CTDB database vacuuming for geniuses

Amitay Isaacs
amitay@samba.org

Samba Team
IBM (Australia Development Labs, Linux Technology Center)

CTDB Project

Motivation: Support for clustered Samba

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

Features:

- Volatile and Persistent databases
- Cluster-side messaging for Samba
- IP failover and load balancing
- Service monitoring

Community:

- `https://wiki.samba.org`
- `git://git.samba.org/samba.git`

Overview

- Database models
- Volatile database
  - Data model
  - Vacuuming
  - Recovery
- Zombie outbreak
  - Resurrection of the dead
  - Keeping 'em dead
- Restructure

Database models

Volatile	Persistent
distributed data	replicated data
single copy	multiple copies
data loss on failure	loss-less
per-node per-db per-chain mutex	clusterwide per-db mutex
disk backed	disk backed
cluster-wide traverse	local traverse
shared access	client-server access

Volatile database

Volatile database – data model – record creation

Volatile database – data model – record creation

- `Node1`: Samba requests `record1`

Volatile database – data model – record creation

- `Node1`: Samba requests `record1`
- `Node1`: Where is `record1`?

Volatile database – data model – record creation

- `Node1`: Samba requests `record1`
- `Node1`: Where is `record1`?

Volatile database – data model – record creation

- `Node1`: Samba requests `record1`
- `Node1`: Where is `record1`?
- `Node1`: Request for `record1`

Volatile database – data model – record creation

- `Node1`: Samba requests `record1`
- `Node1`: Where is `record1`?
- `Node1`: Request for `record1`
- `Node2`: Where is `record1`?

Volatile database – data model – record creation

- `Node1`: Samba requests `record1`
- `Node1`: Where is `record1`?
- `Node1`: Request for `record1`
- `Node2`: Where is `record1`?

Volatile database – data model – record creation

- `Node1`: Samba requests `record1`
- `Node1`: Where is `record1`?
- `Node1`: Request for `record1`
- `Node2`: Where is `record1`?
- `Node2`: Create empty `record1`

Volatile database – data model – record creation

- `Node1`: Samba requests `record1`
- `Node1`: Where is `record1`?
- `Node1`: Request for `record1`
- `Node2`: Where is `record1`?
- `Node2`: Create empty `record1`
- `Node2`: Migrate `record1` to `Node1`

Volatile database – data model – record creation

- `Node1`: Samba requests `record1`
- `Node1`: Where is `record1`?
- `Node1`: Request for `record1`
- `Node2`: Where is `record1`?
- `Node2`: Create empty `record1`
- `Node2`: Migrate `record1` to `Node1`
- `Node1`: Receive `record1`

Volatile database – data model – record creation

- `Node1`: Samba requests `record1`
- `Node1`: Where is `record1`?
- `Node1`: Request for `record1`
- `Node2`: Where is `record1`?
- `Node2`: Create empty `record1`
- `Node2`: Migrate `record1` to `Node1`
- `Node1`: Receive `record1`
- `Node1`: Send reply to samba

Volatile database – data model – local access

Volatile database – data model – local access

- `Node1`: Samba needs `record1`

Volatile database – data model – local access

- `Node1`: Samba needs `record1`
- `Node1`: Samba checks if `record1` is local

Volatile database – data model – local access

- `Node1`: Samba needs `record1`
- `Node1`: Samba checks if `record1` is local
- `Node1`: Samba uses `record1`

Volatile database – data model – local access

- `Node1`: Samba needs `record1`
- `Node1`: Samba checks if `record1` is local
- `Node1`: Samba uses `record1`
- `Node1`: Samba uses `record1`

Volatile database – data model – local access

- `Node1`: Samba needs `record1`
- `Node1`: Samba checks if `record1` is local
- `Node1`: Samba uses `record1`
- `Node1`: Samba uses `record1`
- `Node1`: Samba uses `record1`

Volatile database – data model – local access

- `Node1`: Samba needs `record1`
- `Node1`: Samba checks if `record1` is local
- `Node1`: Samba uses `record1`
- `Node1`: Samba uses `record1`
- `Node1`: Samba uses `record1`
- CTDB is not involved

Volatile database – data model – remote access

Volatile database – data model – remote access

- Node3: Samba requests `record1`

Volatile database – data model – remote access

- `Node3`: Samba requests `record1`
- `Node3`: Where is `record1`?

Volatile database – data model – remote access

- `Node3`: Samba requests `record1`
- `Node3`: Where is `record1`?
- `Node3`: Request for `record1`

Volatile database – data model – remote access

- `Node3`: Samba requests `record1`
- `Node3`: Where is `record1`?
- `Node3`: Request for `record1`
- `Node2`: Where is `record1`?

Volatile database – data model – remote access

- Node3: Samba requests `record1`
- Node3: Where is `record1`?
- Node3: Request for `record1`
- Node2: Where is `record1`?
- Node2: Redirect request for `record1`

Volatile database – data model – remote access

- `Node3`: Samba requests `record1`
- `Node3`: Where is `record1`?
- `Node3`: Request for `record1`
- `Node2`: Where is `record1`?
- `Node2`: Redirect request for `record1`
- `Node2`: Where is `record1`?

Volatile database – data model – remote access

- Node3: Samba requests record1
- Node3: Where is record1?
- Node3: Request for record1
- Node2: Where is record1?
- Node2: Redirect request for record1
- Node2: Where is record1?
- Node1: Migrate record1 to Node2

Volatile database – data model – remote access

- Node3: Samba requests record1
- Node3: Where is record1?
- Node3: Request for record1
- Node2: Where is record1?
- Node2: Redirect request for record1
- Node2: Where is record1?
- Node1: Migrate record1 to Node2
- Node2: Receive record1 and update location

Volatile database – data model – remote access

- Node3: Samba requests record1
- Node3: Where is record1?
- Node3: Request for record1
- Node2: Where is record1?
- Node2: Redirect request for record1
- Node2: Where is record1?
- Node1: Migrate record1 to Node2
- Node2: Receive record1 and update location
- Node2: Migrate record1 to Node3

Volatile database – data model – remote access

- `Node3`: Samba requests `record1`
- `Node3`: Where is `record1`?
- `Node3`: Request for `record1`
- `Node2`: Where is `record1`?
- `Node2`: Redirect request for `record1`
- `Node2`: Where is `record1`?
- `Node1`: Migrate `record1` to `Node2`
- `Node2`: Receive `record1` and update location
- `Node2`: Migrate `record1` to `Node3`
- `Node3`: Receive `record1`

Volatile database – data model – remote access

- `Node3`: Samba requests `record1`
- `Node3`: Where is `record1`?
- `Node3`: Request for `record1`
- `Node2`: Where is `record1`?
- `Node2`: Redirect request for `record1`
- `Node2`: Where is `record1`?
- `Node1`: Migrate `record1` to `Node2`
- `Node2`: Receive `record1` and update location
- `Node2`: Migrate `record1` to `Node3`
- `Node3`: Receive `record1`
- `Node3`: Send reply to samba

Volatile database – data model – record deletion

Volatile database – data model – record deletion

- Node3: Samba deletes `record1`

Volatile database – data model – record deletion

- `Node3`: Samba deletes `record1`
- Record is actually **not deleted**

Volatile database – data model – record deletion

- `Node3`: Samba deletes `record1`
- Record is actually **not deleted**
  - There are copies of the `record1` on all nodes

Volatile database – data model – record deletion

- `Node3`: Samba deletes `record1`
- Record is actually **not deleted**
  - There are copies of the `record1` on all nodes
  - How to delete all copies?

Volatile database – data model – record deletion

- `Node3`: Samba deletes `record1`
- Record is actually **not deleted**
  - There are copies of the `record1` on all nodes
  - How to delete all copies?
- `Node3`: Record is marked to be deleted

Volatile database – data model – record deletion

- `Node3`: Samba deletes `record1`
- Record is actually **not deleted**
  - There are copies of the `record1` on all nodes
  - How to delete all copies?
- `Node3`: Record is marked to be deleted
- **Vacuuming**

Volatile database – vacuuming

Volatile database – vacuuming

- `Node3`: Record is marked for deletion

Volatile database – vacuuming

- `Node3: Record is marked for deletion`
- `Node3: Vacuuming: Send the record to` **lmaster**

Volatile database – vacuuming

- `Node3`: Record is marked for deletion
- `Node3`: Vacuuming: Send the record to **lmaster**
- `Node2`: Receive record & mark for deletion

Volatile database – vacuuming

- `Node3`: Record is marked for deletion
- `Node3`: Vacuuming: Send the record to **lmaster**
- `Node2`: Receive record & mark for deletion
- `Node2`: Vacuuming: Delete record from other nodes

Volatile database – vacuuming

- `Node3`: Record is marked for deletion
- `Node3`: Vacuuming: Send the record to **lmaster**
- `Node2`: Receive record & mark for deletion
- `Node2`: Vacuuming: Delete record from other nodes

Volatile database – vacuuming

- `Node3`: Record is marked for deletion
- `Node3`: Vacuuming: Send the record to **lmaster**
- `Node2`: Receive record & mark for deletion
- `Node2`: Vacuuming: Delete record from other nodes
- `Node2`: Vacuuming: Delete record locally

Volatile database – vacuuming

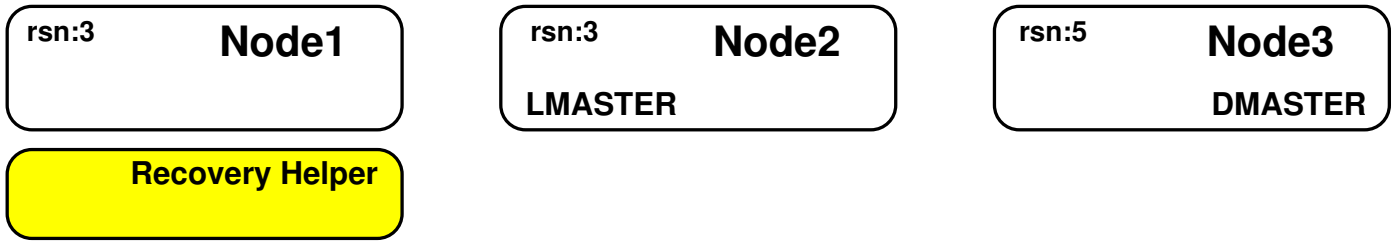
- `Node3: Record is marked for deletion`
- `Node3: Vacuuming: Send the record to` **lmaster**
- `Node2: Receive record & mark for deletion`
- `Node2: Vacuuming: Delete record from other nodes`
- `Node2: Vacuuming: Delete record locally`

Volatile database – recovery

Volatile database – recovery

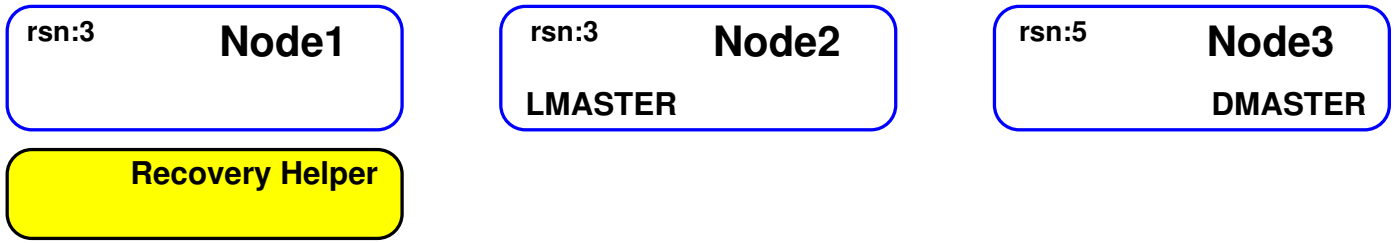
- `Node1`: Starts database recovery

Volatile database – recovery



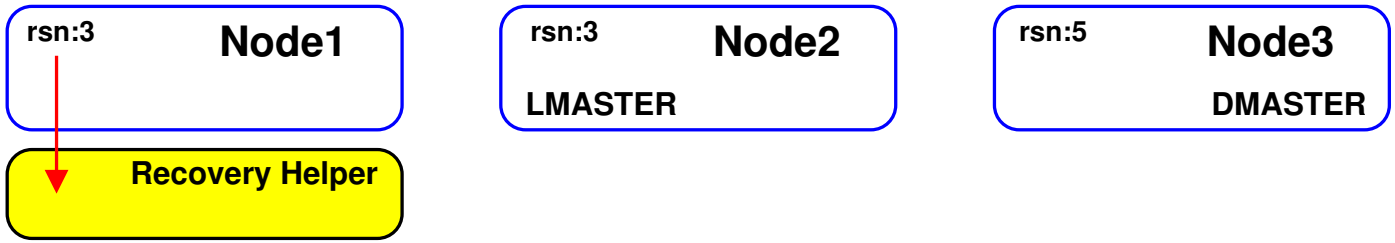
- `Node1`: Starts database recovery

Volatile database – recovery



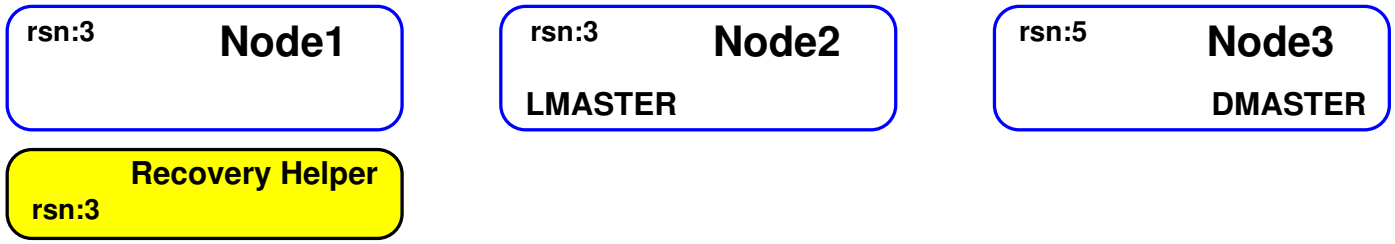
- `Node1`: Starts database recovery
- Recovery: Freeze database

Volatile database – recovery



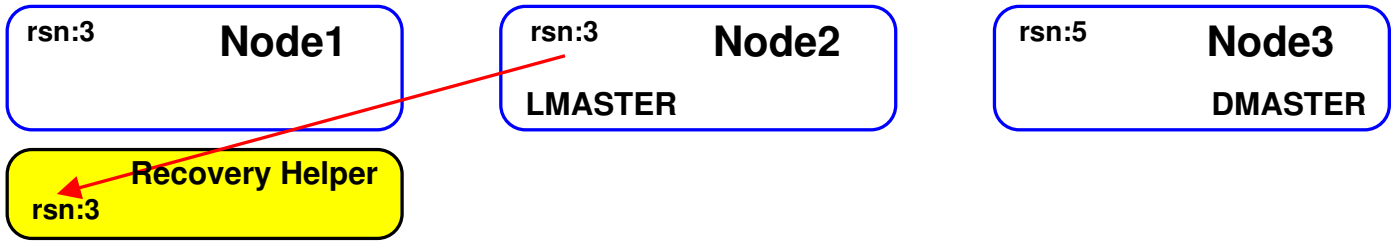
- `Node1`: Starts database recovery
- Recovery: Freeze database
- Recovery: Collect records from `Node1`

Volatile database – recovery



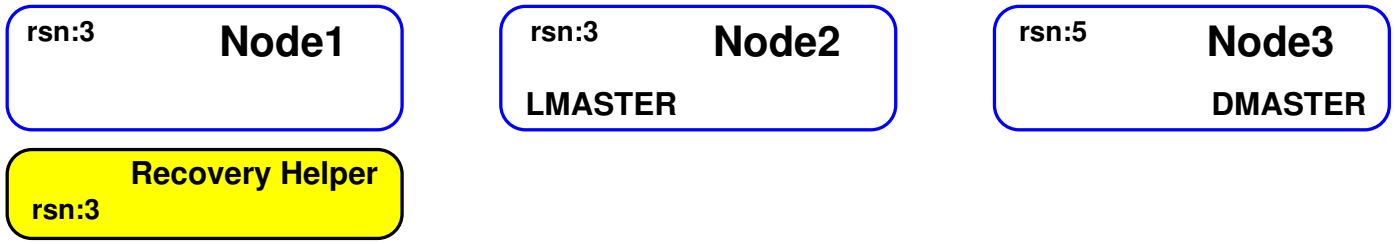
- `Node1`: Starts database recovery
- Recovery: Freeze database
- Recovery: Collect records from `Node1`

Volatile database – recovery



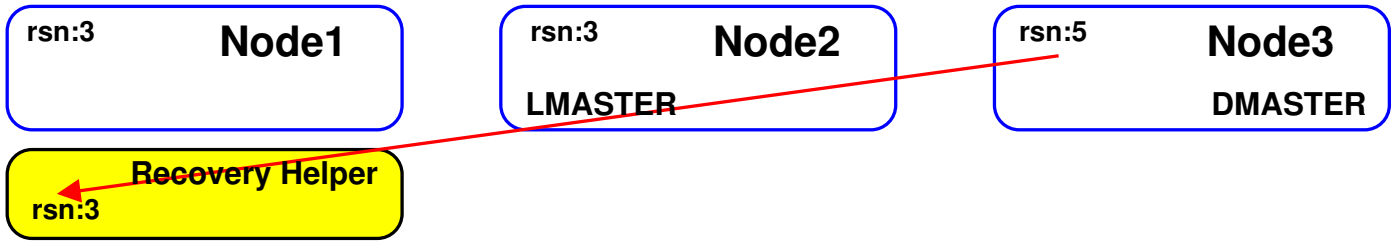
- `Node1`: Starts database recovery
- Recovery: Freeze database
- Recovery: Collect records from `Node1`
- Recovery: Collect records from `Node2`

Volatile database – recovery



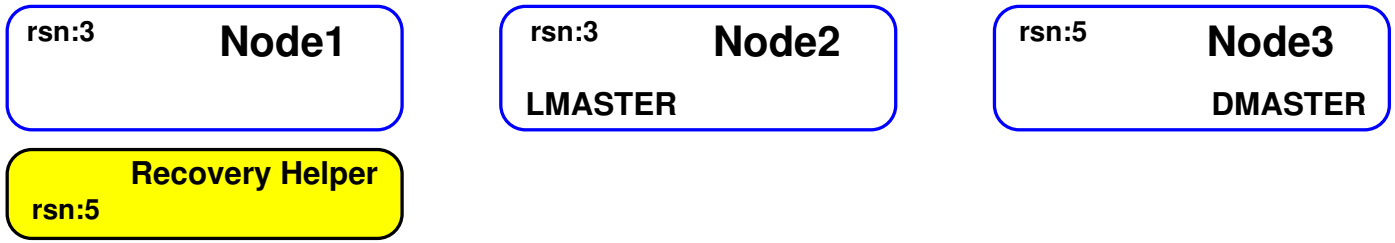
- `Node1`: Starts database recovery
- Recovery: Freeze database
- Recovery: Collect records from `Node1`
- Recovery: Collect records from `Node2`

Volatile database – recovery



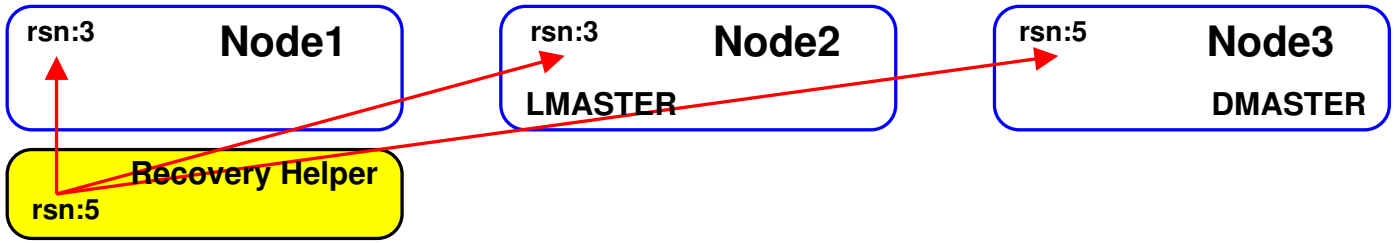
- `Node1`: Starts database recovery
- Recovery: Freeze database
- Recovery: Collect records from `Node1`
- Recovery: Collect records from `Node2`
- Recovery: Collect records from `Node3`

Volatile database – recovery



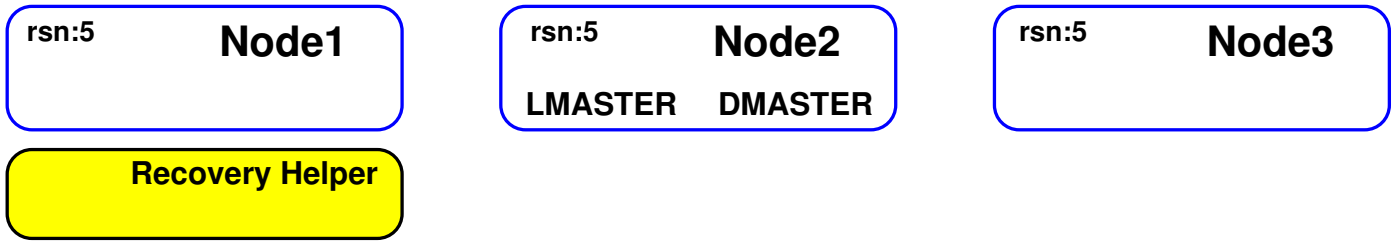
- `Node1`: Starts database recovery
- Recovery: Freeze database
- Recovery: Collect records from `Node1`
- Recovery: Collect records from `Node2`
- Recovery: Collect records from `Node3`

Volatile database – recovery



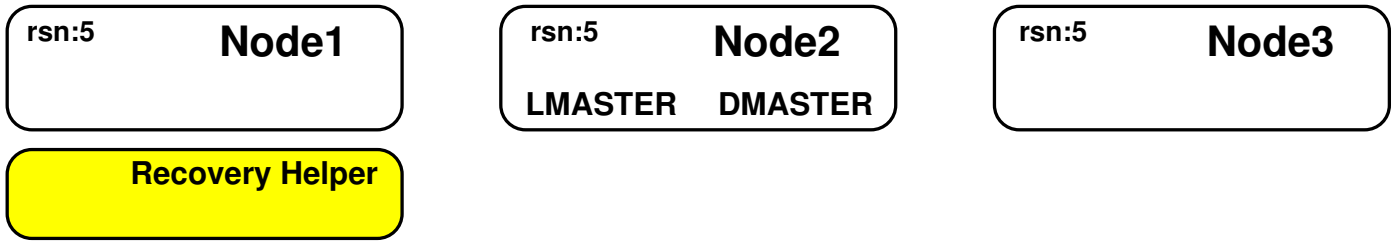
- `Node1`: Starts database recovery
- Recovery: Freeze database
- Recovery: Collect records from `Node1`
- Recovery: Collect records from `Node2`
- Recovery: Collect records from `Node3`
- Recovery: Push the records to all the nodes

Volatile database – recovery



- `Node1`: Starts database recovery
- Recovery: Freeze database
- Recovery: Collect records from `Node1`
- Recovery: Collect records from `Node2`
- Recovery: Collect records from `Node3`
- Recovery: Push the records to all the nodes

Volatile database – recovery



- `Node1`: Starts database recovery
- Recovery: Freeze database
- Recovery: Collect records from `Node1`
- Recovery: Collect records from `Node2`
- Recovery: Collect records from `Node3`
- Recovery: Push the records to all the nodes
- Recovery: Thaw database

Zombie outbreak

Resurrection of the dead – background

Resurrection of the dead – background

- Inactive node(s)

Resurrection of the dead – background

- Inactive node(s)
  - Node is stopped

Resurrection of the dead – background

- Inactive node(s)
  - Node is stopped
  - Node is banned

Resurrection of the dead – background

- Inactive node(s)
  - Node is stopped
  - Node is banned
- What happens?

Resurrection of the dead – background

- Inactive node(s)
  - Node is stopped
  - Node is banned
- What happens?
  - VNN map changes

Resurrection of the dead – background

- Inactive node(s)
  - Node is stopped
  - Node is banned
- What happens?
  - VNN map changes
  - . . . causing database recovery

Resurrection of the dead – background

- Inactive node(s)
  - Node is stopped
  - Node is banned
- What happens?
  - VNN map changes
  - . . . causing database recovery
- What happens when node becomes active again?

Resurrection of the dead – background

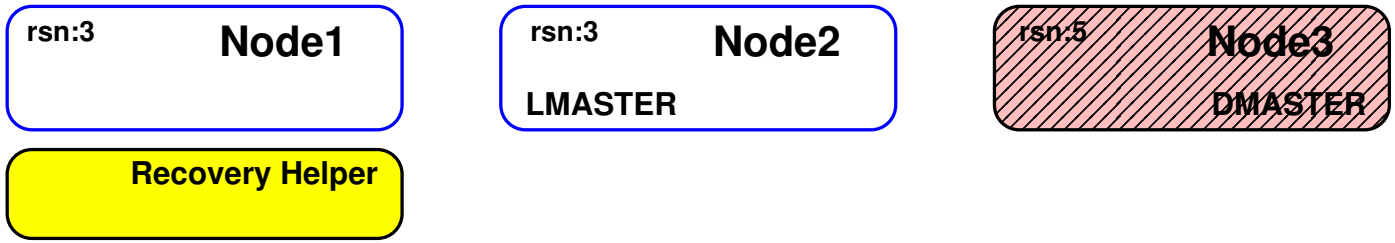
- Inactive node(s)
  - Node is stopped
  - Node is banned
- What happens?
  - VNN map changes
  - . . . causing database recovery
- What happens when node becomes active again?
  - VNN map changes
  - . . . causing database recovery

Resurrection of the dead – example

Resurrection of the dead – example

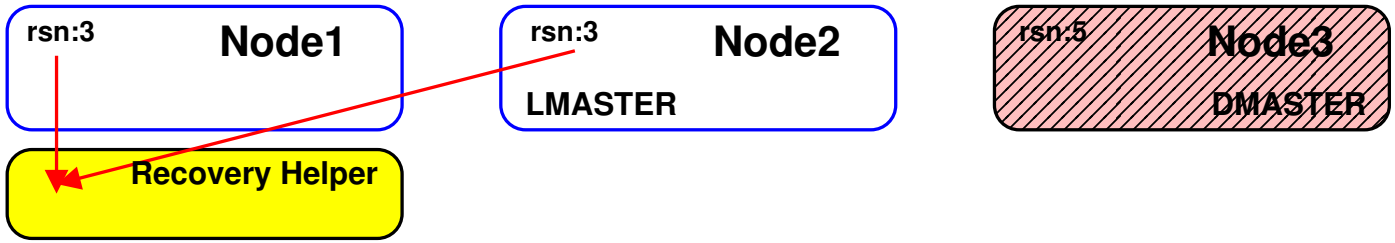
- `Node3` becomes **inactive**

Resurrection of the dead – example



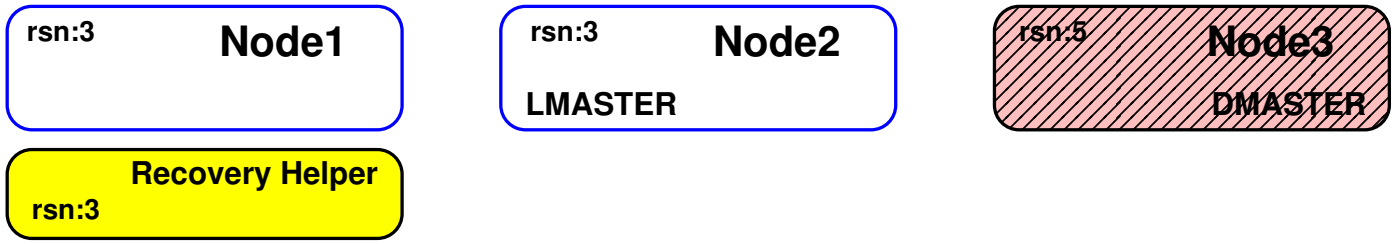
- `Node3` becomes **inactive**
- `Node1` starts recovery

Resurrection of the dead – example



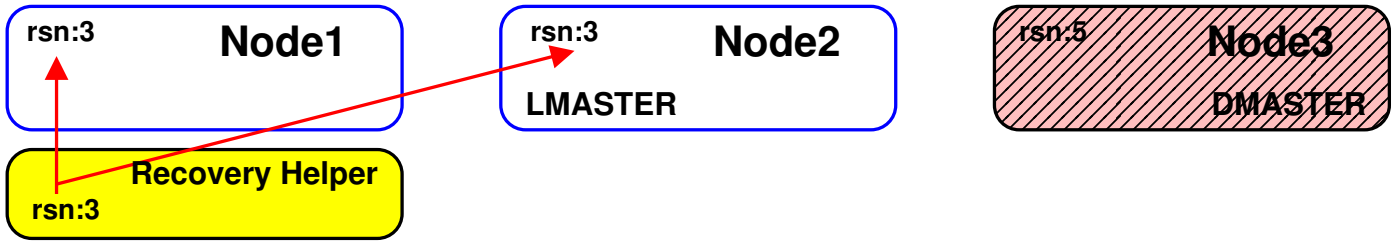
- `Node3` becomes **inactive**
- `Node1` starts recovery
- Recovery: Collect records from `Node1` & `Node2`

Resurrection of the dead – example



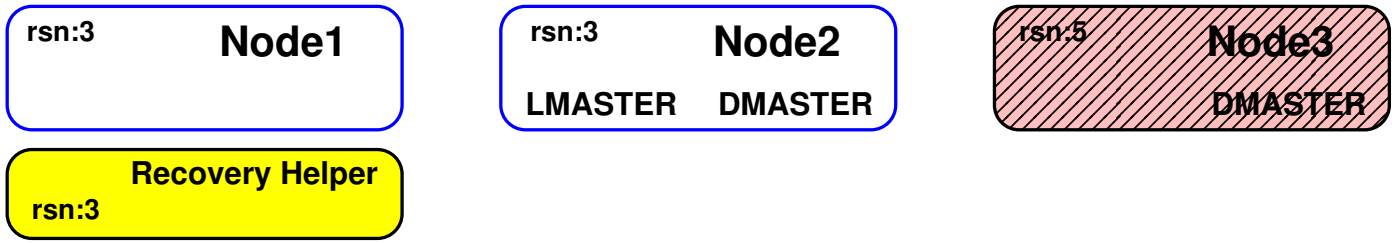
- `Node3` becomes **inactive**
- `Node1` starts recovery
- Recovery: Collect records from `Node1` & `Node2`

Resurrection of the dead – example



- `Node3` becomes **inactive**
- `Node1` starts recovery
- Recovery: Collect records from `Node1` & `Node2`
- Recovery: Push the records to `Node1` & `Node2`

Resurrection of the dead – example



- `Node3` becomes **inactive**
- `Node1` starts recovery
- Recovery: Collect records from `Node1` & `Node2`
- Recovery: Push the records to `Node1` & `Node2`

Resurrection of the dead – example

- `Node3` becomes **inactive**
- `Node1` starts recovery
- Recovery: Collect records from `Node1` & `Node2`
- Recovery: Push the records to `Node1` & `Node2`
- Recovery done

Resurrection of the dead – example

- `Node3` becomes **inactive**
- `Node1` starts recovery
- Recovery: Collect records from `Node1` & `Node2`
- Recovery: Push the records to `Node1` & `Node2`
- Recovery done
- `Node2`: Samba deletes record

Resurrection of the dead – example

- `Node3` becomes **inactive**
- `Node1` starts recovery
- Recovery: Collect records from `Node1` & `Node2`
- Recovery: Push the records to `Node1` & `Node2`
- Recovery done
- `Node2`: Samba deletes record
- `Node2`: Vacuuming removes record

Resurrection of the dead – example

- `Node3` becomes **inactive**
- `Node1` starts recovery
- Recovery: Collect records from `Node1` & `Node2`
- Recovery: Push the records to `Node1` & `Node2`
- Recovery done
- `Node2`: Samba deletes record
- `Node2`: Vacuuming removes record
- `Node3` becomes **active**

Resurrection of the dead – example

- `Node3` becomes **inactive**
- `Node1` starts recovery
- Recovery: Collect records from `Node1` & `Node2`
- Recovery: Push the records to `Node1` & `Node2`
- Recovery done
- `Node2`: Samba deletes record
- `Node2`: Vacuuming removes record
- `Node3` becomes **active**
- Zombie record!

Keeping 'em dead

Keeping 'em dead

- When `Node3` becomes **inactive**,

Keeping 'em dead

- When `Node3` becomes **inactive**,
- . . . invalidate all records

Keeping 'em dead

- When `Node3` becomes **inactive**,
- . . . invalidate all records
- . . . database recovery

Keeping 'em dead

- When `Node3` becomes **inactive**,
- . . . invalidate all records
- . . . database recovery
- Record is deleted and vacuumed

Keeping 'em dead

- When `Node3` becomes **inactive**,
- . . . invalidate all records
- . . . database recovery
- Record is deleted and vacuumed
- `Node3` becomes **active**,

Keeping 'em dead

- When `Node3` becomes **inactive**,
- . . . invalidate all records
- . . . database recovery
- Record is deleted and vacuumed
- `Node3` becomes **active**,
- . . . database recovery

Keeping 'em dead

- When `Node3` becomes **inactive**,
- . . . invalidate all records
- . . . database recovery
- Record is deleted and vacuumed
- `Node3` becomes **active**,
- . . . database recovery
- Patches coming soon

Restructure

Existing issues with vacuuming

Existing issues with vacuuming

- Vacuuming done in a child process

Existing issues with vacuuming

- Vacuuming done in a child process
  - Forks a process from main ctdb daemon

Existing issues with vacuuming

- Vacuuming done in a child process
  - Forks a process from main ctdb daemon
  - Can be very expensive

Existing issues with vacuuming

- Vacuuming done in a child process
  - Forks a process from main ctdb daemon
  - Can be very expensive
- Migrating deleted record to lmaster

Existing issues with vacuuming

- Vacuuming done in a child process
  - Forks a process from main ctdb daemon
  - Can be very expensive
- Migrating deleted record to lmaster
  - CTDB daemon itself cannot start record migration

Existing issues with vacuuming

- Vacuuming done in a child process
  - Forks a process from main ctdb daemon
  - Can be very expensive
- Migrating deleted record to lmaster
  - CTDB daemon itself cannot start record migration
  - Migration is triggered from a client

Existing issues with vacuuming

- Vacuuming done in a child process
  - Forks a process from main ctdb daemon
  - Can be very expensive
- Migrating deleted record to lmaster
  - CTDB daemon itself cannot start record migration
  - Migration is triggered from a client
  - Hook vacuuming migration in recovery daemon (ugh!)

Existing issues with vacuuming

- Vacuuming done in a child process
  - Forks a process from main ctdb daemon
  - Can be very expensive
- Migrating deleted record to lmaster
  - CTDB daemon itself cannot start record migration
  - Migration is triggered from a client
  - Hook vacuuming migration in recovery daemon (ugh!)
- Deleting empty record from other nodes

Existing issues with vacuuming

- Vacuuming done in a child process
  - Forks a process from main ctdb daemon
  - Can be very expensive
- Migrating deleted record to lmaster
  - CTDB daemon itself cannot start record migration
  - Migration is triggered from a client
  - Hook vacuuming migration in recovery daemon (ugh!)
- Deleting empty record from other nodes
  - Special control for deleting records

Existing issues with vacuuming

- Vacuuming done in a child process
  - Forks a process from main ctdb daemon
  - Can be very expensive
- Migrating deleted record to lmaster
  - CTDB daemon itself cannot start record migration
  - Migration is triggered from a client
  - Hook vacuuming migration in recovery daemon (ugh!)
- Deleting empty record from other nodes
  - Special control for deleting records
  - Out-of-band modification to database

Existing issues with vacuuming

- Vacuuming done in a child process
  - Forks a process from main ctdb daemon
  - Can be very expensive
- Migrating deleted record to lmaster
  - CTDB daemon itself cannot start record migration
  - Migration is triggered from a client
  - Hook vacuuming migration in recovery daemon (ugh!)
- Deleting empty record from other nodes
  - Special control for deleting records
  - Out-of-band modification to database
  - Potential race with recovery

Existing issues with vacuuming

- Vacuuming done in a child process
  - Forks a process from main ctdb daemon
  - Can be very expensive
- Migrating deleted record to lmaster
  - CTDB daemon itself cannot start record migration
  - Migration is triggered from a client
  - Hook vacuuming migration in recovery daemon (ugh!)
- Deleting empty record from other nodes
  - Special control for deleting records
  - Out-of-band modification to database
  - Potential race with recovery
- Hopefully, re-structuring will solve these problems!

CTDB restructure

- Continuing from Martin's talk . . .

CTDB restructure

- Continuing from Martin's talk . . .
- Database daemon

- Vacuuming daemon

CTDB restructure

- Continuing from Martin's talk . . .
- Database daemon
  - Focus on database operations
- Vacuuming daemon

CTDB restructure

- Continuing from Martin's talk . . .
- Database daemon
  - Focus on database operations
  - Single daemon – Most operations are done in child processes
- Vacuuming daemon

CTDB restructure

- Continuing from Martin's talk . . .
- Database daemon
  - Focus on database operations
  - Single daemon – Most operations are done in child processes
- Vacuuming daemon
  - Light-weight process

CTDB restructure

- Continuing from Martin's talk . . .
- Database daemon
  - Focus on database operations
  - Single daemon – Most operations are done in child processes
- Vacuuming daemon
  - Light-weight process
  - Keep track of deleted records

CTDB restructure

- Continuing from Martin's talk . . .
- Database daemon
  - Focus on database operations
  - Single daemon – Most operations are done in child processes
- Vacuuming daemon
  - Light-weight process
  - Keep track of deleted records
  - Handle failures better (what if records cannot deleted?)

CTDB restructure

- Continuing from Martin's talk . . .
- Database daemon
  - Focus on database operations
  - Single daemon – Most operations are done in child processes
- Vacuuming daemon
  - Light-weight process
  - Keep track of deleted records
  - Handle failures better (what if records cannot deleted?)
- Hope that someone will write the code

Questions / Comments