A simplified thermal framework for ARM platforms

Amit Daniel

Power Management Working Group, Linaro

Lead Engineer, Samsung electronics

Contributions by:

Amit Daniel Kachhap, Linaro (Samsung)
Vincent Guittot, Linaro (ST-Ericsson)
Robert Lee, Linaro (Freescale)

Why Thermal management on ARM?

- Modern System-on-Chips (SOCs) have considerable higher thermal levels than prior generations.
  - System Integration → more transistors, dense gates in the same area and more leakage.
  - Performance requirements → much higher processor frequencies and bus speeds.
  - More cores → multiple cpu core, multiple gpu core and multiple h/w accelerators.
- Cannot cool most SOCs in a traditional sense
  - Package size limitations.
  - Unavailability of heat sinks, fans, etc.

Why Thermal management on ARM? Continue...

- Knobs which can be used for cooling down SOC
  - Power gating/ clock gating the peripherals and components.
  - Performance reduction → cpu specific thermal management → frequency reduction, longer cpu idle states.
  - P =K * V^2 *I, so voltage reduction of the soc components, battery supply etc important.

Existing kernel Thermal Framework

- Very good definition and basic abstraction concepts (Documentation/thermal/sysfs-api.txt).
- Concepts of thermal zones, trip points and cooling devices.
- Framework to register thermal zone and cooling devices.
- Performs a routing function of generic cooling devices to generic thermal zones with the help of very simple thermal management logic.
-
    –
-

Existing kernel Thermal Framework cont...

- Good userspace hooks and pointers of thermal zone attributes and cooling devices through sysfs.
- Many cooling devices such as processor, LCD etc abstracted inside ACPI specification layer.

Linaro

Enhancement in Thermal Framework

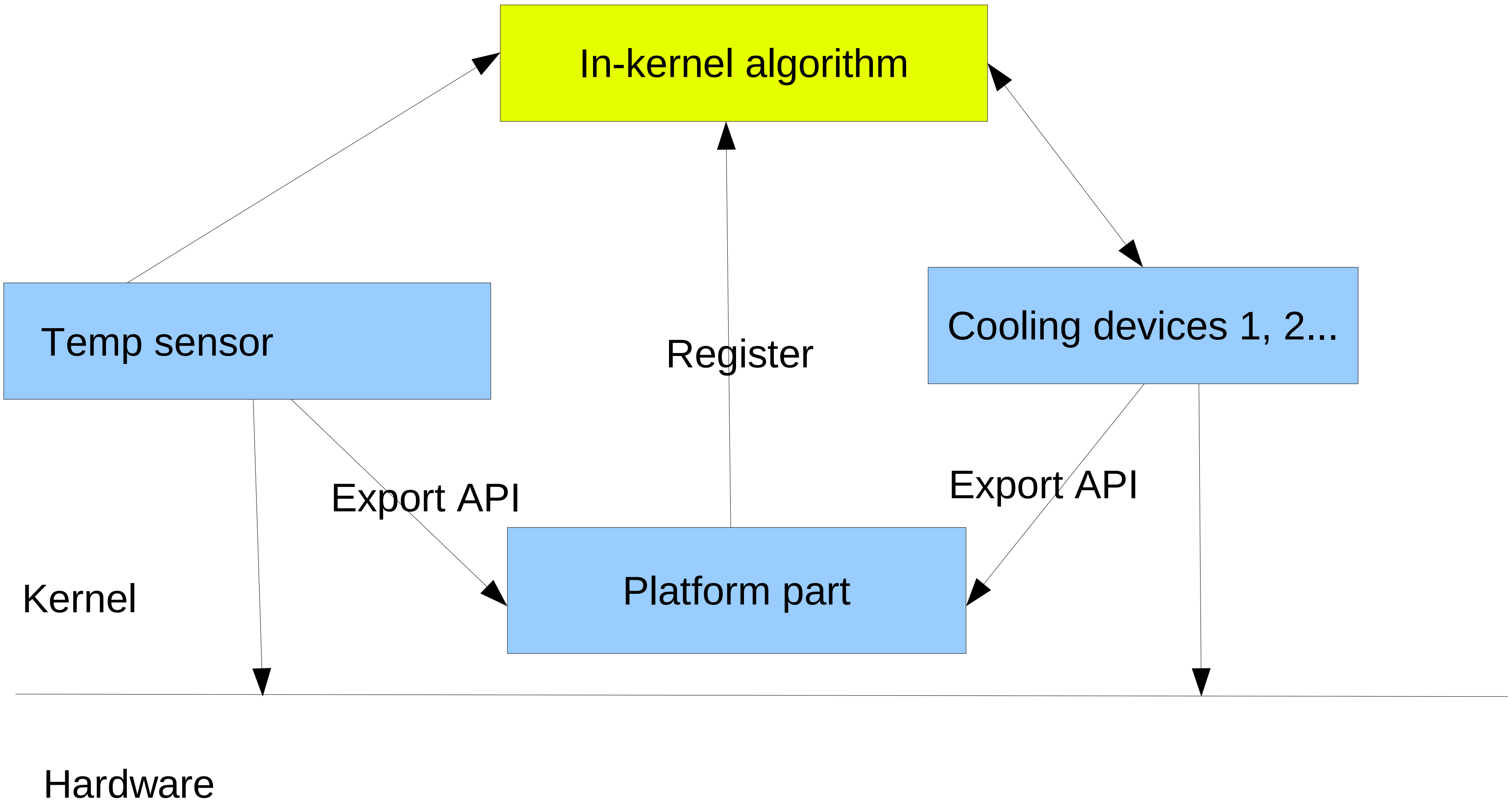
- The in-kernel thermal algorithm is mostly polling and need some modification.
- The cooling devices binded initially to a trip point so some cooling statistics/heuristics may be useful.
- To verify that cooling devices cool properly.
- Provision to add cooling devices dynamically.
- Some ways to use the cooling devices low level handlers(cpufreq, cpuidle, cpu throttling etc) in a generic way.
- The current in-kernel thermal algorithm should lock on a trip point for some cases.

Work implemented till now

In-kernel framework used for thermal solution

- Platform specific temperature sensor driver to export temperature information in necessary format.
- Thermal zone and cooling device binding separated into architecture and non architecture parts.
- Currently they are placed inside driver/thermal/ directory.
- Non architecture specific cooling devices in further slides.
- Tested for samsung and freescale platforms.

Linaro



THERMAL FLOW DIAGRAM

Generic cpu cooling devices

- Patches submitted for generic cpufreq and cpuhotplug cooling devices.
- They are nothing but a simple wrapper and provides a registration/un-registration api's.
- These api's provides cooling device pointers which can be mapped to a trip points as required.
- All these api's multi-instance in behaviour.
- Currently they are placed inside driver/thermal/ directory.
- The link to the patches are https://lkml.org/lkml/2011/12/13/188

Linaro

Generic cpu cooling devices continue...

- The api signatures are,
  - struct thermal_cooling_device *cpufreq_cooling_register(struct freq_pctg_table *tab_ptr, unsigned int tab_size, const struct cpumask *mask_val)
  - void cpufreq_cooling_unregister(struct thermal_cooling_device *cdev)
  - struct thermal_cooling_device *cpuhotplug_cooling_register(const struct cpumask *mask_val)
  - void cpuhotplug_cooling_unregister(struct thermal_cooling_device *cdev)

Generic cpu cooling devices continue...

```
static struct exynos4_tmu_platform_data default_tmu_data = {
        .threshold = 80,
        .trigger_levels[0] = 2,
        .trigger_levels[1] = 5,
        .trigger_levels[2] = 10,
        .trigger_levels[3] = 30,
        .trigger_level0_en = 1,
        .trigger_level1_en = 1,
        .trigger_level2_en = 1,
        .trigger_level3_en = 1,
        .gain = 15,
        .reference_voltage = 7,
        .cal_type = TYPE_ONE_POINT_TRIMMING,
        .freq_tab[0] = {
                .freq_clip_pctg = 30,
                },
        .freq_tab[1] = {
                .freq_clip_pctg = 99,
                },
        .freq_tab_count = 2,
};
```

Support to report cooling statistics

- Add a sysfs node to report cooling achieved by all cooling devices on a single trip points.
- The cooling data reported will be,
  - Absolute if higher temperature trip points are arranged first.
  - Cumulative of the earlier invoked cooling handlers.
- The statistics reported will be fairly correct if the cooling devices added brings down the temperature in a symmetric manner.
- The link to the patches are

  https://lkml.org/lkml/2012/1/18/69

Linaro

Support to report cooling statistics continue….

- The statistics reported looks like

  cat /sys/class/thermal/thermal_zone0/trip_stats

  0 0

  1 11000

  2 5000

- Here, trip point 1 produces a temperature drop of 11 degree C.
- Trip point 2 reports a temperature drop of 5 degree C.
- Clearly trip point 0 threshold is never reached.

Creating a new trip type

- A new trip type created (STATE_ACTIVE).
- This trip combines the benefit of of trip type ACTIVE and PASSIVE into one.
- This is useful for a type of cooling devices which is registered only once but can map it's different cooling state to different trip points.
- The link to the patches are https://lkml.org/lkml/2011/12/13/187

Future work

- Currently the number of trip points fixed to 12. Making it dynamic will be helpful.
- Enhance devfreq driver to expose policy constraints.
- Adding some more trip types which will lock in a trip points.
- More cooling devices with a generic wrapper around them.
- Some PMQOS hooks/notifications in the thermal management.
- Moving the generic code in drivers/acpi ??

THANKS

Q/A ?