A unified solution for SoC idling

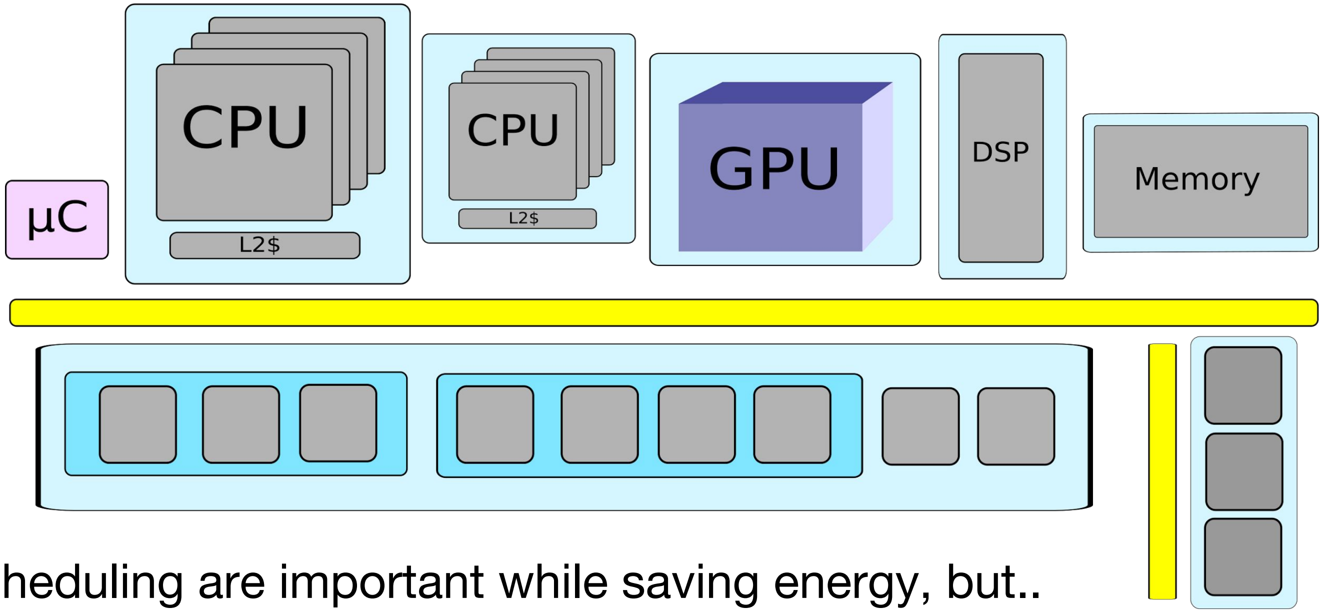
- how far have we come!?

Ulf Hansson, Linaro

Agenda

- Introduction to SoC idling.
- A top-level overview.
- Some important infrastructure changes for ARM SoC PM code.
- The runtime PM centric approach.
- An update on the generic PM domain.
- An update on the unified solution to SoC idling - CPU cluster PM.
- Next challenges.

What is SoC idling?



- CPU scheduling are important while saving energy, but..
- .. lots of other resources needs to enter low power state when idle.

Avoid wasting power when idle!

The involved PM frameworks

- System PM (system wide with all devices)
- Runtime PM (fine grained for any device that is inactive)
- PM domain (genpd, ACPI, etc)
- Device PM QoS (latency constraints per device)
- Device wakeup/wakeirq (wakeup settings for devices)

Enable upstreaming of ARM SoC PM code

- Make collaboration of runtime PM and system PM better.
- Simplify for driver authors to deploy PM support.
- Evolve and modernize the generic PM domain.
- Various optimizations.
- Deployment to provide references (nowadays lots of references).

The runtime PM centric approach - what?

- When the low power state of a device is similar for runtime PM and system PM.
- Re-use runtime PM callbacks for system PM, - hence the “runtime PM centric approach”.
- Don’t unnecessary resume the device from system PM, but defer to runtime PM.

Deploy runtime PM support - get system PM for “free”!

..don’t forget to deal with wakeups.

The runtime PM centric approach - how?

Mydrv.c:

```
static const struct dev_pm_ops mydrv_dev_pm_ops = {
        SET_SYSTEM_SLEEP_PM_OPS(pm_runtime_force_suspend,
                                pm_runtime_force_resume)
        SET_RUNTIME_PM_OPS(mydrv_ runtime_suspend,
                           mydrv_runtime_resume,
                           NULL)
};
```

That’s it!

The runtime PM centric approach - status

Trend of usage:

- 3.15: 1 (introduced pm_runtime_force_suspend|resume())
- 3.18: 7
- 4.5: 16
- 4.11-rc4: 46

Some new ideas:

- Should the helpers respect device links (and not just parent/children)?
- Can drivers using "direct complete" convert to the runtime PM centric approach? Benefit from the deferred resume, requires to adopt the ACPI PM domain (avoid runtime resuming devices for these devices at system PM suspend).

The generic PM domain - status

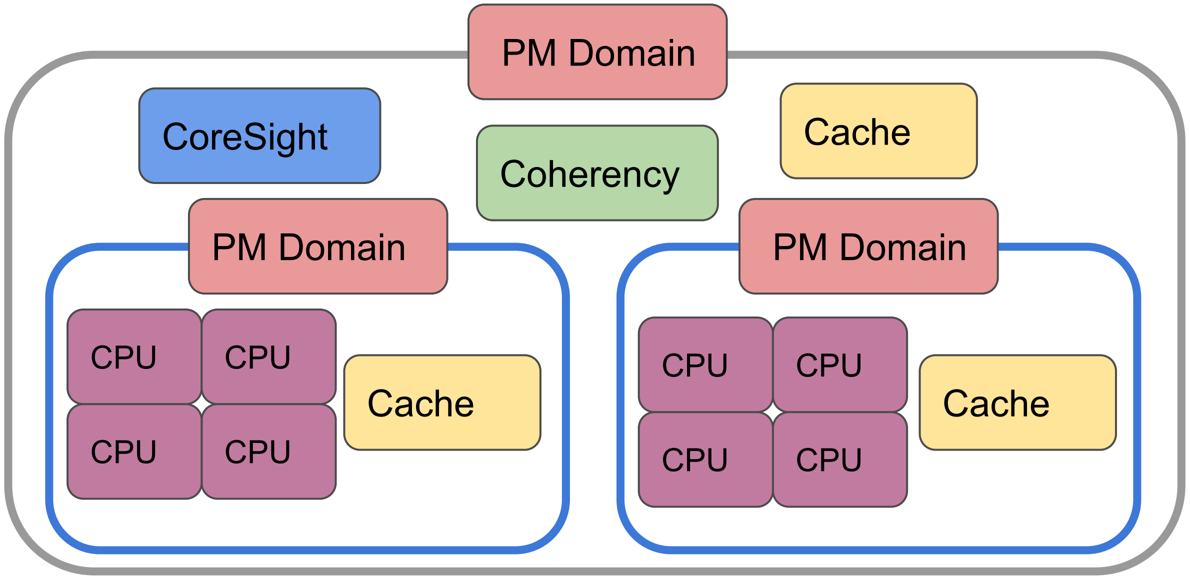
Trend of usage, includes SoC families:

- 3.18: 5
- 4.11-rc4: 19

Recent highlights:

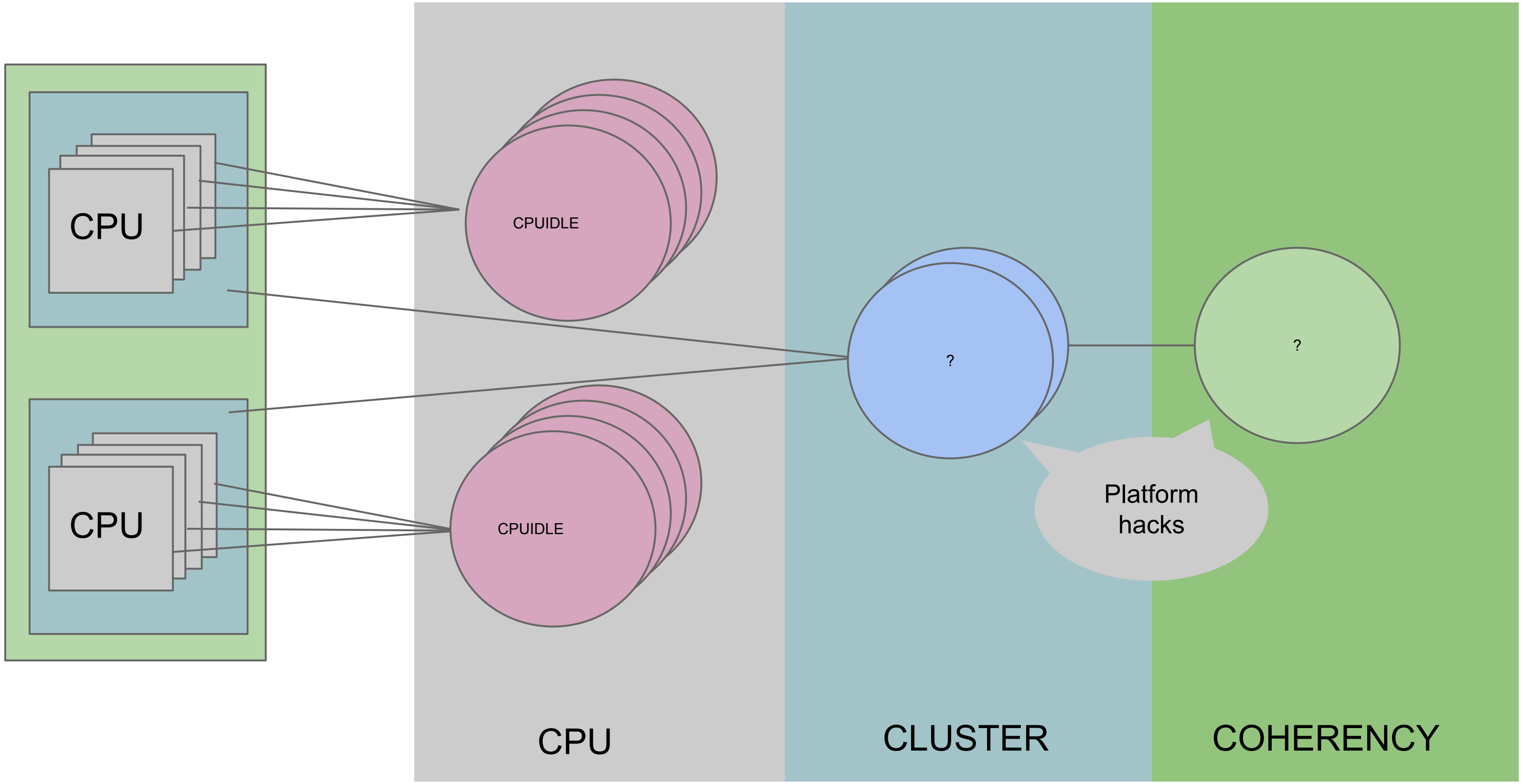
- Minimized latencies in the power off sequence.
- IRQ safe domain support.
- Multiple domain idle states support.

And it gets more complicated...

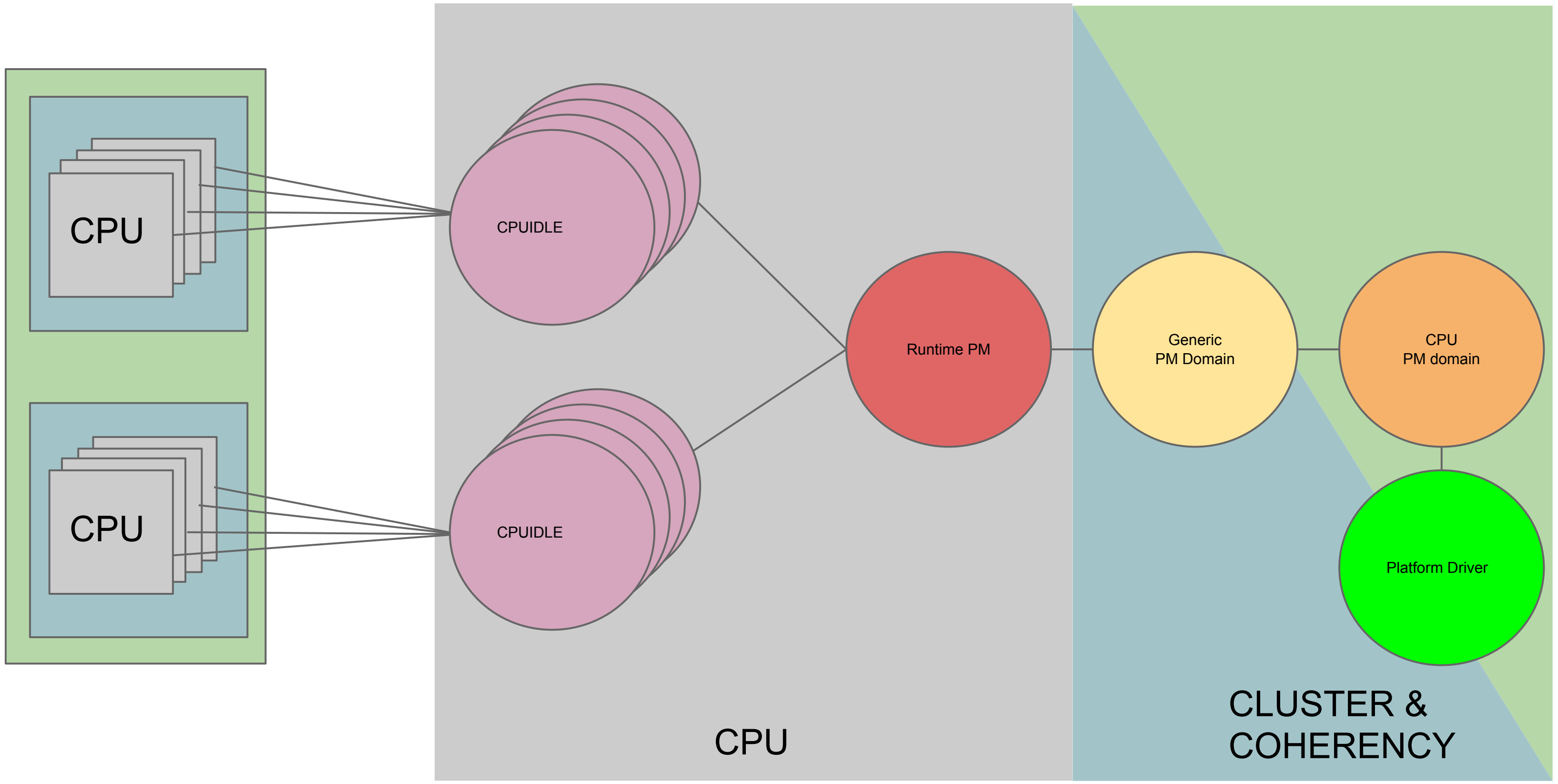


- CPUIdle manages CPUs well, but does not scale for multi-cluster SMP systems and heterogeneous systems like big.LITTLE.

The problem...


CPU
CPU
CPUIDLE
CPUIDLE
?
?
Platform hacks
CPU
CLUSTER
COHERENCY

The solution...



A unified solution to SoC idling - including CPUs 1/2

Completed:

- Infrastructure needed in the generic PM domain.

Under discussion:

- CPU PM domain.
- Deployment of runtime PM to support reference counting on the cluster.
- PSCI changes for OSI.

A unified solution to SoC idling - including CPUs 2/2

Next steps:

- Explore the approach for an ARM32 SoC.
- Deploy the unified SoC idling solution to the entire SoC topology.

Open areas:

- How to get “correct” data for the next wakeup in CPU PM domain governor?
- Is there any constraints for SCHED_DEADLINE we should consider?
- Something else?

SoC idling - next challenges

- More than one PM domain attached per device.
- Performance states of PM domains.
- Improve management of wakeup IRQs in genpd.
- Other things?

Thank you!

Ulf Hansson, Linaro

Linaro