

Lessons Learned in Clustered Samba

sambaXP 2010

Michael Adam

`obnox@samba.org`

Samba Team / SerNet

2010-05-06

Outline

- 1 Refresher on CTDB
- 2 Growing...
- 3 Recent Advances
- 4 Ongoing Tasks

Outline

- 1 Refresher on CTDB
- 2 Growing...
- 3 Recent Advances
- 4 Ongoing Tasks

Refresher on CTDB

- idea: share cluster file system via CIFS
- from multiple nodes simultaneously (active-active)
- need IPC between nodes: messaging and session/locking data
- and need to share some persistent data: passdb, join information, id mapping
- ⇒ need clustered implementation of TDB (and messaging): CTDB

Refresher on CTDB

- idea: share cluster file system via CIFS
- from multiple nodes simultaneously (active-active)
- need IPC between nodes: messaging and session/locking data
- and need to share some persistent data: passdb, join information, id mapping
- ⇒ need clustered implementation of TDB (and messaging): CTDB

Refresher on CTDB

- idea: share cluster file system via CIFS
- from multiple nodes simultaneously (active-active)
- need IPC between nodes: messaging and session/locking data
- and need to share some persistent data: passdb, join information, id mapping
- ⇒ need clustered implementation of TDB (and messaging): CTDB

Refresher on CTDB

- idea: share cluster file system via CIFS
- from multiple nodes simultaneously (active-active)
- need IPC between nodes: messaging and session/locking data
- and need to share some persistent data: passdb, join information, id mapping
- ⇒ need clustered implementation of TDB (and messaging): CTDB

Refresher on CTDB

- idea: share cluster file system via CIFS
- from multiple nodes simultaneously (active-active)
- need IPC between nodes: messaging and session/locking data
- and need to share some persistent data: passdb, join information, id mapping
- ⇒ need clustered implementation of TDB (and messaging): CTDB

Refresher on CTDB – History and Community

- started in 2006 (Volker Lendecke, Andrew Tridgell)
- first usable version of CTDB presented at sambaXP 2007
- Ronnie Sahlberg maintainer
- [git://git.samba.org/sahlberg/ctdb.git](https://git.samba.org/sahlberg/ctdb.git)
- <http://ctdb.samba.org/packages/> (RPMs, Sources)
- warning: there is no elaborate release process
- packagers/integrators: better check with developers
- irc: #ctdb on freenode, samba-technical ML, bugzilla

Refresher on CTDB – History and Community

- started in 2006 (Volker Lendecke, Andrew Tridgell)
- first usable version of CTDB presented at sambaXP 2007
- Ronnie Sahlberg maintainer
- `git://git.samba.org/sahlberg/ctdb.git`
- `http://ctdb.samba.org/packages/` (RPMs, Sources)
- warning: there is no elaborate release process
- packagers/integrators: better check with developers
- irc: #ctdb on freenode, samba-technical ML, bugzilla

Refresher on CTDB – History and Community

- started in 2006 (Volker Lendecke, Andrew Tridgell)
- first usable version of CTDB presented at sambaXP 2007
- **Ronnie Sahlberg maintainer**
- `git://git.samba.org/sahlberg/ctdb.git`
- `http://ctdb.samba.org/packages/` (RPMs, Sources)
- warning: there is no elaborate release process
- packagers/integrators: better check with developers
- irc: #ctdb on freenode, samba-technical ML, bugzilla

Refresher on CTDB – History and Community

- started in 2006 (Volker Lendecke, Andrew Tridgell)
- first usable version of CTDB presented at sambaXP 2007
- Ronnie Sahlberg maintainer
- [git://git.samba.org/sahlberg/ctdb.git](https://git.samba.org/sahlberg/ctdb.git)
- <http://ctdb.samba.org/packages/> (RPMs, Sources)
- warning: there is no elaborate release process
- packagers/integrators: better check with developers
- irc: #ctdb on freenode, samba-technical ML, bugzilla

Refresher on CTDB – History and Community

- started in 2006 (Volker Lendecke, Andrew Tridgell)
- first usable version of CTDB presented at sambaXP 2007
- Ronnie Sahlberg maintainer
- `git://git.samba.org/sahlberg/ctdb.git`
- <http://ctdb.samba.org/packages/> (RPMs, Sources)
- warning: there is no elaborate release process
- packagers/integrators: better check with developers
- irc: #ctdb on freenode, samba-technical ML, bugzilla

Refresher on CTDB – History and Community

- started in 2006 (Volker Lendecke, Andrew Tridgell)
- first usable version of CTDB presented at sambaXP 2007
- Ronnie Sahlberg maintainer
- [git://git.samba.org/sahlberg/ctdb.git](https://git.samba.org/sahlberg/ctdb.git)
- <http://ctdb.samba.org/packages/> (RPMs, Sources)
- **warning: there is no elaborate release process**
- packagers/integrators: better check with developers
- irc: #ctdb on freenode, samba-technical ML, bugzilla

Refresher on CTDB – History and Community

- started in 2006 (Volker Lendecke, Andrew Tridgell)
- first usable version of CTDB presented at sambaXP 2007
- Ronnie Sahlberg maintainer
- `git://git.samba.org/sahlberg/ctdb.git`
- `http://ctdb.samba.org/packages/` (RPMs, Sources)
- warning: there is no elaborate release process
- **packagers/integrators: better check with developers**
- irc: #ctdb on freenode, samba-technical ML, bugzilla

Refresher on CTDB – History and Community

- started in 2006 (Volker Lendecke, Andrew Tridgell)
- first usable version of CTDB presented at sambaXP 2007
- Ronnie Sahlberg maintainer
- [git://git.samba.org/sahlberg/ctdb.git](https://git.samba.org/sahlberg/ctdb.git)
- <http://ctdb.samba.org/packages/> (RPMs, Sources)
- warning: there is no elaborate release process
- packagers/integrators: better check with developers
- irc: #ctdb on freenode, samba-technical ML, bugzilla

Refresher on CTDB – Design

- “normal” databases (volatile):
 - R/W performance critical (locking...)
 - no need to propagate all changes
 - node does only need data related to its sessions
 - session data of a node may (should!) be lost when a node leaves
 - *data master* and *localtion master* roles
- recovery process
- distribution of ip addresses (failover / failback)
- management of services (samba, nfs, vsftpd ...)
- pluggable *event script* architecture

Refresher on CTDB – Design

- “normal” databases (volatile):
 - R/W performance critical (locking...)
 - no need to propagate all changes
 - node does only need data related to its sessions
 - session data of a node may (should!) be lost when a node leaves
 - *data master* and *localtion master* roles
- recovery process
- distribution of ip addresses (failover / failback)
- management of services (samba, nfs, vsftpd ...)
- pluggable *event script* architecture

Refresher on CTDB – Design

- “normal” databases (volatile):
 - R/W performance critical (locking...)
 - **no need to propagate all changes**
 - node does only need data related to its sessions
 - session data of a node may (should!) be lost when a node leaves
 - *data master* and *localtion master* roles
- recovery process
- distribution of ip addresses (failover / failback)
- management of services (samba, nfs, vsftpd ...)
- pluggable *event script* architecture

Refresher on CTDB – Design

- “normal” databases (volatile):
 - R/W performance critical (locking...)
 - no need to propagate all changes
 - **node does only need data related to its sessions**
 - session data of a node may (should!) be lost when a node leaves
 - *data master* and *localtion master* roles
- recovery process
- distribution of ip addresses (failover / failback)
- management of services (samba, nfs, vsftpd ...)
- pluggable *event script* architecture

Refresher on CTDB – Design

- “normal” databases (volatile):
 - R/W performance critical (locking...)
 - no need to propagate all changes
 - node does only need data related to its sessions
 - session data of a node may (should!) be lost when a node leaves
 - *data master* and *localtion master* roles
- recovery process
- distribution of ip addresses (failover / failback)
- management of services (samba, nfs, vsftpd ...)
- plugable *event script* architecture

Refresher on CTDB – Design

- “normal” databases (volatile):
 - R/W performance critical (locking...)
 - no need to propagate all changes
 - node does only need data related to its sessions
 - session data of a node may (should!) be lost when a node leaves
 - *data master and localtion master roles*
- recovery process
- distribution of ip addresses (failover / failback)
- management of services (samba, nfs, vsftpd ...)
- pluggable *event script* architecture

Refresher on CTDB – Design

- “normal” databases (volatile):
 - R/W performance critical (locking...)
 - no need to propagate all changes
 - node does only need data related to its sessions
 - session data of a node may (should!) be lost when a node leaves
 - *data master* and *localtion master* roles
- recovery process
 - distribution of ip addresses (failover / failback)
 - management of services (samba, nfs, vsftpd ...)
 - plugable *event script* architecture

Refresher on CTDB – Design

- “normal” databases (volatile):
 - R/W performance critical (locking...)
 - no need to propagate all changes
 - node does only need data related to its sessions
 - session data of a node may (should!) be lost when a node leaves
 - *data master* and *localtion master* roles
- recovery process
- distribution of ip addresses (failover / failback)
- management of services (samba, nfs, vsftpd ...)
- plugable *event script* architecture

Refresher on CTDB – Design

- “normal” databases (volatile):
 - R/W performance critical (locking...)
 - no need to propagate all changes
 - node does only need data related to its sessions
 - session data of a node may (should!) be lost when a node leaves
 - *data master* and *localtion master* roles
- recovery process
- distribution of ip addresses (failover / failback)
- management of services (samba, nfs, vsftpd ...)
- plugable *event script* architecture

Refresher on CTDB – Design

- “normal” databases (volatile):
 - R/W performance critical (locking...)
 - no need to propagate all changes
 - node does only need data related to its sessions
 - session data of a node may (should!) be lost when a node leaves
 - *data master* and *localtion master* roles
- recovery process
- distribution of ip addresses (failover / failback)
- management of services (samba, nfs, vsftpd ...)
- *pluggable event script* architecture

Outline

- 1 Refresher on CTDB
- 2 Growing...
- 3 Recent Advances
- 4 Ongoing Tasks

contributors: some commit counts (ctdb)

```
6 - Sumit Bose
7 - Wolfgang Mueller-Friedt
11 - Mathieu Parent
20 - Volker Lendecke
24 - Andrew Tridgell
110 - Michael Adam
113 - Rusty Russell
135 - Stefan Metzmacher
145 - Martin Schwenke
369 - Ronnie Sahlberg
-----
~ 1000 past year
```

Stretching the Limits

- building clusters with > 20 nodes (> 30 ?)
- testing with several 10,000 clients (smbtorture)

Stretching the Limits

- building clusters with > 20 nodes (> 30 ?)
- testing with several 10,000 clients (`smbtorture`)

Outline

- 1 Refresher on CTDB
- 2 Growing...
- 3 Recent Advances**
- 4 Ongoing Tasks

some assorted bits – ctdb

- recovery lock has become optional
 - several subcommands added to ctdb (e.g. wipedb)
 - eventscript code (in ctddb) has been reworked
 - vacuuming and repacking has been streamlined and moved into the daemon
 - the tdb code in ctdb synchronized with samba master
 - fixed several race conditions and even deadlock in ctdb/samba
 - local failover and loadbalancing
- initially, just one public interface per node (including bonding)
- new support for distributing public IPs over multiple interfaces per node
- local loadbalancing and failover/fail back

some assorted bits – ctdb

- recovery lock has become optional
- several subcommands added to ctdb (e.g. wipedb)
- eventscrip code (in ctddb) has been reworked
- vacuuming and repacking has been streamlined and moved into the daemon
- the tdb code in ctdb synchronized with samba master
- fixed several race conditions and even deadlock in ctdb/samba
- local failover and loadbalancing
- new optional per node public interface per node (including bonding)
- new support for distributing public IP over multiple interfaces per node
- new support for local loadbalancing and failover/fail back

some assorted bits – ctdb

- recovery lock has become optional
- several subcommands added to ctdb (e.g. wipedb)
- **eventscript code (in ctddb) has been reworked**
- vacuuming and repacking has been streamlined and moved into the daemon
- the tdb code in ctdb synchronized with samba master
- fixed several race conditions and even deadlock in ctdb/samba
- local failover and loadbalancing

some assorted bits – ctdb

- recovery lock has become optional
 - several subcommands added to ctdb (e.g. wipedb)
 - eventscript code (in ctdbd) has been reworked
 - vacuuming and repacking has been streamlined and moved into the daemon
 - the tdb code in ctdb synchronized with samba master
 - fixed several race conditions and even deadlock in ctdb/samba
 - local failover and loadbalancing
- locally, for use with the periodic backup tool
- new support for distributed locks (the new smbclient can now lock files on a remote server)
- local loadbalancing and failover (all back

some assorted bits – ctdb

- recovery lock has become optional
- several subcommands added to ctdb (e.g. wipedb)
- eventscript code (in ctddb) has been reworked
- vacuuming and repacking has been streamlined and moved into the daemon
- the tdb code in ctdb synchronized with samba master
 - fixed several race conditions and even deadlock in ctdb/samba
 - local failover and loadbalancing

some assorted bits – ctdb

- recovery lock has become optional
- several subcommands added to ctdb (e.g. wipedb)
- eventscript code (in ctddb) has been reworked
- vacuuming and repacking has been streamlined and moved into the daemon
- the tdb code in ctdb synchronized with samba master
- fixed several race conditions and even deadlock in ctdb/samba
- local failover and loadbalancing
 - originally, just one public interface per node (including bonding)

some assorted bits – ctdb

- recovery lock has become optional
- several subcommands added to ctdb (e.g. wipedb)
- eventscript code (in ctddb) has been reworked
- vacuuming and repacking has been streamlined and moved into the daemon
- the tdb code in ctdb synchronized with samba master
- fixed several race conditions and even deadlock in ctdb/samba
- **local failover and loadbalancing**
 - originally, just one public interface per node (including bonding)
 - new: support for distributing public ips over multiple interfaces per node
 - local loadbalancing and failover/fail back

some assorted bits – ctdb

- recovery lock has become optional
- several subcommands added to ctdb (e.g. wipedb)
- eventscript code (in ctddb) has been reworked
- vacuuming and repacking has been streamlined and moved into the daemon
- the tdb code in ctdb synchronized with samba master
- fixed several race conditions and even deadlock in ctdb/samba
- local failover and loadbalancing
 - originally, just one public interface per node (including bonding)
 - new: support for distributing public ips over multiple interfaces per node
 - local loadbalancing and failover/fail back

some assorted bits – ctdb

- recovery lock has become optional
- several subcommands added to ctdb (e.g. wipedb)
- eventscript code (in ctddb) has been reworked
- vacuuming and repacking has been streamlined and moved into the daemon
- the tdb code in ctdb synchronized with samba master
- fixed several race conditions and even deadlock in ctdb/samba
- local failover and loadbalancing
 - originally, just one public interface per node (including bonding)
 - new: support for distributing public ips over multiple interfaces per node
 - local loadbalancing and failover/fail back

some assorted bits – ctdb

- recovery lock has become optional
- several subcommands added to ctdb (e.g. wipedb)
- eventscript code (in ctdbd) has been reworked
- vacuuming and repacking has been streamlined and moved into the daemon
- the tdb code in ctdb synchronized with samba master
- fixed several race conditions and even deadlock in ctdb/samba
- local failover and loadbalancing
 - originally, just one public interface per node (including bonding)
 - new: support for distributing public ips over multiple interfaces per node
 - local loadbalancing and failover/fail back

more assorted bits – samba

- **samba-level tools: dbwrap_tool, dbwrap_torture**
- removed messaging storms when (many) clients exit
- extended serverid

– fixed PID problem

– the serverid extended by a 32bit random number

– a new serverid table database

– a new tool: `serverid_wipe_tool` (cluster)

- smb echo responder

– the file system calls can hang for (long) time

– the server stay responsive (answer the requests) while waiting

– the file system calls are not lost

more assorted bits – samba

- samba-level tools: `dbwrap_tool`, `dbwrap_torture`
- removed messaging storms when (many) clients exit
- extended `serverid`
 - recycled PID problem
 - `serverid` extended by a 32bit random number
 - new `serverid` table database
 - `serverid` automatic sync and cleanup
- smb echo responder
 - the system calls can hang for (long) long
 - servers stay responsive (ambushes prevented) while waiting

more assorted bits – samba

- samba-level tools: `dbwrap_tool`, `dbwrap_torture`
- removed messaging storms when (many) clients exit
- **extended serverid**
 - recycled PID problem
 - serverid extended by a 64bit random number
 - new `serverid.tdb` database
 - new `net serverid wipe` tool (cluster)
- smb echo responder

• the system calls can hang for (long) long time
• stay responsive (server's requests) while waiting
• wait for the server to respond

more assorted bits – samba

- samba-level tools: dbwrap_tool, dbwrap_torture
- removed messaging storms when (many) clients exit
- extended serverid
 - recycled PID problem
 - serverid extended by a 64bit random number
 - new serverid.tdb database
 - new net serverid wipe tool (cluster)
- smb echo responder

• `echo` - the system calls can take for (long) long time to return
• `echo` - stay responsive (ambushes requests) while waiting

more assorted bits – samba

- samba-level tools: dbwrap_tool, dbwrap_torture
- removed messaging storms when (many) clients exit
- extended serverid
 - recycled PID problem
 - **serverid extended by a 64bit random number**
 - new serverid.tdb database
 - new net serverid wipe tool (cluster)
- smb echo responder

• *note: the echo responder can also be configured using the smb.conf file*
• *also, stay responsive, samba has recovered while waiting*

more assorted bits – samba

- samba-level tools: `dbwrap_tool`, `dbwrap_torture`
- removed messaging storms when (many) clients exit
- extended `serverid`
 - recycled PID problem
 - `serverid` extended by a 64bit random number
 - **new `serverid.tdb` database**
 - new `net serverid wipe` tool (cluster)
- `smb echo responder`

• `net serverid wipe` tool (cluster)
• `smb echo responder`

more assorted bits – samba

- samba-level tools: dbwrap_tool, dbwrap_torture
- removed messaging storms when (many) clients exit
- extended serverid
 - recycled PID problem
 - serverid extended by a 64bit random number
 - new serverid.tdb database
 - new net serverid wipe tool (cluster)
- smb echo responder
 - file system calls can hang for (too) long
 - may experience sporadic networked while testing

more assorted bits – samba

- samba-level tools: `dbwrap_tool`, `dbwrap_torture`
- removed messaging storms when (many) clients exit
- extended `serverid`
 - recycled PID problem
 - `serverid` extended by a 64bit random number
 - new `serverid.tdb` database
 - new `net serverid wipe` tool (cluster)
- **smb echo responder**
 - file system calls can hang for (tooo) long
 - stay responsive (`smbecho` requests) while waiting
 - fork `smbecho` responder process

more assorted bits – samba

- samba-level tools: `dbwrap_tool`, `dbwrap_torture`
- removed messaging storms when (many) clients exit
- extended `serverid`
 - recycled PID problem
 - `serverid` extended by a 64bit random number
 - new `serverid.tdb` database
 - new `net serverid wipe` tool (cluster)
- `smb echo responder`
 - file system calls can hang for (tooo) long
 - stay responsive (`smbecho` requests) while waiting
 - fork `smbecho` responder process

more assorted bits – samba

- samba-level tools: `dbwrap_tool`, `dbwrap_torture`
- removed messaging storms when (many) clients exit
- extended `serverid`
 - recycled PID problem
 - `serverid` extended by a 64bit random number
 - new `serverid.tdb` database
 - new `net serverid wipe` tool (cluster)
- `smb echo responder`
 - file system calls can hang for (tooo) long
 - stay responsive (`smbecho` requests) while waiting
 - fork `smbecho` responder process

more assorted bits – samba

- samba-level tools: `dbwrap_tool`, `dbwrap_torture`
- removed messaging storms when (many) clients exit
- extended `serverid`
 - recycled PID problem
 - `serverid` extended by a 64bit random number
 - new `serverid.tdb` database
 - new `net serverid wipe` tool (cluster)
- `smb echo responder`
 - file system calls can hang for (tooo) long
 - stay responsive (`smbecho` requests) while waiting
 - **fork `smbecho` responder process**

tdb check infrastructure

- **tdb_check** code added to **tdb**
- integrated into **ctdb**:
- persistent databases get a health status flag
- **ctdb** startup checks for damaged persistent **tdbs** at startup and after recoveries
- **ctdb** either starts or fails depending on `CTDB_MAX_PERSISTENT_CHECK_ERRORS` ($-1/0$)
- in case it starts, startup event / monitoring is deferred until all persistent **tdbs** are healthy
- **db** can become healthy by:
 - integrates with healthy copy existing the cluster
 - recovers from backup
 - recovers from previous backup

tdb check infrastructure

- `tdb_check` code added to `tdb`
- integrated into `ctdb`:
 - persistent databases get a health status flag
 - `ctdb` startup checks for damaged persistent `tdbs` at startup and after recoveries
 - `ctdb` either starts or fails depending on `CTDB_MAX_PERSISTENT_CHECK_ERRORS` ($-1/0$)
 - in case it starts, startup event / monitoring is deferred until all persistent `tdbs` are healthy
 - `db` can become healthy by:
 - syncs with healthy copy across the cluster
 - recovers from corruption

tdb check infrastructure

- tdb_check code added to tdb
- integrated into ctdb:
- persistent databases get a health status flag
- ctdb startup checks for damaged persistent tdb's at startup and after recoveries
- ctdb either starts or fails depending on CTDB_MAX_PERSISTENT_CHECK_ERRORS (-1/0)
- in case it starts, startup event / monitoring is deferred until all persistent dbs are healthy
- db can become healthy by:
 - being repaired by healthy copy (during the cluster recovery)
 - being manually repaired by the user

tdb check infrastructure

- tdb_check code added to tdb
- integrated into ctdb:
- persistent databases get a health status flag
- ctdb startup checks for damaged persistent tdb's at startup and after recoveries
- ctdb either starts or fails depending on `CTDB_MAX_PERSISTENT_CHECK_ERRORS` ($-1/0$)
- in case it starts, startup event / monitoring is deferred until all persistent dbs are healthy
- db can become healthy by:

• becoming healthy again (after the cluster is healthy)

• becoming healthy again (after the cluster is healthy)

tdb check infrastructure

- tdb_check code added to tdb
- integrated into ctdb:
- persistent databases get a health status flag
- ctdb startup checks for damaged persistent tdb's at startup and after recoveries
- ctdb either starts or fails depending on `CTDB_MAX_PERSISTENT_CHECK_ERRORS` ($-1/0$)
- in case it starts, startup event / monitoring is deferred until all persistent dbs are healthy
- db can become healthy by:

tdb check infrastructure

- tdb_check code added to tdb
- integrated into ctdb:
- persistent databases get a health status flag
- ctdb startup checks for damaged persistent tdb's at startup and after recoveries
- ctdb either starts or fails depending on `CTDB_MAX_PERSISTENT_CHECK_ERRORS` ($-1/0$)
- in case it starts, startup event / monitoring is deferred until all persistent db's are healthy
- db can become healthy by:
 - node with healthy copy entering the cluster

tdb check infrastructure

- tdb_check code added to tdb
- integrated into ctdb:
- persistent databases get a health status flag
- ctdb startup checks for damaged persistent tdb's at startup and after recoveries
- ctdb either starts or fails depending on `CTDB_MAX_PERSISTENT_CHECK_ERRORS` ($-1/0$)
- in case it starts, startup event / monitoring is deferred until all persistent dbs are healthy
- **db can become healthy by:**
 - node with healthy copy entering the cluster
 - admin does `ctdb wipedb` or `ctdb restoredb`

tdb check infrastructure

- tdb_check code added to tdb
- integrated into ctdb:
- persistent databases get a health status flag
- ctdb startup checks for damaged persistent tdb's at startup and after recoveries
- ctdb either starts or fails depending on `CTDB_MAX_PERSISTENT_CHECK_ERRORS` ($-1/0$)
- in case it starts, startup event / monitoring is deferred until all persistent dbs are healthy
- db can become healthy by:
 - node with healthy copy entering the cluster
 - admin does `ctdb wipedb` or `ctdb restoredb`

tdb check infrastructure

- tdb_check code added to tdb
- integrated into ctdb:
- persistent databases get a health status flag
- ctdb startup checks for damaged persistent tdb's at startup and after recoveries
- ctdb either starts or fails depending on `CTDB_MAX_PERSISTENT_CHECK_ERRORS` ($-1/0$)
- in case it starts, startup event / monitoring is deferred until all persistent db's are healthy
- db can become healthy by:
 - node with healthy copy entering the cluster
 - admin does `ctdb wipedb` or `ctdb restoredb`

local failover and loadbalancing

- originally, just one public interface per node (including bonding)
- new: support for distributing public ips over multiple interfaces per node
- local loadbalancing and failover/fail back

local failover and loadbalancing

- originally, just one public interface per node (including bonding)
- new: support for distributing public ips over multiple interfaces per node
- local loadbalancing and failover/fail back

local failover and loadbalancing

- originally, just one public interface per node (including bonding)
- new: support for distributing public ips over multiple interfaces per node
- local loadbalancing and failover/fail back

persistent transactions - history

- 1.0.50, September 2007: support for persistent DBs.
- 1.0.58, August 2008: API level transaction for persistent DBs
- 1.0.108, December 2009: Various race fixes for transactions
- 1.0.109, December 2009: Rewrite of transaction code

persistent transactions - history

- 1.0.50, September 2007: support for persistent DBs.
- 1.0.58, August 2008: API level transaction for persistent DBs
- 1.0.108, December 2009: Various race fixes for transactions
- 1.0.109, December 2009: Rewrite of transaction code

persistent transactions - history

- 1.0.50, September 2007: support for persistent DBs.
- 1.0.58, August 2008: API level transaction for persistent DBs
- 1.0.108, December 2009: Various race fixes for transactions
- 1.0.109, December 2009: Rewrite of transaction code

persistent transactions - history

- 1.0.50, September 2007: support for persistent DBs.
- 1.0.58, August 2008: API level transaction for persistent DBs
- 1.0.108, December 2009: Various race fixes for transactions
- 1.0.109, December 2009: Rewrite of transaction code

persistent transactions

- lock entire DB in a global lock
- perform R/W ops in memory (prepare a marshall buffer)
- at commit distribute changes to other nodes and write to LTDB in a local transaction
- finally drop global lock
- note: new `net_g_lock` tool

persistent transactions

- lock entire DB in a global lock
- perform R/W ops in memory (prepare a marshall buffer)
- at commit distribute changes to other nodes and write to LTDB in a local transaction
- finally drop global lock
- note: new `net_g_lock` tool

persistent transactions

- lock entire DB in a global lock
- perform R/W ops in memory (prepare a marshall buffer)
- at commit distribute changes to other nodes and write to LTDB in a local transaction
- finally drop global lock
- note: new `net_g_lock` tool

persistent transactions

- lock entire DB in a global lock
- perform R/W ops in memory (prepare a marshall buffer)
- at commit distribute changes to other nodes and write to LTDB in a local transaction
- finally drop global lock
- note: new `net_g_lock` tool

persistent transactions

- lock entire DB in a global lock
- perform R/W ops in memory (prepare a marshall buffer)
- at commit distribute changes to other nodes and write to LTDB in a local transaction
- finally drop global lock
- note: new `net g_lock` tool

Outline

- 1 Refresher on CTDB
- 2 Growing...
- 3 Recent Advances
- 4 Ongoing Tasks

(re)started: idmap rewrite

- idmap write performance (tdb2)
- several persistent transactions per idmap
- rewrite in the lines of my sambaXP 2009 talk started
 - removes all the allocation methods from winbind's surface
 - replaces the winbind id mapping API and idmap backend methods to take in size and size to side
 - removes the single id allocator
- problems: allocator used in group mapping ldapsam:editposix

(re)started: idmap rewrite

- idmap write performance (tdb2)
- several persistent transactions per idmap
- rewrite in the lines of my sambaXP 2009 talk started
 - remove all the allocation methods from winbindd's surface
 - replace the winbind id mapping API and winbind backend methods by a single `idmap_get` and `idmap_put`
 - removes the single id allocator
- problems: allocator used in group mapping `ldapsam:editposix`

(re)started: idmap rewrite

- idmap write performance (tdb2)
- several persistent transactions per idmap
- **rewrite in the lines of my sambaXP 2009 talk started**
 - remove all the allocation methods from winbindd's surface
 - reduce the winbindd id mapping API and idmap backend methods to `sids_to_xids` and `xids_to_sids`
 - removes the single xid allocator
- problems: allocator used in group mapping `ldapsam:editposix`

(re)started: idmap rewrite

- idmap write performance (tdb2)
- several persistent transactions per idmap
- rewrite in the lines of my sambaXP 2009 talk started
 - remove all the allocation methods from winbindd's surface
 - reduce the winbindd id mapping API and idmap backend methods to `sids_to_xids` and `xids_to_sids`
 - removes the single xid allocator
- problems: allocator used in group mapping `ldapsam:editposix`

(re)started: idmap rewrite

- idmap write performance (tdb2)
- several persistent transactions per idmap
- rewrite in the lines of my sambaXP 2009 talk started
 - remove all the allocation methods from winbindd's surface
 - reduce the winbindd id mapping API and idmap backend methods to `sids_to_xids` and `xids_to_sids`
 - removes the single xid allocator
- problems: allocator used in group mapping `ldapsam:editposix`

(re)started: idmap rewrite

- idmap write performance (tdb2)
- several persistent transactions per idmap
- rewrite in the lines of my sambaXP 2009 talk started
 - remove all the allocation methods from winbindd's surface
 - reduce the winbindd id mapping API and idmap backend methods to `sids_to_xids` and `xids_to_sids`
 - removes the single xid allocator
- problems: allocator used in group mapping `ldapsam:editposix`

(re)started: idmap rewrite

- idmap write performance (tdb2)
- several persistent transactions per idmap
- rewrite in the lines of my sambaXP 2009 talk started
 - remove all the allocation methods from winbindd's surface
 - reduce the winbindd id mapping API and idmap backend methods to `sids_to_xids` and `xids_to_sids`
 - removes the single xid allocator
- problems: allocator used in group mapping `ldapsam:editposix`

ongoing and future tasks

- develop ctdb client library `libctdb`
- develop (more) tools for maintenance and diagnosis
- ...
- SMB2 (?)
- ...

ongoing and future tasks

- develop ctdb client library `libctdb`
- develop (more) tools for maintenance and diagnosis
- ...
- SMB2 (?)
- ...

ongoing and future tasks

- develop ctdb client library `libctdb`
- develop (more) tools for maintenance and diagnosis
- ...
- SMB2 (?)
- ...

ongoing and future tasks

- develop ctdb client library `libctdb`
- develop (more) tools for maintenance and diagnosis
- ...
- SMB2 (?)
- ...

ongoing and future tasks

- develop ctdb client library `libctdb`
- develop (more) tools for maintenance and diagnosis
- ...
- SMB2 (?)
- ...

Questions?