Aggressive Server Consolidation Through Pageable Virtual Machines
Anton Burtsiev, Mike Hibler, Jay Lepreau

Motivation
Traditionally virtual machines choose isolation over sharing
But sharing is important in many cases
- Emulab:
  - Long-running experiments
  - Experiments with unlimited number of nodes
- Honeyport installations
- Datacenters
  - Provide VMs in private use

Goal
Scale to a 100 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 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, which are identical across VMs and share them in a copy-on-write manner.

Minimal Working Set
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.

Implementation
Memory management: VM’s virtual memory consists of a 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
  - Combine with 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 comparison or disk I/O monitoring

File system sharing and migration:
Translation of a virtual block address (VBA) to a physical (PBA). Clustered hash table is used for indexing.

Versioning storage:
- Template golden image
- Set of changes since last migration (RO)
- Write changes (RW)
- Only data needed for migration

Leverage special case:
- Golden image: linear address
  - No need to use radix tree
  - Save metadata memory
- Delta: clustered hash table
  - Guarded page table for large deltas

Network stack virtualization:
- Avoid revisitation after migration (802.11q)
- Support for 100 of networks on a Linux node

V2P Migration:
- Migrate virtual machine to a physical 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 investigate possibilities to provide efficient implementation of paging and CoW memory sharing for Xen. Deriving a sound idleness and working set prediction functions is the most challenging part of this work. V2P and P2V migration is a slick and powerful feature by itself.