



T.J. Watson Research Center

HVM - Hardware Virtual Machine Abstraction Layer (Integrating Intel VT-x and AMD SVM into Xen)

Leendert van Doorn
leendert@watson.ibm.com

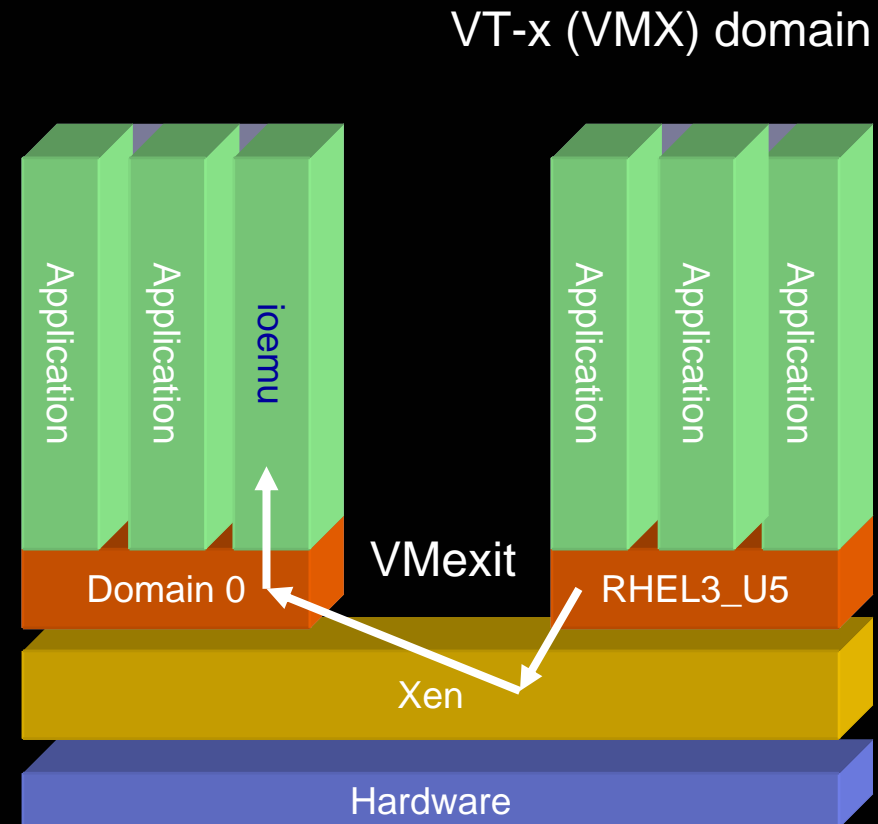
January 2006

Topics

- Full Virtualization Support in xen-unstable
- HVM Motivation and Goals
- HVM Abstraction Layer
 - Top-Level Interface
 - Bottom-Level Interface
- Domain Builder and HVMLoader
- Statistics
- Things to Do

Full Virtualization Support in xen-unstable

- Current xen-unstable contains VT-x/VT-i support
- Most device emulation is implemented in ioemu (PCI, VGA, IDE, NE2100, ...)
- High performance drivers, such as ioapic, lapic, vpit are implemented in Xen
- Firmware (vmxloader) is initially loaded by the builder
- Developed by Intel with contributions from IBM



HVM Motivation

- Two incompatible hardware virtualization abstractions for x86:
 - Intel VT-x
 - AMD SVM
- The architectures are from 10,000 ft. view they are very similar
 - Each has their own advantages/disadvantages
- October situation: two different incompatible Xen trees
- Independent 3rd party (IBM) needed to do the integration
- AMD, IBM, Intel and XenSource (Cambridge) contributed to the HVM design and architecture

HVM Goals

- Integrate both VT-x and SVM support into a single tree
 - Partially shared with VT-i
- Avoid regression (performance/function) of current VT-x functionality
- Factor out major common code
 - I/O emulation (ioemu)
 - MMIO decoding, communication with device model
 - Device models embedded in Xen (vioapic, vlapic, vpic, vpit)
 - Domain builder (xc_hvm_builder)
 - Firmware (Hvmloder was vmxloader)
- Define vendor specific code
 - For the moment we do not factor out common code in vendor specific parts

Top-Level HVM Interface

- Xen invokes VT-x/SVM functions through the top-level HVM interface
- Simple indirection layer that is filled in at CPU initialization time
E.g. Xen calls `hvm_realmode()` which is an indirection to `vmx_realmode()`
- Interface consists of:
 - Construct domain (initialize/relinquish guest domain resources)
 - Load/store guest state (`cpu_user_regs`, control regs, MSRs (x86_64))
 - Auxiliary state (`realmode`, `paging_enabled`, `instruction_length`)

Bottom-Level HVM Interface

- Vendor specific code calls HVM functions through static binding
- Interface consists of calls to functions
 - hvm intercepts (io, mmio)
 - Shadow page handling
 - Vioapic, vlapic, vpit, ...
- This interface is essentially unchanged, except that they have been given the prefix `hvm_` and use the HVM top-level interface to call back into the vendor specific code

Domain Builder and HVMLoader

- Replaced vmx domain builder by hvm domain builder
 - New hvm builder type
 - Builder is shared by VT-x and SVM
 - All user-level software (xend/libxc) is unaware of the difference
- Replaced vmxloader by hvmloader
 - Hvmloader is aware of the difference (uses cpuid AuthenticAMD)
 - Iff AMD then do a VMMSCALL to kick domain into paged realmode
 - Iff intel then invoke vmxassist (realmode emulator)
 - Both run the same BIOS/VGABIOS/ACPI code
- From a user's PoV there is no difference between VT-x or SVM

Statistics

- Lines of Code (C, assembly, headers):
 - Xen: Intel VT-x specific code: 3718 (3.7%)
 - Xen: AMD SVM specific code: 5721 (5.6%)
 - Xen: Common HVM code: 5794 (5.7%)
 - Tools: Common HVM code: 86313 (85%)
- Pull HVM tree from:
 - <http://xenbits.xensource.com/ext/xen-unstable-hvm.hg>
 - (in sync with xen-unstable.hg every evening)

Things to Do

- HVM integration into 3.0.x:
 - VT-i builds but has not been tested
 - Everything can use more testing
- For 3.1:
 - There are 3 instruction decoders in Xen, need to reduce this to one
 - Add batching to the ioemu communication protocol (W batch/R don't)
 - Device model as a partition or PAL code
 - Better device models (one that passes HCT for example)
 - SMP support for fully virtualized guests
 - Suspend/Resume/Migration