The Next Generation of XenLinux

Jun Nakajima

Intel Open Source Technology Center

Agenda

- Reviewing para-virtualization in Linux
- Hybrid-virtualization
- Current status and next steps

Reviewing Para-Virtualization

- Para-virtualization has been proven to be effective and efficient
  - If the guest operating system indicates its intent to the VMM, then it can obtain better performance and efficiency when running in a virtual machine.
- But..

Reviewing Para-Virtualization (cont.)

- Virtual CPU has significant limitations and different behaviors
  - Hard for the kernel developers to fully understand “virtual CPU” because it’s different from the native CPU
    - Ring compression, no privileged instructions, no page-level protection,
  - The functionality/definition of “virtual CPU” can be different on different VMMs (VMware vs. Xen, for example)
  - Or ever changing, not well-defined

Reviewing Para-Virtualization (cont.)

- Virtual CPU has inevitable overheads
  - Fast system calls are no longer fast
  - Efficient TLB usage
    - No global pages for the kernel
- Does not take advantage of HW assists
- Dom0/domU kernel binaries are not compatible with the native

Solution – Hybrid-Virtualization

- Use hardware-assisted virtualization
  - Consistent and well-defined CPU behavior
  - Benefit from future silicon enhancements for hardware-assisted virtualization
    - More features, lower VM exit/entry costs
- Use para-virtualization on the focused areas
  - Starting from hardware-assisted full-virtualization
    - Easier to share the kernel binary with the native
  - Reduce paravirt operations (76 of them for x86 today) significantly
- Reuse Xen's para-virtualization

Solution – Hybrid-Virtualization (cont.)

- The kernel regains the native CPU features lost in software-only para-virtualization
  - Fast system calls
  - Global pages
  - Paging-based protection (U/S), etc.
  - Privileged instructions
  - GDT, IDT, LDT, TSS, cli/sti, etc.
    - Standard exceptions/interrupts
- Efficient handling in hardware-assisted virtualization
  - Example: VMX Error Code Filtering – #PF in user mode is raised directly to the guest kernel without causing a VM exit
  - I/O, MSR bitmap
  - The cost of VM exit/entry will be even lower in the future
    - Cost of VMCALL is lower than other VM exits
  - VPID

Focused Areas for Para-Virtualization

- I/O devices, such as network and disk
- Timer
- idle handling
- interrupt controllers
- MMU
  - Direct page table mode
  - Or hardware-assisted (i.e. EPT or NPT)

Hybrid-Virtualization Linux

- "Next Generation of XenLinux" is hybrid-virtualization Linux
  - Boots from the real mode like the native; the early initialization is identical
  - Switch to the direct mode if hybrid-virtualization is detected
    1. Use CPUID (leaf 0x40000000) to detect it on Xen/KVM
    2. Installs the hypercall page for hypercalls
    3. Execute a hypercall (SWITCH_MMU)
       - MMU is switched from the shadow page table mode to the direct
       - Modified to use 4KB pages for the kernel translation
  - If not, stay in the native mode
  - Use the paravirt for MMU

Current Status and Next Steps

- Initial prototype works well, and performance looks good in limited benchmarks:
  - Some benchmarks from Lmbench are close to native (no way for SW-based x86-64 XenLinux)
  - Kernel build performance is about 98.5% of domU
  - Re-measuring performance using the latest processors
- Next Steps
  - Complete prototype and performance analysis

intel®

Leap ahead™