[v13,09/12] KVM: x86: Introduce a function to initialize the PT configuration

10654377	diff (/patch/10654377/raw/)	mbox (/patch/10654377/mbox/)	series (/series/34463/mbox/)

Message ID	1540368316-12998-10-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

```
Initialize the Intel PT configuration when cpuid update.
Include cpuid inforamtion, rtit_ctl bit mask and the number of
address ranges.

Signed-off-by: Luwei Kang <luwei.kang@intel.com>
---
 arch/x86/kvm/vmx.c | 73 +++++++++++++++++++++++++++++++++++++++++++++++++++++++++
 1 file changed, 73 insertions(+)
```

Patch

10654377	diff (/patch/10654377/raw/)	mbox (/patch/10654377/mbox/)	series (/series/34463/mbox/)

```
diff --git a/arch/x86/kvm/vmx.c b/arch/x86/kvm/vmx.c
index d8480a6..2697618 100644
--- a/arch/x86/kvm/vmx.c
+++ b/arch/x86/kvm/vmx.c
@@ -11921,6 +11921,75 @@  static void nested_vmx_entry_exit_ctls_update(struct kvm_vcpu *vcpu)
        }
 }

+static void update_intel_pt_cfg(struct kvm_vcpu *vcpu)
+{
+       struct vcpu_vmx *vmx = to_vmx(vcpu);
+       struct kvm_cpuid_entry2 *best = NULL;
+       int i;
+
+       for (i = 0; i < PT_CPUID_LEAVES; i++) {
+               best = kvm_find_cpuid_entry(vcpu, 0x14, i);
+               if (!best)
+                       return;
+               vmx->pt_desc.caps[CPUID_EAX + i*PT_CPUID_REGS_NUM] = best->eax;
+               vmx->pt_desc.caps[CPUID_EBX + i*PT_CPUID_REGS_NUM] = best->ebx;
+               vmx->pt_desc.caps[CPUID_ECX + i*PT_CPUID_REGS_NUM] = best->ecx;
+               vmx->pt_desc.caps[CPUID_EDX + i*PT_CPUID_REGS_NUM] = best->edx;
+       }
+
+       /* Get the number of configurable Address Ranges for filtering */
+       vmx->pt_desc.addr_range = intel_pt_validate_cap(vmx->pt_desc.caps,
+                                               PT_CAP_num_address_ranges);
+
+       /* Initialize and clear the no dependency bits */
+       vmx->pt_desc.ctl_bitmask = ~(RTIT_CTL_TRACEEN | RTIT_CTL_OS |
+                       RTIT_CTL_USR | RTIT_CTL_TSC_EN | RTIT_CTL_DISRETC);
+
+       /*
+        * If CPUID.(EAX=14H,ECX=0):EBX[0]=1 CR3Filter can be set otherwise
+        * will inject an #GP
+        */
+       if (intel_pt_validate_cap(vmx->pt_desc.caps, PT_CAP_cr3_filtering))
+               vmx->pt_desc.ctl_bitmask &= ~RTIT_CTL_CR3EN;
+
+       /*
+        * If CPUID.(EAX=14H,ECX=0):EBX[1]=1 CYCEn, CycThresh and
+        * PSBFreq can be set
+        */
+       if (intel_pt_validate_cap(vmx->pt_desc.caps, PT_CAP_psb_cyc))
+               vmx->pt_desc.ctl_bitmask &= ~(RTIT_CTL_CYCLEACC |
+                               RTIT_CTL_CYC_THRESH | RTIT_CTL_PSB_FREQ);
+
+       /*
+        * If CPUID.(EAX=14H,ECX=0):EBX[3]=1 MTCEn BranchEn and
+        * MTCFreq can be set
+        */
+       if (intel_pt_validate_cap(vmx->pt_desc.caps, PT_CAP_mtc))
+               vmx->pt_desc.ctl_bitmask &= ~(RTIT_CTL_MTC_EN |
+                               RTIT_CTL_BRANCH_EN | RTIT_CTL_MTC_RANGE);
+
+       /* If CPUID.(EAX=14H,ECX=0):EBX[4]=1 FUPonPTW and PTWEn can be set */
+       if (intel_pt_validate_cap(vmx->pt_desc.caps, PT_CAP_ptwrite))
+               vmx->pt_desc.ctl_bitmask &= ~(RTIT_CTL_FUP_ON_PTW |
+                                                       RTIT_CTL_PTW_EN);
+
+       /* If CPUID.(EAX=14H,ECX=0):EBX[5]=1 PwrEvEn can be set */
+       if (intel_pt_validate_cap(vmx->pt_desc.caps, PT_CAP_power_event_trace))
+               vmx->pt_desc.ctl_bitmask &= ~RTIT_CTL_PWR_EVT_EN;
+
+       /* If CPUID.(EAX=14H,ECX=0):ECX[0]=1 ToPA can be set */
+       if (intel_pt_validate_cap(vmx->pt_desc.caps, PT_CAP_topa_output))
+               vmx->pt_desc.ctl_bitmask &= ~RTIT_CTL_TOPA;
+
+       /* If CPUID.(EAX=14H,ECX=0):ECX[3]=1 FabircEn can be set */
+       if (intel_pt_validate_cap(vmx->pt_desc.caps, PT_CAP_output_subsys))
+               vmx->pt_desc.ctl_bitmask &= ~RTIT_CTL_FABRIC_EN;
+
+       /* unmask address range configure area */
+       for (i = 0; i < vmx->pt_desc.addr_range; i++)
+               vmx->pt_desc.ctl_bitmask &= ~(0xf << (32 + i * 4));
+}
+
 static void vmx_cpuid_update(struct kvm_vcpu *vcpu)
 {
        struct vcpu_vmx *vmx = to_vmx(vcpu);
@@ -11941,6 +12010,10 @@  static void vmx_cpuid_update(struct kvm_vcpu *vcpu)
                nested_vmx_cr_fixed1_bits_update(vcpu);
                nested_vmx_entry_exit_ctls_update(vcpu);
        }
+
+       if (boot_cpu_has(X86_FEATURE_INTEL_PT) &&
+                       guest_cpuid_has(vcpu, X86_FEATURE_INTEL_PT))
+               update_intel_pt_cfg(vcpu);
 }
 
 static void vmx_set_supported_cpuid(u32 func, struct kvm_cpuid_entry2 *entry)
```