[v13,10/12] KVM: x86: Implement Intel PT MSRs read/write emulation

10654371	diff (/patch/10654371/raw/)	mbox (/patch/10654371/mbox/)	series (/series/34463/mbox/)

Message ID	1540368316-12998-11-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>

This patch implement Intel Processor Trace MSRs read/write
emulation.
Intel PT MSRs read/write need to be emulated when Intel PT
MSRs is intercepted in guest and during live migration.

Signed-off-by: Chao Peng <chao.p.peng@linux.intel.com>
Signed-off-by: Luwei Kang <luwei.kang@intel.com>
---
 arch/x86/include/asm/intel_pt.h |   8 ++
 arch/x86/kvm/vmx.c              | 176 ++++++++++++++++++++++++++++++++++++++++
 arch/x86/kvm/x86.c              |  33 +++++++-
 3 files changed, 216 insertions(+), 1 deletion(-)
```

Patch

10654371	diff (/patch/10654371/raw/)	mbox (/patch/10654371/mbox/)	series (/series/34463/mbox/)

```
diff --git a/arch/x86/include/asm/intel_pt.h b/arch/x86/include/asm/intel_pt.h
index eabbdbc..a1c2080 100644
--- a/arch/x86/include/asm/intel_pt.h
+++ b/arch/x86/include/asm/intel_pt.h
@@ -10,6 +10,14 @@

 #define RTIT_ADDR_RANGE                4

+#define MSR_IA32_RTIT_STATUS_MASK (~(RTIT_STATUS_FILTEREN | \
+               RTIT_STATUS_CONTEXTEN | RTIT_STATUS_TRIGGEREN | \
+               RTIT_STATUS_ERROR | RTIT_STATUS_STOPPED | \
+               RTIT_STATUS_BYTECNT))
+
+#define MSR_IA32_RTIT_OUTPUT_BASE_MASK \
+               (~((1UL << cpuid_query_maxphyaddr(vcpu)) - 1) | 0x7f)
+
 enum pt_capabilities {
        PT_CAP_max_subleaf = 0,
        PT_CAP_cr3_filtering,
diff --git a/arch/x86/kvm/vmx.c b/arch/x86/kvm/vmx.c
index 2697618..a568d49 100644
--- a/arch/x86/kvm/vmx.c
+++ b/arch/x86/kvm/vmx.c
@@ -3350,6 +3350,79 @@  static void vmx_set_interrupt_shadow(struct kvm_vcpu *vcpu, int mask)
                vmcs_write32(GUEST_INTERRUPTIBILITY_INFO, interruptibility);
 }

+static int vmx_rtit_ctl_check(struct kvm_vcpu *vcpu, u64 data)
+{
+       struct vcpu_vmx *vmx = to_vmx(vcpu);
+       unsigned long value;
+
+       /*
+        * Any MSR write that attempts to change bits marked reserved will
+        * case a #GP fault.
+        */
+       if (data & vmx->pt_desc.ctl_bitmask)
+               return 1;
+
+       /*
+        * Any attempt to modify IA32_RTIT_CTL while TraceEn is set will
+        * result in a #GP unless the same write also clears TraceEn.
+        */
+       if ((vmx->pt_desc.guest.ctl & RTIT_CTL_TRACEEN) &&
+               ((vmx->pt_desc.guest.ctl ^ data) & ~RTIT_CTL_TRACEEN))
+               return 1;
+
+       /*
+        * WRMSR to IA32_RTIT_CTL that sets TraceEn but clears this bit
+        * and FabricEn would cause #GP, if
```

```
+		 * CPUID.(EAX=14H, ECX=0):ECX.SNGLRGNOUT[bit 2] = 0
+		 */
+		if ((data & RTIT_CTL_TRACEEN) && !(data & RTIT_CTL_TOPA) &&
+			!(data & RTIT_CTL_FABRIC_EN) &&
+			!intel_pt_validate_cap(vmx->pt_desc.caps,
+					       PT_CAP_single_range_output))
+			return 1;
+
+		/*
+		 * MTCFreq, CycThresh and PSBFreq encodings check, any MSR write that
+		 * utilize encodings marked reserved will casue a #GP fault.
+		 */
+		value = intel_pt_validate_cap(vmx->pt_desc.caps, PT_CAP_mtc_periods);
+		if (intel_pt_validate_cap(vmx->pt_desc.caps, PT_CAP_mtc) &&
+				!test_bit((data & RTIT_CTL_MTC_RANGE) >>
+				RTIT_CTL_MTC_RANGE_OFFSET, &value))
+			return 1;
+		value = intel_pt_validate_cap(vmx->pt_desc.caps,
+					      PT_CAP_cycle_thresholds);
+		if (intel_pt_validate_cap(vmx->pt_desc.caps, PT_CAP_psb_cyc) &&
+				!test_bit((data & RTIT_CTL_CYC_THRESH) >>
+				RTIT_CTL_CYC_THRESH_OFFSET, &value))
+			return 1;
+		value = intel_pt_validate_cap(vmx->pt_desc.caps, PT_CAP_psb_periods);
+		if (intel_pt_validate_cap(vmx->pt_desc.caps, PT_CAP_psb_cyc) &&
+				!test_bit((data & RTIT_CTL_PSB_FREQ) >>
+				RTIT_CTL_PSB_FREQ_OFFSET, &value))
+			return 1;
+
+		/*
+		 * If ADDRx_CFG is reserved or the encodings is >2 will
+		 * cause a #GP fault.
+		 */
+		value = (data & RTIT_CTL_ADDR0) >> RTIT_CTL_ADDR0_OFFSET;
+		if ((value && (vmx->pt_desc.addr_range < 1)) || (value > 2))
+			return 1;
+		value = (data & RTIT_CTL_ADDR1) >> RTIT_CTL_ADDR1_OFFSET;
+		if ((value && (vmx->pt_desc.addr_range < 2)) || (value > 2))
+			return 1;
+		value = (data & RTIT_CTL_ADDR2) >> RTIT_CTL_ADDR2_OFFSET;
+		if ((value && (vmx->pt_desc.addr_range < 3)) || (value > 2))
+			return 1;
+		value = (data & RTIT_CTL_ADDR3) >> RTIT_CTL_ADDR3_OFFSET;
+		if ((value && (vmx->pt_desc.addr_range < 4)) || (value > 2))
+			return 1;
+
+		return 0;
+}
+
+
 static void skip_emulated_instruction(struct kvm_vcpu *vcpu)
```

```
 {
        unsigned long rip;
@@ -4186,6 +4259,7 @@  static int vmx_get_msr(struct kvm_vcpu *vcpu, struct msr_data *msr_info)
 {
        struct vcpu_vmx *vmx = to_vmx(vcpu);
        struct shared_msr_entry *msr;
+       u32 index;

        switch (msr_info->index) {
 #ifdef CONFIG_X86_64
@@ -4250,6 +4324,52 @@  static int vmx_get_msr(struct kvm_vcpu *vcpu, struct msr_data *msr_info)
                        return 1;
                msr_info->data = vcpu->arch.ia32_xss;
                break;
+       case MSR_IA32_RTIT_CTL:
+               if (pt_mode != PT_MODE_HOST_GUEST)
+                       return 1;
+               msr_info->data = vmx->pt_desc.guest.ctl;
+               break;
+       case MSR_IA32_RTIT_STATUS:
+               if (pt_mode != PT_MODE_HOST_GUEST)
+                       return 1;
+               msr_info->data = vmx->pt_desc.guest.status;
+               break;
+       case MSR_IA32_RTIT_CR3_MATCH:
+               if ((pt_mode != PT_MODE_HOST_GUEST) ||
+                       !intel_pt_validate_cap(vmx->pt_desc.caps,
+                                              PT_CAP_cr3_filtering))
+                       return 1;
+               msr_info->data = vmx->pt_desc.guest.cr3_match;
+               break;
+       case MSR_IA32_RTIT_OUTPUT_BASE:
+               if ((pt_mode != PT_MODE_HOST_GUEST) ||
+                       (!intel_pt_validate_cap(vmx->pt_desc.caps,
+                                       PT_CAP_topa_output) &&
+                        !intel_pt_validate_cap(vmx->pt_desc.caps,
+                                       PT_CAP_single_range_output)))
+                       return 1;
+               msr_info->data = vmx->pt_desc.guest.output_base;
+               break;
+       case MSR_IA32_RTIT_OUTPUT_MASK:
+               if ((pt_mode != PT_MODE_HOST_GUEST) ||
+                       (!intel_pt_validate_cap(vmx->pt_desc.caps,
+                                       PT_CAP_topa_output) &&
+                        !intel_pt_validate_cap(vmx->pt_desc.caps,
+                                       PT_CAP_single_range_output)))
+                       return 1;
+               msr_info->data = vmx->pt_desc.guest.output_mask;
+               break;
+       case MSR_IA32_RTIT_ADDR0_A ... MSR_IA32_RTIT_ADDR3_B:
+               index = msr_info->index - MSR_IA32_RTIT_ADDR0_A;
```

```
+               if ((pt_mode != PT_MODE_HOST_GUEST) ||
+                       (index >= 2 * intel_pt_validate_cap(vmx->pt_desc.caps,
+                                       PT_CAP_num_address_ranges)))
+                       return 1;
+               if (index % 2)
+                       msr_info->data = vmx->pt_desc.guest.addr_b[index / 2];
+               else
+                       msr_info->data = vmx->pt_desc.guest.addr_a[index / 2];
+               break;
        case MSR_TSC_AUX:
                if (!msr_info->host_initiated &&
                    !guest_cpuid_has(vcpu, X86_FEATURE_RDTSCP))
@@ -4281,6 +4401,7 @@  static int vmx_set_msr(struct kvm_vcpu *vcpu, struct msr_data *msr_info)
        int ret = 0;
        u32 msr_index = msr_info->index;
        u64 data = msr_info->data;
+       u32 index;

        switch (msr_index) {
        case MSR_EFER:
@@ -4432,6 +4553,61 @@  static int vmx_set_msr(struct kvm_vcpu *vcpu, struct msr_data *msr_info)
                else
                        clear_atomic_switch_msr(vmx, MSR_IA32_XSS);
                break;
+       case MSR_IA32_RTIT_CTL:
+               if ((pt_mode != PT_MODE_HOST_GUEST) ||
+                       vmx_rtit_ctl_check(vcpu, data))
+                       return 1;
+               vmcs_write64(GUEST_IA32_RTIT_CTL, data);
+               vmx->pt_desc.guest.ctl = data;
+               break;
+       case MSR_IA32_RTIT_STATUS:
+               if ((pt_mode != PT_MODE_HOST_GUEST) ||
+                       (vmx->pt_desc.guest.ctl & RTIT_CTL_TRACEEN) ||
+                       (data & MSR_IA32_RTIT_STATUS_MASK))
+                       return 1;
+               vmx->pt_desc.guest.status = data;
+               break;
+       case MSR_IA32_RTIT_CR3_MATCH:
+               if ((pt_mode != PT_MODE_HOST_GUEST) ||
+                       (vmx->pt_desc.guest.ctl & RTIT_CTL_TRACEEN) ||
+                       !intel_pt_validate_cap(vmx->pt_desc.caps,
+                                       PT_CAP_cr3_filtering))
+                       return 1;
+               vmx->pt_desc.guest.cr3_match = data;
+               break;
+       case MSR_IA32_RTIT_OUTPUT_BASE:
+               if ((pt_mode != PT_MODE_HOST_GUEST) ||
+                       (vmx->pt_desc.guest.ctl & RTIT_CTL_TRACEEN) ||
+                       (!intel_pt_validate_cap(vmx->pt_desc.caps,
+                                       PT_CAP_topa_output) &&
```

```
+                        !intel_pt_validate_cap(vmx->pt_desc.caps,
+                                               PT_CAP_single_range_output)) ||
+                       (data & MSR_IA32_RTIT_OUTPUT_BASE_MASK))
+                       return 1;
+               vmx->pt_desc.guest.output_base = data;
+               break;
+       case MSR_IA32_RTIT_OUTPUT_MASK:
+               if ((pt_mode != PT_MODE_HOST_GUEST) ||
+                       (vmx->pt_desc.guest.ctl & RTIT_CTL_TRACEEN) ||
+                       (!intel_pt_validate_cap(vmx->pt_desc.caps,
+                                               PT_CAP_topa_output) &&
+                        !intel_pt_validate_cap(vmx->pt_desc.caps,
+                                               PT_CAP_single_range_output)))
+                       return 1;
+               vmx->pt_desc.guest.output_mask = data;
+               break;
+       case MSR_IA32_RTIT_ADDR0_A ... MSR_IA32_RTIT_ADDR3_B:
+               index = msr_info->index - MSR_IA32_RTIT_ADDR0_A;
+               if ((pt_mode != PT_MODE_HOST_GUEST) ||
+                       (vmx->pt_desc.guest.ctl & RTIT_CTL_TRACEEN) ||
+                       (index >= 2 * intel_pt_validate_cap(vmx->pt_desc.caps,
+                                               PT_CAP_num_address_ranges)))
+                       return 1;
+               if (index % 2)
+                       vmx->pt_desc.guest.addr_b[index / 2] = data;
+               else
+                       vmx->pt_desc.guest.addr_a[index / 2] = data;
+               break;
        case MSR_TSC_AUX:
                if (!msr_info->host_initiated &&
                    !guest_cpuid_has(vcpu, X86_FEATURE_RDTSCP))
diff --git a/arch/x86/kvm/x86.c b/arch/x86/kvm/x86.c
index 66d66d7..603c92a 100644
--- a/arch/x86/kvm/x86.c
+++ b/arch/x86/kvm/x86.c
@@ -69,6 +69,7 @@
 #include <asm/irq_remapping.h>
 #include <asm/mshyperv.h>
 #include <asm/hypervisor.h>
+#include <asm/intel_pt.h>

 #define CREATE_TRACE_POINTS
 #include "trace.h"
@@ -1121,7 +1122,13 @@  bool kvm_rdpmc(struct kvm_vcpu *vcpu)
 #endif
        MSR_IA32_TSC, MSR_IA32_CR_PAT, MSR_VM_HSAVE_PA,
        MSR_IA32_FEATURE_CONTROL, MSR_IA32_BNDCFGS, MSR_TSC_AUX,
-       MSR_IA32_SPEC_CTRL, MSR_IA32_ARCH_CAPABILITIES
+       MSR_IA32_SPEC_CTRL, MSR_IA32_ARCH_CAPABILITIES,
+       MSR_IA32_RTIT_CTL, MSR_IA32_RTIT_STATUS, MSR_IA32_RTIT_CR3_MATCH,
+       MSR_IA32_RTIT_OUTPUT_BASE, MSR_IA32_RTIT_OUTPUT_MASK,
```

```
+       MSR_IA32_RTIT_ADDR0_A, MSR_IA32_RTIT_ADDR0_B,
+       MSR_IA32_RTIT_ADDR1_A, MSR_IA32_RTIT_ADDR1_B,
+       MSR_IA32_RTIT_ADDR2_A, MSR_IA32_RTIT_ADDR2_B,
+       MSR_IA32_RTIT_ADDR3_A, MSR_IA32_RTIT_ADDR3_B,
 };

 static unsigned num_msrs_to_save;
@@ -4842,6 +4849,30 @@  static void kvm_init_msr_list(void)
                        if (!kvm_x86_ops->rdtscp_supported())
                                continue;
                        break;
+               case MSR_IA32_RTIT_CTL:
+               case MSR_IA32_RTIT_STATUS:
+                       if (!kvm_x86_ops->pt_supported())
+                               continue;
+                       break;
+               case MSR_IA32_RTIT_CR3_MATCH:
+                       if (!kvm_x86_ops->pt_supported() ||
+                           !intel_pt_validate_hw_cap(PT_CAP_cr3_filtering))
+                               continue;
+                       break;
+               case MSR_IA32_RTIT_OUTPUT_BASE:
+               case MSR_IA32_RTIT_OUTPUT_MASK:
+                       if (!kvm_x86_ops->pt_supported() ||
+                               (!intel_pt_validate_hw_cap(PT_CAP_topa_output) &&
+                                !intel_pt_validate_hw_cap(PT_CAP_single_range_output)))
+                               continue;
+                       break;
+               case MSR_IA32_RTIT_ADDR0_A ... MSR_IA32_RTIT_ADDR3_B: {
+                       if (!kvm_x86_ops->pt_supported() ||
+                               msrs_to_save[i] - MSR_IA32_RTIT_ADDR0_A >=
+                               intel_pt_validate_hw_cap(PT_CAP_num_address_ranges) * 2)
+                               continue;
+                       break;
+               }
                default:
                        break;
                }
```