Lecture 5 : VM Live Migration

Dr. Bibhas Ghoshal

Assistant Professor

Department of Information Technology

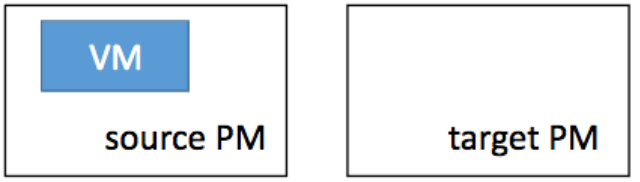
Indian Institute of Information Technology Allahabad

Slides Used in this Lecture have been adapted from slides of Professor Mythilli Vutukuru, Dept. Of CSE, IIT Bombay delivered for the course Virtualization and Cloud Computing

VM Live Migration

Migrate an entire VM from one physical host to another

- All user processes and kernel state
- Without having to shut down the machine



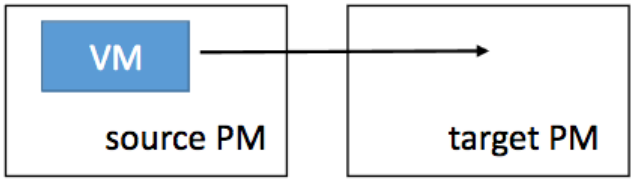
- Why migrate VMs?
  - Distribute VM load efficiently across servers in a cloud
  - System maintenance
- Easier than migrating processes
  - VM has a much narrower interface than a process
- Two main techniques: pre-copy and post-copy

Slide Author : Mythilli Vutukuru

What is migrated ?

CPU context of VM, contents of main memory

- Narrow interface, easier than process migration



Disk: assume NAS (network attached storage) that is accessible from both hosts, or local disk is mirrored

Do not consider migrating disk data

Network: assume both hosts on same LAN

- Migrate IP address, advertise new MAC address to IP mapping via ARP reply
- Migrate MAC address, let switches learn new MAC location
- Network packets redirected to new location (with transient losses)

I/O devices are provisioned at target

- Virtual I/O devices easier to migrate, direct device assignment of physical devices to VMs (device passthrough) makes migration harder

Slide Author : Mythilli Vutukuru

Steps to migrate a VM

Suppose we are migrating a VM from host A to host B

1. Setup target host B, reserve resources for the VM
2. Push phase: push some memory of VM from A to B
3. Stop-and-copy : stop the VM at A, copy CPU context, and some memory
4. Pull phase: Start VM at host B, pull further memory required from A
5. Clean up state from host A, migration complete

- Total migration time: time for steps 2,3,4
- Service downtime: time for step 3
- Other metrics: impact on application performance, network Bandwidth consumed, total pages transferred

Slide Author : Mythilli Vutukuru

Flavours of migration

Pure stop-and-copy :

- **VM stopped, all state transferred to target, VM restarts**
- Too much downtime to be classified as “live” migration

Pre-copy:

- **most state is transferred in the push phase**, followed by a **brief stop-and-copy phase (suited for interactive applications)**

Post-copy:

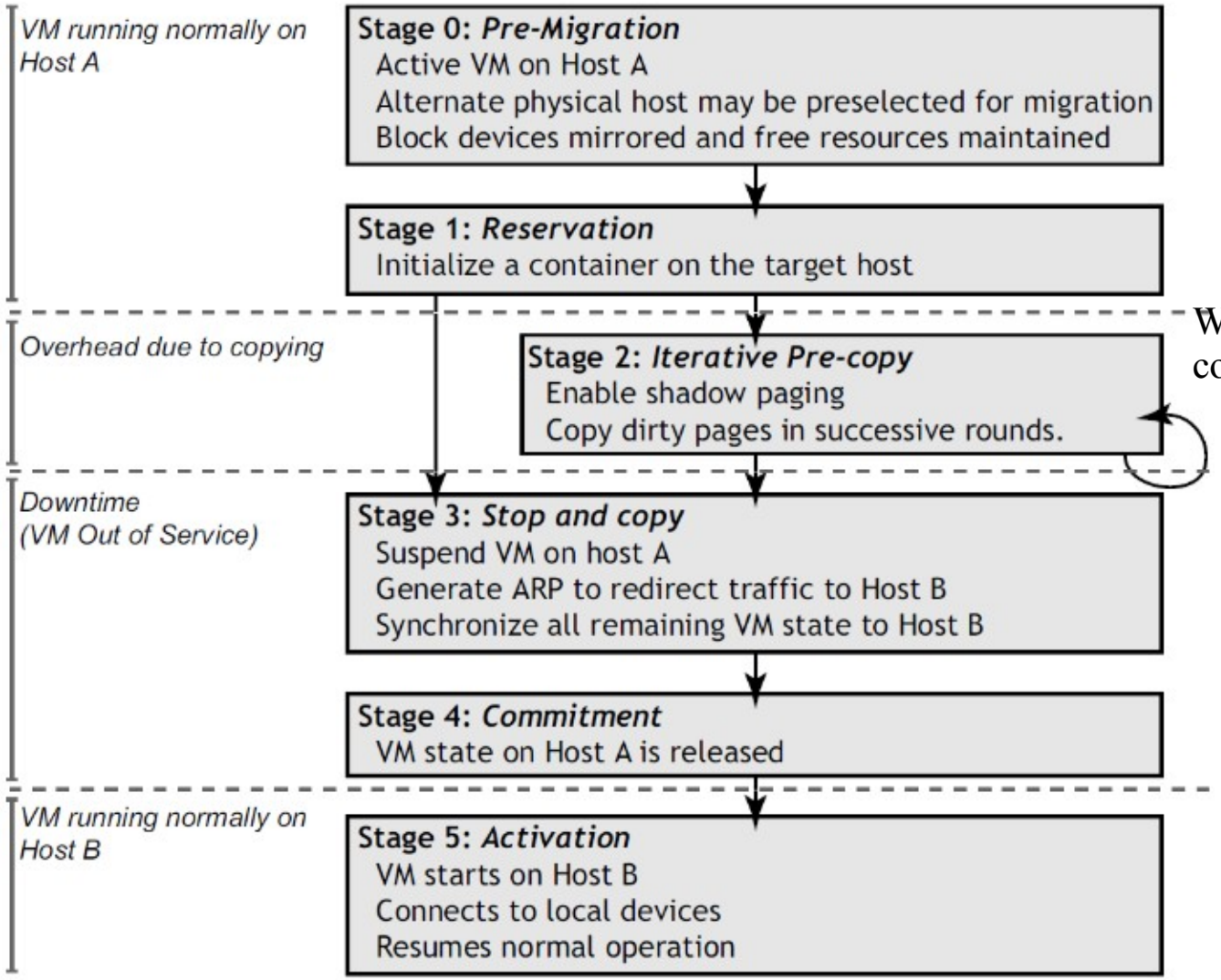
VM stopped, bare minimum state required to run the VM is transferred to the target host. Remaining state is pulled on demand **(suited for memory-intensive applications with large WWS)**

Hybrid:

- a **mix of pre-copy and post-copy**. Some pushing of state followed by stop-and-copy, followed by pulling of state on demand.

Slide Author : Mythilli Vutukuru

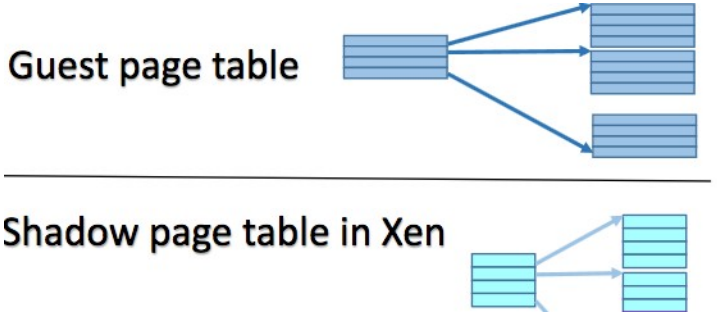
Pre-copy based live migration



Tracking Dirty Pages

Xen-based implementation

- Page tables in Xen maintained by guest
- Move to shadow page tables for migration
- Migration managed by control software in domain0



Shadow page table constructed on demand for every round

- Dirty bitmap maintained for every round
- Any page access by guest page fault to Xen, shadow page table updated
- PTE marked as read-only by default in shadow
- If valid write access, shadow PTE marked writeable, page marked dirty in bitmap
- At end of round, dirty pages are marked for transfer in control software
- Shadow page table and dirty bitmap reinitialized after every round
- Last set of dirty pages copied in stop-and-copy

Guest page table in target host changed based on new physical addresses

Optimizations

Avoid transferring page multiple times

- Before transmitting page, peek into the current round's dirty bitmap
- Skip transmission if page is already dirtied in ongoing round

Move non-interactive processes generating dirty pages to wait queue

- Execution paused until migration completes

Free up page cache and other unnecessary pages

- Reduce memory footprint

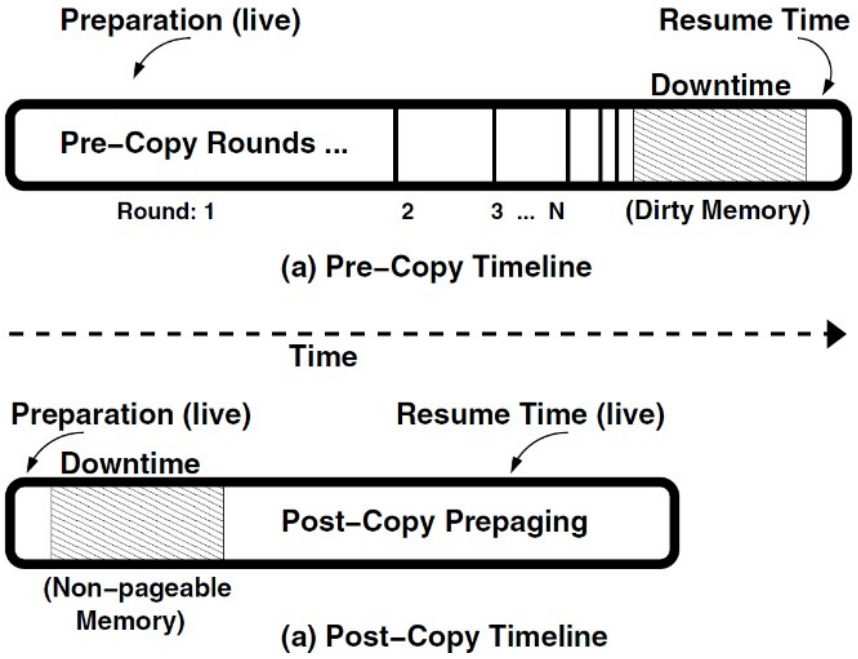
Post Copy Based Live Migration

Avoid multiple transfers of same page as happens in pre-copy

Prepare target, stop VM, copyCPU context and minimum memory to target

Start VM at target, pull memory from source via demand paging

- Memory access at target causes page fault, page fetched from source



Optimizations

Active pushing : source proactively pushes important pages, in addition to pulling pages via page faults

- Pre-paging : a “bubble” of pages around faulted page and proactively pushed, in anticipation of future accesses

- Dynamic self-ballooning : VM periodically frees up unnecessary memory and gives it back to hypervisor
  - Reduces memory footprint, speeds up page transfer
  - Performed carefully without hurting application performance
  - Can be used to optimize pre-copy migration as well
- Hybrid : one pre-copy round, followed by post copy

Page Pull at the Target

Pages at the target are pulled using "Pseudo-paging"

- Page to a pseudo, in-memory, swap device (part of domain0).

  No memory copy, just transfer pages across domains. Guest page table updated suitably.

- Only non-pageable memory transferred during stop-and-copy
- When guest resumes at target, fetch memory from pseudo-paging device via page fault mechanism
- Special swap device driver fetches from source over the network

Alternative: use shadow page tables

- If page fault to non-existent page at target, trap to hypervisor, fetch from source and update

Failures

What if target machine fails during migration?

- Pre-copy can simply abort the migration, restart with another target
    - With pre-copy, latest state is on source only, so can recover
- With post copy, source has stale memory, target has updated memory
    - If target crashes during post copy, cannot recover application data (unless some replication is performed)