d47] rsn[9620] dmaster[1]
UPDATE db[notify_index.tdb]: store: hash[0x0aa13d47] rsn[9621] dmaster[1]
UPDATE db[notify_index.tdb]: store: hash[0x0aa13d47] rsn[9622] dmaster[1]
UPDATE db[notify_index.tdb]: store: hash[0x0aa13d47] rsn[9623] dmaster[1]
```

- Node 0 requests the record. Node 1 updates DMASTER.

```
UPDATE db[notify_index.tdb]: store: hash[0x0aa13d47] rsn[9640] dmaster[1]
UPDATE db[notify_index.tdb]: store: hash[0x0aa13d47] rsn[9641] dmaster[1]
UPDATE db[notify_index.tdb]: store: hash[0x0aa13d47] rsn[9641] dmaster[0]
```

- And Node 1 migrates the record to Node 0
- On Node 0 CTDB tries to grab the record lock
High hopcount bug

- Instrument record request processing code
- Node 1 is the DMASTER for a record (hash 0x0aa13d47)
- Record is getting updated regularly on Node 1

```
UPDATE db[notify_index.tdb]: store: hash[0x0aa13d47] rsn[9620] dmaster[1]
UPDATE db[notify_index.tdb]: store: hash[0x0aa13d47] rsn[9621] dmaster[1]
UPDATE db[notify_index.tdb]: store: hash[0x0aa13d47] rsn[9622] dmaster[1]
UPDATE db[notify_index.tdb]: store: hash[0x0aa13d47] rsn[9623] dmaster[1]
```

- Node 0 requests the record. Node 1 updates DMASTER.

```
UPDATE db[notify_index.tdb]: store: hash[0x0aa13d47] rsn[9640] dmaster[1]
UPDATE db[notify_index.tdb]: store: hash[0x0aa13d47] rsn[9641] dmaster[1]
UPDATE db[notify_index.tdb]: store: hash[0x0aa13d47] rsn[9641] dmaster[0]
```

- And Node 1 migrates the record to Node 0
- On Node 0 CTDB tries to grab the record lock
  - Cannot get a lock in non-blocking mode
High hopcount bug

- Instrument record request processing code
- Node 1 is the DMASTER for a record (hash 0x0aa13d47)
- Record is getting updated regularly on Node 1

```
UPDATE db[notify_index.tdb]: store: hash[0x0aa13d47] rsn[9620] dmaster[1]
UPDATE db[notify_index.tdb]: store: hash[0x0aa13d47] rsn[9621] dmaster[1]
UPDATE db[notify_index.tdb]: store: hash[0x0aa13d47] rsn[9622] dmaster[1]
UPDATE db[notify_index.tdb]: store: hash[0x0aa13d47] rsn[9623] dmaster[1]
```

- Node 0 requests the record. Node 1 updates DMASTER.

```
UPDATE db[notify_index.tdb]: store: hash[0x0aa13d47] rsn[9640] dmaster[1]
UPDATE db[notify_index.tdb]: store: hash[0x0aa13d47] rsn[9641] dmaster[1]
UPDATE db[notify_index.tdb]: store: hash[0x0aa13d47] rsn[9641] dmaster[0]
```

- And Node 1 migrates the record to Node 0
- On Node 0 CTDB tries to grab the record lock
  - Cannot get a lock in non-blocking mode
  - Creates a lock request
High hopcount bug

- Meanwhile, more record requests queue up
High hopcount bug

- Meanwhile, more record requests queue up

Waiting reqid: 732  key: 0x0aa13d47
Waiting reqid: 684  key: 0x0aa13d47
Waiting reqid: 715  key: 0x0aa13d47
Waiting reqid: 701  key: 0x0aa13d47
High hopcount bug

- Meanwhile, more record requests queue up

  Waiting reqid:732 key:0x0aa13d47
  Waiting reqid:684 key:0x0aa13d47
  Waiting reqid:715 key:0x0aa13d47
  Waiting reqid:701 key:0x0aa13d47

- Soon after high hopcount messages are logged on Node 0

  High hopcount 97 key:0x0aa13d47 reqid=00004771 pnn:0 src:1 lmaster:1
  High hopcount 99 key:0x0aa13d47 reqid=00004771 pnn:0 src:1 lmaster:1
  High hopcount 196 key:0x0aa13d47 reqid=000039f9 pnn:0 src:0 lmaster:1
  High hopcount 198 key:0x0aa13d47 reqid=000039f9 pnn:0 src:0 lmaster:1
High hopcount bug

- Meanwhile, more record requests queue up
  
  Waiting reqid:732 key:0x0aa13d47
  Waiting reqid:684 key:0x0aa13d47
  Waiting reqid:715 key:0x0aa13d47
  Waiting reqid:701 key:0x0aa13d47

- Soon after high hopcount messages are logged on Node 0
  
  High hopcount 97 key:0x0aa13d47 reqid=00004771 pnn:0 src:1 lmaster:1
  High hopcount 99 key:0x0aa13d47 reqid=00004771 pnn:0 src:1 lmaster:1
  High hopcount 196 key:0x0aa13d47 reqid=000039f9 pnn:0 src:0 lmaster:1
  High hopcount 198 key:0x0aa13d47 reqid=000039f9 pnn:0 src:0 lmaster:1

- These record requests bounce very quickly. After 2 seconds:
  
  High hopcount 955596 key:0x0aa13d47 reqid=000039f9 pnn:0 src:0 lmaster:1
  High hopcount 955598 key:0x0aa13d47 reqid=000039f9 pnn:0 src:0 lmaster:1
  High hopcount 955597 key:0x0aa13d47 reqid=00004771 pnn:0 src:1 lmaster:1
  High hopcount 955599 key:0x0aa13d47 reqid=00004771 pnn:0 src:1 lmaster:1
High hopcount bug

- Sometime later the migrated record request gets processed

Temporary inconsistency during record migration

Node 0 says Node 1 is DMASTER
Node 1 says Node 0 is DMASTER

Solution

Avoid processing record requests for record in migration
Sometime later the migrated record request gets processed

UPDATE db[notify_index.tdb]: store: hash[0x0aa13d47] rsn[9642] dmaster[0]
UPDATE db[notify_index.tdb]: store: hash[0x0aa13d47] rsn[9643] dmaster[0]
Sometime later the migrated record request gets processed

UPDATE db[notify_index.tdb]: store: hash[0x0aa13d47] rsn[9642] dmaster[0]
UPDATE db[notify_index.tdb]: store: hash[0x0aa13d47] rsn[9643] dmaster[0]

And the bouncing requests stop.
High hopcount bug

- Sometime later the migrated record request gets processed
  UPDATE db[notify_index.tdb]: store: hash[0x0aa13d47] rsn[9642] dmaster[0]
  UPDATE db[notify_index.tdb]: store: hash[0x0aa13d47] rsn[9643] dmaster[0]
  And the bouncing requests stop.
  Temporary inconsistency during record migration
High hopcount bug

- Sometime later the migrated record request gets processed
  
  UPDATE db[notify_index.tdb]: store: hash[0x0aa13d47] rsn[9642] dmaster[0]
  UPDATE db[notify_index.tdb]: store: hash[0x0aa13d47] rsn[9643] dmaster[0]

- And the bouncing requests stop.
- Temporary inconsistency during record migration
  - Node 0 says Node 1 is DMASTER
  - Node 1 says Node 0 is DMASTER
High hopcount bug

Sometime later the migrated record request gets processed

```
UPDATE db[notify_index.tdb]: store: hash[0x0aa13d47] rsn[9642] dmaster[0]
UPDATE db[notify_index.tdb]: store: hash[0x0aa13d47] rsn[9643] dmaster[0]
```

And the bouncing requests stop.

Temporary inconsistency during record migration

- Node 0 says Node 1 is DMASTER
- Node 1 says Node 0 is DMASTER

Solution

Avoid processing record requests for record in migration
Getting Lock Scheduling Right

Locks in CTDB

Record locks
To modify a record, CTDB tries to grab non-blocking lock
If that fails, create a lock request

Database locks
For database recovery, CTDB needs to freeze all databases

Why lock scheduling
Multiple requests for different records
Multiple requests for same record
There are multiple databases
Freeze requests are handled independently

Amitay Isaacs
CTDB Stories
Locks in CTDB

- Record locks
  - To modify a record, CTDB tries to grab non-blocking lock
  - If that fails, create a lock request
Locks in CTDB

- **Record locks**
  - To modify a record, CTDB tries to grab a non-blocking lock.
  - If that fails, it creates a lock request.
- **Database locks**
  - For database recovery, CTDB needs to freeze all databases.
Getting Lock Scheduling Right

Locks in CTDB
- Record locks
  - To modify a record, CTDB tries to grab non-blocking lock
  - If that fails, create a lock request
- Database locks
  - For database recovery, CTDB needs to freeze all databases

Why lock scheduling
Getting Lock Scheduling Right

Locks in CTDB

- Record locks
  - To modify a record, CTDB tries to grab non-blocking lock
  - If that fails, create a lock request
- Database locks
  - For database recovery, CTDB needs to freeze all databases

Why lock scheduling

- Multiple requests for different records
Getting Lock Scheduling Right

Locks in CTDB

- **Record locks**
  - To modify a record, CTDB tries to grab non-blocking lock
  - If that fails, create a lock request
- **Database locks**
  - For database recovery, CTDB needs to freeze all databases

Why lock scheduling

- Multiple requests for different records
- Multiple requests for same record
### Locks in CTDB

- **Record locks**
  - To modify a record, CTDB tries to grab non-blocking lock
  - If that fails, create a lock request
- **Database locks**
  - For database recovery, CTDB needs to freeze all databases

### Why lock scheduling

- Multiple requests for different records
- Multiple requests for same record
- There are multiple databases
Getting Lock Scheduling Right

Locks in CTDB

- Record locks
  - To modify a record, CTDB tries to grab non-blocking lock
  - If that fails, create a lock request
- Database locks
  - For database recovery, CTDB needs to freeze all databases

Why lock scheduling

- Multiple requests for different records
- Multiple requests for same record
- There are multiple databases
- Freeze requests are handled independently
New locking API abstraction - Naive approach
Getting Lock Scheduling Right

- New locking API abstraction - Naive approach
- Same API for record lock request and database lock request

Amitay Isaacs
CTDB Stories
Getting Lock Scheduling Right

- New locking API abstraction - Naive approach
- Same API for record lock request and database lock request
- Queues for active and pending lock requests
- New locking API abstraction - Naive approach
- Same API for record lock request and database lock request
- Queues for active and pending lock requests
- Maximum number of active lock requests
New locking API abstraction - Naive approach
- Same API for record lock request and database lock request
- Queues for active and pending lock requests
- Maximum number of active lock requests
- Create a child process to lock the record
Getting Lock Scheduling Right

- New locking API abstraction - Naive approach
- Same API for record lock request and database lock request
- Queues for active and pending lock requests
- Maximum number of active lock requests
- Create a child process to lock the record
- Mostly works . . .
Getting Lock Scheduling Right

- New locking API abstraction - Naive approach
- Same API for record lock request and database lock request
- Queues for active and pending lock requests
- Maximum number of active lock requests
- Create a child process to lock the record
- Mostly works . . .

Problem

. . . till database recovery is triggered under load
Getting Lock Scheduling Right

- New locking API abstraction - Naive approach
- Same API for record lock request and database lock request
- Queues for active and pending lock requests
- Maximum number of active lock requests
- Create a child process to lock the record
- Mostly works . . .

Problem

. . . till database recovery is triggered under load

Solution

- Active queue is full and freeze lock requests are pending
- Freeze lock requests need to be scheduled immediately
Getting Lock Scheduling Right

Problem
Performance is not good when record locking is in use.

Solution
A single limit on active records kills performance for locking requests across multiple databases.
Implement per database limits for active lock requests.

Problem
There are multiple lock processes waiting for the same record.

Solution
Rely on kernel to do "fair scheduling".
Before scheduling a lock request, check if there is an active lock request for the same record.
Problem

Performance is not good when record locking is in use

Solution

- A single limit on active records kills performance for locking requests across multiple databases
## Problem

Performance is not good when record locking is in use

## Solution

- A single limit on active records kills performance for locking requests across multiple databases
- Implement per database limits for active lock requests
Getting Lock Scheduling Right

Problem
Performance is not good when record locking is in use

Solution
- A single limit on active records kills performance for locking requests across multiple databases
- Implement per database limits for active lock requests

Problem
There are multiple lock processes waiting for the same record

Amitay Isaacs
CTDB Stories
### Problem
Performance is not good when record locking is in use

### Solution
- A single limit on active records kills performance for locking requests across multiple databases
- Implement per database limits for active lock requests

### Problem
There are multiple lock processes waiting for the same record

### Solution
- Rely on kernel to do “fair scheduling”
Getting Lock Scheduling Right

Problem
Performance is not good when record locking is in use

Solution
- A single limit on active records kills performance for locking requests across multiple databases
- Implement per database limits for active lock requests

Problem
There are multiple lock processes waiting for the same record

Solution
- Rely on kernel to do “fair scheduling”
- Before scheduling a lock request, check if there is an active lock request for the same record
Problem

CTDB is consuming 100% CPU under heavy load
Getting Lock Scheduling Right

Problem

CTDB is consuming 100% CPU under heavy load

Solution

- Active and pending lock queues are implemented as linked lists
Problem
CTDB is consuming 100% CPU under heavy load

Solution
- Active and pending lock queues are implemented as linked lists
- CTDB is spinning trying to schedule next request (60k requests in pending queue)
Problem
CTDB is consuming 100% CPU under heavy load

Solution
- Active and pending lock queues are implemented as linked lists
- CTDB is spinning trying to schedule next request (60k requests in pending queue)
- Undo active lock checking?
Problem

CTDB is consuming 100% CPU under heavy load

Solution

- Active and pending lock queues are implemented as linked lists
- CTDB is spinning trying to schedule next request (60k requests in pending queue)
- Undo active lock checking?
- Implement per database queues, not sufficient!
Problem

CTDB is consuming 100% CPU under heavy load

Solution

- Active and pending lock queues are implemented as linked lists
- CTDB is spinning trying to schedule next request (60k requests in pending queue)
- Undo active lock checking?
- Implement per database queues, not sufficient!

Better Solution

- Use better data structure for checking active lock requests
All nodes banned on single node failure

**Observation**

A node becomes INACTIVE (disconnected, stopped or banned)

CTDB tries to freeze databases for recovery and fails

CTDB retries and bans culprit node

Eventually ends up banning all remaining nodes

If locking database fails, CTDB logs useful information

All processes holding locks on CTDB database

Stack traces for all those processes

Relies on parsing /proc/locks

Cannot be used with TDB robust mutexes

Recreate after disabling TDB robust mutexes

Amitay Isaacs

CTDB Stories
Observation

- A node becomes INACTIVE (disconnected, stopped or banned)
All nodes banned on single node failure

Observation

- A node becomes INACTIVE (disconnected, stopped or banned)
- CTDB tries to freeze databases for recovery and fails
All nodes banned on single node failure

Observation

- A node becomes INACTIVE (disconnected, stopped or banned)
- CTDB tries to freeze databases for recovery and fails
- CTDB retries and bans **culprit** node
All nodes banned on single node failure

Observation

- A node becomes INACTIVE (disconnected, stopped or banned)
- CTDB tries to freeze databases for recovery and fails
- CTDB retries and bans culprit node
- Eventually ends up banning all remaining nodes
Observation

- A node becomes INACTIVE (disconnected, stopped or banned)
- CTDB tries to freeze databases for recovery and fails
- CTDB retries and bans **culprit** node
- Eventually ends up banning all remaining nodes

- If locking database fails, CTDB logs useful information
All nodes banned on single node failure

Observation

- A node becomes INACTIVE (disconnected, stopped or banned)
- CTDB tries to freeze databases for recovery and fails
- CTDB retries and bans *culprit* node
- Eventually ends up banning all remaining nodes

- If locking database fails, CTDB logs useful information
  - All processes holding locks on CTDB database
  - Stack traces for all those processes
All nodes banned on single node failure

Observation

- A node becomes INACTIVE (disconnected, stopped or banned)
- CTDB tries to freeze databases for recovery and fails
- CTDB retries and bans culprit node
- Eventually ends up banning all remaining nodes

- If locking database fails, CTDB logs useful information
  - All processes holding locks on CTDB database
  - Stack traces for all those processes
  - Relies on parsing /proc/locks
Observation

- A node becomes INACTIVE (disconnected, stopped or banned)
- CTDB tries to freeze databases for recovery and fails
- CTDB retries and bans **culprit** node
- Eventually ends up banning all remaining nodes

- If locking database fails, CTDB logs useful information
  - All processes holding locks on CTDB database
  - Stack traces for all those processes
  - Relies on parsing /proc/locks

- Cannot be used with TDB robust mutexes
All nodes banned on single node failure

Observation

- A node becomes INACTIVE (disconnected, stopped or banned)
- CTDB tries to freeze databases for recovery and fails
- CTDB retries and bans **culprit** node
- Eventually ends up banning all remaining nodes

- If locking database fails, CTDB logs useful information
  - All processes holding locks on CTDB database
  - Stack traces for all those processes
  - Relies on parsing `/proc/locks`

- Cannot be used with TDB robust mutexes
- Recreate after disabling TDB robust mutexes
All nodes banned on single node failure

- CTDB fails to freeze smbXsrv_session_global.tdb
All nodes banned on single node failure

- CTDB fails to freeze smbXsrv_session_global.tdb

  ctdbd-lock: /usr/bin/ctdb_lock_helper smbXsrv_session_global.tdb.0 168 223318
  ctdbd-lock: /usr/bin/ctdb_lock_helper smbXsrv_tcon_global.tdb.0 168 EOF
  ctdbd-lock: /usr/sbin/smbd smbXsrv_tcon_global.tdb.0 251880 251880 W
  ctdbd-lock: /usr/bin/ctdb_lock_helper locking.tdb.0 168 EOF
  ctdbd-lock: /usr/bin/ctdb_lock_helper smbXsrv_open_global.tdb.0 168 EOF
  ctdbd-lock: /usr/bin/ctdb_lock_helper cnscm_monitoring.tdb.0 168 EOF
  ctdbd-lock: /usr/sbin/smbd smbXsrv_session_global.tdb.0 223320 223320

Amitay Isaacs
CTDB Stories
All nodes banned on single node failure

- CTDB fails to freeze smbXsrv_session_global.tdb
  
  ctdbd-lock: /usr/bin/ctdb_lock_helper smbXsrv_session_global.tdb.0 168 223318
  ctdbd-lock: /usr/bin/ctdb_lock_helper smbXsrv_tcon_global.tdb.0 168 EOF
  ctdbd-lock: /usr/sbin/smbd smbXsrv_tcon_global.tdb.0 251880 251880 W
  ctdbd-lock: /usr/bin/ctdb_lock_helper locking.tdb.0 168 EOF
  ctdbd-lock: /usr/bin/ctdb_lock_helper smbXsrv_open_global.tdb.0 168 EOF
  ctdbd-lock: /usr/bin/ctdb_lock_helper cnscm_monitoring.tdb.0 168 EOF
  ctdbd-lock: /usr/sbin/smbd smbXsrv_session_global.tdb.0 223320 223320

- Samba process is holding a lock
All nodes banned on single node failure

- Stack trace for relevant samba process

```c
#0 0x00007fde05236218 in poll () from /lib64/libc.so.6
#1 0x00007fde0863a93c in poll_one_fd ()
#2 0x00007fde0861146b in ctdb_packet_fd_read_sync_timeout ()
#3 0x00007fde08611c0d in ctdb_packet_fd_read_sync ()
#4 0x00007fde086126fa in ctdb_read_req ()
#5 0x00007fde08612eae in ctdbd_parse ()
#6 0x00007fde0862184d in db_ctdb_parse_record ()
#7 0x00007fde0861d9d4 in dbwrap_parse_record ()
#8 0x00007fde0861dc2a in dbwrap_fetch ()
#9 0x00007fde086250fd in dbwrap_watch_record_stored ()
#10 0x00007fde0861dc86 in dbwrap_record_delete ()
#11 0x00007fde088e41a0 in exit_server_common ()
#12 0x00007fde088e462e in smbd_exit_server_cleanly ()
#13 0x00007fde083609e2 in exit_server_cleanly ()
```
All nodes banned on single node failure

Samba is holding a record lock ([smbXsrv_session_global.tdb]) and waiting for another record ([dbwatchers.tdb]). CTDB is in the process of migrating the record. At this time, CTDB on the remote node becomes INACTIVE.

CTDB has to perform database recovery. CTDB starts to freeze databases. Since CTDB cannot freeze databases, it will ban the culprit.

Multiple Samba processes holding a lock on different nodes. All nodes get banned!
All nodes banned on single node failure

- Samba is holding a record lock
  (smbXsrv_session_global.tdb)

CTDB is in the process of migrating the record
At this time CTDB on the remote node becomes INACTIVE
CTDB has to perform database recovery
CTDB starts to freeze databases
CTDB cannot lock
Deadlock!
Since CTDB cannot freeze databases, it will ban the culprit
Multiple Samba processes holding a lock on different nodes
All nodes get banned!

Amitay Isaacs
CTDB Stories
All nodes banned on single node failure

- Samba is holding a record lock
  (smbXsrv_session_global.tdb)
- And waiting for another record (dbwatchers.tdb)
All nodes banned on single node failure

- Samba is holding a record lock (smbXsrv_session_global.tdb)
- And waiting for another record (dbwatchers.tdb)
- CTDB is in the process of migrating the record
- CTDB on the remote node becomes INACTIVE
- CTDB has to perform database recovery
- CTDB starts to freeze databases
- CTDB cannot lock
- Deadlock!
- Since CTDB cannot freeze databases, it will ban the culprit
- Multiple Samba processes holding a lock on different nodes
- All nodes get banned!
All nodes banned on single node failure

- Samba is holding a record lock (smbXsrv_session_global.tdb)
- And waiting for another record (dbwatchers.tdb)
- CTDB is in the process of migrating the record
- At this time CTDB on the remote node becomes INACTIVE
All nodes banned on single node failure

- Samba is holding a record lock (smbXsrv_session_global.tdb)
- And waiting for another record (dbwatchers.tdb)
- CTDB is in the process of migrating the record
- At this time CTDB on the remote node becomes INACTIVE
- CTDB has to perform database recovery

Amitay Isaacs
CTDB Stories
All nodes banned on single node failure

- Samba is holding a record lock
  (smbXsrv_session_global.tdb)
- And waiting for another record (dbwatchers.tdb)
- CTDB is in the process of migrating the record
- At this time CTDB on the remote node becomes INACTIVE
- CTDB has to perform database recovery
- CTDB starts to freeze databases
All nodes banned on single node failure

- Samba is holding a record lock (smbXsrv_session_global.tdb)
- And waiting for another record (dbwatchers.tdb)
- CTDB is in the process of migrating the record
- At this time CTDB on the remote node becomes INACTIVE
- CTDB has to perform database recovery
- CTDB starts to freeze databases
- CTDB cannot lock smbXsrv_session_global.tdb

Amitay Isaacs

CTDB Stories
All nodes banned on single node failure

- Samba is holding a record lock (smbXsrv_session_global.tdb)
- And waiting for another record (dbwatchers.tdb)
- CTDB is in the process of migrating the record
- At this time CTDB on the remote node becomes INACTIVE
- CTDB has to perform database recovery
- CTDB starts to freeze databases
- CTDB cannot lock smbXsrv_session_global.tdb
- Deadlock!
All nodes banned on single node failure

- Samba is holding a record lock (smbXsrv_session_global.tdb)
- And waiting for another record (dbwatchers.tdb)
- CTDB is in the process of migrating the record
- At this time CTDB on the remote node becomes INACTIVE
- CTDB has to perform database recovery
- CTDB starts to freeze databases
- CTDB cannot lock smbXsrv_session_global.tdb
- Deadlock!
- Since CTDB cannot freeze databases, it will ban the culprit
All nodes banned on single node failure

- Samba is holding a record lock (smbXsrv_session_global.tdb)
- And waiting for another record (dbwatchers.tdb)
- CTDB is in the process of migrating the record
- At this time CTDB on the remote node becomes INACTIVE
- CTDB has to perform database recovery
- CTDB starts to freeze databases
- CTDB cannot lock smbXsrv_session_global.tdb
- Deadlock!
- Since CTDB cannot freeze databases, it will ban the culprit
- Multiple Samba processes holding a lock on different nodes
All nodes banned on single node failure

- Samba is holding a record lock (smbXsrv_session_global.tdb)
- And waiting for another record (dbwatchers.tdb)
- CTDB is in the process of migrating the record
- At this time CTDB on the remote node becomes INACTIVE
- CTDB has to perform database recovery
- CTDB starts to freeze databases
- CTDB cannot lock smbXsrv_session_global.tdb
- Deadlock!
- Since CTDB cannot freeze databases, it will ban the culprit
- Multiple Samba processes holding a lock on different nodes
- All nodes get banned!
### Problem
- CTDB cannot freeze database since Samba is holding a lock
- Samba will not release a lock, till it gets the second lock
All nodes banned on single node failure

Problem

- CTDB cannot freeze database since Samba is holding a lock
- Samba will not release a lock, till it gets the second lock
- CTDB database recovery is serial
All nodes banned on single node failure

Problem
- CTDB cannot freeze database since Samba is holding a lock
- Samba will not release a lock, till it gets the second lock

- CTDB database recovery is serial
  - Freeze all databases
  - Recover databases one by one
  - Unlock all databases
All nodes banned on single node failure

Problem

- CTDB cannot freeze database since Samba is holding a lock
- Samba will not release a lock, till it gets the second lock

- CTDB database recovery is serial
  - Freeze all databases
  - Recover databases one by one
  - Unlock all databases

Solution

- Do database recovery in parallel
Problem

- CTDB cannot freeze database since Samba is holding a lock
- Samba will not release a lock, till it gets the second lock

- CTDB database recovery is serial
  - Freeze all databases
  - Recover databases one by one
  - Unlock all databases

Solution

- Do database recovery in parallel
  - Start freeze of all databases
All nodes banned on single node failure

Problem

- CTDB cannot freeze database since Samba is holding a lock
- Samba will not release a lock, till it gets the second lock

- CTDB database recovery is serial
  - Freeze all databases
  - Recover databases one by one
  - Unlock all databases

Solution

- Do database recovery in parallel
  - Start freeze of all databases
  - As soon as database is frozen, recover database
## Problem
- CTDB cannot freeze database since Samba is holding a lock
- Samba will not release a lock, till it gets the second lock

- CTDB database recovery is serial
  - Freeze all databases
  - Recover databases one by one
  - Unlock all databases

## Solution
- Do database recovery in parallel
  - Start freeze of all databases
  - As soon as database is frozen, recover database
  - Process all pending call requests for that database
Real time or not

Background

CTDB runs with real-time priority
CTDB creates lots of processes.
ctdb_fork()
fork() is found to be expensive on busy systems
Replace fork() with vfork() and exec*
Introduce helper processes - ctdb_event_helper

Regression
All event scripts now run with real-time priority
CTDB_MANAGES_SAMBA=yes
In 50.samba, startup event starts smbd

Amitay Isaacs
CTDB Stories
CTDB runs with real-time priority
Real time or not

Background

- CTDB runs with real-time priority
- CTDB creates lots of processes.
CTDB runs with real-time priority
CTDB creates lots of processes.
`ctdb_fork()` - reset process priority
Real time or not

Background

- CTDB runs with real-time priority
- CTDB creates lots of processes.
- `ctdb_fork()` - reset process priority
- `fork()` is found to be expensive on busy systems

Replace `fork()` with `vfork()` and `exec*()`

Introduce helper processes - `ctdb_event_helper`

Regression

All event scripts now run with real-time priority

CTDB_MANAGES_SAMBA=yes

In 50.samba, startup event starts smbd
Real time or not

Background

- CTDB runs with real-time priority
- CTDB creates lots of processes.
- `ctdb_fork()` - reset process priority
- `fork()` is found to be expensive on busy systems
- Replace `fork()` with `vfork()` and `exec*()`
Background

- CTDB runs with real-time priority
- CTDB creates lots of processes.
- `ctdb_fork()` - reset process priority
- `fork()` is found to be expensive on busy systems
- Replace `fork()` with `vfork()` and `exec*()`
- Introduce helper processes - `ctdb_event_helper`
Real time or not

Background

- CTDB runs with real-time priority
- CTDB creates lots of processes.
- `ctdb_fork()` - reset process priority
- `fork()` is found to be expensive on busy systems
- Replace `fork()` with `vfork()` and `exec*()`
- Introduce helper processes - `ctdb_event_helper`

Regression
Real time or not

Background

- CTDB runs with real-time priority
- CTDB creates lots of processes.
- `ctdb_fork()` - reset process priority
- `fork()` is found to be expensive on busy systems
- Replace `fork()` with `vfork()` and `exec*()`
- Introduce helper processes - `ctdb_event_helper`

Regression

- All event scripts now run with real-time priority
Real time or not

Background
- CTDB runs with real-time priority
- CTDB creates lots of processes.
- `ctdb_fork()` - reset process priority
- `fork()` is found to be expensive on busy systems
- Replace `fork()` with `vfork()` and `exec*()`
- Introduce helper processes - `ctdb_event_helper`

Regression
- All event scripts now run with real-time priority
- `CTDB_MANAGES_SAMBA=yes`
Real time or not

**Background**

- CTDB runs with real-time priority
- CTDB creates lots of processes.
- `ctdb_fork()` - reset process priority
- `fork()` is found to be expensive on busy systems
- Replace `fork()` with `vfork()` and `exec*()`
- Introduce helper processes - `ctdb_event_helper`

**Regression**

- All event scripts now run with real-time priority
- `CTDB_MANAGES_SAMBA=yes`
- In 50.samba, startup event starts smbd
Fixing compiler warnings

Background

CTDB sets up pipe from a child process so the child process can send the status via pipe. If pipe close indicates failure of the child, there are many read()/write() calls without checking return values. Replace all read()/write() with sys_read()/sys_write().

Regression

While testing on VMs, CTDB was consuming 100% CPU. Tracing showed CTDB was busy stuck in sys_write() as Samba was not getting scheduled to read from CTDB. If write() calls fail with EAGAIN, back off.
Fixing compiler warnings

Background

- CTDB sets up pipe from a child process

Regression

While testing on VMs, CTDB consuming 100% CPU. Tracing shows CTDB is busy stuck in `sys_write()`. Samba not getting scheduled to read from CTDB. If `write()` calls fails with `EAGAIN`, back off.
Fixing compiler warnings

Background

- CTDB sets up pipe from a child process
  - So child process can send the status via pipe

Regression

While testing on VMs, CTDB consuming 100% CPU
Tracing shows CTDB is busy stuck in sys_write()
Samba not getting scheduled to read from CTDB
If write() calls fails with EAGAIN, back off
Fixing compiler warnings

Background

- CTDB sets up pipe from a child process
  - So child process can send the status via pipe
  - Pipe close indicates failure of child
Fixing compiler warnings

Background

- CTDB sets up pipe from a child process
  - So child process can send the status via pipe
  - Pipe close indicates failure of child
- Many `read()`/`write()` calls without checking return values

Regression

While testing on VMs, CTDB consuming 100% CPU

Tracing shows CTDB is busy stuck in `sys_write()`

Samba not getting scheduled to read from CTDB

If `write()` calls fail with `EAGAIN`, back off

Amitay Isaacs

CTDB Stories
Fixing compiler warnings

Background

- CTDB sets up pipe from a child process
  - So child process can send the status via pipe
  - Pipe close indicates failure of child
- Many `read()`/`write()` calls without checking return values
- Replace all `read()`/`write()` with `sys_read()`/`sys_write()`

Regression

While testing on VMs, CTDB consuming 100% CPU. Tracing shows CTDB is busy stuck in `sys_write()`. Samba not getting scheduled to read from CTDB.

If `write()` calls fails with `EAGAIN`, back off.
Fixing compiler warnings

Background
- CTDB sets up pipe from a child process
  - So child process can send the status via pipe
  - Pipe close indicates failure of child
- Many read()/write() calls without checking return values
- Replace all read()/write() with sys_read()/sys_write()

Regression

Amitay Isaacs
CTDB Stories
Fixing compiler warnings

Background

- CTDB sets up pipe from a child process
  - So child process can send the status via pipe
  - Pipe close indicates failure of child
- Many `read()`/`write()` calls without checking return values
- Replace all `read()`/`write()` with `sys_read()`/`sys_write()`

Regression

- While testing on VMs, CTDB consuming 100% CPU
Fixing compiler warnings

Background
- CTDB sets up pipe from a child process
  - So child process can send the status via pipe
  - Pipe close indicates failure of child
- Many `read()`/`write()` calls without checking return values
- Replace all `read()`/`write()` with `sys_read()`/`sys_write()`

Regression
- While testing on VMs, CTDB consuming 100% CPU
- Tracing shows CTDB is busy stuck in `sys_write()`
Fixing compiler warnings

Background

- CTDB sets up pipe from a child process
  - So child process can send the status via pipe
  - Pipe close indicates failure of child
- Many `read()/write()` calls without checking return values
- Replace all `read()/write()` with `sys_read()/sys_write()`

Regression

- While testing on VMs, CTDB consuming 100% CPU
- Tracing shows CTDB is busy stuck in `sys_write()`
- Samba not getting scheduled to read from CTDB
Fixing compiler warnings

Background

- CTDB sets up pipe from a child process
  - So child process can send the status via pipe
  - Pipe close indicates failure of child
- Many `read()`/`write()` calls without checking return values
- Replace all `read()`/`write()` with `sys_read()`/`sys_write()`

Regression

- While testing on VMs, CTDB consuming 100% CPU
- Tracing shows CTDB is busy stuck in `sys_write()`
- Samba not getting scheduled to read from CTDB
- If `write()` calls fails with EAGAIN, back off
Questions/Comments?