Support for Fast and Reliable VMM Live Upgrades in Libvirt

Soham Ghosh, Prachatos Mitra
Agenda

1. Problems with VMM upgrades
2. Design and implementation
3. Results and Future work
Problems with VMM upgrades
Problems with VMM upgrades

- VMs are live migrated to another host and migrated back
  - Requires a long maintenance window

- Issues we face with live migration
  - Non-deterministic
  - Guest impact
  - Resource contention

- As a result, VMM upgrades are deferred by system admins
Our solution

• Upgrading VMM through **Local live migration**
  • Using existing Libvirt migration workflow
  • No memory copy required
  • No dirty-logging and throttling

• Can upgrade VMMs with near-zero downtime

• Minimal maintenance window required
Local migration workflow

Host 1

- libvirt
- Local migration
- VM1
- qemu (6.1.0)
- qemu (6.1.1)

/usr/bin/qemu-x86-6.1.0 (yum update)
Design and implementation
Design challenges

• Keeping the machine ABI unchanged
  • libvirt expects name and UUID of a VM to be unique
  • Handling absolute path dependencies on UUID / name

• Modifying migration phases to work on the same host
  • Modifying remote migration phases
  • Resolving the correct domain object

• Avoiding the memory copy
Bypassing memory copy

Fetch fd for memory backend
fetch-backingfd
QMP command

VM1
qemu 6.0.0

Transfer memory backend fd

libvirt

VM1
qemu 6.1.0

Local migration
Handling Qemu monitor and log files

VM1 paths (domain id 1)
/var/lib/libvirt/qemu/domain-<VM1 UUID>/monitor.sock

Monitor socket path

VM1# paths (domain id 2)
/var/lib/libvirt/qemu/domain-<domain-id-2>-<VM1# UUID>/monitor.sock

Monitor socket path (new format)

Begin Phase (Source)  Prepare Phase (Remote)  Perform Phase (Source)
Finish Phase (Remote)  Confirm Phase (Source)
Changes to migration flow

<table>
<thead>
<tr>
<th>Source hash table</th>
<th>Remote hash table (New)</th>
</tr>
</thead>
<tbody>
<tr>
<td>VM1 UUID</td>
<td>VM1#</td>
</tr>
<tr>
<td>VM2 UUID</td>
<td>VM2</td>
</tr>
<tr>
<td>VM3 UUID</td>
<td>VM3</td>
</tr>
</tbody>
</table>

Begin Phase (Source) | Prepare Phase (Remote) | Perform Phase (Source) | Finish Phase (Remote) | Confirm Phase (Source)
Results and future work
Demo

yum update qemu-kvm

virsh migrate --local <domain id>

Initiate qemu upgrade from package manager

Local migrate VM to new qemu binary
Results - migration time

VM1 - 2 vCPUs, 8 GB memory (nested)
VM2 - 4 vCPUs, 16 GB memory (nested)
VM3 - 6 vCPUs, 24 GB memory (nested)
VM4 - 8 vCPUs, 32 GB memory (nested)

Workload: High write-throughput
(n=x, m=y, l=z) = x threads dirtying y GB of memory at z GB/s
Results - downtime

VM1 - 2 vCPUs, 8 GB memory (nested)
VM2 - 4 vCPUs, 16 GB memory (nested)
VM3 - 6 vCPUs, 24 GB memory (nested)
VM4 - 8 vCPUs, 32 GB memory (nested)

Workload: High write-throughput
(n=x, m=y, l=z) = x threads dirtying y GB of memory at z GB/s
Conclusion and Future Work

- We have enabled Qemu upgrade using local migration
  - ~1s migration time
  - < 50ms downtime

- Extending FD transfer framework to all types of devices
  - Passthrough devices

- Continuing to upstream the patches
Thank you

Contact
soham.ghosh@nutanix.com
prachatos.mitra@nutanix.com