CS 695: Virtualization and Cloud Computing

Lecture 9: VM Live Migration

Mythili Vutukuru
IIT Bombay
Spring 2021
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

“Live Migration of Virtual Machines”, Christopher Clark, Keir Fraser, Steven Hand, Jacob Gorm Hansen, Eric Jul, Christian Limpach, Ian Pratt, Andrew Warfield
“Post-Copy Based Live Virtual Machine Migration Using Adaptive Pre-Paging and Dynamic Self-Ballooning”, Michael R. Hines and Kartik Gopalan
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
  - We 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
Steps to migrate a VM

• Broad steps in any migration technique: 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 any 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
Flavors of migration techniques

• **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

• **Post-copy**: VM stopped, bare minimum state required to run the VM is transferred to the target host. Remaining state is pulled on demand while the VM is running at the new location.

• **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.
Pre-copy based live migration

- Iterative pre-copy + stop-and-copy for remaining memory
- First push round copies all pages
- Every round copies pages dirtied in previous round
  - A page maybe copied multiple times
- Writable Working Set (WWS): pages commonly written to
  - WWS will be copied multiple times
  - Finally transferred in stop-and-copy
- How many rounds? Stop when rate of dirtying > rate of transfer
  - Diminishing returns with more than few rounds
Impact of iterative pre-copy

- If stop-and-copy, 512MB VM, 128 Mbps network, downtime = 32 sec
- With one pre-copy round, downtime goes to 2-3 sec
  - ~1 second for 2 or more rounds

Effect of Bandwidth and Pre-Copy Iterations on Migration Downtime
(Based on a page trace of OLTP Database Benchmark)
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 \(\rightarrow\) 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
Some 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
  • Much like ballooning
Pre-copy performance

- Downtime: ~100 millisecond, total migration time of few tens of seconds
- Worse for memory-intensive applications, better for interactive apps
Post-copy based live migration

- Avoid multiple transfers of same page as happens in pre-copy
- Prepare target, stop VM, copy CPU 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
Implementation details (Xen)

- How are pages pulled at target? “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
What about 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)
Post copy performance

- Longer downtime as compared to pre-copy, but lower total migration time, fewer page transfers, lesser disruption to application.
Summary

• VM live migration techniques
  • Iterative pre-copy vs post-copy via demand paging
  • Implementation details on Xen
  • Performance comparison

• Which is better?
  • Pre-copy suited for interactive application
  • Post copy is better for memory-intensive applications with large WWS
  • Hybrid techniques are also used