Deep Insights: High Availability VMs via a Simple Host-to-Guest Interface

OpenStack Masakari
Greg Waines (Wind River Systems)
Wind River’s products focus on:
- embedded software,
- cloud software and
- software for the internet of things.

Our software has been deployed in over 2 billion devices; into environments, systems, and applications subject to the highest standards of safety, security and performance.
Titanium Cloud in Networking and Industrial Applications

Providing secure, reliable, low-latency, high-bandwidth private-cloud-platforms for mission-critical market applications:

Networking

Energy

Manufacturing

Smart Buildings

Healthcare
BACKGROUND:

OpenStack Masakari – Virtual Machine High Availability

- Recovering KVM-based VMs from various failure events:
  - KVM Process down,
  - Critical Service Process down,
  - Compute Host failure.

- ‘masakari-monitors’ on Compute Host monitor for failures
  - Reporting failures via ‘masakari-api’ to the ‘masakari-engine’.

- ‘masakari-engine’ on Controller Node takes appropriate actions to recover affected VMs.
OpenStack Masakari – Introspective Instance Monitoring - NEW -

- **Existing** masakari instance-monitor, Black Box Monitoring of VM thru LibVirt.

- **NEW** masakari *introspective*-instance-monitor, White Box Monitoring of VM thru QEMU Guest Agent.

- Enables detection of Guest VM Failures that existing Black Box instance-monitor does NOT detect

- Simple QEMU Guest Agent guest-ping() can indirectly detect failures such as:
  - Hung Guest OS,
  - Failure of Guest OS to schedule Application Processes,
  - Failure to route basic IO within the Guest,
  - Severe resource exhaustion, etc.

- Enabled via Flavor Extra-Spec,

- Leverages open-source Qemu Guest Agent for messaging to/from VM.

https://wiki.libvirt.org/page/Qemu_guest_agent
OpenStack Masakari – Other Introspective Possibilities -NEXT-

Other ways to leverage 'reachthru capability inside Guest VM' ... i.e. Host → Guest APIs to enable Higher Availability VMs.

- Graceful Guest Application Handling of VM Operations,
  - Introduce NOVA PROXY to intercept non-passive VM Operations (stop/start, reboot, pause/unpause,, resize begin/end, migrate begin/end) and notify Guest VM Application to "gracefully" prepare for the event.

- Ability for VM to NACK VM Operation to avoid Application Out-Of-Service Condition,
  - Introduce NOVA PROXY to intercept non-passive VM Operations (stop/start, reboot, pause/unpause,, resize begin/end, migrate begin/end) such that Guest Application can WARN / NACK OpenStack that this would result in Application Out-Of-Service.

- Lightweight Broadcast Notification and State-Change-Notifications within Nova Server-Group,
  - Provide Lightweight Broadcast messaging between VMs of the same Nova Server-Group, for purposes of bootstrap communication between VMs prior to even networking within Guest even coming up,
  - Provide VM State Change Notifications to all VMs of the same Nova Server Group, for a potentially faster, more reliable, backup notification path for peer VM failures to Guest Applications.

- Ability to fully support Dynamic Resource Scaling between NOVA and Guest VMs.
  - E.g. Ability to notify Guest OS of VM that a new virtual vCPU core was assigned/removed to the VM, in order for the Guest OS to properly online / offline the cpu within the Guest OS.