Lessons Learned in Clustered Samba
sambaXP 2010

Michael Adam

obnox@samba.org

Samba Team / SerNet

2010-05-06

Outline

Outline

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
- $\Rightarrow$ 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
- $\Rightarrow$ 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
- $\Rightarrow$ 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
- $\Rightarrow$ 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
- $\Rightarrow$ 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`
- `http://ctdb.samba.org/packages/` (RPMs, Sources)
- warning: there is no elaborate release process
- packagers/integrators: better check with developers
- irc: `#ctdb` ond 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` ond 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` ond 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 ond 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 ond 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` ond 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 ond 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` ond 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 ...)
- 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 ...)
- 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 ...)
- 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 ...)
- plugable *event script* architecture

Outline

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

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

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/fallback

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 (no bonding, bridges)
  - 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 (no bonding, no vlans)
  - new: support for distributing public ips over multiple interfaces per node
  - local loadbalancing and failover/fallback

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 (no bonding)
  - new: support for distributing public ips over multiple interfaces per node
  - local loadbalancing and failover fall 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

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 lodabalancing 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 lodabalancing 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 lodabalancing 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 lodabalancing and failover/fail back

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 (dwarer)
- smb echo responder
  - file system calls can hang for (too?) 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 (dwerner)
- smb echo responder
  - file system calls can hang for (long) time
  - 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 (too) 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 (long) time
  - 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 (long) time
  - 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 (long) time
  - 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

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 damadged 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 dbs are healthy
- db can become healthy by:
  - a node with healthy copy entering the cluster
  - by admin (`ctdb wipedb` or `ctdb cleardbhealth`)

tdb check infrastructure

- `tdb_check` code added to `tdb`
- integrated into ctdb:
- persistent databases get a health status flag
- ctdb startup checks for damadged 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 dbs are healthy
- db can become healthy by:
  - a node with healthy copy entering the cluster
  - the 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 damadged 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 dbs are healthy
- db can become healthy by:
  - a node with healthy copy entering the cluster
  - the admin does `ctdb wipedb` or `ctdb tdbtool wipe`

tdb check infrastructure

- `tdb_check` code added to `tdb`
- integrated into ctdb:
- persistent databases get a health status flag
- ctdb startup checks for damadged 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 dbs are healthy
- db can become healthy by:
  - a node with healthy copy entering the cluster
  - or admin does `ctdb wipedb` or `ctdb backupdb`

tdb check infrastructure

- `tdb_check` code added to `tdb`
- integrated into ctdb:
- persistent databases get a health status flag
- ctdb startup checks for damadged 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 dbs are healthy
- db can become healthy by:
  - a node with healthy copy entering the cluster
  - an 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 damadged 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 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 damadged 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 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 damadged 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 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 damadged 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 dbs 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 lodabalancing 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 lodabalancing 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 lodabalancing 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

(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 id allocators
- 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 id allocators
- 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?