Mathias Dietz

Christian Ambach

Centralized configuration management using registry tdb in a CTDB cluster

Introduction

- Christian Ambach and Mathias Dietz

  Working for *IBM Research and Development* in Mainz on the IBM SONAS product since 2008. Have experiences with Samba as part of the IBM SoFS offering since 2006 and the IBM OESV offering since 2003.

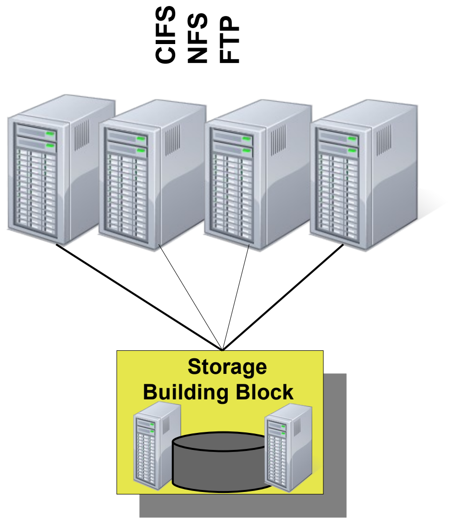
IBM SONAS - Scale Out Network Attached Storage

Modular high performance storage with massive scalability and high availability.

Supports multiple petabytes of storage for organizations that need billions of files in a single file system.

Based on Open Source technologies (Samba, CTDB, Linux, ...)

Link: http://www.ibm.com/systems/storage/network/sonas/



Why do you need a Centralized Configuration

In large cluster environments it is hard to keep the configuration in sync

IBM SONAS supports up to 96 cluster nodes of which 31 participate in a CTDB cluster

Writing configuration changes to many cluster nodes can be tricky

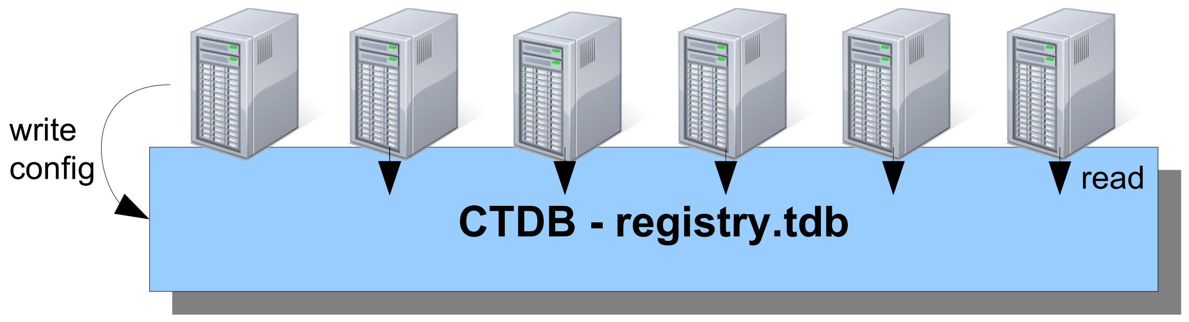
- If something fails during the change some of the nodes might be updated already some not.
- If cluster nodes are down during configuration changes they will not be changed at all
- Administrators are in favor of changing config files on a single node directly
- When adding new nodes to the cluster the configuration must be synced manually

CTDB already distributes persistent TDBs like secrets.tdb, registry.tdb, etc. across all nodes,so why not use this mechanism for configuration data ?!

General Approach

The registry format is well structured and supports many data types

- Store cluster wide configuration in the registry tdb.
- Let CTDB distribute the changes across all cluster nodes.
- Always get the most current configuration out of the registry.



This approach has been introduced in IBM SoFS 1.5 (August 2008)

Use the registry for Samba configuration

- Samba can store its configuration in the clustered registry
  - “`include=registry`” option in smb.conf
- No need to keep smb.conf synchronized across all cluster nodes any more
  - Updates are automatically pushed via CTDB
- Command line tools for editing the samba configuration in registry exist (`net conf`)
- You could even modify your Samba configuration with regedit from a Windows box :)
- Caveats
  - vi cannot be used anymore to modify the configuration
  - SWAT will not work any more
  - CTDB must be up and running to allow access to the registry

How to convert your existing Samba config into the registry

- Make sure that CTDB is up and running
- Import your existing smb.conf with `net conf import`
  - Will overwrite existing registry contents in HKLM\Software\Samba\smbconf
  - Requires to have `clustering=yes` in smb.conf to enable CTDB clustering
- Minimize your smb.conf to just include the registry contents and previous includes

```
[global]
  clustering = yes
  include=registry
  include=/etc/samba/smb.conf.%I
```

- Restart Samba daemons (smbd, winbind) to pick up the change
- Check with `net conf list` and `testparm`

From now on each change to the registry will be automatically distributed to all CTDB nodes

What about the other configuration files?

Other services than Samba require local config files and cannot access TDBs directly.

Besides CIFS, IBM SONAS supports FTP, HTTP, NFS and SCP access.

- Store configuration files for other services (e.g. `/etc/exports`) in custom registry keys
  - CTDB keeps them synchronized across the nodes of your cluster
- Synchronize local configuration files with the registry configuration (one-way)
  - Read from registry and write out to configuration files
- Use the `net registry` command family to put your own arbitrary values into the registry

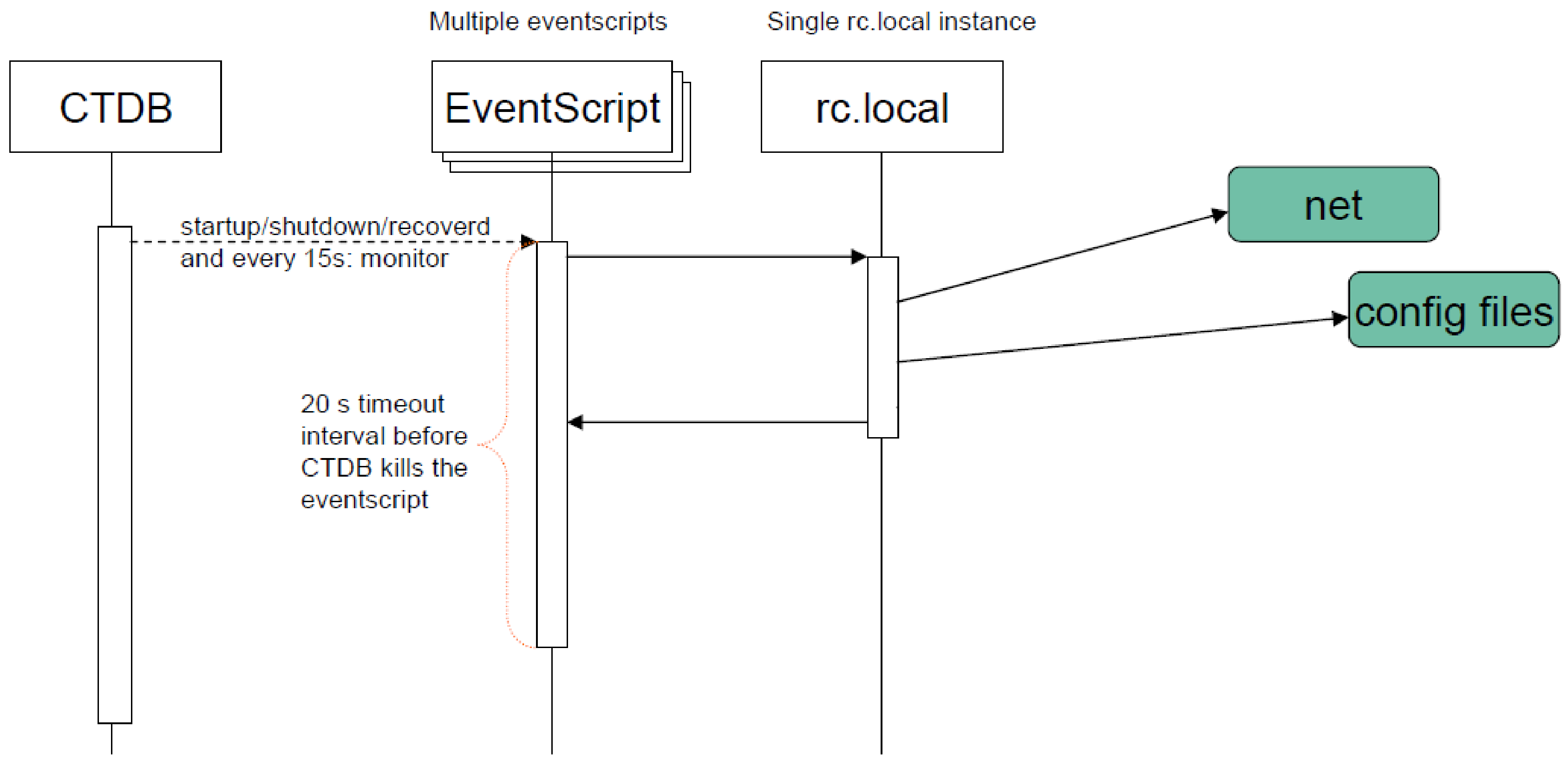
```
– net registry createkey 'HKLM\Software\ACME'
– net registry setvalue 'HKLM\Software\ACME' nfsexports sz
    "/shared/export1 *(ro,fsid=4711)"
```

Using the CTDB rc.local hook (hack)

You can use rc.local to extend CTDB eventscript behavior, e.g. to pull contents from the registry into local files

- By default, CTDB will call `/etc/ctdb/rc.local` (when present) each time an eventscript is called
- rc.local must finish quickly, its execution time counts into the general script timeout
  - Try to do things in the background
- rc.local is called for each and every eventscript on each event
  - only handle events that you are interested in
- You can introduce “fake” events that only have a meaning for rc.local and are ignored by the CTDB eventscripts
  - use ctdb eventscript <myevent> to trigger execution
- Do not use echo for debugging outputs

rc.local


Multiple eventscripts
Single rc.local instance
CTDB
EventScript
rc.local
net
config files
startup/shutdown/recoverd
and every 15s: monitor
20 s timeout
interval before
CTDB kills the
eventscript

Simple rc.local which manages NFS exports in /etc/exports

```
updatenfsconfig() {
  net registry getvalueraw 'HKLM\Software\ACME' 'nfsexports' > /etc/exports
  exportfs -r
  logger -t rc.local -p info "updated NFS config"
}

if [ "x$1" == "xstartup" -o "x$1" == "xreload" ]; then
 if [ "x$0" == "x/etc/ctdb/events.d/60.nfs" ]; then
  updatenfsconfig &
 fi
fi
```

DEMO

DEMO

What are the problems with this implementation?

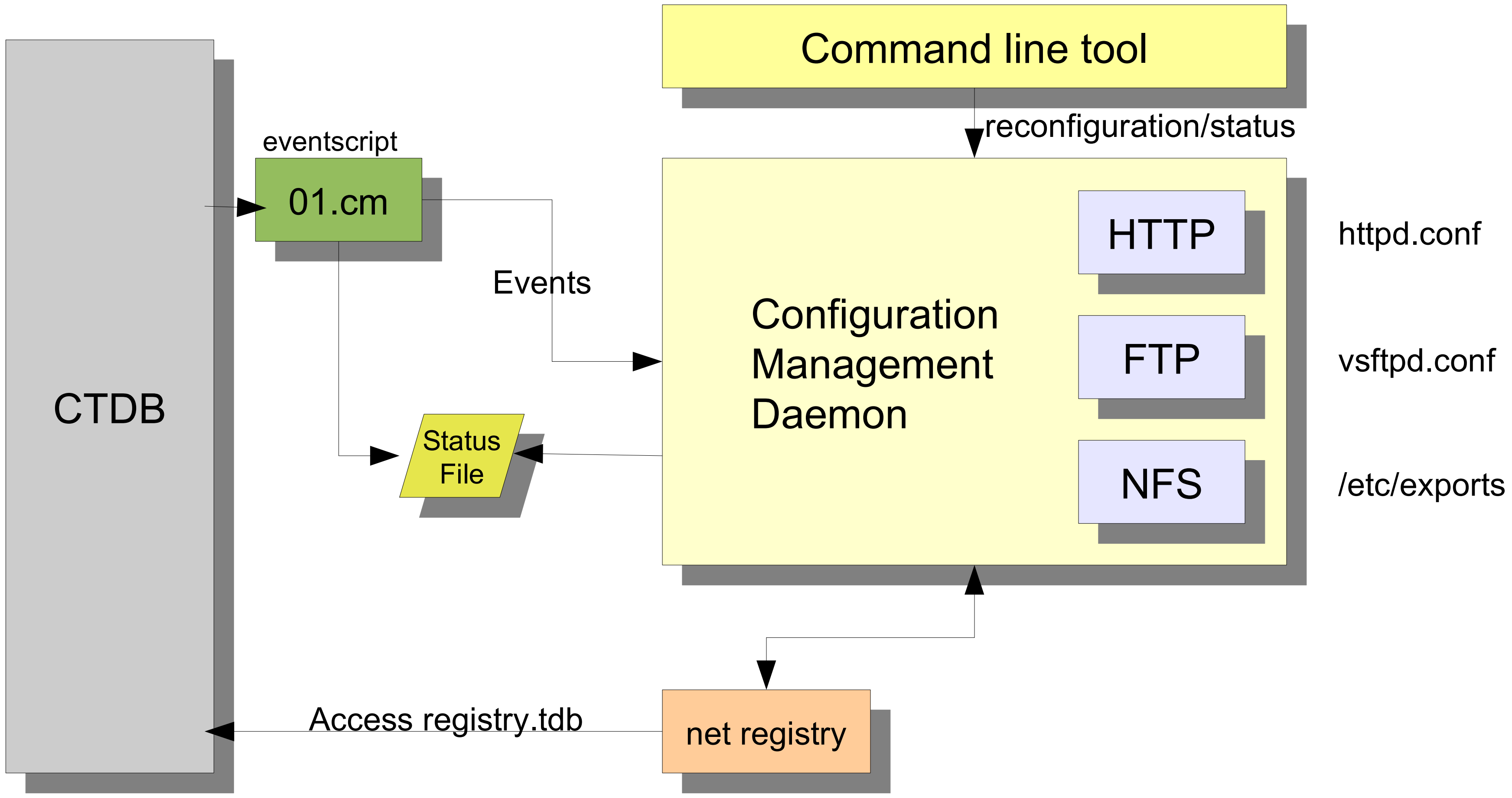
- No error code checking
- Maybe listen on each 10th monitoring interval and check if contents in the registry have changed?
- Implementation can get overly complex quickly when lots of services are to be configured
  - Complex rc.local can lead to general script timeouts
- Called implicitly from <u>every eventscript</u> for <u>every event</u>
- High risk of deadlocks (net registry wait for CTDB , CTDB waits for eventscript complete)
- CTDB cannot differentiate between eventscript errors and rc.local errors
  - Hard to debug
  - Sometimes false error reporting
- All eventscripts are called and managed by CTDB only
  - No external interface for triggering configuration updates
  - All services handled the same way although they may have different requirements
  - Configuration can only be reloaded for all services (all or nothing)
- There is no separation of duty: CTDB wasn't designed for service configuration management

Configuration Management Daemon

A “**Configuration Management Daemon**” can be used to keep the configuration files in sync

- Introduce a daemon that runs independently from CTDB
- Can run periodic tasks independently from CTDB events
- Can listen on CTDB events (like startup, recovered,...)
- Provides command line interface for administrators
- Can respond back to CTDB and influence CTDB status (unhealthy, banned)
  - New status file handling added to CTDB

Configuration Management Daemon



Configuration Management Daemon

The concept of a "Configuration Daemon" can solve the issues of the rc.local solution.

- Introduce a daemon that runs independently from CTDB
  - No risk of deadlocks (CTDB does not wait for the Daemon)
- Can run periodic tasks independently from CTDB events
  - No need for e.g. 10th monitoring interval logic
- Interfaces with CTDB in order get notified about events and to report service status back
  - Not called implicitly from every eventscript for every event
  - Configuration for a single service can be reloaded
- Provides interfaces for administrators and management code to trigger re-configuration
  - External interface for triggering configuration updates
- Handles service configurations individually, checks that reconfiguration was successful
  - Not limited by CTDB script timeout
  - Allow different requirements for different services (e.g. configuration timeouts)
- Clear separation of duty and therefore easier to debug

Hints

- Define registry security descriptors so that not everybody can read the configuration
  - `net registry setsd`
- Split large configuration files into separate registry values to avoid Linux argument limits
  - e.g. for 1000 nfs exports use one registry value per export below one common key
  - Use `net registry getvaluesraw` to efficiently read the whole configuration
- Make sure to have a backup of your registry
  - `ctdb backupdb registry.tdb <tofile>`
- You can even store some of the CTDB configurations in the registry.
  - `CTDB_MANAGES_XXXX` variables
- But many CTDB configuration options cannot be stored in registry, yet
  - public_addresses, static_routes, reclock directory, nat gw, ...

Questions ?

Thank you !