KVM: ARM64: Add guest PMU support

From:	Shannon Zhao <zhaoshenglong@huawei.com>
To:	<kvmarm@lists.cs.columbia.edu>, <marc.zyngier@arm.com>, <christoffer.dall@linaro.org>
Subject:	[PATCH v9 00/21] KVM: ARM64: Add guest PMU support
Date:	Fri, 15 Jan 2016 14:27:34 +0800
Message-ID:	<1452839275-19368-1-git-send-email-zhaoshenglong@huawei.com>
Cc:	kvm@vger.kernel.org, will.deacon@arm.com, linux-arm-kernel@lists.infradead.org, shannon.zhao@linaro.org
Archive-link:	Article

```
From: Shannon Zhao <shannon.zhao@linaro.org>

This patchset adds guest PMU support for KVM on ARM64. It takes
trap-and-emulate approach. When guest wants to monitor one event, it
will be trapped by KVM and KVM will call perf_event API to create a perf
event and call relevant perf_event APIs to get the count value of event.

Use perf to test this patchset in guest. When using "perf list", it
shows the list of the hardware events and hardware cache events perf
supports. Then use "perf stat -e EVENT" to monitor some event. For
example, use "perf stat -e cycles" to count cpu cycles and
"perf stat -e cache-misses" to count cache misses.

Below are the outputs of "perf stat -r 5 sleep 5" when running in host
and guest.

Host:
 Performance counter stats for 'sleep 5' (5 runs):

          0.529248      task-clock (msec)         #    0.000 CPUs utilized           ( +-  1.65% )
                 1      context-switches          #    0.002 M/sec
                 0      cpu-migrations            #    0.000 K/sec
                49      page-faults               #    0.092 M/sec                   ( +-  1.05% )
           1104279      cycles                    #    2.087 GHz                     ( +-  1.65% )
   <not supported>      stalled-cycles-frontend
   <not supported>      stalled-cycles-backend
            528112      instructions              #    0.48  insns per cycle         ( +-  1.12% )
   <not supported>      branches
              9579      branch-misses             #   18.099 M/sec                   ( +-  2.40% )

       5.000851904 seconds time elapsed                                          ( +-  0.00% )

Guest:
 Performance counter stats for 'sleep 5' (5 runs):

          0.695412      task-clock (msec)         #    0.000 CPUs utilized           ( +-  1.26% )
                 1      context-switches          #    0.001 M/sec
                 0      cpu-migrations            #    0.000 K/sec
                49      page-faults               #    0.070 M/sec                   ( +-  1.29% )
           1430471      cycles                    #    2.057 GHz                     ( +-  1.25% )
   <not supported>      stalled-cycles-frontend
   <not supported>      stalled-cycles-backend
            659173      instructions              #    0.46  insns per cycle         ( +-  2.64% )
   <not supported>      branches
             10893      branch-misses             #   15.664 M/sec                   ( +-  1.23% )

       5.001277044 seconds time elapsed                                          ( +-  0.00% )

Have a cycle counter read test like below in guest and host:

static void test(void)
{
        unsigned long count, count1, count2;
        count1 = read_cycles();
        count++;
        count2 = read_cycles();
}

Host:
count1: 3046505444
count2: 3046505575
delta: 131

Guest:
count1: 5932773531
count2: 5932773668
delta: 137

The gap between guest and host is very small. One reason for this I
think is that it doesn't count the cycles in EL2 and host since we add
exclude_hv = 1. So the cycles spent to store/restore registers which
happens at EL2 are not included.

This patchset can be fetched from [1] and the relevant QEMU version for
test can be fetched from [2].

The results of 'perf test' can be found from [3][4].
The results of perf_event_tests test suite can be found from [5][6].
```

Also, I have tested "perf top" in two VMs and host at the same time. It
works well.

Thanks,
Shannon

[1] https://git.linaro.org/people/shannon.zhao/linux-mainline... KVM_ARM64_PMU_v9
[2] https://git.linaro.org/people/shannon.zhao/qemu.git PMU
[3] http://people.linaro.org/~shannon.zhao/PMU/perf-test-host...
[4] http://people.linaro.org/~shannon.zhao/PMU/perf-test-gues...
[5] http://people.linaro.org/~shannon.zhao/PMU/perf_event_tes...
[6] http://people.linaro.org/~shannon.zhao/PMU/perf_event_tes...

Changes since v8:
* Fix the wrong use of r->reg in register accessors for 32bit part
* Rewrite the handle of PMUSERENR based on the new inject UND patch
* Drop the inline attribute
* Introduce SET/GET/HAS_DEVICE_ATTR for vcpu iotcl and set the PMU
  overflow interrupt via this API
* Use one feature bit for PMUv3

Changes since v7:
* Rebase on kvm-arm next
* Fix the handler of PMUSERENR and add a helper to forward trap to guest
  EL1
* Fix some small bugs found by Marc

Changes since v6:
* Rebase on v4.4-rc5
* Drop access_pmu_cp15_regs() so that it could use same handler for both
  arch64 and arch32. And it could drop the definitions of CP15 register
  offsets, also avoid same codes added twice
* Use vcpu_sys_reg() when accessing PMU registers to avoid endian things
* Add handler for PMUSERENR and some checkers for other registers
* Add kvm_arm_pmu_get_attr()

Changes since v5:
* Rebase on new linux kernel mainline
* Remove state duplications and drop PMOVSCLR, PMCNTENCLR, PMINTENCLR,
  PMXEVCNTR, PMXEVTYPER
* Add a helper to check if vPMU is already initialized
* remove kvm_vcpu from kvm_pmc

Changes since v4:
* Rebase on new linux kernel mainline
* Drop the reset handler of CP15 registers
* Fix a compile failure on arch ARM due to lack of asm/pmu.h
* Refactor the interrupt injecting flow according to Marc's suggestion
* Check the value of PMSELR register
* Calculate the attr.disabled according to PMCR.E and PMCNTENSET/CLR
* Fix some coding style
* Document the vPMU irq range

Changes since v3:
* Rebase on new linux kernel mainline
* Use ARMV8_MAX_COUNTERS instead of 32
* Reset PMCR.E to zero.
* Trigger overflow for software increment.
* Optimize PMU interrupt inject logic.
* Add handler for E,C,P bits of PMCR
* Fix the overflow bug found by perf_event_tests
* Run 'perf test', 'perf top' and perf_event_tests test suite
* Add exclude_hv = 1 configuration to not count in EL2

Changes since v2:
* Directly use perf raw event type to create perf_event in KVM
* Add a helper vcpu_sysreg_write
* remove unrelated header file

Changes since v1:
* Use switch...case for registers access handler instead of adding
  alone handler for each register
* Try to use the sys_regs to store the register value instead of adding
  new variables in struct kvm_pmc
* Fix the handle of cp15 regs
* Create a new kvm device vPMU, then userspace could choose whether to
  create PMU
* Fix the handle of PMU overflow interrupt

Shannon Zhao (21):
  ARM64: Move PMU register related defines to asm/pmu.h
  KVM: ARM64: Define PMU data structure for each vcpu
  KVM: ARM64: Add offset defines for PMU registers
  KVM: ARM64: Add access handler for PMCR register
  KVM: ARM64: Add access handler for PMSELR register
  KVM: ARM64: Add access handler for PMCEID0 and PMCEID1 register
  KVM: ARM64: PMU: Add perf event map and introduce perf event creating
    function
  KVM: ARM64: Add access handler for event type register
  KVM: ARM64: Add access handler for event counter register
  KVM: ARM64: Add access handler for PMCNTENSET and PMCNTENCLR register
  KVM: ARM64: Add access handler for PMINTENSET and PMINTENCLR register
  KVM: ARM64: Add access handler for PMOVSSET and PMOVSCLR register
  KVM: ARM64: Add access handler for PMSWINC register
  KVM: ARM64: Add helper to handle PMCR register bits
  KVM: ARM64: Add access handler for PMUSERENR register
  KVM: ARM64: Add PMU overflow interrupt routing
  KVM: ARM64: Reset PMU state when resetting vcpu
  KVM: ARM64: Free perf event of PMU when destroying vcpu
  KVM: ARM64: Add a new feature bit for PMUv3
  KVM: ARM: Introduce per-vcpu kvm device controls

```
    KVM: ARM64: Add a new vcpu device control group for PMUv3

  Documentation/virtual/kvm/api.txt          |  12 +-
  Documentation/virtual/kvm/devices/vcpu.txt |  32 ++
  arch/arm/include/asm/kvm_host.h            |  15 +
  arch/arm/kvm/arm.c                         |  61 +++
  arch/arm64/include/asm/kvm_host.h          |  25 +-
  arch/arm64/include/asm/pmu.h               |  81 ++++
  arch/arm64/include/uapi/asm/kvm.h          |   6 +
  arch/arm64/kernel/perf_event.c             |  36 +-
  arch/arm64/kvm/Kconfig                     |   7 +
  arch/arm64/kvm/Makefile                    |   1 +
  arch/arm64/kvm/guest.c                     |  51 +++
  arch/arm64/kvm/hyp/hyp.h                   |   1 +
  arch/arm64/kvm/hyp/switch.c                |   3 +
  arch/arm64/kvm/reset.c                     |   7 +
  arch/arm64/kvm/sys_regs.c                  | 570 ++++++++++++++++++++++++++--
  include/kvm/arm_pmu.h                      | 102 ++++++
  include/uapi/linux/kvm.h                   |   2 +
  virt/kvm/arm/pmu.c                         | 513 +++++++++++++++++++++++++
  18 files changed, 1456 insertions(+), 69 deletions(-)
  create mode 100644 Documentation/virtual/kvm/devices/vcpu.txt
  create mode 100644 arch/arm64/include/asm/pmu.h
  create mode 100644 include/kvm/arm_pmu.h
  create mode 100644 virt/kvm/arm/pmu.c

 --
 2.0.4
```