Linaro Connect
MADRID 2024 | MAY 14-17 2024

MAD24-205 Align the upstream kernel and the userspace PM

Daniel Lezcano <daniel.lezcano@linaro.org>

Android GKI

- Goal is to reduce fragmentation
  - Let’s upstream

- Improve maintainability
  - Allow LTS fixes to be carried on
  - Reduce cost when moving to next LTS

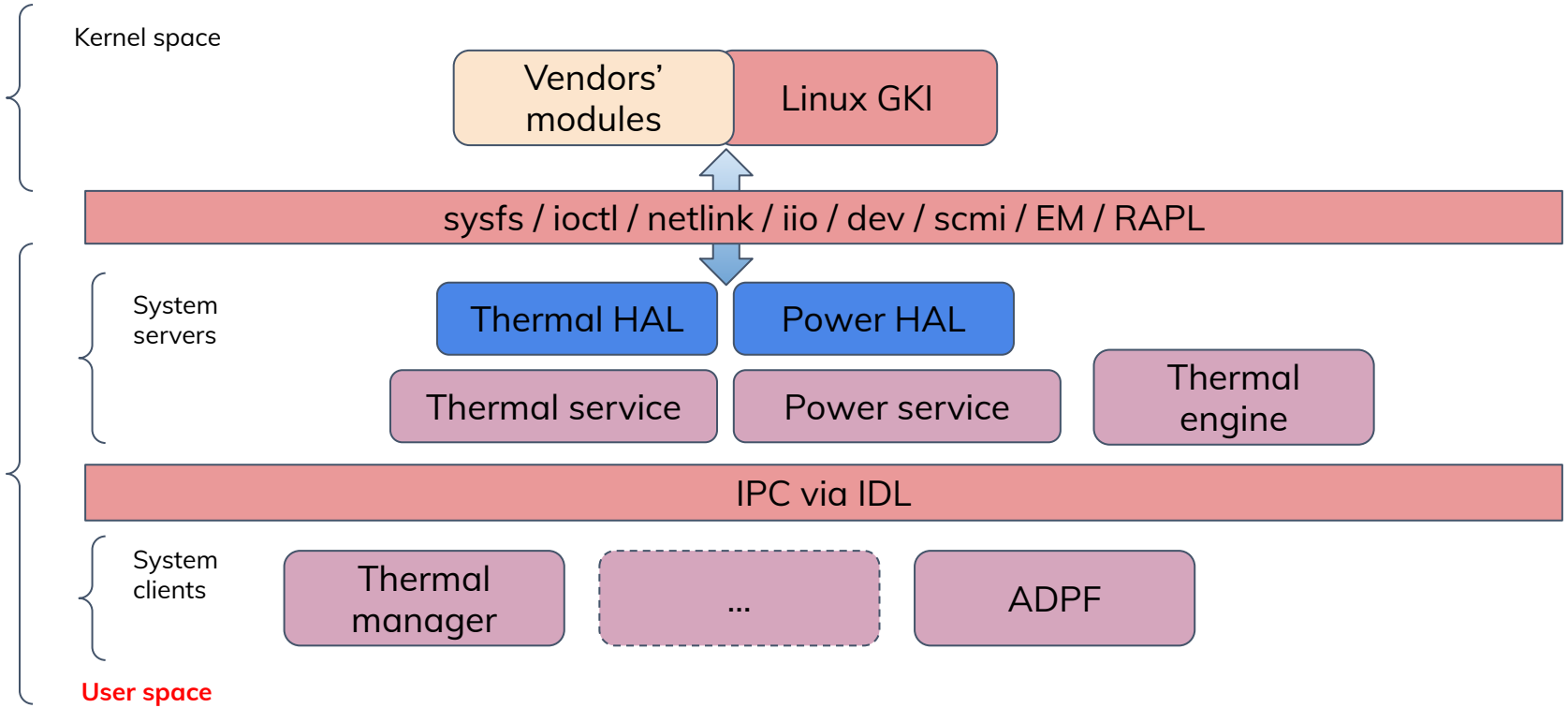
⇒ Finally SoC vendors are upstreaming the code

Vendor’s modules

- Out of tree modules implement platform power management features
- Interfaces are not normalized
- Behavior is not normalized
- Units are not normalized

⇒ Userspace has to be aligned to the interfaces

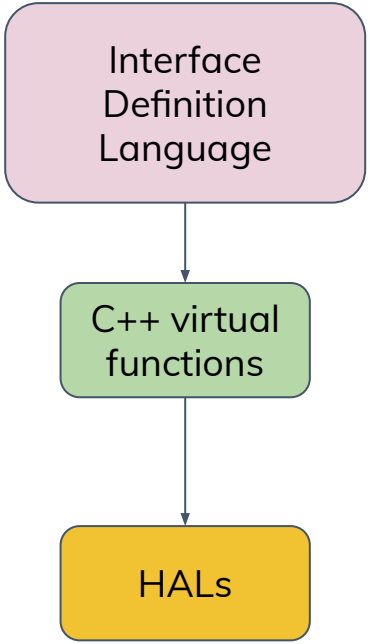
Overall components



What is “userspace” ?

- All components using the exported kernel services
  - Thermal HAL implementation
  - Power HAL implementation
- Optionally a thermal engine

Thermal / Power HALs

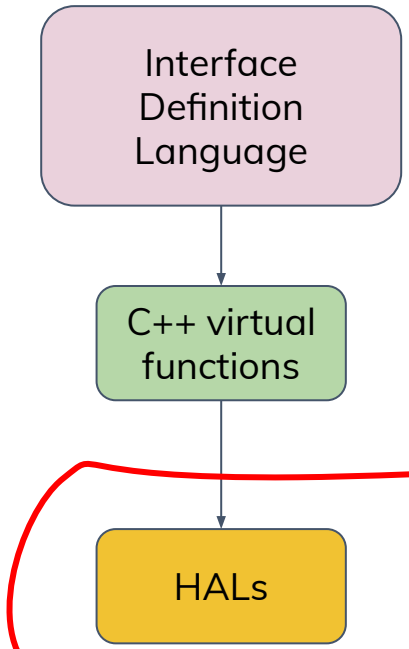



```
interface IPower {
  ...
  boolean isBoostSupported(in Boost type);
  ...
}
```

```
class IPower {
  ...
  virtual ScopedAStatus isBoostSupported(Boost in_type, bool* _aidl_return) = 0;
  ...
}
```

```
class Power : public IPower {
  ...
  ScopedAStatus isBoostSupported(Boost in_type, bool* _aidl_return)
          { <code> };
  ...
}
```

Thermal / Power HALs




```
interface IPower {
  ...
  boolean isBoostSupported(in Boost type);
  ...
}
```

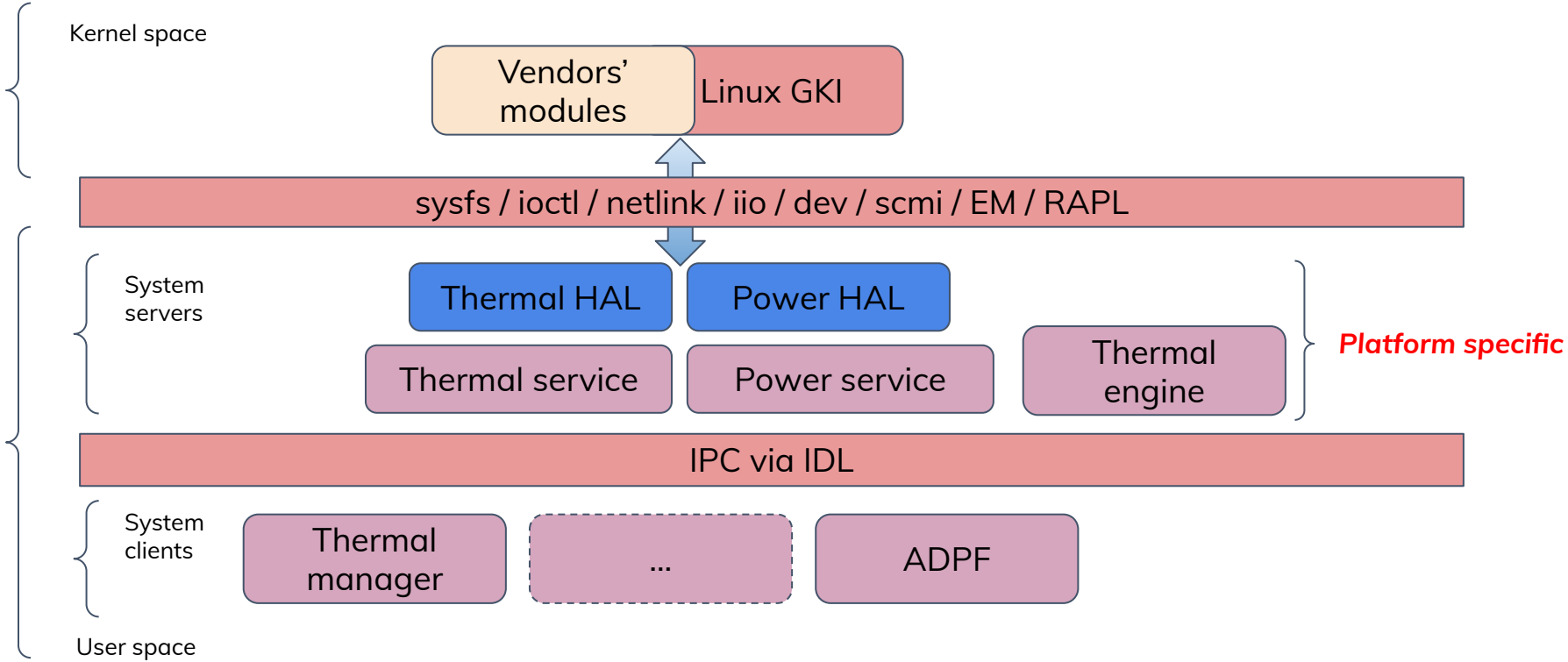
```
class IPower {
  ...
  virtual ScopedAStatus isBoostSupported(Boost in_type, bool* _aidl_return) = 0;
  ...
}
```

```
class Power : public IPower {
  ...
  ScopedAStatus isBoostSupported(Boost in_type, bool* _aidl_return)
          { <code> };
  ...
}
```

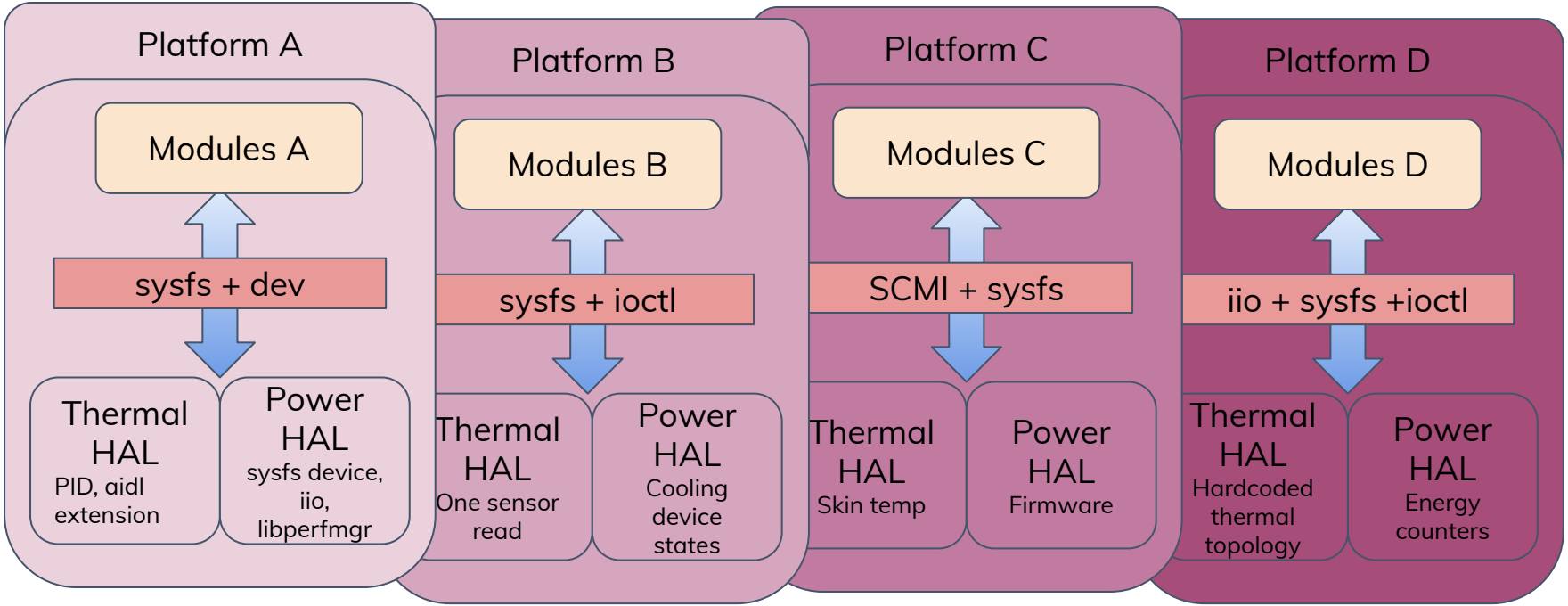
Thermal engine

- Additional logic to manage the thermal envelope of the system
- Usually stand-alone
- Contains:
  - Thermal logic (pid, decisions, ...)
  - System programing (mainloop, timers, kernel interfaces, ...)

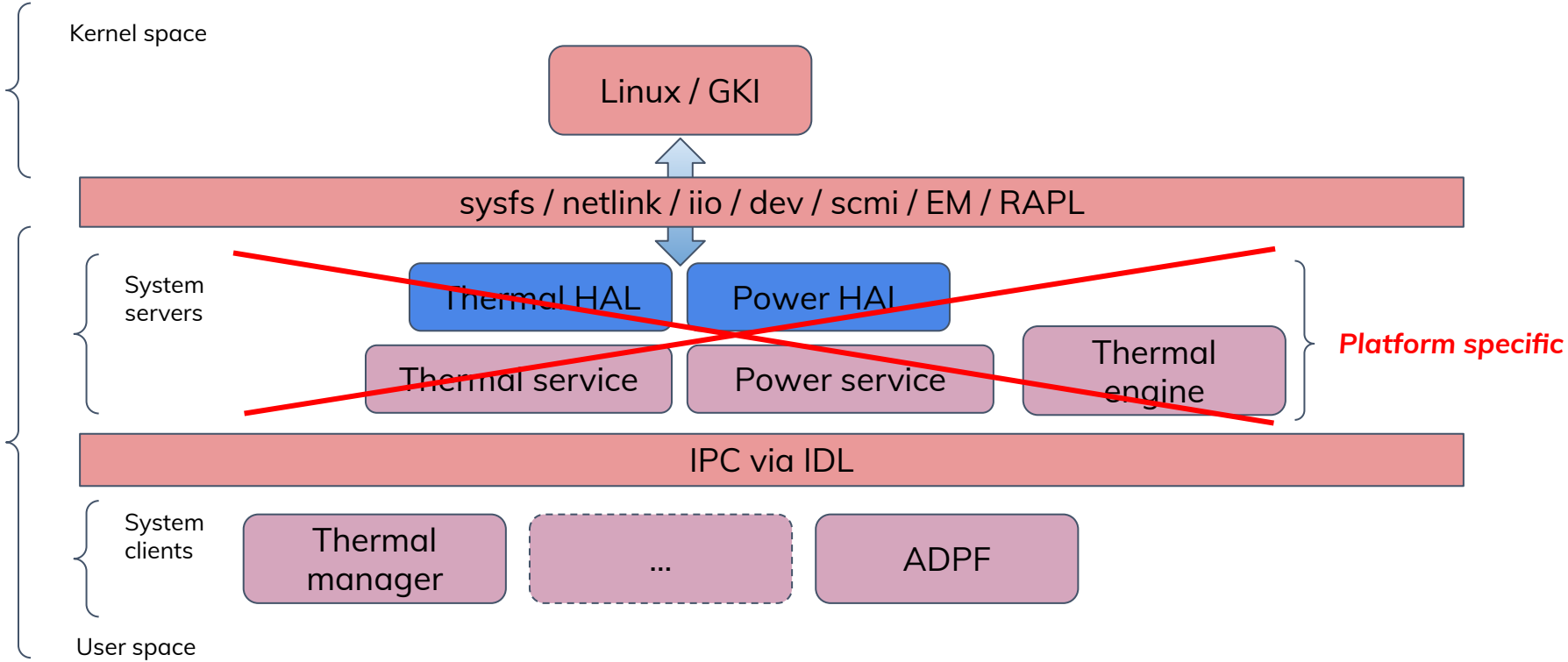
Overall components



Again and again



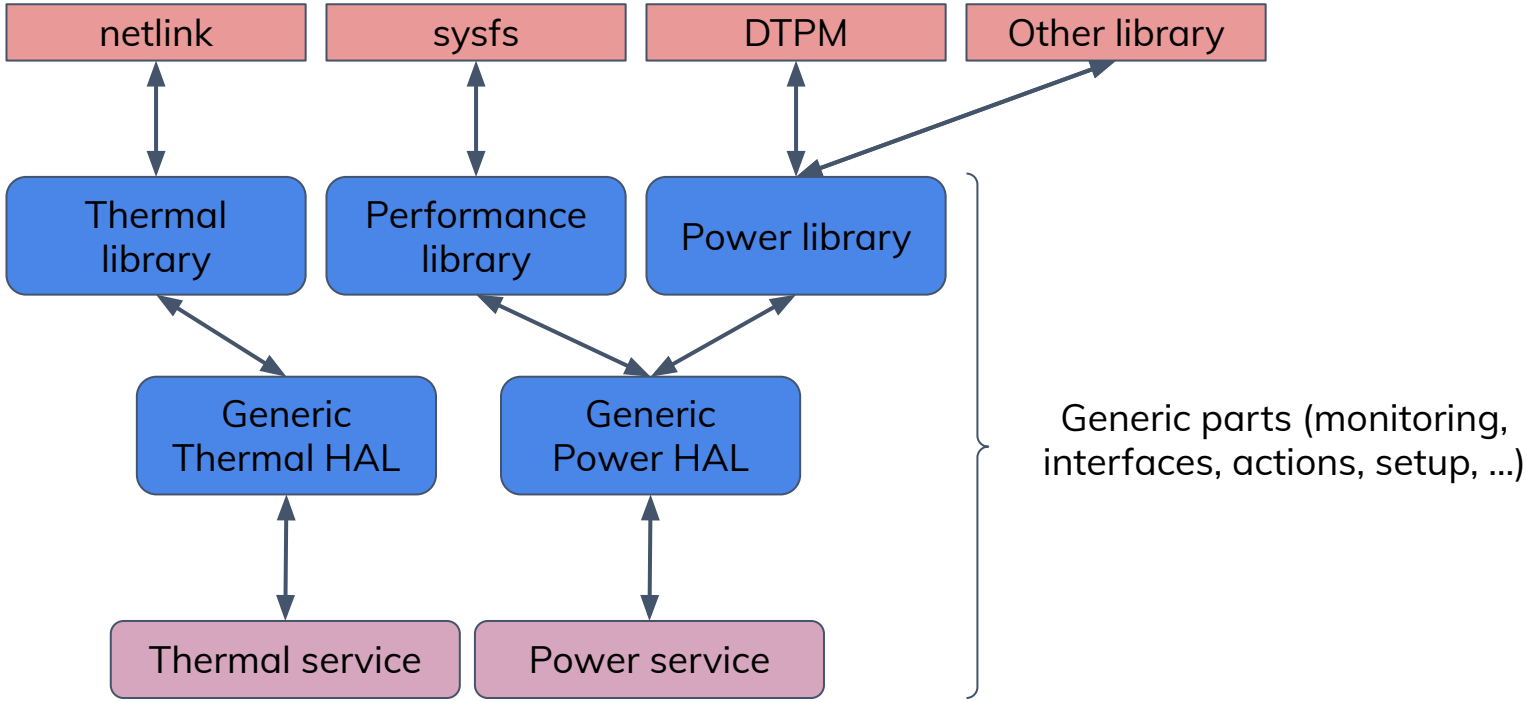
Impact of the upstreaming



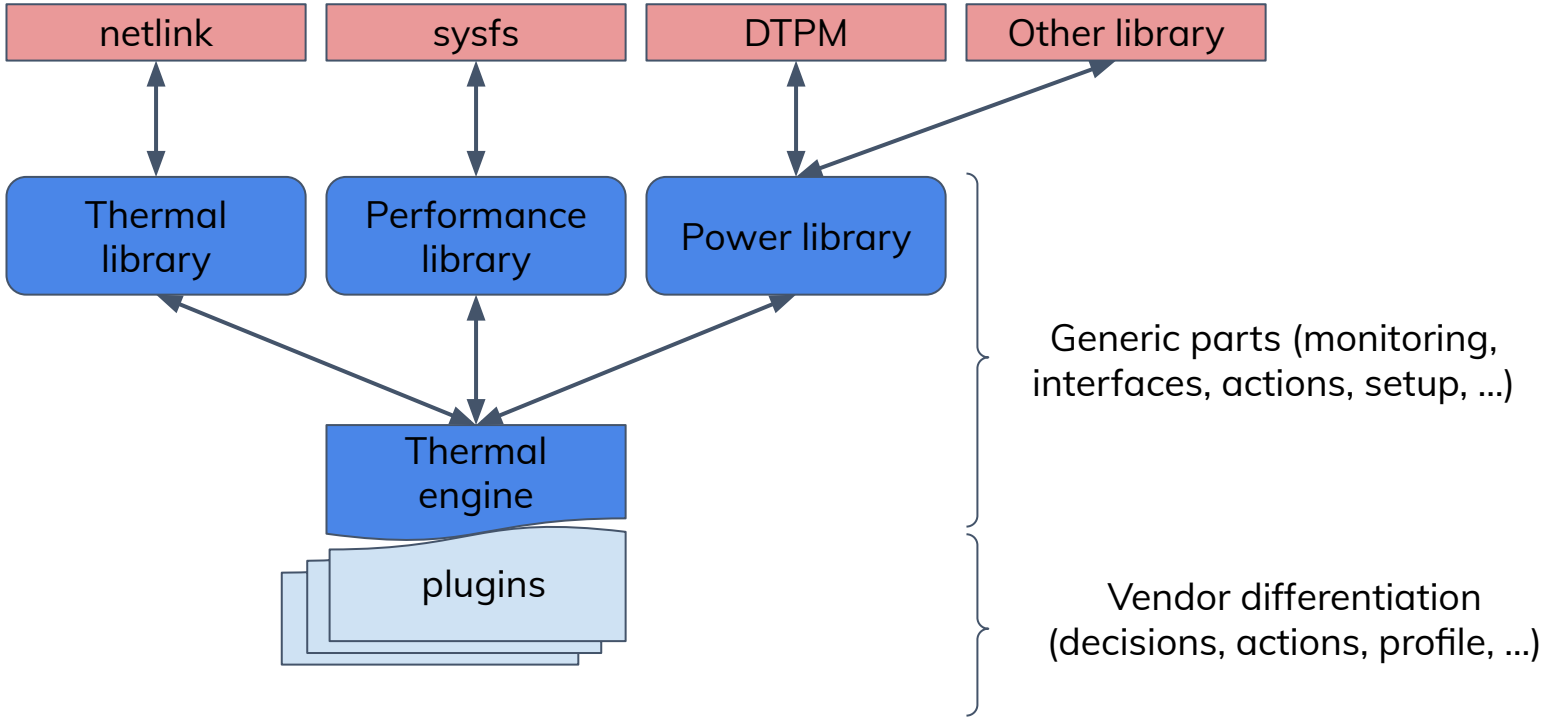
Consequences by ***not*** upstreaming

- Knowledge can be lost across LTS versions (turn over)
- Logic can be complex in the HAL and lost across versions (turn over, interface changes, ...)
- Repeating code pattern in the HAL (timer management, task period, sysfs access, temporary constraints/boosts, etc ...)
- Kernel ⇔user space exotic interfaces
    - Custom code to take into account
    - Different units (opaque values, volt, intensity, watts, joules, ...)
    - Different behavior

New components to solve these problems



Thermal engine also has a role



Work saving for each PM HAL

Feature to be implemented	Thermal HAL	Power HAL	Generic based Thermal HAL	Generic based Power HAL
Knowledge of the platform	YES	YES	NO	YES
Mainloop for events	YES	YES	NO	NO
Temporary constraints	NO	YES	NO	NO
IDL extension	OPTIONAL	OPTIONAL	OPTIONAL	OPTIONAL
Logic	NO	YES	NO	YES
Performance limits	NO	YES	NO	NO
Profile management	NO	YES	NO	NO
Workload analysis	YES	YES	NO	NO

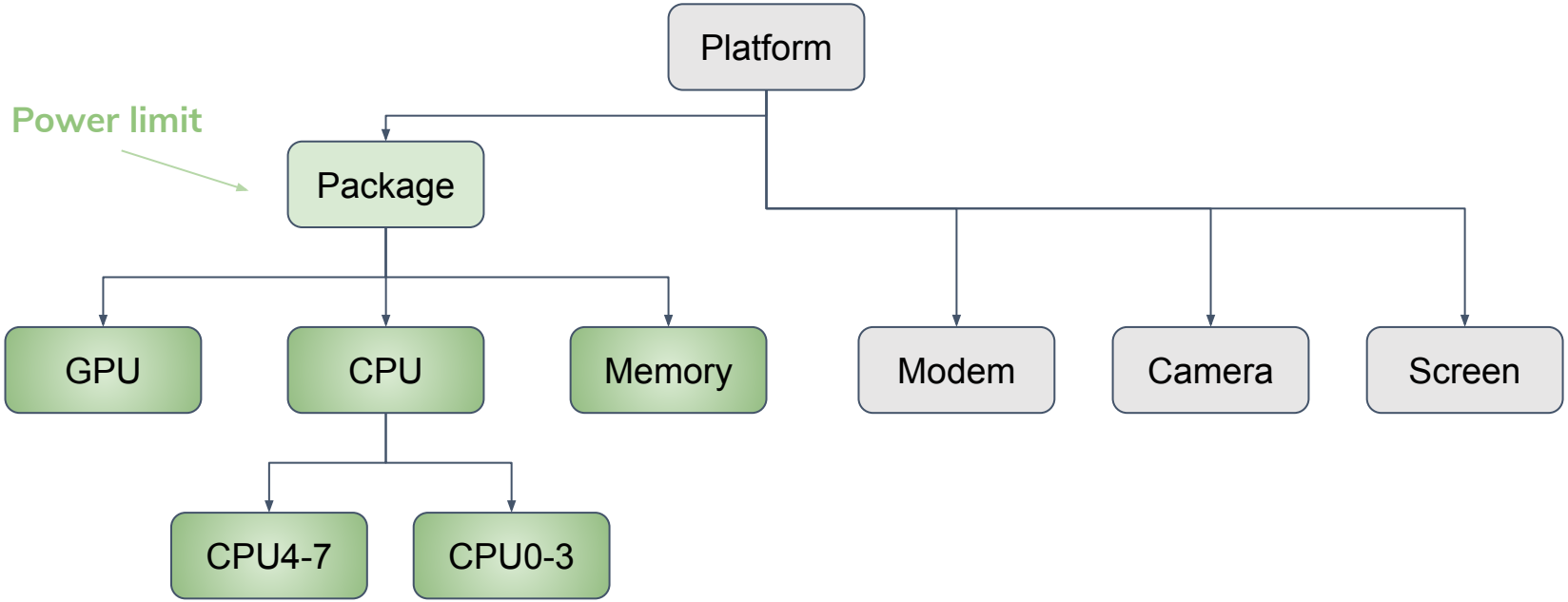
Dynamic Thermal Power Management

- Unification of the interfaces
  - Get power consumption / energy consumption
  - Set power limit

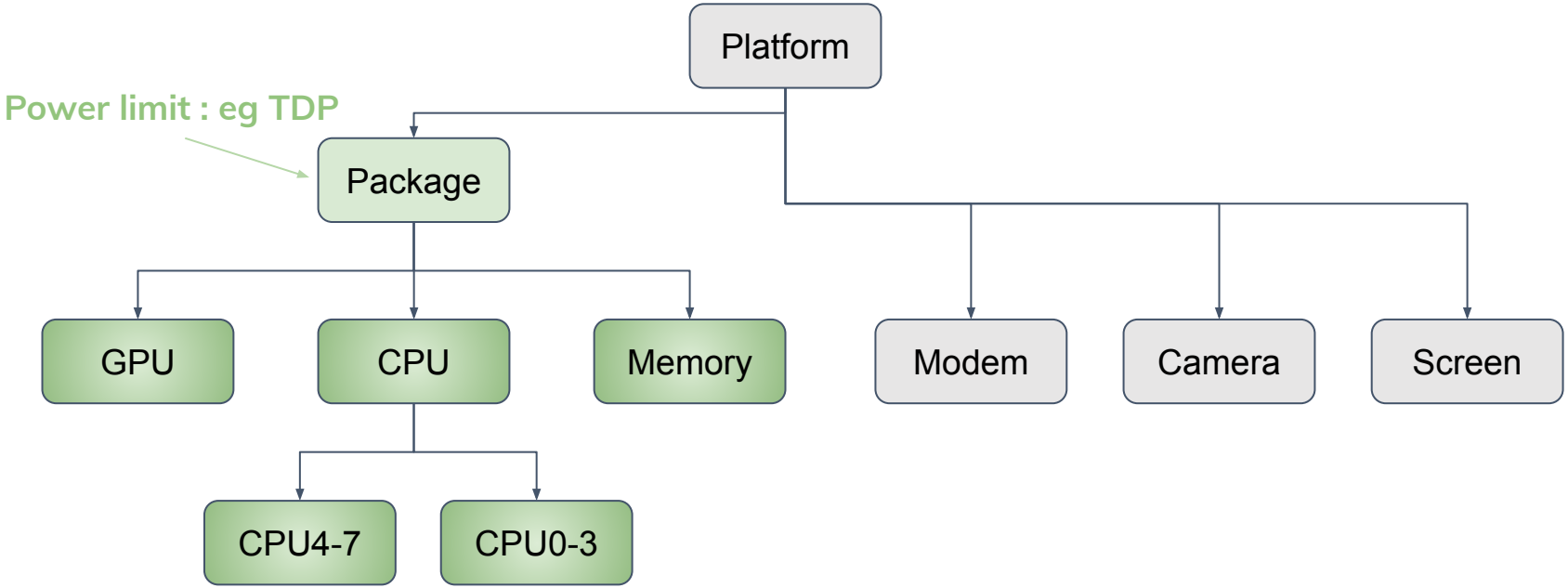
- Aggregation of the power information

- More to come if the project is supported:
    - Power budget automatic rebalancing
    - Temporary constraints

Dynamic Thermal Power Management



Dynamic Thermal Power Management



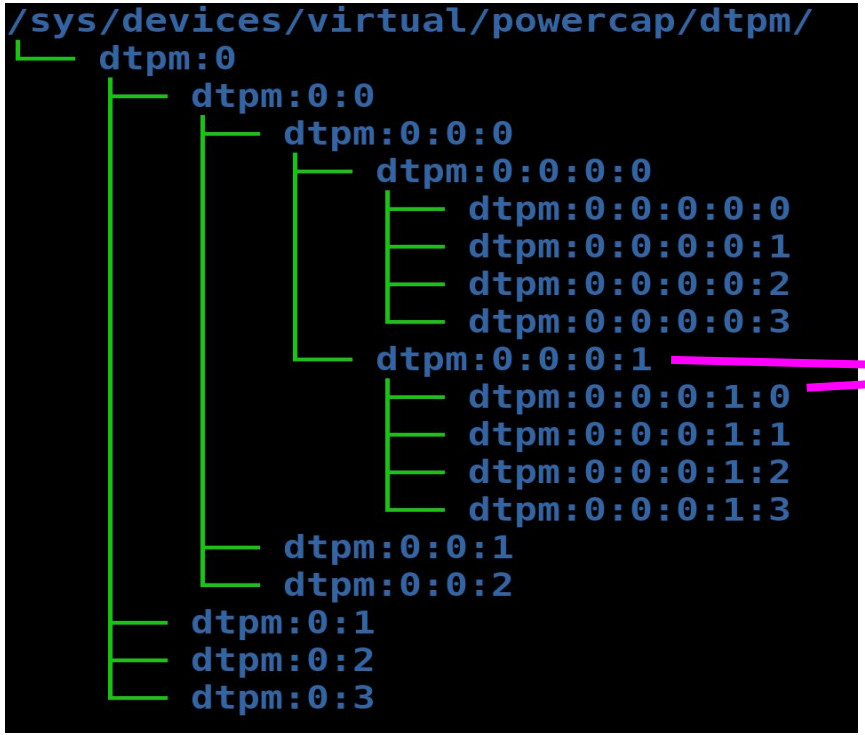
Power(GPU) + Power(CPU) + Power(Memory) <= Power(Package)

Trivial description

```
static struct dtpm_node __initdata db845c_hierarchy[] = {
        [0]{ .name = "SoC",
             .type = DTPM_NODE_VIRTUAL },
        [1]{ .name = "package",
             .type = DTPM_NODE_VIRTUAL,
             .parent = &db845c_hierarchy[0] },
        [2]{ .name = "cpu",
             .type = DTPM_NODE_VIRTUAL,
             .parent = &db845c_hierarchy[1] },
        [3]{ .name = "cluster0",
             .type = DTPM_NODE_VIRTUAL,
             .parent = &db845c_hierarchy[2] },
        [4]{ .name = "cluster1",
             .type = DTPM_NODE_VIRTUAL,
             .parent = &db845c_hierarchy[2] },
        [5]{ .name = "cpu0",
             .type = DTPM_NODE_VIRTUAL,
             .parent = &db845c_hierarchy[3] },

                [ ... ]

        [17]{ .name = "backlight",
             .type = DTPM_NODE_VIRTUAL,
             .parent = &db845c_hierarchy[0] },
        [18]{ /* sentinel */ }
};

static struct of_device_id __initdata qcom_dtpm_match_table[] = {
        { .compatible = "thundercomm,db845c", .data = db845c_hierarchy },
        {},
};

static int __init qcom_dtpm_init(void)
{
        return dtpm_create_hierarchy(qcom_dtpm_match_table);
}
module_init(qcom_dtpm_init);
```

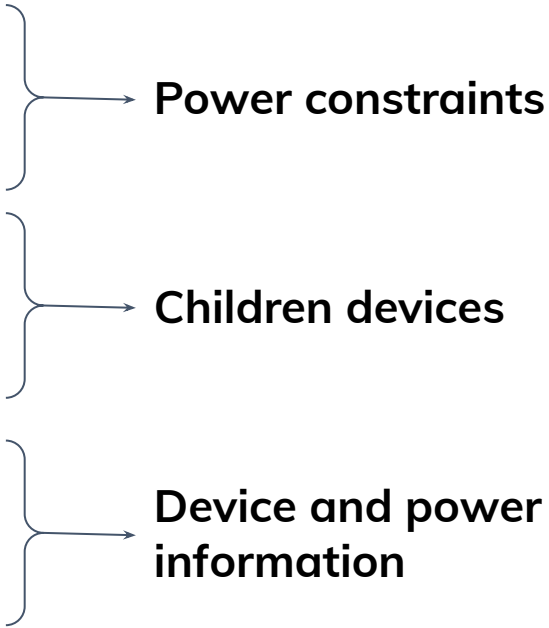
- Two types of node:
  - Virtual : for aggregation purpose
  - DT : a string pointing to the device tree path
- A flat array, easy to define
- A simple function receiving the array describing the devices hierarchy

Dynamic Thermal Power Management



Dynamic Thermal Power Management

```
constraint_0_max_power_uw
constraint_0_name
constraint_0_power_limit_uw
constraint_0_time_window_us
device -> ../../dtpm
dtpm:0:0
dtpm:0:1
dtpm:0:2
dtpm:0:3
enabled
max_power_range_uw
name
power
power_uw
subsystem -> ../../../../../class/powercap
uevent
```



Conclusion

- SoC vendors are upstreaming
  - Userspace power management no longer aligned
  - Extra work to enable the platform

- Thermal / Performance / Power libraries
  - Interface unification

- Generic Thermal HAL / Power HAL / Thermal engine
  - Reduce the amount of work to re-enable the PM in userspace

- Dynamic Thermal Power Management
  - Unified interface for power capable devices

Linaro Connect
MADRID 2024 | MAY 12-17 2024

Thank you