[v13,11/12] KVM: x86: Set intercept for Intel PT MSRs read/write

10654373	diff (/patch/10654373/raw/)	mbox (/patch/10654373/mbox/)	series (/series/34463/mbox/)

Message ID	1540368316-12998-12-git-send-email-luwei.kang@intel.com
State	New
Headers	show
Series	Intel Processor Trace virtualization enabling
Related	show

Commit Message

Kang, Luwei (/project/kvm/list/?submitter=168537) Oct. 24, 2018, 8:05 a.m. UTC

```
From: Chao Peng <chao.p.peng@linux.intel.com>

To save performance overhead, disable intercept Intel PT MSRs
read/write when Intel PT is enabled in guest.
MSR_IA32_RTIT_CTL is an exception that will always be intercepted.

Signed-off-by: Chao Peng <chao.p.peng@linux.intel.com>
Signed-off-by: Luwei Kang <luwei.kang@intel.com>
---
 arch/x86/kvm/vmx.c | 23 ++++++++++++++++++++++
 1 file changed, 23 insertions(+)
```

Patch

10654373	diff (/patch/10654373/raw/)	mbox (/patch/10654373/mbox/)	series (/series/34463/mbox/)

```
diff --git a/arch/x86/kvm/vmx.c b/arch/x86/kvm/vmx.c
index a568d49..ed247dd 100644
--- a/arch/x86/kvm/vmx.c
+++ b/arch/x86/kvm/vmx.c
@@ -1333,6 +1333,7 @@  static bool nested_vmx_is_page_fault_vmexit(struct vmcs12 *vmcs12,
 static void vmx_update_msr_bitmap(struct kvm_vcpu *vcpu);
 static void __always_inline vmx_disable_intercept_for_msr(unsigned long *msr_bitmap,
                                                            u32 msr, int type);
+static void pt_set_intercept_for_msr(struct vcpu_vmx *vmx, bool flag);

 static DEFINE_PER_CPU(struct vmcs *, vmxarea);
 static DEFINE_PER_CPU(struct vmcs *, current_vmcs);
@@ -4558,6 +4559,7 @@  static int vmx_set_msr(struct kvm_vcpu *vcpu, struct msr_data *msr_info)
                        vmx_rtit_ctl_check(vcpu, data))
                        return 1;
                vmcs_write64(GUEST_IA32_RTIT_CTL, data);
+               pt_set_intercept_for_msr(vmx, !(data & RTIT_CTL_TRACEEN));
                vmx->pt_desc.guest.ctl = data;
                break;
        case MSR_IA32_RTIT_STATUS:
@@ -6414,6 +6416,27 @@  static void vmx_update_msr_bitmap(struct kvm_vcpu *vcpu)
        vmx->msr_bitmap_mode = mode;
 }

+static void pt_set_intercept_for_msr(struct vcpu_vmx *vmx, bool flag)
+{
+       unsigned long *msr_bitmap = vmx->vmcs01.msr_bitmap;
+       u32 i;
+
+       vmx_set_intercept_for_msr(msr_bitmap, MSR_IA32_RTIT_STATUS,
+                                                       MSR_TYPE_RW, flag);
+       vmx_set_intercept_for_msr(msr_bitmap, MSR_IA32_RTIT_OUTPUT_BASE,
+                                                       MSR_TYPE_RW, flag);
+       vmx_set_intercept_for_msr(msr_bitmap, MSR_IA32_RTIT_OUTPUT_MASK,
+                                                       MSR_TYPE_RW, flag);
+       vmx_set_intercept_for_msr(msr_bitmap, MSR_IA32_RTIT_CR3_MATCH,
+                                                       MSR_TYPE_RW, flag);
+       for (i = 0; i < vmx->pt_desc.addr_range; i++) {
+               vmx_set_intercept_for_msr(msr_bitmap,
+                       MSR_IA32_RTIT_ADDR0_A + i * 2, MSR_TYPE_RW, flag);
+               vmx_set_intercept_for_msr(msr_bitmap,
+                       MSR_IA32_RTIT_ADDR0_B + i * 2, MSR_TYPE_RW, flag);
+       }
+}
+
 static bool vmx_get_enable_apicv(struct kvm_vcpu *vcpu)
 {
        return enable_apicv;
```