Qdevice

Jan Friesse <jfriesse@redhat.com>

Clusterlabs Summit 2017
September 13, 2017

Qdevice x Booth x SBD

- All of them use third-party arbitrator
- Each of them operates on different stack layer and has different purpose so use all of them in one deployment is perfectly valid
- SBD Is fencing device
- Booth
  - Operates (usually) at Pacemaker level
  - It's primary use case is geo-clustering - spans multiple "independent" clusters
- Qdevice
  - Operates at Corosync (quorum) layer - it is adding vote to single cluster quorum
  - It's primary use case is for even-node clusters/LMS/stretch clusters

Qdevice

- Independent arbiter for solving split-brain situations, stretch cluster
- Daemon running on every node of the cluster and using `Corosync votequorum` API
- Modular architecture - `model` API
  - Currently only `net` model is implemented (`qdevice-net`)
- qdevice-net has support for multiple `algorithms`
  - Currently LMS, FFSplit and 2 Node LMS
  - Test algorithm provided as template

Qnetd

- 3rd side for Qdevice-net
- It is “clever” - responsible for decisions
- Supports TLS with both server and client (per cluster) certificates
- It’s able to handle multiple clusters
- No configuration file - all required information provided by cluster nodes
- No persistent state
- TCP based protocol designed with backwards/forwards compatibility in the mind since the very beginning

Algorithms

- LMS
  - Provides *NumberOfNodes* − 1 votes
  - If node is last one connected to qnetd, node gets votes
  - If more nodes exists vote is provided to largest partition
  - Useful when cluster with only one node should remain quorate
  - With `wait_for_all` enabled it can survive qnetd disconnect (but no other change can happen)
- FFSplit
  - Provides one vote
  - Behaves like just another node
  - Make sense only for even-node clusters
  - Useful for 2 node cluster (or a lot of them), cluster where qnetd can disappear or where it's not intended to let qnetd overvote cluster membersip

Heuristics

- Execute arbitrary number of commands
- If all of them success whole heuristics sucess → no scoring
- Result sent to the `corosync-qnetd` and there it is used as tie-breaker
- 3 modes of operation: disabled, sync (heuristics executed only during the sync phase → only when membership changes), enabled (sync and regular heuristics)

Future

- Clustered Qnetd
  - Idea is to implement active/passive RA for Qnetd
  - Qnetd doesn't have persistent state
  - Qdevice tries to reconnect when connection to Qnetd is lost
  - Only nssdb is subject to synchronization
- Heuristics only model
  - Idea is to base vote only on heuristics result
  - Should be used in situations where 3rd side arbiter is already deployed
- Allow more than 1 vote for FFSplit
  - For situations when LMS is too strong and current FFSplit too weak
  - Be able to set arbitrary votes

Future cont.

- Redundant connections to Qnetd
  - Better reliability
  - Quite important for LMS
- Disk model?
  - Probably use SBD as arbiter
  - Closer replacement of `qdisk`

Part II

Extending Qdevice

Algorithms - implementing new one

- Add `TLV_DECISION_ALGORITHM_TYPE_*` into `tlv.[ch]`
- Add handling of this new type into helper functions in `tlv.c`, `qdevice-net-instance.c`, ... (compiler will tell you)
- Qdevice side
  - Qdevice-net side callbacks are mostly empty (default should be good enough)
  - Copy `qdevice-net-algo-test.[ch]` and use them as a template
  - Add to `Makefile.am`
- Qnetd side
  - Much harder because qnetd side is the "clever" one
  - Use `qnetd-algo-test.[ch]` as template

Model - implementing new one

- Add `QDEVICE_MODEL_TYPE_*` into `qdevice-model-type.h`
- Use `qdevice-model-net.[ch]` as template
- Add to `Makefile.am`
- Implement required functions
- Model is responsible for main loop and periodical calling of `qdevice_votequorum_poll`