NUTANIX™

Support for Fast and Reliable VMM Live Upgrades in Libvirt

Soham Ghosh, Prachatos Mitra

SEPTEMBER 2021 | KVM FORUM

Agenda

1 Problems with VMM upgrades

2 Design and implementation

3 Results and Future work

Problems with VMM upgrades

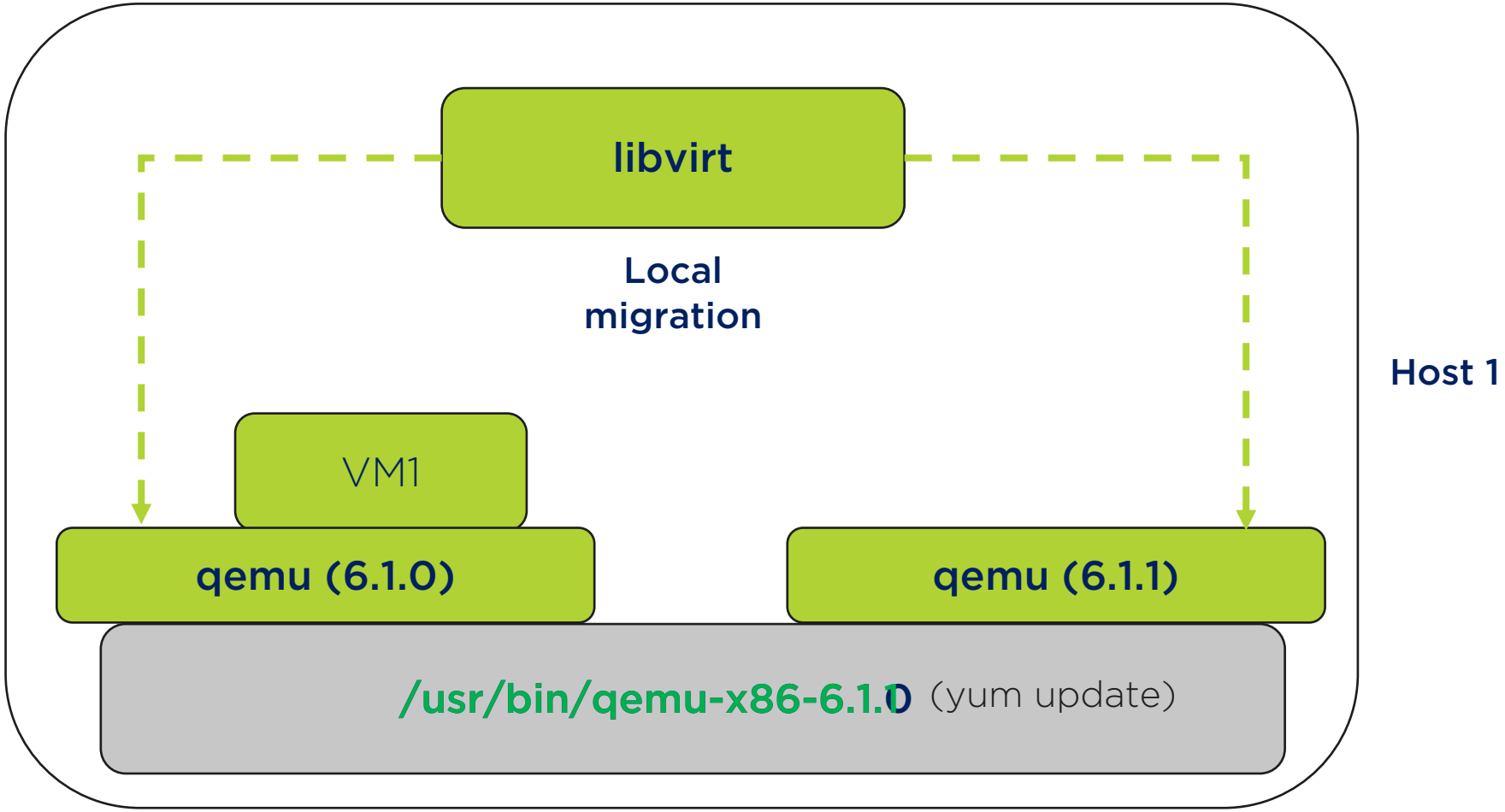
Problems with VMM upgrades

- 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

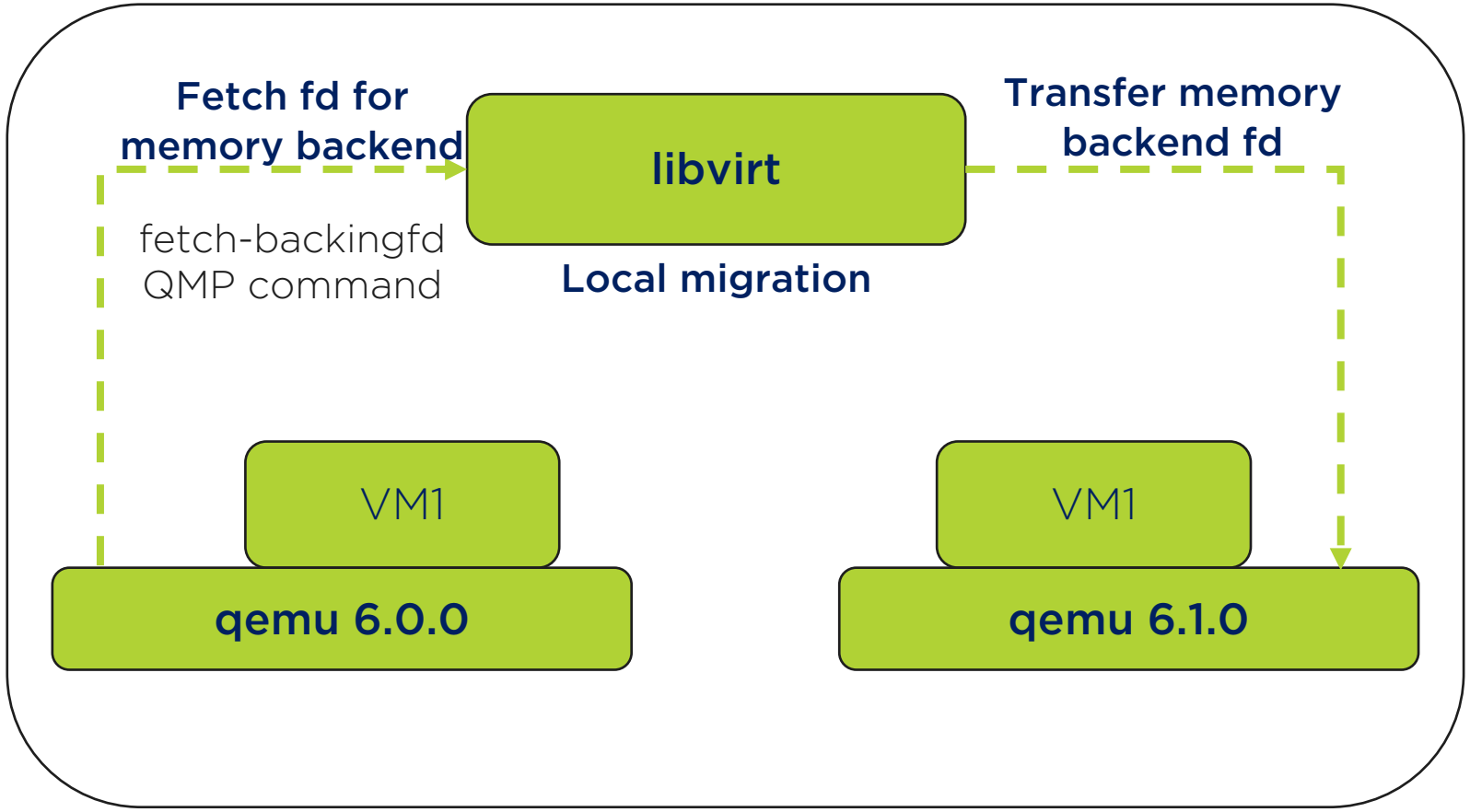


Design and implementation

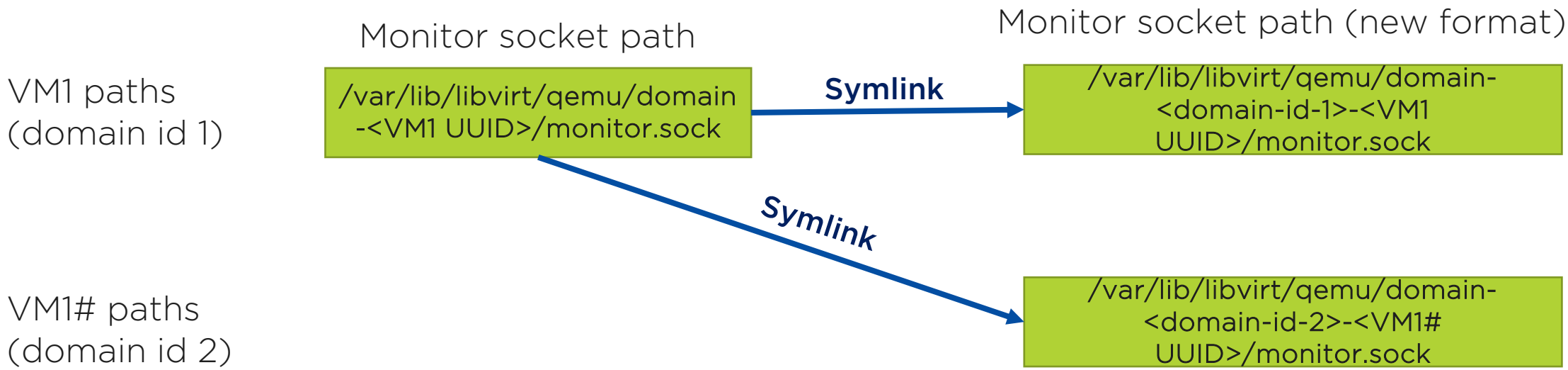
Design challenges

- 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



Handling Qemu monitor and log files



Begin Phase (Source)

Prepare Phase (Remote)

Perform Phase (Source)

Finish Phase (Remote)

Confirm Phase (Source)

Changes to migration flow

Source hash table

VM1 UUID	VM1
VM2 UUID	VM2
VM3 UUID	VM3

Remote hash table (New)

VM1 UUID	VM1#

Begin Phase (Source)

Prepare Phase (Remote)

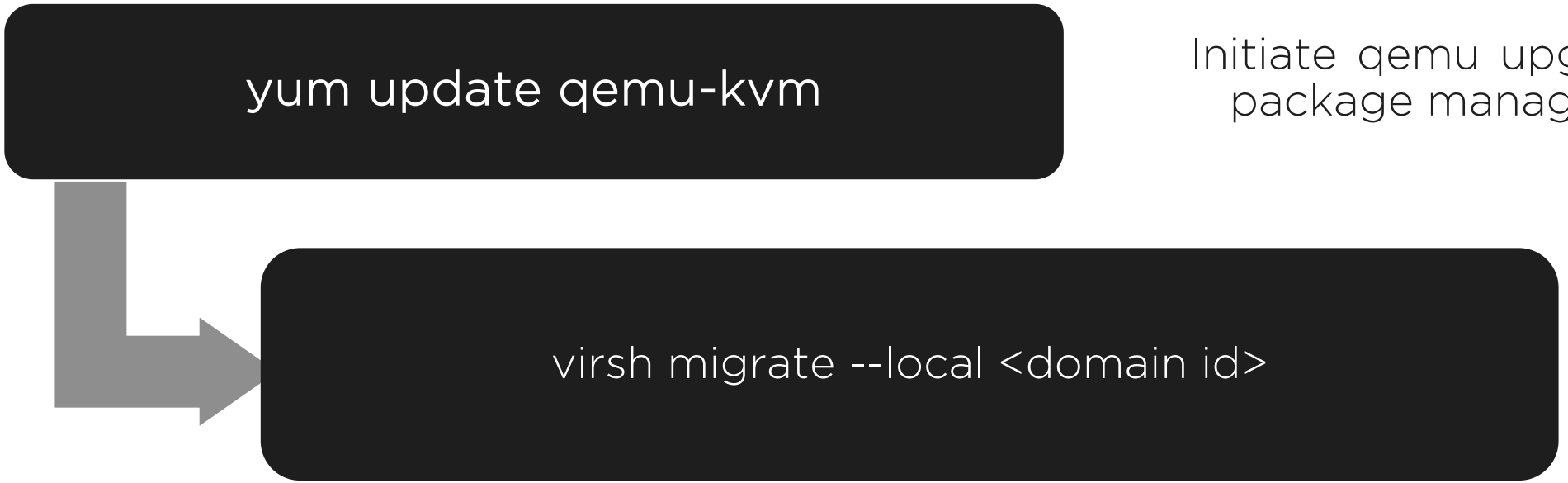
Perform Phase (Source)

Finish Phase (Remote)

Confirm Phase (Source)

Results and future work

Demo

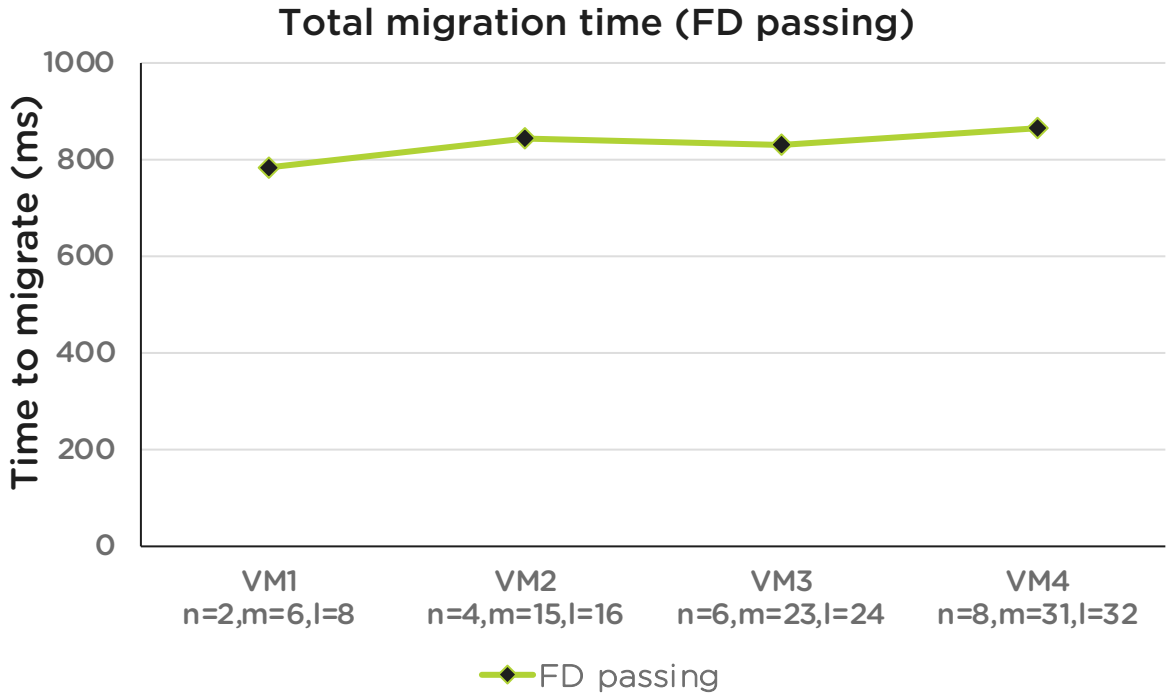


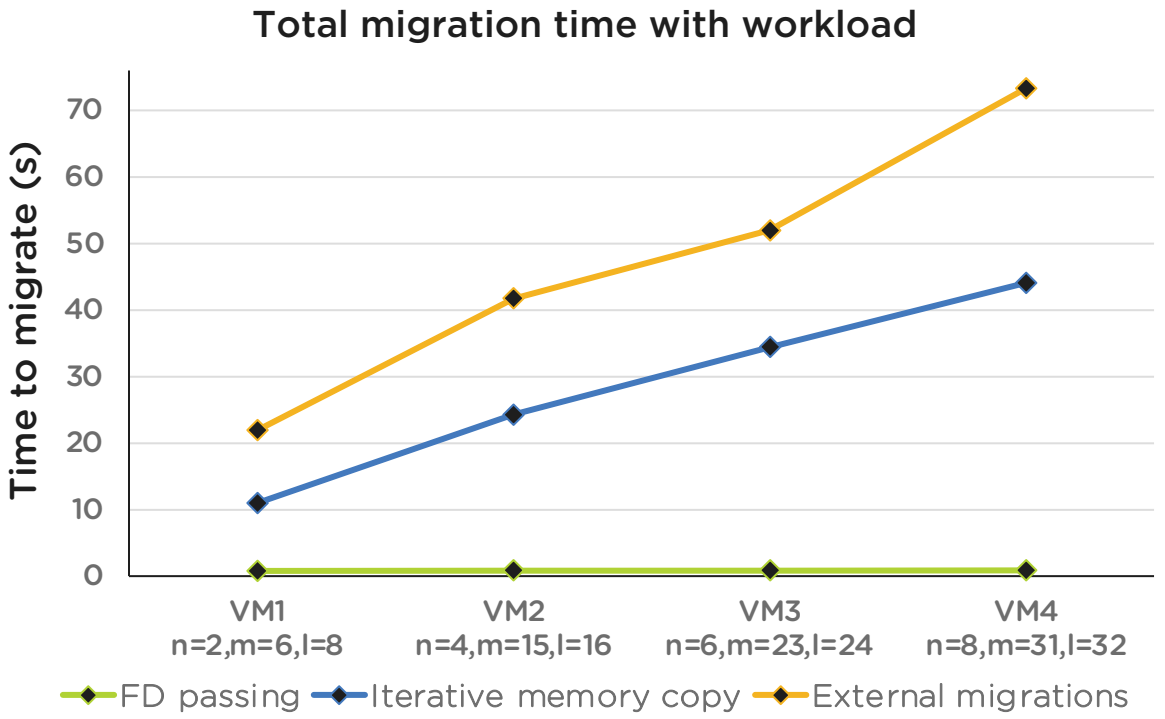
Initiate qemu upgrade from package manager

Local migrate VM to new qemu binary

Demo link

Results – migration time

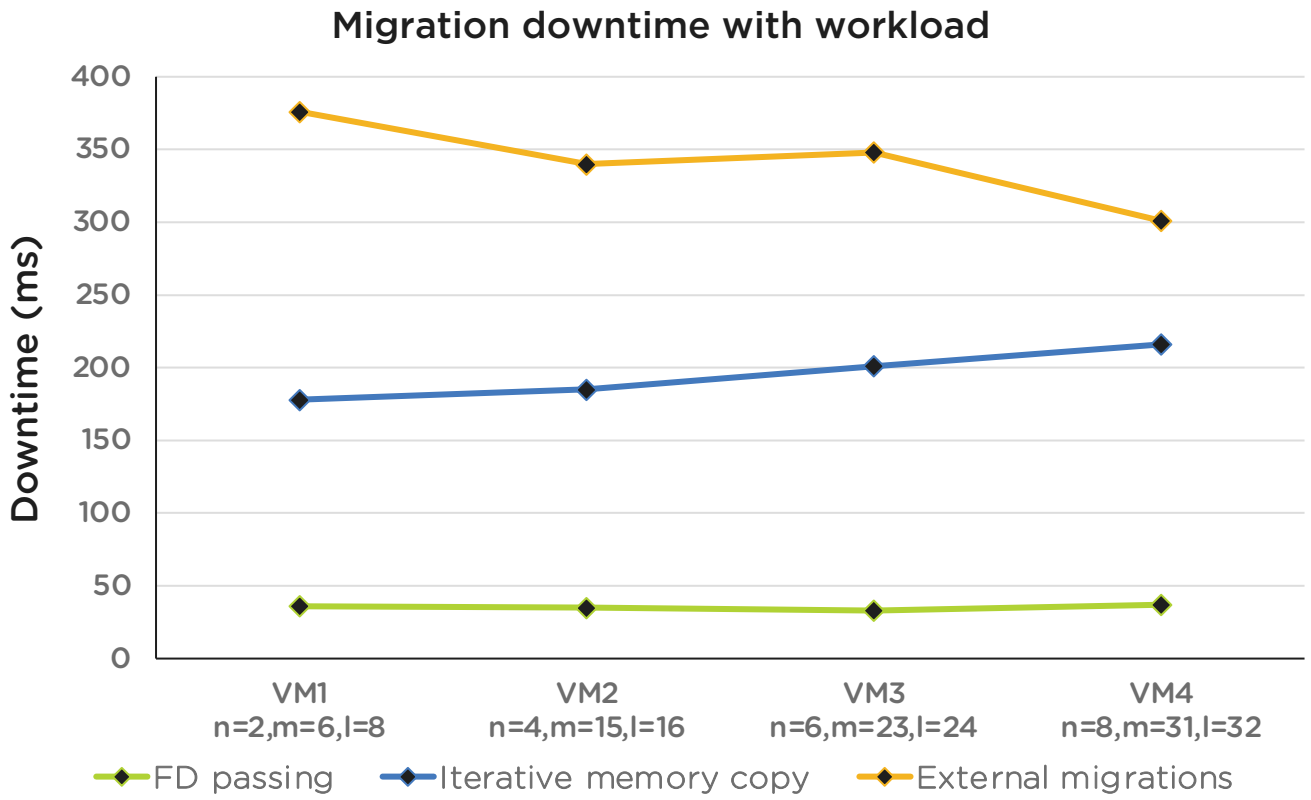




VM1 – 2 vCPUs, 8 GB memory (nested)
VM2 – 4 vCPUs, 16 GB memory (nested)
VM3 – 6 vCPUs, 24 GB memory (nested)
VM4 – 8 vCPUs, 32 GB memory (nested)

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