Table of Contents

Kernel-based Virtual Machine **1**
- Monolithic and "Micro"-hypervisors 1
- KVM compared to Xen 1
- Performance 2
  - CPU benchmark results 2
  - I/O benchmark results 2
  - Geant4 benchmark results 2

Kernel-based Virtual Machine

KVM is an alternative VMM to Xen that supports full virtualization with Intel VT and AMD SVM hardware. Similarly to HVM (VT and SVM) virtualized Xen, it uses a modified version of Qemu to emulate and virtualize hardware.

Monolithic and "Micro"-hypervisors

In Xen, the virtual drivers run in a different address space than the hypervisor, which essentially makes it a "micro"-hypervisor. There have been many discussions about monolithic versus micro-kernels, and often they boil down to security/stability versus performance - since switching address spaces (context switch) takes extra time micro-kernels are slower, but running many processes in the same address space can be hazardous. The same arguments apply to monolithic and micro-hypervisors.

The virtual drivers that Xen exposes to its domains run in a special VM, dom0. Each time the virtual drivers in dom0 needs to manage for example multiplexing of a packet from the network driver, the VMM scheduler switches to dom0 kernel space, and the packet is multiplexed in dom0 userspace. Receiving this packet in domU thus incurs at least six context switches: domU -> VMM scheduler -> dom0 kernel space -> dom0 user space -> dom0 kernel space -> VMM scheduler -> domU

KVM is not really a monolithic hypervisor, either. But it's closer: VM -> Linux kernel space -> Linux user space -> Linux kernel space -> VM

A monolithic hypervisor would multiplex drivers in the same address space as the VMM, and only two context switches would be incurred: VM -> VMM space -> VM

KVM compared to Xen

- Qemu, and, in effect, KVM and HVM Xen, use a particular kind of OS Image to boot a VM, which includes the kernel and a boot sector, similar to a bootable CD. This is different from a paravirt Xen OS image, which only includes the userspace parts of the OS image and is a raw copy of a volume with a root filesystem.
- In KVM, a VM runs on top of a normal Linux kernel as a normal process. For example, using the 'kill' command on a KVM process kills the VM.
- There is no logical network between the VMs and the VMM's physical network interface. This can be an advantage or a disadvantage:
    - There is less isolation between VMs and VMM, which can affect stability and security.
    - In terms of performance, it is hard to say if one approach is better than the other, but having a virtual network driver between VMs gives a performance penalty, and direct access to hardware is always faster. In both KVM and Xen it should be possible to bypass the driver domain (dom0 in Xen or the KVM VMM) and, with Xen paravirt, this has already been done, allowing VMs to have direct access to hardware. Bypassing the driver domain, however, is harder to implement in full virtualization (Xen HVM and KVM), although this may be solved with newer virtualization hardware extensions (e.g. VT-d).
- KVM does not yet support SMP in VMs.
- KVM is in the vanilla Linux 2.6.20 kernel and can be patched against older kernels. The source code profile of KVM is less intrusive on the kernel source and should be more easy to backport to older versions of Linux.
- KVM relies on the availability of VT or SVM hardware extensions of x86 CPUs, which should be available in most new PC computers.

Performance

The benchmark results below are not as good as some other results found http://www.phoronix.com/scan.php?page=article&item=623&num=1 and http://linux.inet.hr/finally-user-friendly-virtualization-for-linux.html, some of which show that KVM in certain cases performs better than Xen.

The following results are adapted from http://cern.ch/hbjerke/benchmark.xml

CPU benchmark results

CPU: time factor 3546946890454528713

Native:
real 0m4.858s
user 0m4.336s
sys 0m0.032s

KVM:
real 0m7.076s
user 0m6.980s
sys 0m0.014s

KVM CPU performance: 69 %

I/O benchmark results

IO: dd if=/dev/zero of=test bs=1M count=1000
time dd if=test of=testout

Native:
real 0m12.159s
user 0m0.740s
sys 0m10.829s

KVM:
real 8m21.899s
user 0m2.549s
sys 0m47.874s

KVM I/O performance: 23 %

Geant4 benchmark results

Native: 51.8 s

KVM: 55.8 s

Performance: 93 %

-- Main.hbjerke - 02 Feb 2007

This topic: Virtualization > KVM
Topic revision: r5 - 2010-11-05 - JuanManuelGuijarro