

[PATCH 13/40] KVM: VMX: Enable Virtual Processor Identification (VPID)

Source: <http://linux.derkeiler.com/Mailing-Lists/Kernel/2008-03/msg12006.html>

- *From:* Avi Kivity <avi@xxxxxxxxxxxxx>
 - *Date:* Mon, 31 Mar 2008 17:36:57 +0300
-

From: Sheng Yang <sheng.yang@xxxxxxxxx>

To allow TLB entries to be retained across VM entry and VM exit, the VMM can now identify distinct address spaces through a new virtual-processor ID (VPID) field of the VMCS.

[avi: drop vpid_sync_all()]
[avi: add "cc" to asm constraints]

Signed-off-by: Sheng Yang <sheng.yang@xxxxxxxxx>
Signed-off-by: Avi Kivity <avi@xxxxxxxxxxxxx>

arch/x86/kvm/vmx.c | 80 ++++++-----
arch/x86/kvm/vmx.h | 6 +++
include/asm-x86/kvm_host.h | 1 +
3 files changed, 82 insertions(+), 5 deletions(-)

diff --git a/arch/x86/kvm/vmx.c b/arch/x86/kvm/vmx.c
index 8e14628..1157e8a 100644

--- a/arch/x86/kvm/vmx.c
+++ b/arch/x86/kvm/vmx.c
@@ -37,6 +37,9 @@ MODULE_LICENSE("GPL");
static int bypass_guest_pf = 1;
module_param(bypass_guest_pf, bool, 0);

+static int enable_vpid = 1;
+module_param(enable_vpid, bool, 0);
+
struct vmcs {
u32 revision_id;
u32 abort;
@@ -71,6 +74,7 @@ struct vcpu_vmx {
unsigned rip;
} irq;
} rmode;
+ int vpid;
};

[PATCH 13/40] KVM: VMX: Enable Virtual Processor Identification (VPID)

```
static inline struct vcpu_vmx *to_vmx(struct kvm_vcpu *vcpu)
@@ -86,6 +90,9 @@ static DEFINE_PER_CPU(struct vmcs *, current_vmcs);
static struct page *vmx_io_bitmap_a;
static struct page *vmx_io_bitmap_b;

+static DECLARE_BITMAP(vmx_vpid_bitmap, VMX_NR_VPIDS);
+static DEFINE_SPINLOCK(vmx_vpid_lock);
+
static struct vmcs_config {
int size;
int order;
@@ -204,6 +211,12 @@ static inline int vm_need_virtualize_apic_accesses(struct kvm *kvm)
(irqchip_in_kernel(kvm));
}

+static inline int cpu_has_vmx_vpid(void)
+{
+ return (vmcs_config.cpu_based_2nd_exec_ctrl &
+ SECONDARY_EXEC_ENABLE_VPID);
+}
+
static int __find_msr_index(struct vcpu_vmx *vmx, u32 msr)
{
int i;
@@ -214,6 +227,20 @@ static int __find_msr_index(struct vcpu_vmx *vmx, u32 msr)
return -1;
}

+static inline void __invvpid(int ext, u16 vpid, gva_t gva)
+{
+ struct {
+ u64 vpid : 16;
+ u64 rsvd : 48;
+ u64 gva;
+ } operand = { vpid, 0, gva };
+
+ asm volatile (ASM_VMX_INVVPID
+ /* CF==1 or ZF==1 --> rc = -1 */
+ " ; ja 1f ; ud2 ; 1:"
+ : : "a"(&operand), "c"(ext) : "cc", "memory");
+}
+
static struct kvm_msr_entry *find_msr_entry(struct vcpu_vmx *vmx, u32 msr)
{
int i;
@@ -257,6 +284,14 @@ static void vcpu_clear(struct vcpu_vmx *vmx)
vmx->launched = 0;
}

+static inline void vpid_sync_vcpu_all(struct vcpu_vmx *vmx)
+{
```

[PATCH 13/40] KVM: VMX: Enable Virtual Processor Identification (VPID)

```
+ if (vmx->vpid == 0)
+ return;
+
+ __invvpid(VMX_VPID_EXTENT_SINGLE_CONTEXT, vmx->vpid, 0);
+}
+
static unsigned long vmcs_readl(unsigned long field)
{
    unsigned long value;
    @@ -490,6 +525,7 @@ static void vmx_vcpu_load(struct kvm_vcpu *vcpu, int cpu)
    if (vcpu->cpu != cpu) {
        vcpu_clear(vmx);
        kvm_migrate_apic_timer(vcpu);
        + vpid_sync_vcpu_all(vmx);
    }

    if (per_cpu(current_vmcs, cpu) != vmx->vmcs) {
        @@ -971,7 +1007,8 @@ static __init int setup_vmcs_config(struct vmcs_config *vmcs_conf)
        if (_cpu_based_exec_control & CPU_BASED_ACTIVATE_SECONDARY_CONTROLS) {
            min = 0;
            opt = SECONDARY_EXEC_VIRTUALIZE_APIC_ACCESSES |
                - SECONDARY_EXEC_WBINVD_EXITING;
            + SECONDARY_EXEC_WBINVD_EXITING |
            + SECONDARY_EXEC_ENABLE_VPID;
            if (adjust_vmcs_controls(min, opt, MSR_IA32_VMX_PROCBASED_CTL2,
                &_cpu_based_2nd_exec_control) < 0)
                return -EIO;
            @@ -1239,6 +1276,11 @@ static void exit_lmode(struct kvm_vcpu *vcpu)

#endif

+static void vmx_flush_tlb(struct kvm_vcpu *vcpu)
+{
+    + vpid_sync_vcpu_all(to_vmx(vcpu));
+}
+
static void vmx_decache_cr4_guest_bits(struct kvm_vcpu *vcpu)
{
    vcpu->arch.cr4 &= KVM_GUEST_CR4_MASK;
    @@ -1275,6 +1317,7 @@ static void vmx_set_cr0(struct kvm_vcpu *vcpu, unsigned long cr0)

static void vmx_set_cr3(struct kvm_vcpu *vcpu, unsigned long cr3)
{
    + vmx_flush_tlb(vcpu);
    vmcs_writel(GUEST_CR3, cr3);
    if (vcpu->arch.cr0 & X86_CR0_PE)
        vmx_fpu_deactivate(vcpu);
    @@ -1494,6 +1537,22 @@ out:
    return r;
}
}
```

[PATCH 13/40] KVM: VMX: Enable Virtual Processor Identification (VPID)

```
+static void allocate_vpid(struct vcpu_vmx *vmx)
+{
+ int vpid;
+
+ vmx->vpid = 0;
+ if (!enable_vpid || !cpu_has_vmx_vpid())
+ return;
+ spin_lock(&vmx_vpid_lock);
+ vpid = find_first_zero_bit(vmx_vpid_bitmap, VMX_NR_VPIDS);
+ if (vpid < VMX_NR_VPIDS) {
+ vmx->vpid = vpid;
+ __set_bit(vpid, vmx_vpid_bitmap);
+ }
+ spin_unlock(&vmx_vpid_lock);
+}
+
+/*
+ * Sets up the vmcs for emulated real mode.
+ */
@@ -1532,6 +1591,8 @@ static int vmx_vcpu_setup(struct vcpu_vmx *vmx)
if (!vm_need_virtualize_apic_accesses(vmx->vcpu.kvm))
exec_control &=
~SECONDARY_EXEC_VIRTUALIZE_APIC_ACCESSES;
+ if (vmx->vpid == 0)
+ exec_control &= ~SECONDARY_EXEC_ENABLE_VPID;
vmcs_write32(SECONDARY_VM_EXEC_CONTROL, exec_control);
}

@@ -1704,6 +1765,9 @@ static int vmx_vcpu_reset(struct kvm_vcpu *vcpu)
vmcs_write64(APIC_ACCESS_ADDR,
page_to_phys(vmx->vcpu.kvm->arch.apic_access_page));

+ if (vmx->vpid != 0)
+ vmcs_write16(VIRTUAL_PROCESSOR_ID, vmx->vpid);
+
vmx->vcpu.arch.cr0 = 0x60000010;
vmx_set_cr0(&vmx->vcpu, vmx->vcpu.arch.cr0); /* enter rmode */
vmx_set_cr4(&vmx->vcpu, 0);
@@ -1713,6 +1777,8 @@ static int vmx_vcpu_reset(struct kvm_vcpu *vcpu)
vmx_fpu_activate(&vmx->vcpu);
update_exception_bitmap(&vmx->vcpu);

+ vpid_sync_vcpu_all(vmx);
+
return 0;

out:
@@ -2221,10 +2287,6 @@ static int kvm_handle_exit(struct kvm_run *kvm_run, struct kvm_vcpu *vcpu)
return 0;
}
}
```

[PATCH 13/40] KVM: VMX: Enable Virtual Processor Identification (VPID)

```

-static void vmx_flush_tlb(struct kvm_vcpu *vcpu)
- {
- }
-
static void update_tpr_threshold(struct kvm_vcpu *vcpu)
{
int max_irr, tpr;
@@ -2489,6 +2551,10 @@ static void vmx_free_vcpu(struct kvm_vcpu *vcpu)
{
struct vcpu_vmx *vmx = to_vmx(vcpu);

+ spin_lock(&vmx_vpid_lock);
+ if (vmx->vpid != 0)
+ __clear_bit(vmx->vpid, vmx_vpid_bitmap);
+ spin_unlock(&vmx_vpid_lock);
vmx_free_vmcs(vcpu);
kfree(vmx->host_msrs);
kfree(vmx->guest_msrs);
@@ -2505,6 +2571,8 @@ static struct kvm_vcpu *vmx_create_vcpu(struct kvm *kvm, unsigned int id)
if (!vmx)
return ERR_PTR(-ENOMEM);

+ allocate_vpid(vmx);
+
err = kvm_vcpu_init(&vmx->vcpu, kvm, id);
if (err)
goto free_vcpu;
@@ -2652,6 +2720,8 @@ static int __init vmx_init(void)
memset(iova, 0xff, PAGE_SIZE);
kunmap(vmx_io_bitmap_b);

+ set_bit(0, vmx_vpid_bitmap); /* 0 is reserved for host */
+
r = kvm_init(&vmx_x86_ops, sizeof(struct vcpu_vmx), THIS_MODULE);
if (r)
goto out1;
diff --git a/arch/x86/kvm/vmx.h b/arch/x86/kvm/vmx.h
index d52ae8d..436ce0f 100644
--- a/arch/x86/kvm/vmx.h
+++ b/arch/x86/kvm/vmx.h
@@ -49,6 +49,7 @@
* Definitions of Secondary Processor-Based VM-Execution Controls.
*/
#define SECONDARY_EXEC_VIRTUALIZE_APIC_ACCESSES 0x00000001
+#define SECONDARY_EXEC_ENABLE_VPID 0x00000020
#define SECONDARY_EXEC_WBINVD_EXITING 0x00000040

@@ -65,6 +66,7 @@

/* VMCS Encodings */

```

[PATCH 13/40] KVM: VMX: Enable Virtual Processor Identification (VPID)

```
enum vmcs_field {
+ VIRTUAL_PROCESSOR_ID = 0x00000000,
GUEST_ES_SELECTOR = 0x00000800,
GUEST_CS_SELECTOR = 0x00000802,
GUEST_SS_SELECTOR = 0x00000804,
@@ -321,4 +323,8 @@ enum vmcs_field {

#define APIC_ACCESS_PAGE_PRIVATE_MEMSLOT 9

+#define VMX_NR_VPIDS (1 << 16)
+#define VMX_VPID_EXTENT_SINGLE_CONTEXT 1
+#define VMX_VPID_EXTENT_ALL_CONTEXT 2
+
#endif
diff --git a/include/asm-x86/kvm_host.h b/include/asm-x86/kvm_host.h
index d6db0de..67ae307 100644
--- a/include/asm-x86/kvm_host.h
+++ b/include/asm-x86/kvm_host.h
@@ -600,6 +600,7 @@ static inline void kvm_inject_gp(struct kvm_vcpu *vcpu, u32 error_code)
#define ASM_VMX_VMWRITE_RSP_RDX ".byte 0x0f, 0x79, 0xd4"
#define ASM_VMX_VMXOFF ".byte 0x0f, 0x01, 0xc4"
#define ASM_VMX_VMXON_RAX ".byte 0xf3, 0x0f, 0xc7, 0x30"
+#define ASM_VMX_INVVPID ".byte 0x66, 0x0f, 0x38, 0x81, 0x08"

#define MSR_IA32_TIME_STAMP_COUNTER 0x010

--
1.5.4.5

--
To unsubscribe from this list: send the line "unsubscribe linux-kernel" in
the body of a message to majordomo@xxxxxxxxxxxxxxxxxxx
More majordomo info at http://vger.kernel.org/majordomo-info.html
Please read the FAQ at http://www.tux.org/lkml/
```