SBD Recent Changes and Future Plans

Well - since around Summit 2017

Klaus Wenninger

Senior Software Engineer, Red Hat

3 Pillars of SBD fencing

A quick overview ...

‘Poison Pill’ - Messaging

Communicate fencing-requests via Messaging based on defined data-slots on shared block device(s).

Suicide based on Quorum & Health

- Collect Node Quorum & Health - State
- Periodically Poll Cluster-Components
- Periodically Poll Disk(s) for Accessibility

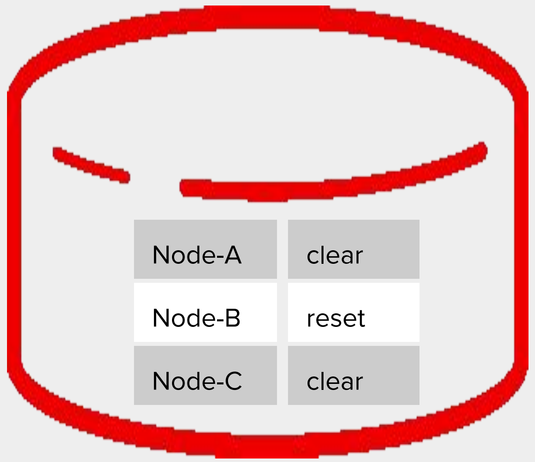
Watchdog Observation

SBD itself is observed by a hardware-watchdog to assure node to be taken out of service in case SBD and/or reboot-mechanisms are stuck.

POISON-PILL MESSAGING

Node-A fencing Node-B via shared Disk

Node-A puts Poison-Pill into Messaging-Slot of Node-B



Node-A	clear
Node-B	reset
Node-C	clear

Node-B periodically checks Messaging-Slot

Watchdog Handling

cmd-line-tool for query & test of watchdog-devices

-
```
[root@node2 ~]# sbd query-watchdog
Discovered 2 watchdog devices:
[1] /dev/watchdog
Identity: i6300ESB timer
Driver: <unknown>

[2] /dev/watchdog0
Identity: i6300ESB timer
Driver: <unknown>
```

-
```
[root@node2 ~]# sbd test-watchdog -w /dev/watchdog
WARNING: This operation is expected to force-reboot this system
without following any shutdown procedures.

Proceed? [NO/Proceed] Proceed

Initializing /dev/watchdog with a reset countdown of 5 seconds …

NOTICE: The watchdog device is expected to reset the system
        in 5 seconds. If system remains active beyond that time,
        watchdog may not be functional.

Reset countdown ... 5 seconds
Reset countdown ... 4 seconds
Reset countdown ... 3 seconds
Reset countdown ... 2 seconds
System expected to reset any moment …
System expected to reset any moment …
```

Red Hat

Watchdog Handling continued ...

Consistent handling of timeouts detected by watchdog-device and daemon

- Make timeout-action executed by sbd configurable to match behavior of watchdog-device

```
comma-separated combination of
noflush|flush plus reboot|crashdump|off
```

- Fix some bugs regarding shutdown/reboot issues

Help preventing watchdog-device from triggering in crash-dump case

- Fix bug to actually use crashdump-timeout if configured
- Have in mind that setting a high crashdump-timeout may not be consistent with Stonith-watchdog-timeout set in Pacemaker

Realtime Scheduling

With CPU-Accounting active system-slice doesn't have RT-Budget.

Generic way around is to introduce realtime-slice and assign service (e.g. SBD) to that from unit-file.

```
[Service]
Slice=realtime.slice
```

Alternative: Bypass slicing done in systemd and move to root-slice (current corosync implementation).

SBD-Implementation

- Check for RT-Budget in current slice - not necessarily system-slice
- Move to root-slice just if not enough RT-Budget found

```
## Type: yesno / auto
## Default: auto
SBD_MOVE_TO_ROOT_CGROUP=auto
```

Todo

Possibly port Implementation to corosync

- Maybe common implementation in libqb ...
- No BSD / GPL issues if I do it
- Control Group V2

Robustness in Interaction between Daemons

SBD does couple of crucial things at startup

- Lock to memory
- Open hardware-watchdog
- Set rt-scheduling

pacemaker shouldn't be started if they fail

- Adaptions to unit-file to keep pacemaker down

```
[Install]
RequiredBy=pacemaker.service
```

- Todo: Pacemaker-remoted still starts if sbd is failing

If corosync-daemon is frozen

- CIB doesn't get updated
- No update about quorum-state
- No update about node becoming unclean

periodically ping corosync-daemon for liveness

Robustness in Interaction between Daemons continued ...

Just go back to relaxed watching state (initial) if pacemaker went down gracefully

- Check if all resources are down before pacemaker disappears via CIB
- Todo: Implement state-reporting via Pacemaker-API
  - Use 'shutdown -completed' in SBD

On startup pacemaker-detection solely via CIB isn't robust enough

Todo: make Pacemaker wait to be contacted by SBD before starting resources

Certain combinations of pacemaker-sub-daemons frozen lead to

- Slow recovery of the cluster via fencing
- Totally frozen cluster (e.g. Scheduler frozen)

Todo

- Use new state-reporting mechanism in Pacemaker For periodic liveness-check
- Implement hierarchical liveness-check of Sub-Daemons inside Pacemaker

Robustness in Interaction between Daemons continued ...

Todo - Improve Validation and Automatic-Synchronization of Timeouts

- Pacemaker 2.x allows stonith-watchdog-timeout = -1

  Dangerous because taken from local node → needs bookkeeping of watchdog-timeout on all nodes
- Corosync in qdevice-setup can be stalled for substantial time

  Need to look deeper into Corosync-timeouts and use-cases like token-loss, maximum stall-time, ...

  Maybe need some kind of graceful shutdown detection for Corosync

Todo - open PRs

- Feature: service: add pre-start configuration validator PR#99
- sbd-cluster: Simplify cluster connection loss handling PR#81
- sbd-cluster: stop dispatching cmap if disconnected PR#80
- systemd: make corosync wait for sbd-start to complete PR#74

Improve Build/Test for CI-Friendlyness

All sorts of things require raised-privileges

- /dev/watchdog
- Blockdevices

  either real ones or emulation via loop-mount & device-mapper
- /proc/sysrq-trigger

  As well as any other means to trigger shutdown/reboot/crashdump
- Non-cached-files-access
- asynchronous-IO

Or Test nasty to implement because of reboots

- /dev/watchdog
- /proc/sysrq-trigger

Preload Library Approach

```
LD_PRELOAD=libsbdtestbed.so
```

- Interception of all crucial stuff
- Log for checks in CI
- Skip or map to something non-crucial
- Simulate behavior (e.g. watchdog)

Implementation

- Straight forward from scratch implementation as part of SBD-repo
- None of the existing frameworks found useful out of the box (missing features, availability on build-targets)
- Todo: Integrate into umockdev (Martin Pitt)

Red Hat

Improve Build/Test for CI-Friendlyness continued ...

What we get

- tests in environments that wouldn’t allow block-device simulation (loop+devmapper)
- tests in sbd daemon-mode (watch)
- test of all kinds of reboot-causes
  - reboot via hardware-watchdog
  - various types triggered via sysrq
  - detection of issues with accessing content of block-devices
- tests can run in all sorts of container environments
  - LXD on travis
  - inside mock-containers
  - even on foreign architectures ...

... .e.g. Inside mock with userspace-qemu:

```
$> make PACKAGE=sbd -f Makefile.am srpm

$> mock --forcearch aarch64 -r fedora-28-aarch64
   … sbd*.src.rpm

$> mock -r fedora-28-aarch64 --shell
   … --forcearch aarch64

<mock-chroot> sh-4.4# cd builddir/build/BUILD/sbd…

<mock-chroot> sh-4.4# SBD_TRANSLATE_AIO=yes make check

     SUCCESS: All tests completed
     PASS: tests/regressions.sh
     =============
     1 test passed
     =============
```

Red Hat

SBD and Pacemaker-remote

- Skip sbd-check on guest-containers & bundles eventhough watchdog-fencing is enabled
  Actually a Pacemaker fix where sbd-check is unnecessary as fencing done via hypervisor
- Todo: limit sbd to certain nodes
  Actually to be implemented with Pacemaker
- Todo: full support of pacemaker-integration on remote-nodes
- Todo: prevent startup of pacemaker-remoted if sbd doesn’t come up properly

Documentation & Logging

fixes, overhauls improvements

- overhaul log-levels
- More reasonable stdout/stderr distribution
  + dependency to fence-agents using both
- add man section for query-watchdog & test-watchdog
- auto-generate man section for environment from sbd.sysconfig
- Todo: Updates to manual page and usage message PR#54

Thank you

See you online ...

https://github.com/ClusterLabs/sbd

https://clusterlabs.org

Red Hat