Aggressive Server Consolidation through Pageable Virtual Machines

Anton Burtsev Mike Hibler Jay Lepreau

University of Utah, School of Computing

Historically, virtual machine monitors (VMMs) have chosen isolation as a primary goal. This design choice prohibits almost any form of resource sharing. The only shared resource is a physical CPU. Memory and disk space are statically preallocated upon creation of a virtual machine (VM). As a result, 5-10 VMs can easily exhaust memory and disk resources of a physical host.

There are many situations, however, in which it would be desirable to keep hundreds of VMs around. In many cases VMs are used only occasionally, but need to be available on demand. Examples include long-running experiments in network testbeds like Emulab, months-long debugging and development sessions, experiments with unlimited number of nodes (with VMs allocated on demand), honeypot installations, and public machines in data centers and organizations, where users are allowed to keep a running copy of a VM.

We have started to extend the Xen VMM with the ability to host hundreds of VMs on a single physical node. Similar to demand paging of virtual memory, we will page-out idle VMs, making them available on demand. Paging is transparent. An idle VM remains operational: it is able to process timer events, maintain network connections, and respond to requests from remote machines and local devices.

We will detect when a VM becomes idle and aggressively reclaim its resources. We will identify and leave in memory only a minimal working set of pages required to maintain the illusion of running VM. To keep the number of active pages small without harming performance dramatically, we intend to build a correspondence between every event and its working set. Reducing the working set further, we plan to implement copy-on-write page sharing across VMs running on the same host.

If we predict a long period of inactivity, we release the physical node entirely by migrating the VM along with its local file system to a special node hosting idle VMs. If we predict a period of active execution, we proactively start process of swapping the VM in.

Several existing systems increase server consolidation by using memory sharing across VMs. For example, Potemkin implements a virtual honeypot capable of hosting hundreds of virtual machines on a single physical host (Vrable et al., SOSP'05). However, it limits all VMs to be started from the same memory image. The VMware VMM implements copy-on-write memory sharing across VMs running on the same node. However, it is unable to achieve our desired degree of consolidation because it does not attempt to page-out the memory of an idle VM.

Idleness detection: A good indication that a VM has become idle is that it returns to the hypervisor before exhausting its scheduling quantum. We can stress the VM further by depriving it of CPU time. For that, we gradually reduce the length and frequency of the VM's CPU quantum. We intend to combine these techniques with existing approaches for detecting and predicting periods of idleness (Golding et al., USENIX Winter'95).

Timely response from a swapped VM: By keeping a small working set in memory, we create the illusion of a running VM. To optimize event processing in the swapped state, we try to predict the next event and swap in the corresponding working set proactively. Our default heuristic is to prepare for processing of the most frequent events, which are incoming network packets and timer interrupts. We keep in memory an active set corresponding to these events. If we predict that the next event is a wake-up of a guest task, we try to extract information about the task from the guest OS and swap in its working set.

Memory management: Virtual installations run VMs from a small set of preinstalled operating system images. As a result, many identical memory pages can be found and shared across VMs. Shared pages can be identified either by comparing page table hashes, or, more efficiently, by monitoring the VM's disk I/O (Bugnion et al., ACM TOCS'97).

File system sharing and migration: Similar to memory, the file systems of all VMs initialized from the same system image are almost identical, and vary only with a small set of write changes. To reflect this fact, we use a "golden" file system image from which we start all VMs. The golden image remains unchanged and thus can be shared across any number of VMs. To provide the VM with a write access, we have developed a fast versioning block-level storage solution.

Network stack virtualization: To host hundreds of VMs on a single node, we need adequate support from a VMM's network stack. The Linux protocol stack used currently by Xen is not designed to provide this support efficiently. Further, for migrating multiple idle VMs on a single host, we will need to support emulation of a non-trivial network topologies on a single network stack.

Status: We have implemented the copy-on-write branching storage system mentioned above. Everything else is at the earliest stage of research. Deriving sound idleness and working set prediction functions is the most challenging part of this work.

Students: Burtsev. Corresponding author: aburtsev@cs.utah.edu. No demo is planned.