Aggressive Server Consolidation Through Pageable Virtual Machines

Anton Burtsev, Mike Hibler, Jay Lepreau

emulab

THE UNIVERSITY OF UTAH

Motivation

Many environments host a large number of VMs
- Testbeds like Emulab
  - Long-running experiments
  - Large-scale experiments
- Honeypots
- Datacenters
- VMs for individual use

Historically, VMs choose isolation as primary goal
Sharing opportunities are neglected
- VMs run similar OS and FS image,
- spend a fair amount of time idle

Goal

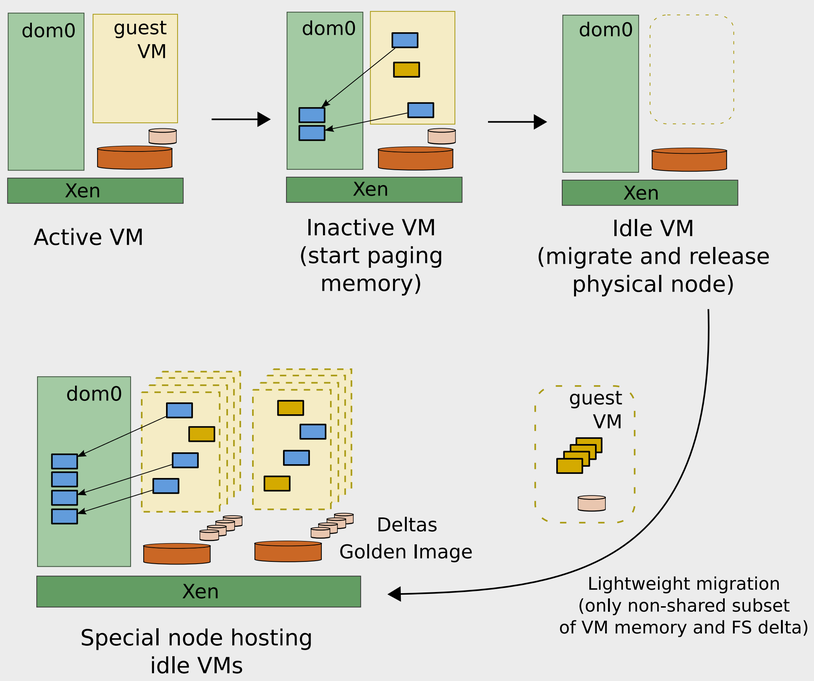
Scale to hundreds of virtual machines on a single node
- Keep idle VMs forever
- Available on demand

Key Techniques

Page out idle VMs
- Detect idleness
- Page out VM memory
  - Predict minimal working set
- Migrate

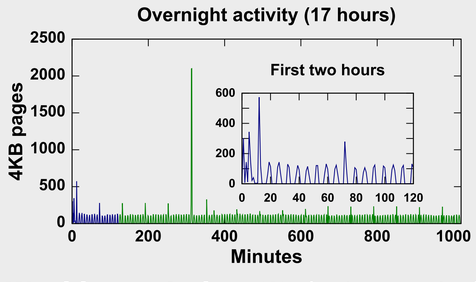
Share resources across VMs
- Memory sharing
  - Copy-on-write (CoW) content-based sharing
- File system sharing
  - Versioning storage
  - Golden imaging

Swapping Overview



VM's lifecycle: Detect when virtual machine becomes inactive. Infer a minimal working set needed for a continuous operation and swap out the rest of the VM's memory. If we predict a long period of inactivity, we can release the physical node entirely by migrating the virtual machine along with its local file system to a special node hosting idle VMs. Here we detect all pages that are identical across VMs and share them in a copy-on-write manner.

Minimal Working Set

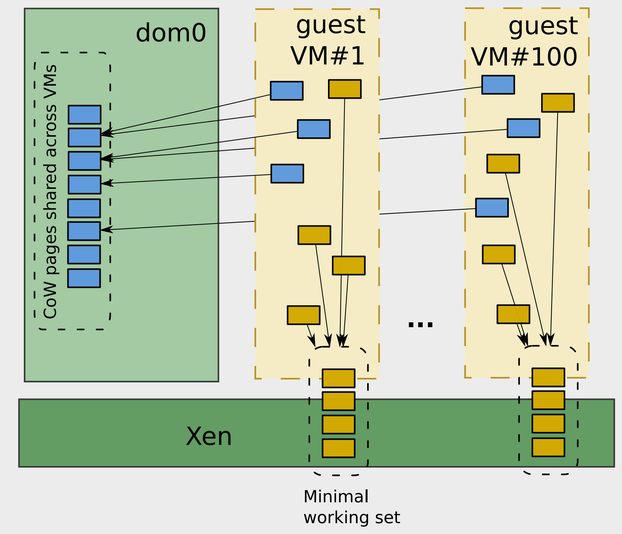


	Working Set Size	
	4KB Pages	MBs
Idle VM	445	1.8
Idle Bash shell	725	2.9
Ping response	776	3.1
Idle ssh	805	3.2
SCP out 20MB 348KB/s	856	3.5
SCP in 20MB 348KB/s	7723	31.6
Find / -name *	6098	24.9
One minute Vim editing session	1432	5.8

Working set size: We instrumented the Xen VMM to measure a working set size of an idle Linux VM. The plot depicts changes in a working set size during a 17 hour "idle" experiment. Inner plot zooms into the first two hours of experiment. The table provides working set sizes for typical "idle" OS activities.

This material is based upon work supported by the National Science Foundation under Grant No. CNS-0524096. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation

Implementation



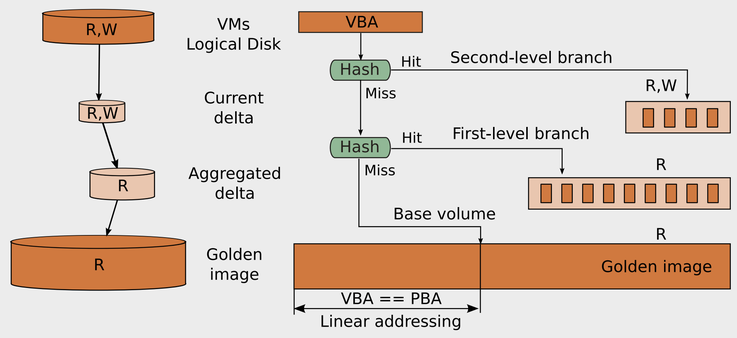
Memory management: VM's virtual memory consists of pages shared across all VMs on a single node (*blue*), and a minimal working set (*yellow*).

Idleness detection:
- Heuristic:
  - VM doesn't use its scheduling quantum entirely
  - Plus combination of existing techniques [Golding et al.]

Timely response:
- Build a minimal working set for each event
- Keep working sets of frequent events in memory
  - Timer interrupts
  - Network packets
- Swap in predicted working set before delivering event

Memory management:
- Extend Xen to support paging and copy-on-write
- Content-based sharing across VMs
  - Identical pages: hash comparision or disk I/O monitoring

File system sharing and migration:



Translation of a virtual block address (VBA) to a physical (PBA). Hash table is used for indexing.

Three-level storage
- Template golden image (R)
- Set of changes since last migration (R)
- Set of write changes since last active period (RW)

Leverage special case:
- Golden image: linear address
  - No need to use radix tree
  - Disk access locality
- Delta: clustered hash table
  - Guarded page table for large delta

Network stack virtualization:
- Support for hundreds of networks on a Linux node
- Emulation of non-trivial network topologies on a single node

V2P Migration:
- Migrate virtual machine to a physical machine on the fly
  - Realism and performance of real hardware
  - Consolidation when idle

Status

We have implemented CoW branching storage. Everything else is at the earliest stage of research. We are investigating possibilities for providing efficient implementations of paging and CoW memory sharing for Xen. Deriving sound idleness and working set prediction functions is the most challenging part of this work.