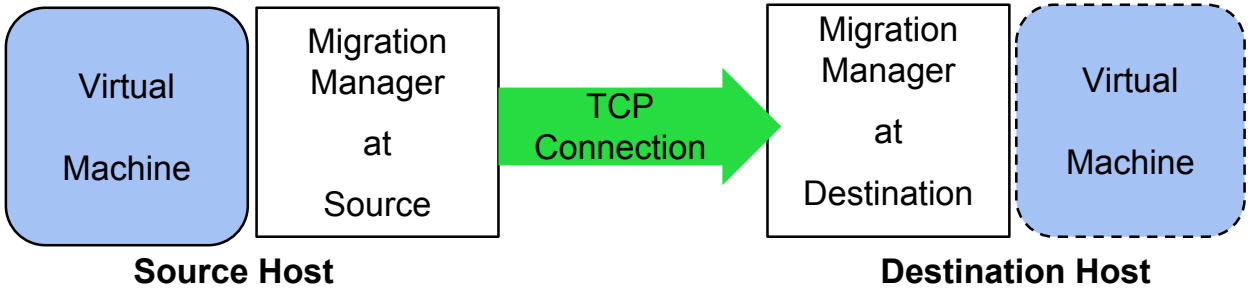
Live Migration of Virtual Machines

Pre-copy :Christopher Clarke, Keir Fraser, et. al. NSDI 2005

Post-copy: Hines, Deshpande, Gopalan, VEE 2009

What is live VM migration?



- Move a VM from one physical machine to another even as its applications continue to execute during migration

- Live VM migration usually involves
  - Migrating memory state
  - Migrating CPU state
  - Optionally, migrating virtual disk state

- Migration managers at source and destination
  - Connect via TCP connection
  - At source, the migration manager maps the guest VM's memory and execution state
  - Transfers VM's pages to the target migration manager over TCP connection.
  - At destination, the migration manager restores the VM's state and resumes execution
  - Migration manager examples: xend for Xen, QEMU for KVM

Why Live VM Migration?

- Why Migrate?
  - Load Balancing
    - Move VMs from highly loaded servers to lightly loaded servers
  - Server maintenance
    - When server needs to be upgraded
  - Energy savings
    - Move out VMs before shutting down servers to reduce energy usage
- Why live?
  - To keep long-running jobs alive
  - To keep network connections alive
  - Broadly, to avoid disruptions to users of VM
- Why VM?
  - Why not migrate individual processes?
  - Process migration may leave residual dependencies (state) at source host
    - E.g. system call redirection, shared memory, open files, inter-process communication, etc.

Performance Goals in Live Migration

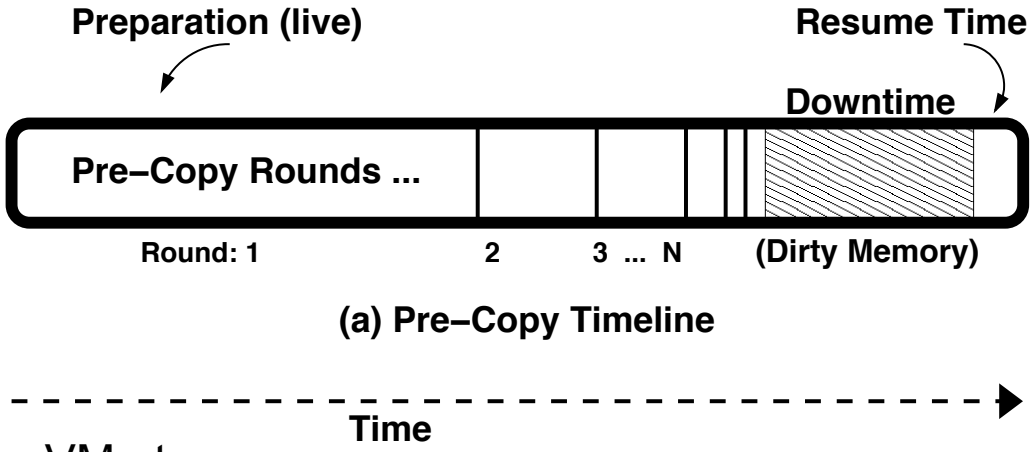
- Minimizing Downtime
- Reducing total migration time
- Avoiding interference with normal system activity
- Minimizing network activity

Migrating Memory

- Pure stop-and-copy
  - Freeze VM at source,
  - Copy the VM's pseudo-physical memory contents to target,
  - Restart VM at target
  - Long downtime.
  - Minimal total migration time = downtime

- Pure Demand Paging:
  - Freeze VM at source,
  - Copy minimal execution context to target
    - PC, Registers, non-pageable memory
  - Restart VM at target,
  - Pull memory contents form source as and when needed
  - Smaller downtime
  - Sloooow warm-up phase at target during page-faults across network

Pre-copy migration



- DON'T freeze VM at source
  - Let the VM continue to run

- Copy VM's pseudo-physical memory contents to target over multiple iterations
  - First iteration ➔ copy all pages.
  - Each subsequent iteration ➔ copy pages that were dirtied by the VM during the previous iteration

- Do a short stop-and-copy when number of dirty pages is "small enough".

- But what if number of dirty pages never converges to a small enough number?
  - After a fixed number of iterations, give up and stop-and-copy.

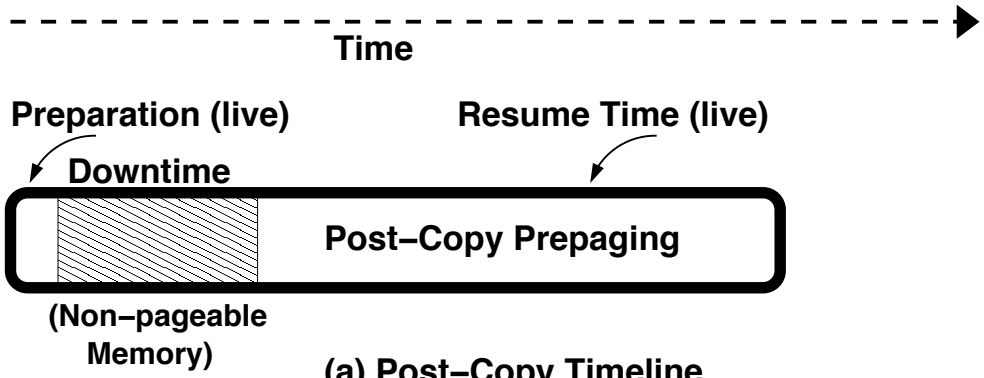
So what's the catch? How do we track dirtied pages?

- Mark the VM's memory pages as read-only after each iteration.
- Trap write operations via hypervisor to xend and track dirtied pages.
- Reset after each iteration
- Works well as long as writes are infrequent

Optimizations

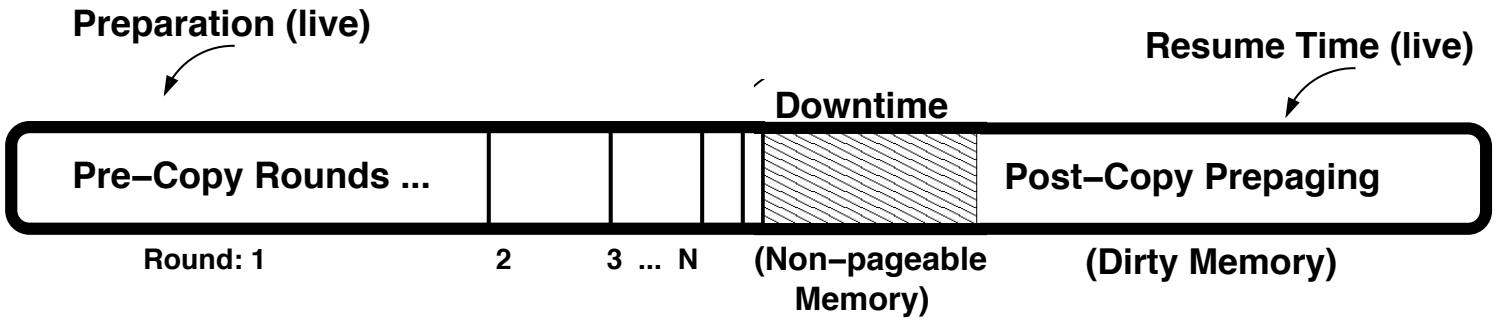
- Limit the bandwidth used by migration
  - To minimize impact on running services
- Stun Rogue Processes
  - Those that don’t stop dirtying memory
- Free Page Cache Pages
  - Can be re-cached at target
  - Potential performance hit

Post-copy migration



- Freeze the VM first
- Migrate CPU state and minimum state to destination
- Start VM at the target, but without its memory!
- Transfer memory by concurrently doing the following
  - Demand paging over network
  - Actively pushing from source
  - Hopefully most pages will be pushed BEFORE they are demand paged.
- Advantage:
  - Each page transferred over the network only once.
  - Deterministic total migration time
- Disadvantage:
  - Cold start penalty at the destination
  - If migration fails, then VM is lost.

Hybrid pre/post-copy



- Combines the benefits & drawbacks of both

1. Perform one or more rounds of live pre-copy rounds
2. Pause VM and transfer execution state
3. Use post-copy to transfer any remaining dirty pages from source

Migrating Network Connections

- Within a LAN,
  - the migrated VM carries its IP address, MAC address, and all protocol state, including any open sockets
  - Backward (re)learning delay at the network switches
    - Switches needs to re-learn the new location of migrated VM's MAC address
    - Solution: Send an unsolicited ARP reply from the target host.
    - Intermediate switches will re-learn automatically.
    - Few in-flight packets might get lost.
- Across a WAN (wide-area network)
  - Source and destination subnets may have different IP addresses.
  - Active network connections may need to be tunneled via VPN or similar mechanisms.

Storage Migration

- Many gigabytes of local disk image possible.

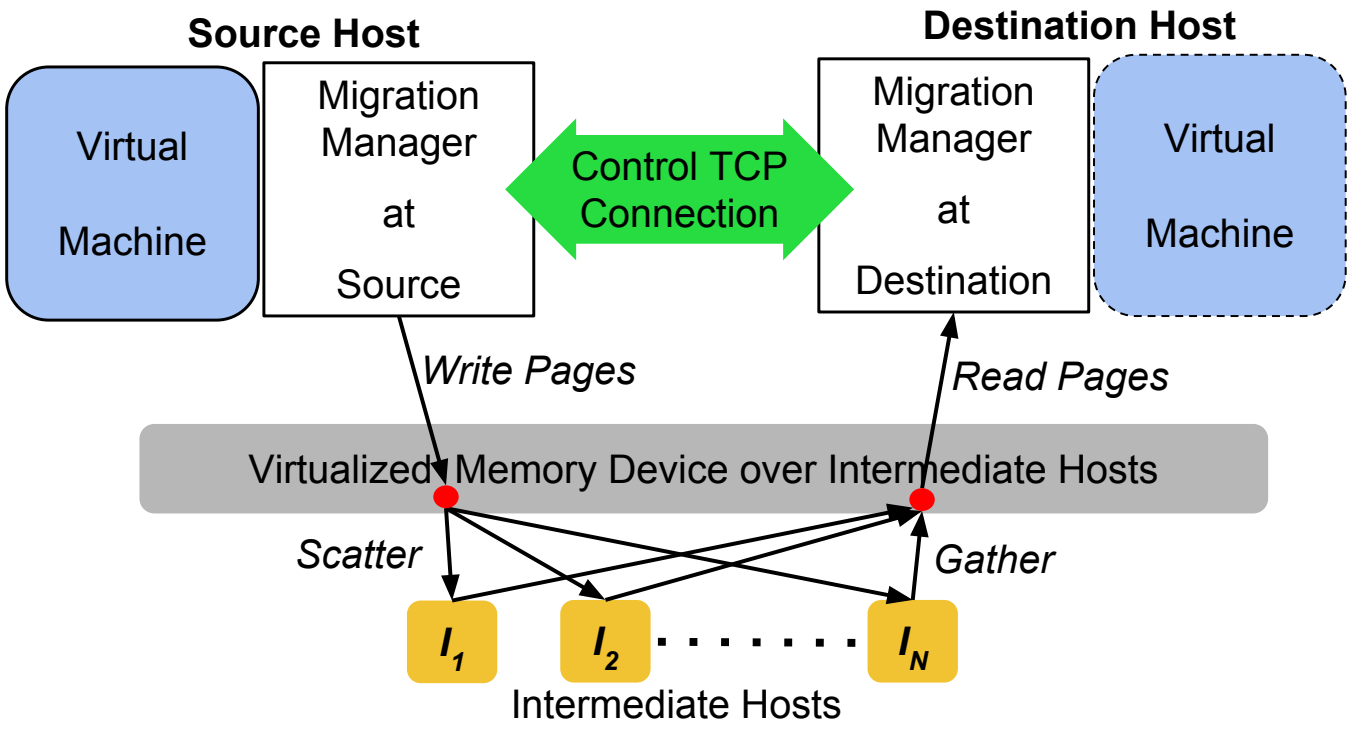
- For LAN
  - Assume the storage is over the network and remains accessible from the new target machine.
  - E.g. Network File System (NFS), or Network Block Device(NBD), or iSCSI etc.

- For WAN
  - Disk image may need to be transferred.
  - Can use pre-copy or post-copy for disk images,
  - Combined bandwidth saving optimizations such as compression, and/or de-duplication.

Self Migration

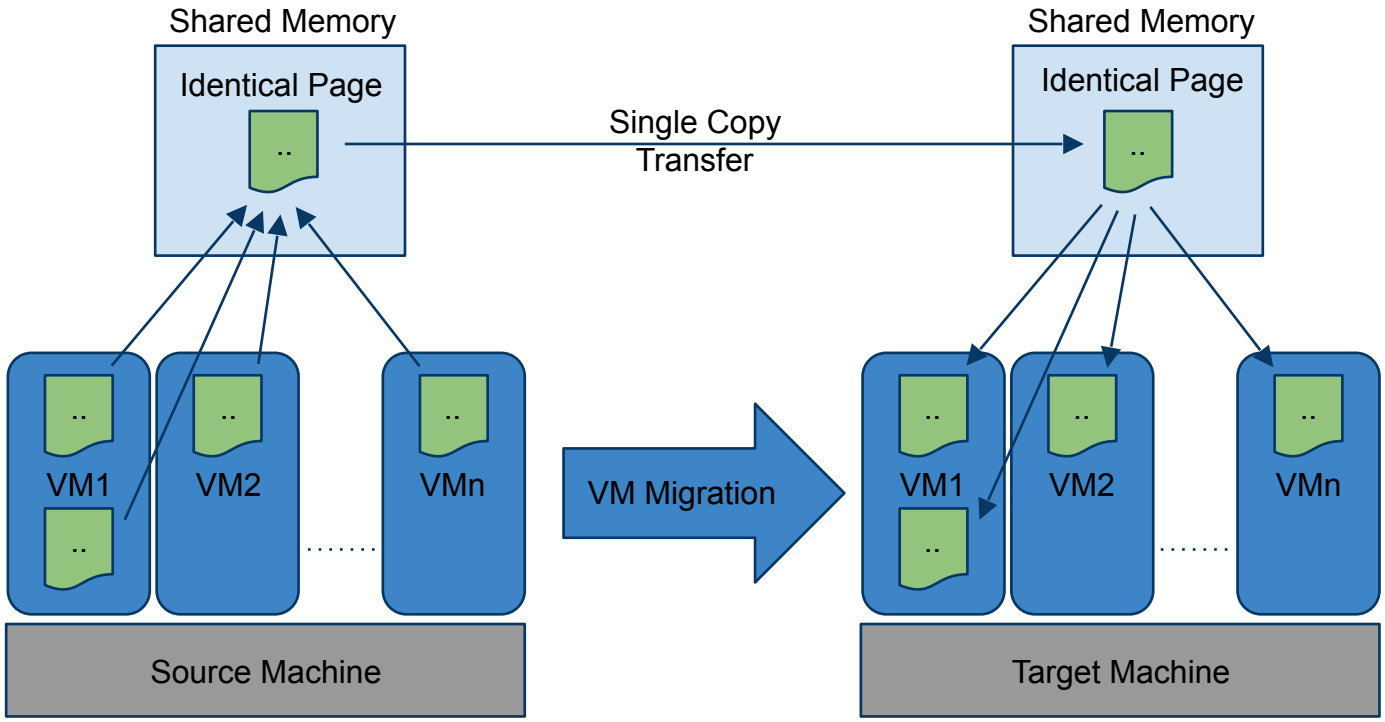
- Guest OS migrates itself (mostly)
  - No migration manager needed at source!
- Migration stub needed at destination
  - Fresh VM that receives guest state and restores migrated VM's state.
- Challenge:
  - OS must continue to execute while transferring its final state.
  - Perform a careful (complicated) 2-stage checkpoint and copy.

Scatter-Gather migration



Scatter-Gather migration: The VM's state is transferred through intermediaries. A direct connection between the source and destination carries control information, faulted pages, and some actively pushed pages.

Multi-VM (Gang) Migration



- De-duplicate memory pages to reduce network traffic.
- Identify identical pages across multiple VMs
  - By comparing byte-wise (expensive), or checksum (cheaper)
- Send only one copy of identical page to destination node
- Destination Node replicates the pages to multiple VMs.