[v13,02/12] perf/x86/intel/pt: Export pt_cap_get()

Message ID	1540368316-12998-3-git-send-email-luwei.kang@intel.com
State	New
Headers	show
Series	Intel Processor Trace virtualization enabling
Related	show

10654359	diff (/patch/10654359/raw/)	mbox (/patch/10654359/mbox/)	series (/series/34463/mbox/)

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>

pt_cap_get() is required by the upcoming PT support in KVM guests.

Export it and move the capabilites enum to a global header.

As a global functions, "pt_*" is already used for ptrace and
other things, so it makes sense to use "intel_pt_*" as a prefix.

Acked-by: Song Liu <songliubraving@fb.com>
Signed-off-by: Chao Peng <chao.p.peng@linux.intel.com>
Signed-off-by: Luwei Kang <luwei.kang@intel.com>
---
 arch/x86/events/intel/pt.c      | 49 ++++++++++++++++++++++++++-------------
 arch/x86/events/intel/pt.h      | 21 ------------------
 arch/x86/include/asm/intel_pt.h | 23 +++++++++++++++++++
 3 files changed, 49 insertions(+), 44 deletions(-)
```

Patch

10654359	diff (/patch/10654359/raw/)	mbox (/patch/10654359/mbox/)	series (/series/34463/mbox/)

```
diff --git a/arch/x86/events/intel/pt.c b/arch/x86/events/intel/pt.c
index 8d016ce..309bb1d 100644
--- a/arch/x86/events/intel/pt.c
+++ b/arch/x86/events/intel/pt.c
@@ -75,7 +75,7 @@
        PT_CAP(psb_periods,             1, CPUID_EBX, 0xffff0000),
 };

-static u32 pt_cap_get(enum pt_capabilities cap)
+u32 intel_pt_validate_hw_cap(enum pt_capabilities cap)
 {
        struct pt_cap_desc *cd = &pt_caps[cap];
        u32 c = pt_pmu.caps[cd->leaf * PT_CPUID_REGS_NUM + cd->reg];
@@ -83,6 +83,7 @@  static u32 pt_cap_get(enum pt_capabilities cap)

        return (c & cd->mask) >> shift;
 }
+EXPORT_SYMBOL_GPL(intel_pt_validate_hw_cap);

 static ssize_t pt_cap_show(struct device *cdev,
                           struct device_attribute *attr,
@@ -92,7 +93,7 @@  static ssize_t pt_cap_show(struct device *cdev,
                container_of(attr, struct dev_ext_attribute, attr);
        enum pt_capabilities cap = (long)ea->var;

-       return snprintf(buf, PAGE_SIZE, "%x\n", pt_cap_get(cap));
+       return snprintf(buf, PAGE_SIZE, "%x\n", intel_pt_validate_hw_cap(cap));
 }

 static struct attribute_group pt_cap_group = {
@@ -310,16 +311,16 @@  static bool pt_event_valid(struct perf_event *event)
                return false;

        if (config & RTIT_CTL_CYC_PSB) {
-               if (!pt_cap_get(PT_CAP_psb_cyc))
+               if (!intel_pt_validate_hw_cap(PT_CAP_psb_cyc))
                        return false;

-               allowed = pt_cap_get(PT_CAP_psb_periods);
+               allowed = intel_pt_validate_hw_cap(PT_CAP_psb_periods);
                requested = (config & RTIT_CTL_PSB_FREQ) >>
                        RTIT_CTL_PSB_FREQ_OFFSET;
                if (requested && (!(allowed & BIT(requested))))
                        return false;

-               allowed = pt_cap_get(PT_CAP_cycle_thresholds);
+               allowed = intel_pt_validate_hw_cap(PT_CAP_cycle_thresholds);
                requested = (config & RTIT_CTL_CYC_THRESH) >>
                        RTIT_CTL_CYC_THRESH_OFFSET;
                if (requested && (!(allowed & BIT(requested))))
```

```
@@ -334,10 +335,10 @@  static bool pt_event_valid(struct perf_event *event)
                 * Spec says that setting mtc period bits while mtc bit in
                 * CPUID is 0 will #GP, so better safe than sorry.
                 */
-               if (!pt_cap_get(PT_CAP_mtc))
+               if (!intel_pt_validate_hw_cap(PT_CAP_mtc))
                        return false;

-               allowed = pt_cap_get(PT_CAP_mtc_periods);
+               allowed = intel_pt_validate_hw_cap(PT_CAP_mtc_periods);
                if (!allowed)
                        return false;

@@ -349,11 +350,11 @@  static bool pt_event_valid(struct perf_event *event)
        }

        if (config & RTIT_CTL_PWR_EVT_EN &&
-           !pt_cap_get(PT_CAP_power_event_trace))
+           !intel_pt_validate_hw_cap(PT_CAP_power_event_trace))
                return false;

        if (config & RTIT_CTL_PTW) {
-               if (!pt_cap_get(PT_CAP_ptwrite))
+               if (!intel_pt_validate_hw_cap(PT_CAP_ptwrite))
                        return false;

                /* FUPonPTW without PTW doesn't make sense */
@@ -598,7 +599,7 @@  static struct topa *topa_alloc(int cpu, gfp_t gfp)
         * In case of singe-entry ToPA, always put the self-referencing END
         * link as the 2nd entry in the table
         */
-       if (!pt_cap_get(PT_CAP_topa_multiple_entries)) {
+       if (!intel_pt_validate_hw_cap(PT_CAP_topa_multiple_entries)) {
                TOPA_ENTRY(topa, 1)->base = topa->phys >> TOPA_SHIFT;
                TOPA_ENTRY(topa, 1)->end = 1;
        }
@@ -638,7 +639,7 @@  static void topa_insert_table(struct pt_buffer *buf, struct topa *topa)
        topa->offset = last->offset + last->size;
        buf->last = topa;

-       if (!pt_cap_get(PT_CAP_topa_multiple_entries))
+       if (!intel_pt_validate_hw_cap(PT_CAP_topa_multiple_entries))
                return;

        BUG_ON(last->last != TENTS_PER_PAGE - 1);
@@ -654,7 +655,7 @@  static void topa_insert_table(struct pt_buffer *buf, struct topa *topa)
 static bool topa_table_full(struct topa *topa)
 {
        /* single-entry ToPA is a special case */
-       if (!pt_cap_get(PT_CAP_topa_multiple_entries))
+       if (!intel_pt_validate_hw_cap(PT_CAP_topa_multiple_entries))
```

```
                return !!topa->last;

        return topa->last == TENTS_PER_PAGE - 1;
@@ -690,7 +691,8 @@  static int topa_insert_pages(struct pt_buffer *buf, gfp_t gfp)

        TOPA_ENTRY(topa, -1)->base = page_to_phys(p) >> TOPA_SHIFT;
        TOPA_ENTRY(topa, -1)->size = order;
-       if (!buf->snapshot && !pt_cap_get(PT_CAP_topa_multiple_entries)) {
+       if (!buf->snapshot &&
+           !intel_pt_validate_hw_cap(PT_CAP_topa_multiple_entries)) {
                TOPA_ENTRY(topa, -1)->intr = 1;
                TOPA_ENTRY(topa, -1)->stop = 1;
        }
@@ -725,7 +727,7 @@  static void pt_topa_dump(struct pt_buffer *buf)
                                topa->table[i].intr ? 'I' : ' ',
                                topa->table[i].stop ? 'S' : ' ',
                                *(u64 *)&topa->table[i]);
-                       if ((pt_cap_get(PT_CAP_topa_multiple_entries) &&
+                       if ((intel_pt_validate_hw_cap(PT_CAP_topa_multiple_entries) &&
                             topa->table[i].stop) ||
                            topa->table[i].end)
                                break;
@@ -828,7 +830,7 @@  static void pt_handle_status(struct pt *pt)
                 * means we are already losing data; need to let the decoder
                 * know.
                 */
-               if (!pt_cap_get(PT_CAP_topa_multiple_entries) ||
+               if (!intel_pt_validate_hw_cap(PT_CAP_topa_multiple_entries) ||
                    buf->output_off == sizes(TOPA_ENTRY(buf->cur, buf->cur_idx)->size)) {
                        perf_aux_output_flag(&pt->handle,
                                             PERF_AUX_FLAG_TRUNCATED);
@@ -840,7 +842,8 @@  static void pt_handle_status(struct pt *pt)
                 * Also on single-entry ToPA implementations, interrupt will come
                 * before the output reaches its output region's boundary.
                 */
-               if (!pt_cap_get(PT_CAP_topa_multiple_entries) && !buf->snapshot &&
+               if (!intel_pt_validate_hw_cap(PT_CAP_topa_multiple_entries) &&
+                   !buf->snapshot &&
                    pt_buffer_region_size(buf) - buf->output_off <= TOPA_PMI_MARGIN) {
                        void *head = pt_buffer_region(buf);

@@ -931,7 +934,7 @@  static int pt_buffer_reset_markers(struct pt_buffer *buf,


        /* single entry ToPA is handled by marking all regions STOP=1 INT=1 */
-       if (!pt_cap_get(PT_CAP_topa_multiple_entries))
+       if (!intel_pt_validate_hw_cap(PT_CAP_topa_multiple_entries))
                return 0;

        /* clear STOP and INT from current entry */
@@ -1082,7 +1085,7 @@  static int pt_buffer_init_topa(struct pt_buffer *buf, unsigned long nr_pages,
```

```
        pt_buffer_setup_topa_index(buf);

        /* link last table to the first one, unless we're double buffering */
-       if (pt_cap_get(PT_CAP_topa_multiple_entries)) {
+       if (intel_pt_validate_hw_cap(PT_CAP_topa_multiple_entries)) {
                TOPA_ENTRY(buf->last, -1)->base = buf->first->phys >> TOPA_SHIFT;
                TOPA_ENTRY(buf->last, -1)->end = 1;
        }
@@ -1153,7 +1156,7 @@  static int pt_addr_filters_init(struct perf_event *event)
        struct pt_filters *filters;
        int node = event->cpu == -1 ? -1 : cpu_to_node(event->cpu);

-       if (!pt_cap_get(PT_CAP_num_address_ranges))
+       if (!intel_pt_validate_hw_cap(PT_CAP_num_address_ranges))
                return 0;

        filters = kzalloc_node(sizeof(struct pt_filters), GFP_KERNEL, node);
@@ -1202,7 +1205,7 @@  static int pt_event_addr_filters_validate(struct list_head *filters)
                                return -EINVAL;
                }

-               if (++range > pt_cap_get(PT_CAP_num_address_ranges))
+               if (++range > intel_pt_validate_hw_cap(PT_CAP_num_address_ranges))
                        return -EOPNOTSUPP;
        }

@@ -1507,12 +1510,12 @@  static __init int pt_init(void)
        if (ret)
                return ret;

-       if (!pt_cap_get(PT_CAP_topa_output)) {
+       if (!intel_pt_validate_hw_cap(PT_CAP_topa_output)) {
                pr_warn("ToPA output is not supported on this CPU\n");
                return -ENODEV;
        }

-       if (!pt_cap_get(PT_CAP_topa_multiple_entries))
+       if (!intel_pt_validate_hw_cap(PT_CAP_topa_multiple_entries))
                pt_pmu.pmu.capabilities =
                        PERF_PMU_CAP_AUX_NO_SG | PERF_PMU_CAP_AUX_SW_DOUBLEBUF;

@@ -1530,7 +1533,7 @@  static __init int pt_init(void)
        pt_pmu.pmu.addr_filters_validate = pt_event_addr_filters_validate;
        pt_pmu.pmu.nr_addr_filters       =
-               pt_cap_get(PT_CAP_num_address_ranges);
+               intel_pt_validate_hw_cap(PT_CAP_num_address_ranges);

        ret = perf_pmu_register(&pt_pmu.pmu, "intel_pt", -1);

diff --git a/arch/x86/events/intel/pt.h b/arch/x86/events/intel/pt.h
```

```
index 0050ca1..269e15a 100644
--- a/arch/x86/events/intel/pt.h
+++ b/arch/x86/events/intel/pt.h
@@ -45,30 +45,9 @@  struct topa_entry {
        u64     rsvd4   : 16;
 };

-#define PT_CPUID_LEAVES                2
-#define PT_CPUID_REGS_NUM      4 /* number of regsters (eax, ebx, ecx, edx) */
-
 /* TSC to Core Crystal Clock Ratio */
 #define CPUID_TSC_LEAF         0x15

-enum pt_capabilities {
-       PT_CAP_max_subleaf = 0,
-       PT_CAP_cr3_filtering,
-       PT_CAP_psb_cyc,
-       PT_CAP_ip_filtering,
-       PT_CAP_mtc,
-       PT_CAP_ptwrite,
-       PT_CAP_power_event_trace,
-       PT_CAP_topa_output,
-       PT_CAP_topa_multiple_entries,
-       PT_CAP_single_range_output,
-       PT_CAP_payloads_lip,
-       PT_CAP_num_address_ranges,
-       PT_CAP_mtc_periods,
-       PT_CAP_cycle_thresholds,
-       PT_CAP_psb_periods,
-};
-
 struct pt_pmu {
        struct pmu              pmu;
        u32                     caps[PT_CPUID_REGS_NUM * PT_CPUID_LEAVES];
diff --git a/arch/x86/include/asm/intel_pt.h b/arch/x86/include/asm/intel_pt.h
index b523f51..fa4b4fd 100644
--- a/arch/x86/include/asm/intel_pt.h
+++ b/arch/x86/include/asm/intel_pt.h
@@ -2,10 +2,33 @@
 #ifndef _ASM_X86_INTEL_PT_H
 #define _ASM_X86_INTEL_PT_H

+#define PT_CPUID_LEAVES                2
+#define PT_CPUID_REGS_NUM      4 /* number of regsters (eax, ebx, ecx, edx) */
+
+enum pt_capabilities {
+       PT_CAP_max_subleaf = 0,
+       PT_CAP_cr3_filtering,
+       PT_CAP_psb_cyc,
+       PT_CAP_ip_filtering,
+       PT_CAP_mtc,
```

```
+	PT_CAP_ptwrite,
+	PT_CAP_power_event_trace,
+	PT_CAP_topa_output,
+	PT_CAP_topa_multiple_entries,
+	PT_CAP_single_range_output,
+	PT_CAP_payloads_lip,
+	PT_CAP_num_address_ranges,
+	PT_CAP_mtc_periods,
+	PT_CAP_cycle_thresholds,
+	PT_CAP_psb_periods,
+};
+
 #if defined(CONFIG_PERF_EVENTS) && defined(CONFIG_CPU_SUP_INTEL)
 void cpu_emergency_stop_pt(void);
+extern u32 intel_pt_validate_hw_cap(enum pt_capabilities cap);
 #else
 static inline void cpu_emergency_stop_pt(void) {}
+static inline u32 intel_pt_validate_hw_cap(enum pt_capabilities cap) { return 0; }
 #endif

 #endif /* _ASM_X86_INTEL_PT_H */
```