redhat.

INTRODUCING storhaug: High-Availability Storage for Linux

An Operetta in Three Parts

José A. Rivera
Software Engineer, redhat.
Team Member, SAMBA TEAM

2016.05.12

redhat.

HALLO NOCHMAL

sambaXP 2016

Overture

OVERTURE

Who's this guy?

José helps hack and package Samba full-time for Red Hat. He also likes to talk a lot.

- 9-ish years of working with Microsoft protocols
  - Even wrote some of the definitive documentation!
- Two years on the Samba Team
  - Has yet to run screaming
- Never driven a motorcycle

OVERTURE

Looking ahead

ACT I. HISTORY

- The need for CTDB
- Where are we going?

ACT II. INTEGRATION

- Introducing storhaug
- Pacemaker overview
- Dialing back CTDB
- Filling in the gaps
- Playing nice with others

ACT III. STATUS

- Recent developments
- Planned enhancements

OVERTURE

Starting on the same page

HA - High Availability

- A characteristic of a system which says the system can be reliably used with a minimum of downtime.

Failover

- Switching from a failed service to a redundant service due to abnormal termination of the initial service.

Active/Active

- An HA cluster configuration in which failover of services occurs between always-on and (typically) homogenous software nodes.

TDB – Trivial Database

- Samba's primary DB backend.

CTDB – Clustered TDB

- A Samba project that provides a way of distributing its TDBs across clustered nodes.

VIPs – Virtual IP Addresses

- Also known as public IP addresses, these are IP addresses which clients will use to connect to the clustered services and can typically change which node they are assigned to.

Act I. Raccontare

A BRIEF HISTORY

HISTORY

The need for CTDB

- Samba wanted a way to serve the same data from multiple nodes simultaneously.
- CTDB provided a number of other things, including:
  - A common identity for all Samba instances
  - Synchronization of SMB/Windows metadata
  - Cross-node messaging
- In 2012, CTDB version 2.0 was released. This started the road towards modularization.
  - Huge thanks to Amitay Isaacs <amitay@samba.org> and Martin Schwenke <martin@meltin.net>!

HISTORY

Where are we going and why am I in this handbasket?

Modularization facilitates integration!

- Allows for individual features of CTDB to be turned off without disrupting other components.
- This eases integration into other clustered environments, as long as we provide those features elsewhere.

- Why not integrate Samba/CTDB into a fully open source, Linux-based clustered environment?
  - Note the logo in the lower right-hand corner. :)

Act II. Integrare

INTEGRATION

INTEGRATION

Introducing storhaug

storhaug (formerly Storage-HA) is an HA provider for Linux-based clustered storage systems

- Simplifies the setup and configuration of a storage cluster
  - NOTE: You'll still need to configure your access methods (SMB, NFS) to fit your requirements
- Provides many of the non-DB features of CTDB
- Mostly a big pile of shell script
  - “stor haug” roughly means “big pile” in Norwegian

INTEGRATION

Introducing storhaug

```
# storhaug -h
Usage: storhaug [<OPTIONS>] <COMMAND> [<ARGUMENTS>]
Manage a storhaug high-availability (HA) storage cluster.

Global OPTIONS:
  -h, --help            Output this useful help message

COMMANDS:
  status                Check the status of the cluster
  setup                 Setup a new cluster
  teardown              Teardown an existing cluster
  cleanup               Cleanup local cluster config
  cleanup-all           Cleanup cluster config on all nodes
  add                   Add a node to the cluster
  delete, remove        Remove a node from the cluster

Command ARGUMENTS:
  add <NODE>            Add hostname NODE to the cluster
  remove <NODE>         Remove hostname NODE from the cluster
  delete <NODE>         Synonym for 'remove'


Configuration is read from the following locations:
  /etc/sysconfig/storhaug.conf
  /etc/sysconfig/storhaug.d/*.conf
```

```
# Name of the HA cluster created.
HA_NAME="storhaug"

# Password of the hacluster user
HA_PASSWORD="hacluster"

# The server on which cluster-wide configuration is managed.
# IP/Hostname
HA_SERVER="server1"

# The set of nodes that forms the HA cluster.
# Comma-deliminated IP/Hostname list
HA_CLUSTER_NODES="server1,server2,server3,..."

# [OPTIONAL] A subset of HA nodes that will serve as storage servers.
# Comma-deliminated IP/Hostname list
STORAGE_NODES="server2,server3,..."

# [OPTIONAL] Mount point for shared volumes used by HA resources.
HA_MNT_DIR="/var/run/gluster"

# Virtual IPs of each of the nodes specified above.
# Whitespace-deliminated IP address list
HA_VIPS="10.x.x.x 10.x.x.x"
```

INTEGRATION

Pacemaker overview

storhaug's main workhorse, **Pacemaker** is a flexible and extensible HA resource manager...

- A “resource” is defined via a resource agent (RA).
  - RAs can be defined as anything from storage volumes to IP addresses to daemon processes.
- Resources can be centrally managed from a single interface, either from any node in the Pacemaker cluster or a remote management node.
- Resources (and nodes!) can have automated logging of and recovery from failures.

...and it's all extremely configurable.

INTEGRATION

Pacemaker overview

```
# pcs resource show ctdb
 Resource: ctdb (class=ocf provider=heartbeat type=CTDB)
  Attributes: ctdb_recovery_lock=/gluster/lock/lockfile
              ctdb_socket=/var/run/ctdb/ctdbd.socket
              ctdb_manages_winbind=no
              ctdb_manages_samba=no
              ctdb_logfile=/var/log/log.ctdb
  Operations: monitor interval=10 timeout=30 (ctdb-monitor-interval-10)
              start interval=0 timeout=90 (ctdb-start-interval-0)
              stop interval=0 timeout=100 (ctdb-stop-interval-0)
```

```
# pcs resource
 Clone Set: ctdb_lock-clone [ctdb_lock]
     Started: [ buddhi ganesh riddhi siddhi ]
 Clone Set: ganesha_state-clone [ganesha_state]
     Started: [ buddhi ganesh riddhi siddhi ]
 Clone Set: ctdb-clone [ctdb]
     Started: [ buddhi ganesh riddhi siddhi ]
 Clone Set: samba-group-clone [samba-group]
     Started: [ buddhi ganesh riddhi siddhi ]
 Clone Set: ganesha-clone [ganesha]
     Started: [ buddhi ganesh riddhi siddhi ]
 vip1   (ocf::heartbeat:IPaddr2):       Started
 vip1_trigger   (ocf::heartbeat:ganesha_trigger):       Started
 vip2   (ocf::heartbeat:IPaddr2):       Started
 vip2_trigger   (ocf::heartbeat:ganesha_trigger):       Started
 vip3   (ocf::heartbeat:IPaddr2):       Started
 vip3_trigger   (ocf::heartbeat:ganesha_trigger):       Started
 vip4   (ocf::heartbeat:IPaddr2):       Started
 vip4_trigger   (ocf::heartbeat:ganesha_trigger):       Started
```

Pacemaker CLI Examples

INTEGRATION

Pacemaker overview

```
ctdb_start() {
        # Do nothing if already running
        ctdb_monitor && return $OCF_SUCCESS

        # Make sure config is adequate
        ctdb_validate
        rv=$?
        [ $rv -ne 0 ] && return $rv

        # Die if databases are corrupted
        persistent_db_dir="${OCF_RESKEY_ctdb_dbdir}/persistent"
        mkdir -p $persistent_db_dir 2>/dev/null
        for pdbase in $(ls $persistent_db_dir/*.tdb.[0-9] 2>/dev/null$) ; do
                /usr/bin/tdbdump $pdbase >/dev/null 2>/dev/null || {
                        ocf_log err "Persistent database $pdbase is corrupted!  CTDB will not start."
                        return $OCF_ERR_GENERIC
                }
        done

        # Add necessary configuration to smb.conf
        init_smb_conf
        if [ $? -ne 0 ]; then
                ocf_log err "Failed to update $OCF_RESKEY_smb_conf."
                return $OCF_ERR_GENERIC
        fi

        # Generate new CTDB sysconfig
        generate_ctdb_sysconfig
        enable_event_scripts

        # Use logfile by default (and create the logdir if needed), or syslog if asked for
        local log_option
        if [ "$OCF_RESKEY_ctdb_logfile" = "syslog" ]; then
                log_option="--syslog"
        else
                log_option="--logfile=$OCF_RESKEY_ctdb_logfile"
                [ -d $(dirname "$OCF_RESKEY_ctdb_logfile") ] || \
                        mkdir -p $(dirname "$OCF_RESKEY_ctdb_logfile")
                [ -f "$OCF_RESKEY_ctdb_logfile" ] || \
                        touch "$OCF_RESKEY_ctdb_logfile"
        fi
        # public addresses file (should not be present, but need to set for correctness if it is)
        local pub_addr_option=""
        [ -f "${OCF_RESKEY_ctdb_config_dir}/public_addresses" ] && \
                pub_addr_option="--public-addresses=${OCF_RESKEY_ctdb_config_dir}/public_addresses"
        # start as disabled
        local start_as_disabled="--start-as-disabled"
        ocf_is_true "$OCF_RESKEY_ctdb_start_as_disabled" || start_as_disabled=""
        # create the socket/run dir, if needed
        [ -d $(dirname "$OCF_RESKEY_ctdb_socket") ] || mkdir -p $(dirname "$OCF_RESKEY_ctdb_socket")

        # Start her up
        $OCF_RESKEY_ctdbd_binary \
                --reclock=$OCF_RESKEY_ctdb_recovery_lock \
                --nlist=$OCF_RESKEY_ctdb_config_dir/nodes \
                --socket=$OCF_RESKEY_ctdb_socket \
                --dbdir=$OCF_RESKEY_ctdb_dbdir \
                --dbdir-persistent=$OCF_RESKEY_ctdb_dbdir/persistent \
                --event-script-dir=$OCF_RESKEY_ctdb_config_dir/events.d \
                --notification-script=$OCF_RESKEY_ctdb_config_dir/notify.sh \
                --transport=tcp \
                $start_as_disabled $log_option $pub_addr_option \
                -d $OCF_RESKEY_ctdb_debuglevel
        if [ $? -ne 0 ]; then
                # cleanup smb.conf
                cleanup_smb_conf

                ocf_log err "Failed to execute $OCF_RESKEY_ctdbd_binary."
                return $OCF_ERR_GENERIC
        else
                # Wait a bit for CTDB to stabilize
                # (until start times out if necessary)
```

CTDB Resource Agent Samples

INTEGRATION

Pacemaker overview

```
# CTDB
pcs resource create ctdb ocf:heartbeat:CTDB \
        params \
                ctdb_recovery_lock="/gluster/lock/lockfile" \
                ctdb_socket="/var/run/ctdb/ctdbd.socket" \
                ctdb_manages_winbind="no" \
                ctdb_manages_samba="no" \
                ctdb_logfile="/var/log/log.ctdb" \
        op monitor interval="10" timeout="30" \
        op start interval="0" timeout="90" \
        op stop interval="0" timeout="100" \
        --clone ctdb-clone ctdb meta interleave="true" globally-unique="false"

# CTDB: We need our shared recovery lock file
pcs constraint colocation add ctdb-clone with ctdb_lock-clone INFINITY
pcs constraint order ctdb_lock-clone then ctdb-clone INFINITY
```

CTDB Resource Definition

INTEGRATION

Dialing back CTDB

INTEGRATION

Dialing back CTDB

Configuring CTDB so that it only serves as a distributed database backend provider is as simple as not telling it to do other things.

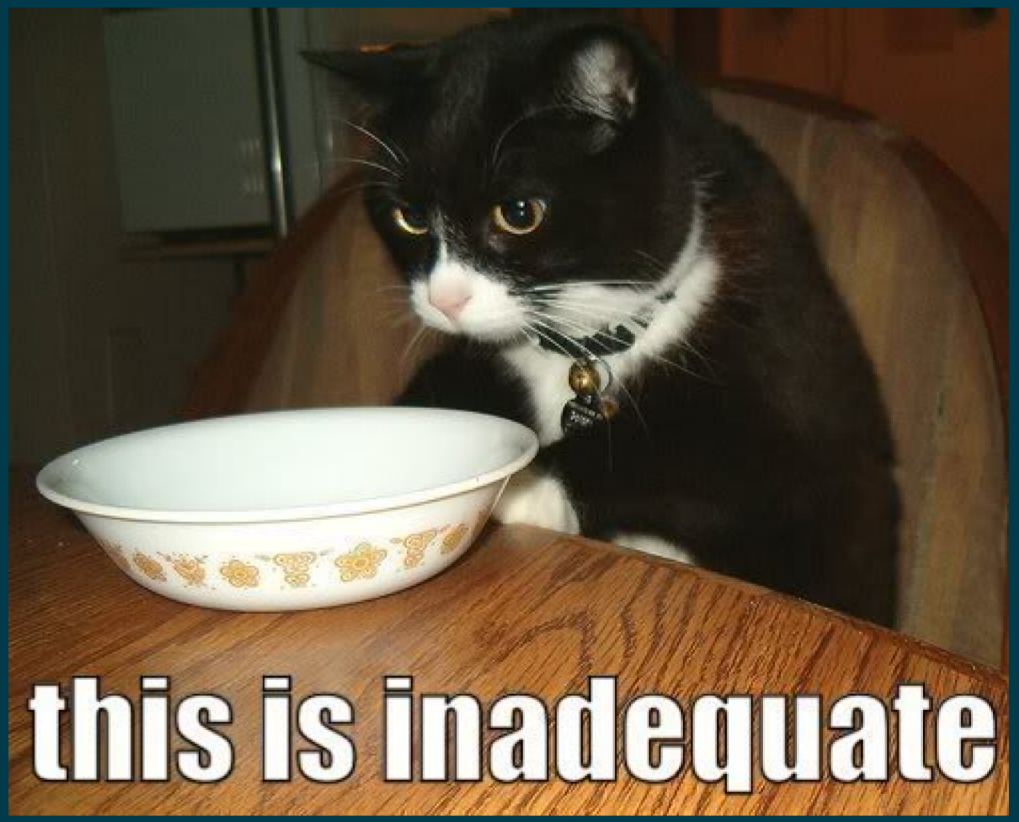
- Don't configure CTDB_PUBLIC_ADDRESSES
  - Disables VIP management
- Don't configure CTDB_MANAGES_SAMBA
  - Disables management of smbd and nmbd
- Don't configure CTDB_MANAGES_WINBIND
  - Disables management of winbindd

Hat tip: Michael Adam <obnox@samba.org>

INTEGRATION

Filling in the gaps

Don't fret, Mr. Snuggleton



We just need to define a few more resources.

INTEGRATION

Filling in the gaps

```
# Virtual IPs
pcs resource create vip${ipcount} ocf:heartbeat:IPaddr2 \
        params \
                ip=${ip} \
                flush_routes="true" \
        op monitor interval=60s \
        meta resource-stickiness="0"
```

VIP Management: IPaddr2

- One resource per address.
- Pacemaker moves the resource for failover.
- Only fails back if resource is not evenly distributed.

- Daemons are a grouped resource and cloned to all nodes.
- Colocate the group with a CTDB instance and start it after CTDB start.

```
# Samba
pcs resource create nmb lsb:nmb \
        op start timeout="60" interval="0" \
        op stop timeout="60" interval="0" \
        op monitor interval="60" timeout="60"
pcs resource create smb lsb:smb \
        op start timeout="60" interval="0" \
        op stop timeout="60" interval="0" \
        op monitor interval="60" timeout="60"
pcs resource group add samba-group nmb smb
pcs resource clone samba-group meta interleave="true"

pcs constraint colocation add samba-group-clone with ctdb-clone INFINITY
pcs constraint order ctdb-clone then samba-group-clone INFINITY
```

Daemon Management

INTEGRATION

Playing nice with others

Finally, we can define other services which will be managed by Pacemaker.

```
# GANESHA: shared state volume
pcs -f ${cibfile} resource create ganesha_state ocf:heartbeat:Filesystem \
    params \
        device="localhost:/${HA_NFS_VOL}" \
        directory="${HA_MNT_DIR}/${HA_NFS_MNT_DIR}" \
        fstype="glusterfs" \
                  options="_netdev,defaults,direct-io-mode=enable,transport=tcp,xlator-option=*client*.ping-timeout=10" \
           --clone ganesha_state-clone ganesha_state meta interleave="true" clone-max="${STORAGE_NUM_SERVERS}"

pcs -f ${cibfile} constraint location ganesha_state-clone rule resource-discovery=exclusive score=0 role eq storage


# GANESHA: NFS-Ganesha daemons
pcs -f ${cibfile} resource create nfs-ganesha ocf:heartbeat:ganesha \
    params \
        config="${HA_NFS_CONF}" \
        state_mnt="${HA_NFS_STATE_MNT}" \
    --clone nfs-ganesha-clone ganesha meta interleave="true" \
                                      globally-unique="false" \
                                      notify="true"

# GANESHA: We need our shared state FS
pcs -f ${cibfile} constraint colocation add nfs-ganesha-clone with ganesha_state-clone INFINITY
pcs -f ${cibfile} constraint order ganesha_state-clone then nfs-ganesha-clone INFINITY
```

Example: NFS-Ganesha

Intermezzo

A SHORT DEMO

Act III. Stato

CURRENT STATUS

STATUS

Recent developments

- Plays nice with Vagrant and Ansible!
  - Project contains a Vagrant+Ansible configuration to allow quick deployment of a sample VM cluster
- Improving integration with NFS-Ganesha
  - Streamlined configuration of shared state
  - Properly trigger GRACE
  - Be slightly less Red Hat about where configs are :)
- Add new nodes to the cluster at runtime
  - Populates/distributes resources as needed
  - Currently in “tech preview”

storhaug v1.0 is just around the corner

STATUS

Planned enhancements

- Manage storage volumes
  - At least monitor status
  - Possibly start/stop or mount/unmount
- Develop new tickle ACK implementation for Pacemaker
  - Current implementation could be improved
  - Use conntrackd/iptables
  - A new RA? Maybe integrate in IPaddr2? A new IPaddr3?
- Natively support more filesystems, e.g.:
  - GFS2
  - Ceph FS

Catch the presentation by
Ira Cooper <ira@samba.org>

redhat.

Fine
(Das Ende)

THANK YOU!

https://github.com/linux-ha-storage/storhaug

jarrpa@samba.org || jarrpa@redhat.com
IRC: jarrpa in #samba-technical on irc.freenode.net
Twitter: @jarrpa