Reliable Host Fencing In CloudStack

Rohit Yadav (Software Architect)

Boris Stoyanov (Sr. Software Test Engineer)

rohit.yadav@shapeblue.com

boris.stoyanov@shapeblue.com

@rhtyd / @bsstoyanov

About Me

Rohit Yadav

- Software Architect @ ShapeBlue
- Contributor and Committer since 2012
- Author and maintainer of CloudMonkey

Boris Stoyanov

- Senior Software Engineer Test @ ShapeBlue
- Contributor since 2016

About ShapeBlue

"ShapeBlue are expert builders of public & private clouds. They are the leading global CloudStack services company."

ShapeBlue customers

ShapeBlue customers

ShapeBlue customers

What is HA?

High availability is a characteristic of a system, which aims to ensure an agreed level of operational performance, usually uptime, for a higher than normal period. [source: wikipedia]

HA in CloudStack: Status Quo

- Currently **HA** is only supported for VMs by CloudStack.
- VM HA mechanism works for VMs that are marked *HA.*
- Implementation tied to VM as a first class resource, asynchronously scheduled, limited to VM investigation/fencing/restart on new host.

HA in Production: Status Quo

- Investigations are VM centric and not host centric.
- Limited fencing of host, highly unreliable.
- VM HA may end up starting VMs on another host, while the VMs may be running on the *faulty.* Large environments see corrupt VMs and disks.
- Unchecked faulty hosts and faulty neighbors, with no automatic-recovery.
- Real world issues seen in a very large KVM environment.

Attempted Solutions: KVM

- Check VM for disk activities based on a timeout/threshold before re/starting VM.
- (Wall) Clocks are not reliable
- Maintenance and management issues
- No recovery mechanism, fencing still remains unreliable

References:
https://issues.apache.org/jira/browse/CLOUDSTACK-8762
https://github.com/apache/cloudstack/pull/753

Long Term Solution?

- CloudStack needs a way to perform power management tasks for hosts
- Solve issues of corrupt disks due to VM HA and unreliable host fencing
- Improve experience for admins: granular configuration, feature *kill-switch*, maintenance, management, reporting, alerts, investigations, reliable fencing and recovery etc.

Host Power Management for CloudStack

- Implemented a pluggable out-of-band management framework for CloudStack
- Granular configuration per host, kill switch at zone/cluster/host level
- Default plugin for IPMI 2.0 compliant hosts to support power operations: on, off, reboot, shutdown, status etc.
- High quality tests, end-to-end testing based on ipmisim
- DIY oobm plugin

Reference:
https://cwiki.apache.org/confluence/display/CLOUDSTACK/Out-of-band+Management+for+CloudStack

Building Blocks for Host HA

- Solve reliably fence/recover a host: use the new shiny out-of-band management subsystem
- What's missing:
  - Granular HA configuration
  - Host HA kill-switch: at zone/cluster/host level
  - Tuning: Threshold based investigation, activity checks, timeouts etc.
  - Task/Load management, circuit breakers, constraint based state transitions and operations

Reference:

https://cwiki.apache.org/confluence/display/CLOUDSTACK/KVM+HA+with+IPMI+Fencing

Rethink HA

- CloudStack organization units as partitions: Zone, Pod, Cluster, Host, VM.
- Separate policy from mechanism:
  Implement framework/managers to enforce policies, have plugins to carry out mechanisms
- Define HA for a general resource, pluggable HA provider implementations.
- Operational simplicity.
  - Granular configuration, kill-switch at zone/cluster/host level. Disabled by default.
  - Threshold based investigations, checking, fencing and recovery.
- Leverage existing abstractions.
- Integrated resource management.

Host HA: Design and Implementation

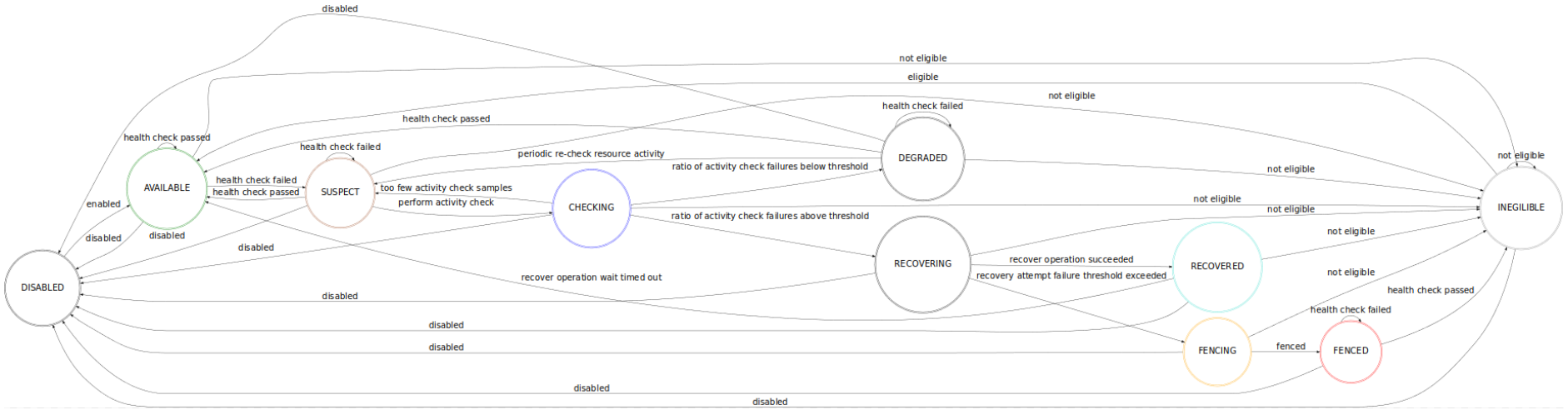
- **HA Resource Management Service**
  - HA resource lifecycle management
  - HA resource type agnostic
  - Disabled by default, granular configurations, zone/cluster/host kill-switch, tuning
- **HA Provider**
  - Resource specific HA plugin
  - Defines partition and resource type
  - DIY HA provider for partition: host/hypervisor/etc
  - One HA provider per resource type, per partition

Reference:
https://cwiki.apache.org/confluence/display/CLOUDSTACK/Host+HA

Host HA: FSM States Explained

- **HA Resource FSM States**
  - Available
  - Suspect
  - Checking
  - Degraded
  - Recovering, Recovered
  - Fencing, Fenced
  - Disabled
  - Ineligible

Host HA: FSM State Transitions



Reference:
https://cwiki.apache.org/confluence/display/CLOUDSTACK/Host+HA

Host HA: Lifecycle management

- Granular HA configuration
- Kill switch: enable/disable for a partition (zone/cluster/host)
- HA validation and ownership management
- New Background Polling Manager for executor service management
- Tasks executor, bounded (ephemeral) queue management
- HA Polling tasks: Health Checks, Activity Checks, Recovery Task and Fence Task
- FSM transitions based on task execution result
- HA resource counter management: track investigation rounds, thresholds, timestamps, recover/fence operations

Host HA: KVM HA Provider

- STONITH (Shoot The Other Node In The Head) fencing model
- Activity check operations, checks for disk access activities on NFS storage
- Configurable activity check interval and activity checks
- Tunable timeouts and thresholds
- Request-reply model to check activity checks via adjacent eligible and healthy host(s)
- Uses out-of-band management subsystem to carry out recover and fence operations
- Recovery is attempted before fencing of the host
- Alerting and reporting of operations

Host HA: VM HA – HAProvider Coordination

- Remaps VM-HA host state returned to VM-HA framework based on Host HA states, **only for hosts with Host HA enabled**.
- For Host HA to work effectively, existing VM HA framework to work in tandem with Host HA.
- By default Host HA is disabled, no explicit configuration changes needed for existing users pre/post upgrade.
- Currently, done for KVM HAProvider

Host HA state (KVM)	VM-HA host state returned
Available	Up
Suspect/Checking	Up (Investigating)
Degraded	Alert
Recovering/Recovered/Fencing	Disconnected
Fenced	Down
Ineligible/Disabled	--

Host HA: Testing with Simulator HA Provider

- HA Provider for Simulator provides means and instrumentation to perform end-to-end deterministic testing of the framework.
- Provides means of validation of the feature and shows pluggability of the framework.
- New Simulator APIs provides means of validating FSM sequences and instrumenting internal data structures.
- Marvin based integration test, covers FSM transitions, HA operations, validations, configurations, HA ownership.

Host HA: Testing in nested CloudStack environment

- Recently, nested CloudStack environments such as Trillian, Bubble etc have tremendously helped with QA efforts. In such environments, hypervisor hosts are VMs in another CloudStack environments.
- As part of the FR, we've implemented a new out-of-band management plugin for nested CloudStack environment.
- This plugin can perform power management operations to start/stop/reboot the host VMs.
- The new oobm plugin allows for scalability and load testing of the Host HA feature in nested CloudStack environment. Currently being tested for a large KVM based environment.

Host HA: Current State & Future Plans

- Pull request: https://github.com/apache/cloudstack/pull/1960
- FS: https://cwiki.apache.org/confluence/display/CLOUDSTACK/Host+HA
- Currently supports two HA Provider implementations:
  - KVM: Out-of-band management, NFS supported
  - Simulator: QA/testing
- Available out-of-band management plugins: ipmitool and nested-cloudstack
- Likely available in Apache CloudStack 4.11 or above
- Future Plans:
  - Multiple HA Provider implementations for other hypervisors, support for other storage
  - Scope for extension to support HA for other resources/partitions

Host HA: Thanks & Credits

- Abhinandan Prateek: KVM HA Provider implementation
- Boris Stoyanov: Reviews and QA
- Ilya Musayev, Marcus Sorensen and John Burwell: Requirements, feedback and design
- Rohit Yadav: Overall design and implementation
- Team ShapeBlue, Paul, Dag, Daan – Reviews, discussions, testing, Trillian setups

Q & A

- Comments, questions welcome!
- Discuss on dev ML or on the PR.