Perf

Perf Event Attributes

Author: Andrew Murray <andrew.murray@arm.com>
Date: 2019-03-06

exclude_user

This attribute excludes userspace.

Userspace always runs at EL0 and thus this attribute will exclude EL0.

exclude_kernel

This attribute excludes the kernel.

The kernel runs at EL2 with VHE and EL1 without. Guest kernels always run at EL1.

For the host this attribute will exclude EL1 and additionally EL2 on a VHE system.

For the guest this attribute will exclude EL1. Please note that EL2 is never counted within a guest.

exclude_hv

This attribute excludes the hypervisor.

For a VHE host this attribute is ignored as we consider the host kernel to be the hypervisor.

For a non-VHE host this attribute will exclude EL2 as we consider the hypervisor to be any code that runs at EL2 which is predominantly used for guest/host transitions.

For the guest this attribute has no effect. Please note that EL2 is never counted within a guest.

exclude_host / exclude_guest

These attributes exclude the KVM host and guest, respectively.

The KVM host may run at EL0 (userspace), EL1 (non-VHE kernel) and EL2 (VHE kernel or non-VHE hypervisor).

The KVM guest may run at EL0 (userspace) and EL1 (kernel).

Due to the overlapping exception levels between host and guests we cannot exclusively rely on the PMU's hardware exception filtering - therefore we must enable/disable counting on the entry and exit to the guest. This is performed differently on VHE and non-VHE systems.

For non-VHE systems we exclude EL2 for exclude_host - upon entering and exiting the guest we disable/enable the event as appropriate based on the exclude_host and exclude_guest attributes.

For VHE systems we exclude EL1 for exclude_guest and exclude both EL0,EL2 for exclude_host. Upon entering and exiting the guest we modify the event to include/exclude EL0 as appropriate based on the exclude_host and exclude_guest attributes.

The statements above also apply when these attributes are used within a non-VHE guest however please note that EL2 is never counted within a guest.

Accuracy

On non-VHE hosts we enable/disable counters on the entry/exit of host/guest transition at EL2 - however there is a period of time between enabling/disabling the counters and entering/exiting the guest. We are able to eliminate counters counting host events on the boundaries of guest entry/exit when counting guest events by filtering out EL2 for exclude_host. However when using !exclude_hv there is a small blackout window at the guest entry/exit where host events are not captured.

On VHE systems there are no blackout windows.

Perf Userspace PMU Hardware Counter Access

Overview

The perf userspace tool relies on the PMU to monitor events. It offers an abstraction layer over the hardware counters since the underlying implementation is cpu-dependent. Arm64 allows userspace tools to have access to the registers storing the hardware counters' values directly.

This targets specifically self-monitoring tasks in order to reduce the overhead by directly accessing the registers without having to go through the kernel.

How-to

The focus is set on the armv8 PMUv3 which makes sure that the access to the pmu registers is enabled and that the userspace has access to the relevant information in order to use them.

In order to have access to the hardware counters, the global sysctl kernel/perf_user_access must first be enabled:

```
echo 1 > /proc/sys/kernel/perf_user_access
```

It is necessary to open the event using the perf tool interface with config1:1 attr bit set: the sys_perf_event_open syscall returns a fd which can subsequently be used with the mmap syscall in order to retrieve a page of memory containing information about the event. The PMU driver uses this page to expose to the user the hardware counter's index and other necessary data. Using this index enables the user to access the PMU registers using the *mrs* instruction. Access to the PMU registers is only valid while the sequence lock is unchanged. In particular, the PMSELR_EL0 register is zeroed each time the sequence lock is changed.

The userspace access is supported in libperf using the perf_evsel__mmap() and perf_evsel__read() functions. See tools/lib/perf/tests/test-evsel.c for an example.

About heterogeneous systems

On heterogeneous systems such as big.LITTLE, userspace PMU counter access can only be enabled when the tasks are pinned to a homogeneous subset of cores and the corresponding PMU instance is opened by specifying the 'type' attribute. The use of generic event types is not supported in this case.

Have a look at tools/perf/arch/arm64/tests/user-events.c for an example. It can be run using the perf tool to check that the access to the registers works correctly from userspace:

```
perf test -v user
```

About chained events and counter sizes

The user can request either a 32-bit (config1:0 == 0) or 64-bit (config1:0 == 1) counter along with userspace access. The sys_perf_event_open syscall will fail if a 64-bit counter is requested and the hardware doesn't support 64-bit counters. Chained events are not supported in conjunction with userspace counter access. If a 32-bit counter is requested on hardware with 64-bit counters, then userspace must treat the upper 32-bits read from the counter as UNKNOWN. The 'pmc_width' field in the user page will indicate the valid width of the counter and should be used to mask the upper bits as needed.