

Devfreq for uncore DVFS

Jie Zhan - Huawei

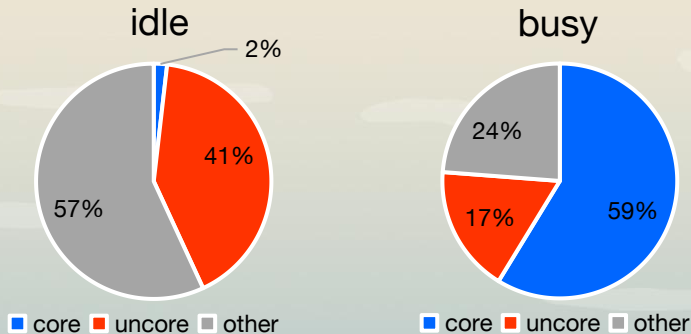


LINUX
PLUMBERS
CONFERENCE

PRAGUE, CZECHIA / OCT. 5-7, 2026

Background and Motivation

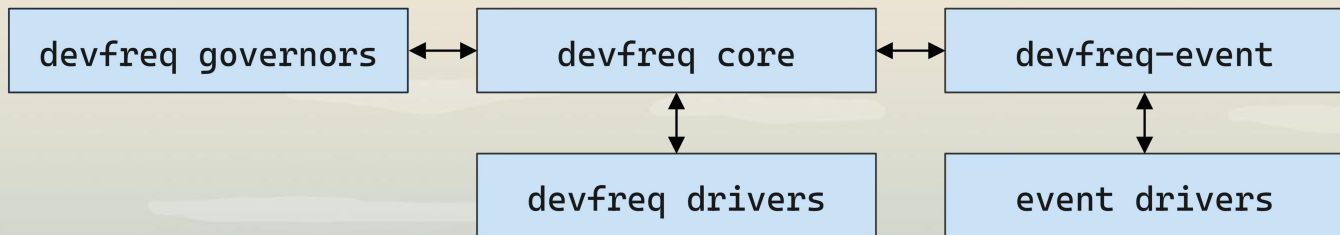
- What's "uncore"?
 - In practice, SoC common components, such as L3 cache, interconnects, memory controllers, excluding I/O devices (network, PCIe), GPU, etc.
- Uncore DVFS
 - Significant power saving under low system utilization.



- Kernel support needed
 - Some metrics only available in kernel, e.g. PMUs, software events
 - Tunables for userspace to meet variable scenarios

Upstream framework - devfreq

- devfreq: Generic DVFS framework for non-CPU devices
 - Typical devices: interconnect / bus, memory controller, GPU
 - Components: core, drivers, governors, events
 - Metrics / events come from devfreq-event drivers.



Metric selection

- Challenges of measuring uncore “utilization”
 - No in-kernel utilization metrics - unlike cpu, uncore device not perceivable to kernel.
 - Hardware cycles can't reflect utilization - uncore can be constantly clocked.
 - No generic / portable definitions of such metrics.

Option	Pros	Cons
CPU utilization	Ready, portable	Uncore performance demand can go opposite to CPU utilization
ARM AMU or something similar	Architecturally defined	Difficult to translate to uncore frequency demand
Uncore PMU (current attempt)	Relevant to uncore components	Hard to cover all traffics or loads across SoC

perf core ABI

- Existing perf ABI - `perf_event_create_kernel_counter()`
 - In-kernel perf event creation
 - Callers today: KVM (x86, arm64, riscv), perf watchdog, hw_breakpoint
 - Passing `attr.type`: compile-time type, or a dynamic PMU ID passed from the driver (arm64 KVM)
 - **Gap**: no reliable in-kernel ABI to look up a dynamic PMU ID

- new ABI?

`perf_pmu_type_by_parent_dev(parent)`

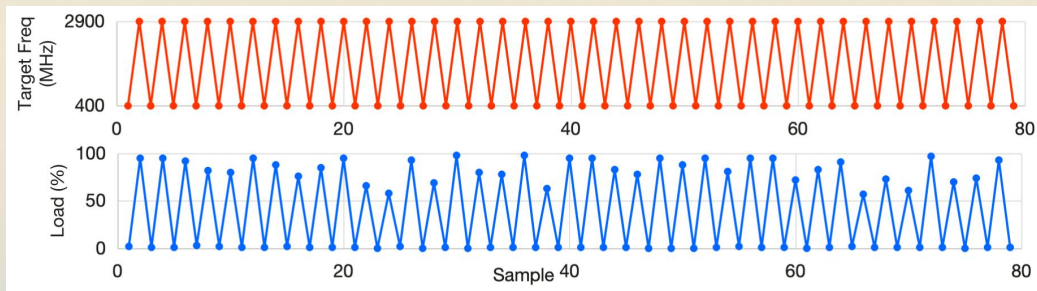
- Find the PMU type ID by its parent device
- Issue: an ID can be invalid after PMU unregistration

`perf_event_create_kernel_counter_for_dev(attr, parent, ...)`

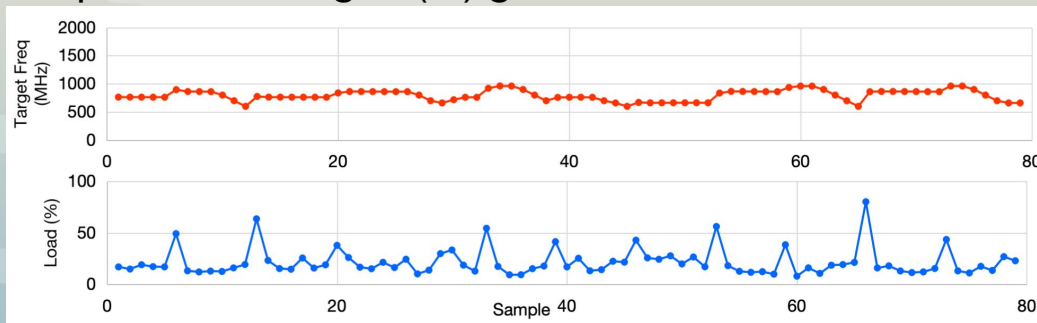
- Create a perf event on the PMU of the parent in a `pmus_srcu` section
- Issue: sort of overlapped with the existing one

In-kernel governor mechanism

- devfreq dynamic scaling governor - `simple_ondemand`
 - set to max, if $\text{load\%} > \text{up_th}$
 - keep current freq, if $\text{down_th} < \text{load\%} < \text{up_th}$
 - set to $\text{freq_curr} * \text{load\%} / \text{mid_th}$, if $\text{load\%} < \text{down_th}$
 - a test (without carefully tweaking parameters)



- Proportional-Integral (PI) governor?



Current status and discussion

- RFC to post
- Some plumbing work for `devfreq` core
- QoS voting from CPU or IO devices?

