Hollis Blanchard <hollisb@us.ibm.com>
15 Apr 2008

Various notes on the implementation of KVM for PowerPC 440:

To enforce isolation, host userspace, guest kernel, and guest userspace all run at user privilege level. Only the host kernel runs in supervisor mode. Executing privileged instructions in the guest traps into KVM (in the host kernel), where we decode and emulate them. Through this technique, unmodified 440 Linux kernels can be run (slowly) as guests. Future performance work will focus on reducing the overhead and frequency of these traps.

The usual code flow is started from userspace invoking an "run" ioctl, which causes KVM to switch into guest context. We use IVPR to hijack the host interrupt vectors while running the guest, which allows us to direct all interrupts to kvmppc_handle_interrupt(). At this point, we could either
- handle the interrupt completely (e.g. emulate "mtspr SPRG0"), or
- let the host interrupt handler run (e.g. when the decrementer fires), or
- return to host userspace (e.g. when the guest performs device MMIO)

Address spaces: We take advantage of the fact that Linux doesn't use the AS=1 address space (in host or guest), which gives us virtual address space to use for guest mappings. While the guest is running, the host kernel remains mapped in AS=0, but the guest can only use AS=1 mappings.

TLB entries: The TLB entries covering the host linear mapping remain present while running the guest. This reduces the overhead of lightweight exits, which are handled by KVM running in the host kernel. We keep three copies of the TLB:
 - guest TLB: contents of the TLB as the guest sees it
 - shadow TLB: the TLB that is actually in hardware while guest is running
 - host TLB: to restore TLB state when context switching guest -> host
When a TLB miss occurs because a mapping was not present in the shadow TLB, but was present in the guest TLB, KVM handles the fault without invoking the guest. Large guest pages are backed by multiple 4KB shadow pages through this mechanism.

IO: MMIO and DCR accesses are emulated by userspace. We use virtio for network and block IO, so those drivers must be enabled in the guest. It's possible that some qemu device emulation (e.g. e1000 or rtl8139) may also work with little effort.