Asymmetric 32-bit SoCs

Author: Will Deacon <will@kernel.org>

This document describes the impact of asymmetric 32-bit SoCs on the execution of 32-bit (`AArch32`) applications.

Date: 2021-05-17

Introduction

Some Armv9 SoCs suffer from a big.LITTLE misfeature where only a subset of the CPUs are capable of executing 32-bit user applications. On such a system, Linux by default treats the asymmetry as a "mismatch" and disables support for both the `PER_LINUX32` personality and `execve(2)` of 32-bit ELF binaries, with the latter returning `-ENOEXEC`. If the mismatch is detected during late onlining of a 64-bit-only CPU, then the onlining operation fails and the new CPU is unavailable for scheduling.

Surprisingly, these SoCs have been produced with the intention of running legacy 32-bit binaries. Unsurprisingly, that doesn't work very well with the default behaviour of Linux.

It seems inevitable that future SoCs will drop 32-bit support altogether, so if you're stuck in the unenviable position of needing to run 32-bit code on one of these transitionary platforms then you would be wise to consider alternatives such as recompilation, emulation or retirement. If neither of those options are practical, then read on.

Enabling kernel support

Since the kernel support is not completely transparent to userspace, allowing 32-bit tasks to run on an asymmetric 32-bit system requires an explicit "opt-in" and can be enabled by passing the `allow_mismatched_32bit_el0` parameter on the kernel command-line.

For the remainder of this document we will refer to an *asymmetric system* to mean an asymmetric 32-bit SoC running Linux with this kernel command-line option enabled.

Userspace impact

32-bit tasks running on an asymmetric system behave in mostly the same way as on a homogeneous system, with a few key differences relating to CPU affinity.

sysfs

The subset of CPUs capable of running 32-bit tasks is described in `/sys/devices/system/cpu/aarch32_el0` and is documented further in `Documentation/ABI/testing/sysfs-devices-system-cpu`.

Note: CPUs are advertised by this file as they are detected and so late-onlining of 32-bit-capable CPUs can result in the file contents being modified by the kernel at runtime. Once advertised, CPUs are never removed from the file.

`execve(2)`

On a homogeneous system, the CPU affinity of a task is preserved across `execve(2)`. This is not always possible on an asymmetric system, specifically when the new program being executed is 32-bit yet the affinity mask contains 64-bit-only CPUs. In this situation, the kernel determines the new affinity mask as follows:

1. If the 32-bit-capable subset of the affinity mask is not empty, then the affinity is restricted to that subset and the old affinity mask is saved. This saved mask is inherited over `fork(2)` and preserved across `execve(2)` of 32-bit programs.

   Note: This step does not apply to `SCHED_DEADLINE` tasks. See SCHED_DEADLINE.

2. Otherwise, the cpuset hierarchy of the task is walked until an ancestor is found containing at least one 32-bit-capable CPU. The affinity of the task is then changed to match the 32-bit-capable subset of the cpuset determined by the walk.

3. On failure (i.e. out of memory), the affinity is changed to the set of all 32-bit-capable CPUs of which the kernel is aware.

A subsequent `execve(2)` of a 64-bit program by the 32-bit task will invalidate the affinity mask saved in (1) and attempt to restore the CPU affinity of the task using the saved mask if it was previously valid. This restoration may fail due to intervening changes to the deadline policy or cpuset hierarchy, in which case the `execve(2)` continues with the affinity unchanged.

Calls to `sched_setaffinity(2)` for a 32-bit task will consider only the 32-bit-capable CPUs of the requested affinity mask. On success, the affinity for the task is updated and any saved mask from a prior `execve(2)` is invalidated.

SCHED_DEADLINE

Explicit admission of a 32-bit deadline task to the default root domain (e.g. by calling `sched_setattr(2)`) is rejected on an asymmetric 32-bit system unless admission control is disabled by writing -1 to `/proc/sys/kernel/sched_rt_runtime_us`.

`execve(2)` of a 32-bit program from a 64-bit deadline task will return `-ENOEXEC` if the root domain for the task contains any 64-bit-only CPUs and admission control is enabled. Concurrent offlining of 32-bit-capable CPUs may still necessitate the procedure described in execve(2), in which case step (1) is skipped and a warning is emitted on the console.

Note: It is recommended that a set of 32-bit-capable CPUs are placed into a separate root domain if `SCHED_DEADLINE` is to be used with 32-bit tasks on an asymmetric system. Failure to do so is likely to result in missed deadlines.

Cpusets

The affinity of a 32-bit task on an asymmetric system may include CPUs that are not explicitly allowed by the cpuset to which it is attached. This can occur as a result of the following two situations:

- A 64-bit task attached to a cpuset which allows only 64-bit CPUs executes a 32-bit program.
- All of the 32-bit-capable CPUs allowed by a cpuset containing a 32-bit task are offlined.

In both of these cases, the new affinity is calculated according to step (2) of the process described in execve(2) and the cpuset hierarchy is unchanged irrespective of the cgroup version.

CPU hotplug

On an asymmetric system, the first detected 32-bit-capable CPU is prevented from being offlined by userspace and any such attempt will return `-EPERM`. Note that suspend is still permitted even if the primary CPU (i.e. CPU 0) is 64-bit-only.

KVM

Although KVM will not advertise 32-bit EL0 support to any vCPUs on an asymmetric system, a broken guest at EL1 could still attempt to execute 32-bit code at EL0. In this case, an exit from a vCPU thread in 32-bit mode will return to host userspace with an `exit_reason` of `KVM_EXIT_FAIL_ENTRY` and will remain non-runnable until successfully re-initialised by a subsequent `KVM_ARM_VCPU_INIT` operation.