CTDB 2.0 and Beyond

Amitay Isaacs
amitay@samba.org

Samba Team
IBM, Australia Development Labs, Linux Technology Center

The Journey

1. Unraveling CTDB

2. Current Development

3. Testing

4. Future

What is CTDB?

Motivation: Support for clustered Samba

- Multiple nodes active simultaneously
- Communication between nodes (heartbeat, failover)
- Share databases between nodes

What is CTDB?

Motivation: Support for clustered Samba

- Multiple nodes active simultaneously
- Communication between nodes (heartbeat, failover)
- Share databases between nodes

CTDB: Clustered implementation of TDB

- Volatile and Persistent databases
- IP failover and load balancing
- Service monitoring

CTDB Project

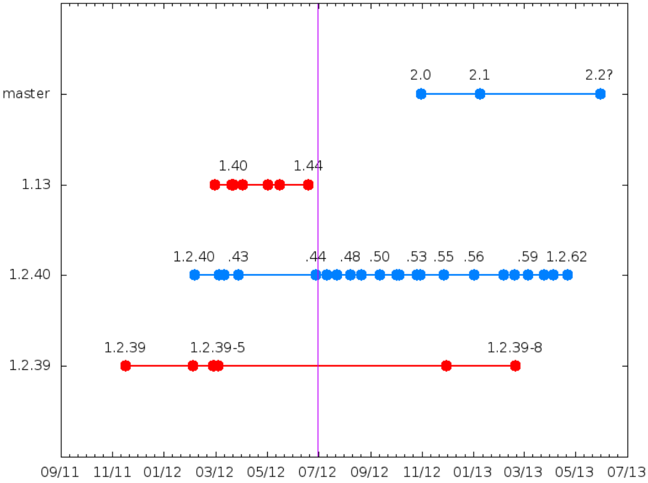
- `http://ctdb.samba.org`
- `git://git.samba.org/ctdb.git`

- Took over maintainership from Ronnie Salhberg (July 2012)
- Sketchy/Missing documentation - Updates to website/wiki

Branches

1.0.44
1.0.45
1.0.46
1.0.47
1.0.48
1.0.49
1.0.50
1.0.52
1.0.53
1.0.54
1.0.55
1.0.56
1.0.64
1.0.69
1.0.82
1.0.89
1.0.108
1.0.112
1.0.112b
1.0.114
1.13
1.2
1.2.27
1.2.27-PTF1
1.2.38
1.2.39
1.2.39-28
1.2.40
1.3
master

Branches & Releases



CTDB Releases

- 1.44 (June 2012)
  - Last release by Ronnie Sahlberg
- 2.0 (November 2012)
  - 147 patches since 1.44
  - locking, tevent logging, building and packaging
- 2.1 (January 2013)
  - 61 patches since 2.0
  - support for Samba 4
- 2.2 (May 2013?)
  - 150+ patches since 2.1
  - performance improvements, recovery/vacuum database corruption fixes

Developers

Contributions in 2012

221	Martin Schwenke
94	Amitay Isaacs
82	Ronnie Sahlberg
13	Michael Adam
11	Volker Lendecke
3	Stefan Metzmacher
3	Mathieu Parent
1	Gregor Beck
1	David Disseldorp

Developers

Contributions since Jan 2013

116	Martin Schwenke
35	Amitay Isaacs
31	Michael Adam
7	Mathieu Parent
4	Volker Lendecke
1	Sumit Bose
1	Srikrishan Malik

Current Development

Bug fixes

- Persistent database corruption in recovery
- Non-persistent database corrupation (record migration and recovery interaction)
- Vacuum and Recovery interaction causing database corrpution
- Race condition when running monitor and other events
- Close unix domain socket in syslog daemon
- Fixing Statd callout for CTDB (RHEL6 runs statd as rpcuser)

Features / Changes

- Improved status checking using PID file
    - Avoid race condition due to timeouts in `ctdb ping`
- Startup sequence serialization using runstate
  INIT → SETUP → FIRST_RECOVERY → STARTUP → RUNNING → SHUTDOWN
- `ctdb getlog [recoverd]`
- New tunable – NoIPHostOnAllDisabled
- Locking changes with deadlock detection

Performance Improvements

Problem

CTDB consuming 100% CPU and causing OOM with 5000 SMB connections

Performance Improvements

Problem

CTDB consuming 100% CPU and causing OOM with 5000 SMB connections

- Improve handling of socket I/O
- Free log ringbuffer in child processes
- Tevent changes to deal with lots of zero timeval events
- Replace message handler linked list with hash table
- Use lightweight helper process for locking records

Testing

Testing Infrastructure

- Unit tests
  - eventscripts - test eventscripts using stubs
  - onnode - tests for **onnode** tool
  - takeover - tests for IP allocation algorithm
  - tool - tests for **ctdb** tool
- Integration tests
  - simple - tests that can be run locally and on cluster
  - complex - tests that can be only run on cluster

Testing with Local daemons

- Allow developer testing without building clusters
- Using stubs to allow non-root execution (e.g. `ip` command)
- CTDB Test environment
  - Run 3 CTDB daemons locally
  - Simple eventscript
- Flexible test framework to run specific testsuites

Test runner

```
ctdb-2.1$ tests/scripts/run_tests --help
Usage: run_tests [OPTIONS] [TESTS]

Options:
  -s         Print a summary of tests results after running all tests
  -l         Use local daemons for integration tests
  -e         Exit on the first test failure
  -V <dir>   Use <dir> as TEST_VAR_DIR
  -C         Clean up - kill daemons and remove TEST_VAR_DIR when done
  -v         Verbose - print test output for non-failures (only some tests)
  -A         Use "cat -A" to print test output (only some tests)
  -D         Show diff between failed/expected test output (some tests only)
  -X         Trace certain scripts run by tests using -x (only some tests)
  -d         Print descriptions of tests instead of filenames (dodgy!)
  -H         No headers - for running single test with other wrapper
  -q         Quiet - don't show tests being run (hint: use with -s)
  -x         Trace this script with the -x option
```

Testing - examples

- Run unit test testsuite

```
$ tests/run_tests -V tests/var eventscripts
```

- Start local daemons

```
$ tests/run_tests -V tests/var tests/simple/00_ctdb_init.sh
```

- Run a test

```
$ tests/run_tests -V tests/var tests/simple/51_ctdb_bench.sh
```

- Shutdown daemons and cleanup

```
$ tests/run_tests -V tests/var -C tests/simple/99_daemons_shutdown.sh
```

- Running tests on cluster
  - ctdb_run_tests
  - ctdb_run_cluster_tests

Autocluster

Problem

How to easily test CTDB and Clustered Samba?

- Disposable clusters
  - Hardware is not always available
  - Hard to reproduce exact setups
  - Clusters tend to degrade
- Steps for new cluster
  1. Choose configuration
  2. Create base image (one time)
  3. Create cluster (setup AD, GPFS + clustered Samba)
  4. Boot it
- Autobuild for CTDB
- `git://git.samba.org/autocluster.git`

Future

Wish List

- Split monolithic code into separate daemons
  - Logging, IP handling, Services monitoring
- Proper CTDB library - libctdb
  - Database operations are missing
  - Thread-safe (avoid talloc/tevent?)
- CTDB Protocol
  - Version tracking
  - Auto-generated marshalling/unmarshalling code
- Scalability – large number of nodes
  - Database recovery
  - Handling record contention
- Pluggable Monitoring and Failover
  - Integration with 3rd party HA

Logging Daemon

- Text protocol - easier to debug
- Prototype - Server (python), Client (shell script)

```
SYSLOG <debuglevel>
SYSLOG OFF
LOGFIL <debuglevel> <filepath>
LOGFIL OFF
BUFFER <debuglevel>
BUFCLR
BUFSIZ <size>
BUFGET <tag> <debuglevel>
STATUS
LOGMSG <pid> <tag> <debuglevel> <msg>
```

Project direction

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

Questions?