[v13,07/12] KVM: x86: Add Intel Processor Trace cpuid emulation

10654367	diff (/patch/10654367/raw/)	mbox (/patch/10654367/mbox/)	series (/series/34463/mbox/)

Message ID	1540368316-12998-8-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>

Expose Intel Processor Trace to guest only when
the PT works in Host-Guest mode.

Signed-off-by: Chao Peng <chao.p.peng@linux.intel.com>
Signed-off-by: Luwei Kang <luwei.kang@intel.com>
---
 arch/x86/include/asm/kvm_host.h |  1 +
 arch/x86/kvm/cpuid.c            | 22 ++++++++++++++++++++--
 arch/x86/kvm/svm.c              |  6 ++++++
 arch/x86/kvm/vmx.c              |  6 ++++++
 4 files changed, 33 insertions(+), 2 deletions(-)
```

Patch

10654367	diff (/patch/10654367/raw/)	mbox (/patch/10654367/mbox/)	series (/series/34463/mbox/)

```
diff --git a/arch/x86/include/asm/kvm_host.h b/arch/x86/include/asm/kvm_host.h
index 55e51ff..9ab7ac0 100644
--- a/arch/x86/include/asm/kvm_host.h
+++ b/arch/x86/include/asm/kvm_host.h
@@ -1105,6 +1105,7 @@  struct kvm_x86_ops {
        bool (*mpx_supported)(void);
        bool (*xsaves_supported)(void);
        bool (*umip_emulated)(void);
+       bool (*pt_supported)(void);

        int (*check_nested_events)(struct kvm_vcpu *vcpu, bool external_intr);
        void (*request_immediate_exit)(struct kvm_vcpu *vcpu);
diff --git a/arch/x86/kvm/cpuid.c b/arch/x86/kvm/cpuid.c
index 7bcfa61..05b8fb4 100644
--- a/arch/x86/kvm/cpuid.c
+++ b/arch/x86/kvm/cpuid.c
@@ -337,6 +337,7 @@  static inline int __do_cpuid_ent(struct kvm_cpuid_entry2 *entry, u32 function,
        unsigned f_mpx = kvm_mpx_supported() ? F(MPX) : 0;
        unsigned f_xsaves = kvm_x86_ops->xsaves_supported() ? F(XSAVES) : 0;
        unsigned f_umip = kvm_x86_ops->umip_emulated() ? F(UMIP) : 0;
+       unsigned f_intel_pt = kvm_x86_ops->pt_supported() ? F(INTEL_PT) : 0;

        /* cpuid 1.edx */
        const u32 kvm_cpuid_1_edx_x86_features =
@@ -395,7 +396,7 @@  static inline int __do_cpuid_ent(struct kvm_cpuid_entry2 *entry, u32 function,
                F(BMI2) | F(ERMS) | f_invpcid | F(RTM) | f_mpx | F(RDSEED) |
                F(ADX) | F(SMAP) | F(AVX512IFMA) | F(AVX512F) | F(AVX512PF) |
                F(AVX512ER) | F(AVX512CD) | F(CLFLUSHOPT) | F(CLWB) | F(AVX512DQ) |
-               F(SHA_NI) | F(AVX512BW) | F(AVX512VL);
+               F(SHA_NI) | F(AVX512BW) | F(AVX512VL) | f_intel_pt;

        /* cpuid 0xD.1.eax */
        const u32 kvm_cpuid_D_1_eax_x86_features =
@@ -426,7 +427,7 @@  static inline int __do_cpuid_ent(struct kvm_cpuid_entry2 *entry, u32 function,

        switch (function) {
        case 0:
-               entry->eax = min(entry->eax, (u32)0xd);
+               entry->eax = min(entry->eax, (u32)(f_intel_pt ? 0x14 : 0xd));
                break;
        case 1:
                entry->edx &= kvm_cpuid_1_edx_x86_features;
@@ -603,6 +604,23 @@  static inline int __do_cpuid_ent(struct kvm_cpuid_entry2 *entry, u32 function,
                }
                break;
        }
+       /* Intel PT */
+       case 0x14: {
+               int t, times = entry->eax;
+
+               if (!f_intel_pt)
+                       break;
+
+               entry->flags |= KVM_CPUID_FLAG_SIGNIFCANT_INDEX;
+               for (t = 1; t <= times; ++t) {
+                       if (*nent >= maxnent)
+                               goto out;
+                       do_cpuid_1_ent(&entry[t], function, t);
+                       entry[t].flags |= KVM_CPUID_FLAG_SIGNIFCANT_INDEX;
+                       ++*nent;
+               }
+               break;
+       }
        case KVM_CPUID_SIGNATURE: {
                static const char signature[12] = "KVMKVMKVM\0\0";
                const u32 *sigptr = (const u32 *)signature;
diff --git a/arch/x86/kvm/svm.c b/arch/x86/kvm/svm.c
index f416f5c7..6e8a61b 100644
--- a/arch/x86/kvm/svm.c
+++ b/arch/x86/kvm/svm.c
@@ -5904,6 +5904,11 @@  static bool svm_umip_emulated(void)
        return false;
 }

+static bool svm_pt_supported(void)
+{
+       return false;
+}
+
 static bool svm_has_wbinvd_exit(void)
 {
        return true;
@@ -7139,6 +7144,7 @@  static int nested_enable_evmcs(struct kvm_vcpu *vcpu,
        .mpx_supported = svm_mpx_supported,
        .xsaves_supported = svm_xsaves_supported,
        .umip_emulated = svm_umip_emulated,
+       .pt_supported = svm_pt_supported,

        .set_supported_cpuid = svm_set_supported_cpuid,

diff --git a/arch/x86/kvm/vmx.c b/arch/x86/kvm/vmx.c
index c4c4b76..692154c 100644
--- a/arch/x86/kvm/vmx.c
+++ b/arch/x86/kvm/vmx.c
@@ -11013,6 +11013,11 @@  static bool vmx_xsaves_supported(void)
                SECONDARY_EXEC_XSAVES;
 }

+static bool vmx_pt_supported(void)
+{
+       return (pt_mode == PT_MODE_HOST_GUEST);
```

```
+}
+
 static void vmx_recover_nmi_blocking(struct vcpu_vmx *vmx)
 {
 	u32 exit_intr_info;
@@ -15127,6 +15132,7 @@  static int vmx_set_nested_state(struct kvm_vcpu *vcpu,
 	.mpx_supported = vmx_mpx_supported,
 	.xsaves_supported = vmx_xsaves_supported,
 	.umip_emulated = vmx_umip_emulated,
+	.pt_supported = vmx_pt_supported,
 
 	.check_nested_events = vmx_check_nested_events,
 	.request_immediate_exit = vmx_request_immediate_exit,
```