Thermal developer guide

Release version: 1.0

Author email: finley.xiao@rock-chips.com

Date: 2019.01.22

Security Classification: Public

Preface

Overview

This document mainly describes the related concept, configuration method and user mode interface of thermal.

Product version

Chipset name	Kernel version
All chipsets	Linux4.4

Application object

Software development engineers

Field application engineers

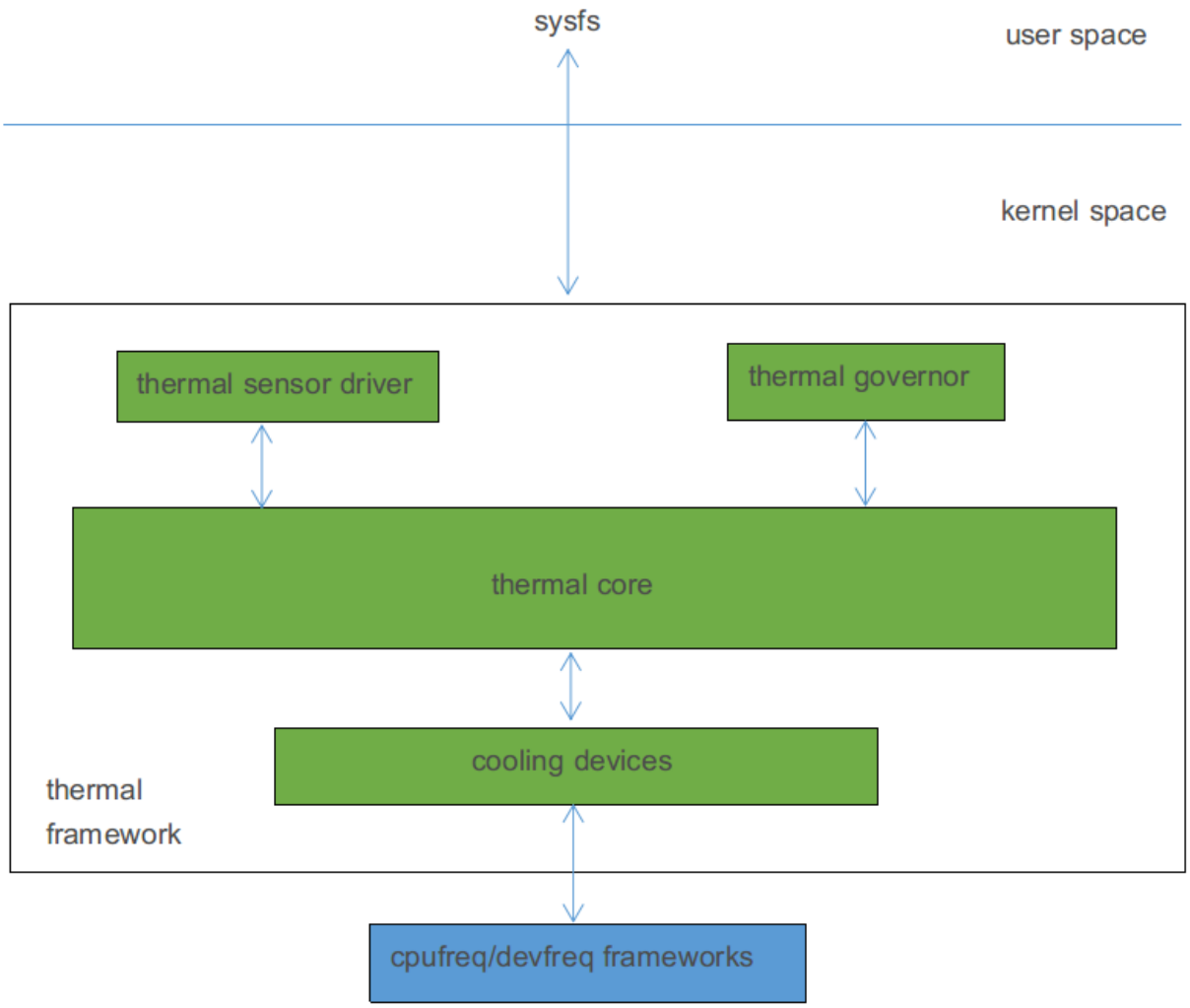
Revision history

Date	Version	Author	Revision description
2019-01-22	V1.0	Finley Xiao	The initial version

5.2 Acquire Current Temperature

1 Overview

Thermal is a framework model defined by kernel developers supporting to control the system temperature according to the specific governor in order to prevent the chipset from overheating. Thermal framework consists of governor, core, cooling device and sensor driver. The software architecture is as below:



Thermal governor: used to decide whether the cooling device needs to decrease frequency, and decrease to what extent. Currently Linux4.4 kernel includes several kinds of governor as below:

- power_allocator: Introduce PID (Proportion-Integral-Differential) control to dynamically allocate power for the cooling devices according to current temperature and convert power into frequency so as to achieve the effect of limiting the frequency through temperature.
- step_wise: Decrease the frequency of cooling device step by step according to current temperature.
- fair share: Prefer to decrease the frequency of cooling device with more frequencies.
- userspace: Only notify user space about thermal events.

Thermal core: Package and abstract the thermal governors and thermal driver and define the clear interface.

Thermal sensor driver: sensor driver, used to acquire temperature, such as tsadc.

Thermal cooling device: heating source or cooling device, such as CPU, GPU, DDR etc.

2 Code Path

Governor related code:

```
drivers/thermal/power_allocator.c      /* power allocator governor */
drivers/thermal/step_wise.c            /* step wise governor */
drivers/thermal/fair_share.c           /* fair share governor */
drivers/thermal/user_space.c           /* userspace governor */
```

Cooling device related code:

```
drivers/thermal/devfreq_cooling.c
drivers/thermal/cpu_cooling.c
```

Core related code:

```
drivers/thermal/thermal_core.c
```

Driver related code:

```
drivers/thermal/rockchip_thermal.c    /* all platforms tsadc driver except RK3368*/
drivers/thermal/rk3368_thermal.c      /* tsadc driver for RK3368 */
```

3 Configuration Method

3.1 Menuconfig Configuration

```
<*> Generic Thermal sysfs driver  --->
    --- Generic Thermal sysfs driver
    [*]   APIs to parse thermal data out of device tree
    [*]   Enable writable trip points
          Default Thermal governor (power_allocator)  --->  /* default thermal governor
*/
    [ ]   Fair-share thermal governor
    [ ]   Step_wise thermal governor                    /* step_wise governor */
    [ ]   Bang Bang thermal governor
    [*]   User_space thermal governor                   /* user_space governor */
    -*-   Power allocator thermal governor              /* power_allocator governor
*/
    [*]   generic cpu cooling support                   /* cooling device */
    [ ]   Generic clock cooling support
    [*]   Generic device cooling support                /* cooling device */
    [ ]   Thermal emulation mode support
    < >   Temperature sensor driver for Freescale i.MX SoCs
    <*>   Rockchip thermal driver                       /* thermal sensor driver */
```

```
    < >     rk_virtual thermal driver
    <*>     rk3368 thermal driver legacy                /* thermal sensor driver */
```

It is able to select thermal governor through "Default Thermal governor" configuration option. Developers can change it according to the actual product requirement.

3.2 Tsadc Configuration

Tsadc as thermal sensor in the thermal control is used to acquire temperature. Usually need to do the configuration in both DTSI and DTS.

Take RK3399 as an example, DTSI includes below configuration:

```
tsadc: tsadc@ff260000 {
    compatible = "rockchip,rk3399-tsadc";
    /* the basic address of register and the total length of register address */
    reg = <0x0 0xff260000 0x0 0x100>;
    /* interrupt number and interrupt trigger method */
    interrupts = <GIC_SPI 97 IRQ_TYPE_LEVEL_HIGH 0>;
    /* working clock, 750KHz*/
    assigned-clocks = <&cru SCLK_TSADC>;
    assigned-clock-rates = <750000>;
    /* working clock and configuration clock */
    clocks = <&cru SCLK_TSADC>, <&cru PCLK_TSADC>;
    clock-names = "tsadc", "apb_pclk";
    /* reset signal*/
    resets = <&cru SRST_TSADC>;
    reset-names = "tsadc-apb";
    /* invoke grf module, some platforms need*/
    rockchip,grf = <&grf>;
    /* the threshold temperature of reboot, 120 degree*/
    rockchip,hw-tshut-temp = <120000>;
    /* tsadc output pin configuration, support two modes*/
    pinctrl-names = "gpio", "otpout";
    pinctrl-0 = <&otp_gpio>;
    pinctrl-1 = <&otp_out>;
    /*
     * thermal sensor symble, means tsadc can be as a thermal sensor,
     * and specify how many parameters are needed when invoking tsadc node,
     * if SoC only has one tsadc, can be set as 0, if more than one,
     * it must be set as 1.
     */
    #thermal-sensor-cells = <1>;
    status = "disabled";
};

/* IO port configuration*/
pinctrl: pinctrl {
    ...
    tsadc {
        /* configure it to be gpio mode*/
        otp_gpio: otp-gpio {
            rockchip,pins = <1 6 RK_FUNC_GPIO &pcfg_pull_none>;
```

```
        };
        /* configure it to be over temperature protection mode*/
        otp_out: otp-out {
            rockchip,pins = <1 6 RK_FUNC_1 &pcfg_pull_none>;
        };
    };
    ....
}
```

DTS configuration is mainly used to select CRU reset or GPIO reset, low voltage reset or high voltage reset. Need to pay attention to that, if want to configure as GPIO reset, hardware needs to connect tsadc output pin to PMIC reset pin, otherwise it only can be configured as CRU reset.

```
&tsadc {
    rockchip,hw-tshut-mode = <1>;     /* tshut mode 0:CRU 1:GPIO */
    rockchip,hw-tshut-polarity = <1>; /* tshut polarity 0:LOW 1:HIGH */
    status = "okay";
};
```

Refer to the document "Documentation/devicetree/bindings/thermal/rockchip-thermal.txt".

3.3 Power Allocator Governor Configuration

Power allocator thermal governor introduces PID (Proportion-Integral-Differential) control to dynamically allocate power for cooling devices according to current temperature. When the temperature is low, the allocatable power is relatively large, that is, the operating frequency is high. As the temperature rises, the allocatable power gradually decreases and the operating frequency also gradually decreases, so as to limit the frequency according to the temperature.

3.3.1 CPU Configuration

CPU as cooling device in thermal control needs to include "#cooling-cells", "dynamic-power-coefficient" attributes in the node.

Take RK3399 as an example:

```
cpu_l0: cpu@0 {
    device_type = "cpu";
    compatible = "arm,cortex-a53", "arm,armv8";
    reg = <0x0 0x0>;
    enable-method = "psci";
    /* cooling device symbol, means the device can be as a cooling device */
    #cooling-cells = <2>;
    clocks = <&cru ARMCLKL>;
    cpu-idle-states = <&CPU_SLEEP &CLUSTER_SLEEP>;
     /*
      * dynamic power consumption constant C,
      * dynamic power consumption formula is Pdyn=C*V^2*F
      */
    dynamic-power-coefficient = <100>;
};
...
```

```
cpu_b0: cpu@100 {
    device_type = "cpu";
    compatible = "arm,cortex-a72", "arm,armv8";
    reg = <0x0 0x100>;
    enable-method = "psci";
    /* cooling device symbol, means the device can be as a cooling device */
    #cooling-cells = <2>;
    clocks = <&cru ARMCLKB>;
    cpu-idle-states = <&CPU_SLEEP &CLUSTER_SLEEP>;
    /* the parameter is used to compute the dynamic power consumption */
    dynamic-power-coefficient = <436>;
};
```

3.3.2 GPU Configuration

GPU as cooling device in thermal control needs to include "#cooling-cells" attribute in the node and power_model subnode.

Take RK3399 as an example:

```
gpu: gpu@ff9a0000 {
    compatible = "arm,malit860",
    "arm,malit86x",
    "arm,malit8xx",
    "arm,mali-midgard";

    reg = <0x0 0xff9a0000 0x0 0x10000>;

    interrupts = <GIC_SPI 19 IRQ_TYPE_LEVEL_HIGH 0>,
    <GIC_SPI 20 IRQ_TYPE_LEVEL_HIGH 0>,
    <GIC_SPI 21 IRQ_TYPE_LEVEL_HIGH 0>;
    interrupt-names = "GPU", "JOB", "MMU";

    clocks = <&cru ACLK_GPU>;
    clock-names = "clk_mali";
    /* cooling device symbol, means the device can be as a cooling device */
    #cooling-cells = <2>;
    power-domains = <&power RK3399_PD_GPU>;
    power-off-delay-ms = <200>;
    status = "disabled";

    gpu_power_model: power_model {
        compatible = "arm,mali-simple-power-model";
        /* the parameter used to compute the static power consumption */
        static-coefficient = <411000>;
        /* the parameter used to compute the dynamic power consumption */
        dynamic-coefficient = <733>;
        /* the parameter used to compute the static power consumption */
        ts = <32000 4700 (-80) 2>;
         /*
          * the temperature acquired from gpu-thermal,
          * used to compute the static power consumption
          */
```

```
        thermal-zone = "gpu-thermal";
    };
};
```

3.3.3 Thermal Zone Configuration

Thermal zone node is mainly used to configure the related parameters of thermal governor and generate the corresponding user mode interface.

Take RK3399 as an example:

```
thermal_zones: thermal-zones {
    /*
     * one node corresponds to one thermal zone,
     * and include the related parameters of thermal governor
     */
    soc_thermal: soc-thermal {
        /*
         * when the temperature is higher than the trip-point-0,
         * acquire the temperature every 20ms
         */
        polling-delay-passive = <20>; /* milliseconds */
        /*
         * when the temperature is lower than the trip-point-0,
         * acquire the temperature every 1000ms
         */
        polling-delay = <1000>; /* milliseconds */
        /*
         * the total power allocated to cooling device,
         * when the temperature is equal to the trip-point-1
         */
        sustainable-power = <1000>; /* milliwatts */
        /* current thermal zone acquired through tsadc0 */
        thermal-sensors = <&tsadc 0>;

        /*
         * trips node includes different temperature threshod.
         * different thermal governor need different configuration
         */
        trips {
            /*
             * thermal control threshold, thermal governor starts to work when the
             * temperature is over this value, but may not limit the frequency at
once,
             * start to limit the frequency only when power is small enough
             */
            threshold: trip-point-0 {
                /*
                 * thermal control governor starts to work when temperature is over
70
                 * degree, and 70 degree is also a threshold for tsadc to trigger the
                 * interrupt
                 */
```

```
                temperature = <70000>; /* millicelsius */
                 /* unuseful, but need to fill as required by framework*/
                hysteresis = <2000>; /* millicelsius */
                /*
                 * indicate use polling-delay-passive time to acquire
                 * when temperature over trip-point-0
                 */
                type = "passive";
            };
            /*
             * thermal target temperature, expect the chipset not to exceed
             * the value by decreasing the frequency
             */
            target: trip-point-1 {
                /*
                 * expect the chipset not to exceed 85 degree by decreasing the
                 * frequency, and 85 degree is also a threshold for tsadc to trigger
                 * the interrupt
                 */
                temperature = <85000>; /* millicelsius */
                 /* unuseful, but need to fill as required by framework*/
                hysteresis = <2000>; /* millicelsius */
                /*
                 * indicate use polling-delay-passive time to acquire
                 * when temperature over trip-point-1
                 */
                type = "passive";
            };
            /*
             * overtemperature protection threshold, if the temperature is still
rising
             * after the frequency is decreased, when the temperature is over the
value,
             * the system will reboot
             */
            soc_crit: soc-crit {
                /*
                 * reboot when over 115 degree, and 115 degree is also a threshold
for
                 * tsadc to trigger the interrupt
                 */
                temperature = <115000>; /* millicelsius */
                 /* unuseful, but need to fill as required by framework*/
                hysteresis = <2000>; /* millicelsius */
                /* reboot when temperature is over soc-crit */
                type = "critical";
            };
        };

        /*
         * cooling device configuration node,
         * each subnode represents a cooling device
         */
```

```
        cooling-maps {
            map0 {
                /*
                 * indicate the cooling device only works in target trip,
                 * it must be target for power allocater governor
                 */
                trip = <&target>;
                /*
                 * A53 is a cooling device, THERMAL_NO_LIMIT is unuseful,
                 * but must be filled in
                 */
                cooling-device =
                    <&cpu_l0 THERMAL_NO_LIMIT THERMAL_NO_LIMIT>;
                /*
                 * when computing the power consumption multiply it by 4096/1024 times,
                 * used to adjust the sequence and extent while decreasing the
                 * frequency
                 */
                contribution = <4096>;
            };
            map1 {
                /*
                 * indicate the cooling device only works in target trip,
                 * it must be target for power allocater governor
                 */
                trip = <&target>;
                /*
                 * A72 is a cooling device, THERMAL_NO_LIMIT is unuseful,
                 * but must be filled in
                 */
                cooling-device =
                    <&cpu_b0 THERMAL_NO_LIMIT THERMAL_NO_LIMIT>;
                /*
                 * when computing the power consumption multiply it by 1024/1024 times,
                 * used to adjust the sequence and extent while decreasing the
                 * frequency
                 */
                contribution = <1024>;
            };
            map2 {
                /*
                 * indicate the cooling device only works in target trip,
                 * it must be target for power allocater governor
                 */
                trip = <&target>;
                /*
                 * GPU is a cooling device, THERMAL_NO_LIMIT is unuseful,
                 * but must be filled in
                 */
                cooling-device =
                    <&gpu THERMAL_NO_LIMIT THERMAL_NO_LIMIT>;
```

```
                /*
                 * when computing the power consumption multiply it by 4096/1024
times,
                 * used to adjust the sequence and extent while decreasing the
                 * frequency
                 */
                contribution = <4096>;
            };
        };
    };
    /*
     * one node corresponds to one thermal zone, and includes related parameters
     * of thermal governor, but current thermal zone is only used to acquire the
     * temperature
     */
    gpu_thermal: gpu-thermal {
        /*
         * it is usefull only when add thermal governor configuration,
         * and must be filled in as required by framework
         */
        polling-delay-passive = <100>; /* milliseconds */
        /* acquire the temperature every 1000ms */
        polling-delay = <1000>; /* milliseconds */

        /* current thermal zone acquires temperature through tsadc1 */
        thermal-sensors = <&tsadc 1>;
    };
};
```

Refer to the document "Documentation/devicetree/bindings/thermal/thermal.txt"、"Documentation/thermal/power_allocator.txt".

3.3.4 Thermal Parameter Adjustment

Some parameters are related with the chipset and usually no need to change. Some parameters need to be adjusted according to the actual product requirement. Generally you can do as below steps:

1. Confirm the target temperature.

Assume that we want the thermal control starts to work when the temperature is over 70 degree (acquire the temperature more frequently), the highest temperature not to exceed 85 degree, and reboot the system when over 115 degree. Then we can do below configuration:

```
thermal_zones: thermal-zones {
    soc_thermal: soc-thermal {
        ....
        trips {
            threshold: trip-point-0 {
                /*
                 * thermal control starts to work when over 70 degree,
                 * decrease the time interval to acquire the temperature,
                 * but not decrease the frequency at once, also related
                 * with sustainable-power
```

```
                 */
                temperature = <70000>; /* millicelsius */
                hysteresis = <2000>; /* millicelsius */
                type = "passive";
            };
            target: trip-point-1 {
                /* expect the highest temperature not to exceed 85 degree */
                temperature = <85000>; /* millicelsius */
                hysteresis = <2000>; /* millicelsius */
                type = "passive";
            };
            soc_crit: soc-crit {
                /* reboot the system when over 115 degree*/
                temperature = <115000>; /* millicelsius */
                hysteresis = <2000>; /* millicelsius */
                type = "critical";
            };
        };
        ...
    }
};
```

2. Confirm the cooling device.

Take RK3399 as an example, some products need to use CPU and GPU, we can do below configuration:

```
thermal_zones: thermal-zones {
    soc_thermal: soc-thermal {
        ...
        /* A53, A72 and GPU three modules are all as cooling device */
        cooling-maps {
            map0 {
                trip = <&target>;
                cooling-device =
                    <&cpu_l0 THERMAL_NO_LIMIT THERMAL_NO_LIMIT>;
                contribution = <4096>;
            };
            map1 {
                trip = <&target>;
                cooling-device =
                    <&cpu_b0 THERMAL_NO_LIMIT THERMAL_NO_LIMIT>;
                contribution = <1024>;
            };
            map2 {
                trip = <&target>;
                cooling-device =
                    <&gpu THERMAL_NO_LIMIT THERMAL_NO_LIMIT>;
                contribution = <4096>;
            };
        };
        ...
    };
};
```

Some products only use CPU, we can do below configuration:

```
thermal_zones: thermal-zones {
    soc_thermal: soc-thermal {
        ...
        /* only A53 and A72 two modules are as cooling device */
        cooling-maps {
            map0 {
                trip = <&target>;
                cooling-device =
                    <&cpu_l0 THERMAL_NO_LIMIT THERMAL_NO_LIMIT>;
                contribution = <4096>;
            };
            map1 {
                trip = <&target>;
                cooling-device =
                    <&cpu_b0 THERMAL_NO_LIMIT THERMAL_NO_LIMIT>;
                contribution = <1024>;
            };
        };
        ...
    };
};
```

3. Adjust sustainable-power.

The range from 70 degree to 85 degree set in step (1) means the system will provide a relatively large power value with 70 degree, when the temperature rises, power gradually decreases, when power decreases to a certain extent, it will start to decrease the frequency, if the temperature keeps rising, power keeps decreasing, and the frequency also keeps decreasing. So it only shorten the time interval to acquire the temperature when over 70 degree, not definitely decrease the frequency, you can change the sustainable value to adjust the time to decrease the frequency.

Assume that when over 70 degree the thermal control strategy starts to work, that is to shorten the time interval to acquire the temperature, starts to limit the frequency when 75 degree (this setting can reduce the frequency fluctuation at the beginning of the thermal control), not to exceed 85 degree at most. Then we can set the power value with 75 degree equal to the sum of the max power consumption of all cooling devices, and then gradually decrease to debug, until it meets our requirement.

The power consumption contains static power consumption and dynamic power consumption. The compution formulas are as below:

```
Static power consumption formula:
    /*
     * a, b, c, d, C are constants, in DTSI configuration, just use the default
values,
     * T is the temperature, V is the voltage, need to adjust them according to the
     * actual situation.
     */
    t_scale = (a * T^3) + (b * T^2) + (c * T) + d
    v_scale = V^3
    P(s)= C * T_scale * V_scale
```

```
Dynamic power consumption formula:
    /*
     * C is the constant, in DTSI configuration, just use the default value,
     * V is the voltage, F is the frequency, need to adjust them according to
     * the actual situation.
     */
    P(d)= C * V^2 * F
```

Take RK3399 as an example, assume that A53, A72 and GPU are all working and need to be limited, the actually used max frequencies are 1416MHz(1125mV), 1800MHz(1200mV) and 800MHz(1100mV), then we can compute the power consumption as below:

```
A53 dynamic power consumption: C = 100 (dynamic-power-coefficient is configured as 100 in DTSI), V = 1125mV, F = 1416MHz, quad cores
    P_d_a53 = 100 * 1125 * 1125 * 1416 * 4 / 1000000000 = 716 mW

A72 dynamic power consumption: C = 436 (dynamic-power-coefficient is configured as 436 in DTSI), V = 1200mV, F = 1800MHz, dual cores
    P_d_a72 = 436 * 1200 * 1200 * 1800 * 2 / 1000000000 = 2260 mW

GPU dynamic power consumption: C = 733 (dynamic-coefficient is configured as 733 in DTSI), V = 1100mV, F = 800MHz
    P_d_gpu = 733 * 1100 * 1100 * 800 / 1000000000 = 709 mW

GPU static power consumption: static-coefficient is configured as 411000 and ts is configured as 32000 4700 -80 2 in DTSI, then C = 411000, a = 2, b = -80, c = 4700, d = 32000, the temperature starting to decrease the frequency T = 75000mC, V = 1100mV
    t_scale = ( 2 * 75000 * 75000 * 75000 / 1000000 ) + ( -80 * 75000 * 75000 / 1000) +
    ( 4700 * 75000 ) + 32000 * 1000 =  778250
    v_scale = 1100 * 1100 * 1100 / 1000000 = 1331
    P_s_gpu = 411000 * 778250 / 1000000 * 1331 / 1000000 = 425mW

    P_max = P_d_a53 + P_d_a72 + P_d_gpu + P_s_gpu = 4110mW

    Note: currently only GPU computes the static power consumption. Here only list the computing method, actually it is more convenient to compute through excel.
```

We expect to decrease the frequency over 75 degree, so we can set the power with 75 degree as the biggest power, and then compute through below formula to get the sustainable value:

```
sustainable + 2 * sustainable / (target- threshold) * (target- 75) = P_75
sustainable + 2 * sustainable / (85 - 70) * (85 - 75) = 4110
sustainable = 1761mW
```

Firstly set sustainable-power as 1761 in DTSI, actually test the different scenarios such as Antutu, Geekbench and so on, capture the trace data, analyze the change of the frequency and temperature, or use lisa tool to draw picture to analyze, to see if match with the expectation or not, if not match with the expectation, decrease the value, continue debugging, until it meets the expectation.

4. Adjust contribution.

Adjust the corresponding contribution of cooling device can adjust the sequence and extent while decreasing the frequency, even if not configured, it also will be set as 1024. If in high temperature environment, both A53 and A72 are running with full load, it is found that the frequency of A53 is easier to be decreased, if now want to decrease the frequency of A72 with priority, you can increase the contribution of A53, for example:

```
thermal_zones: thermal-zones {
    soc_thermal: soc-thermal {
        ...
        cooling-maps {
            map0 {
                trip = <&target>;
                cooling-device =
                    <&cpu_l0 THERMAL_NO_LIMIT THERMAL_NO_LIMIT>;
                contribution = <4096>; /* change from the default value 1024 to 4096
*/
            };
            map1 {
                trip = <&target>;
                cooling-device =
                    <&cpu_b0 THERMAL_NO_LIMIT THERMAL_NO_LIMIT>;
                contribution = <1024>;
            };
        };
        ...
    };
};
```

5. Acquire trace data to analyze.

Firstly, need to enable the trace related configurations in menuconfig.

```
Kernel hacking  --->
    [*] Tracers  --->
        --- Tracers
        [ ]   Kernel Function Tracer
        [ ]   Enable trace events for preempt and irq disable/enable
        [ ]   Interrupts-off Latency Tracer
        [ ]   Preemption-off Latency Tracer
        [ ]   Scheduling Latency Tracer
        [*]   Trace process context switches and events
        [ ]   Trace syscalls
        [ ]   Create a snapshot trace buffer
        Branch Profiling (No branch profiling)  --->
        [ ]   Trace max stack
        [ ]   Support for tracing block IO actions
        [ ]   Add tracepoint that benchmarks tracepoints
        < >   Ring buffer benchmark stress tester
        [ ]   Ring buffer startup self test
        [ ]   Show enum mappings for trace events
        [*]   Trace gpio events
```

Method 1: capture log through trace-cmd, there is trace-cmd in lisa tool package, refer to lisa related documents for lisa environment installation. Push trace-cmd to the target board through adb, and then acquire thermal control related log through below commands:

```
/*
 * -b specifies the buffer size, unit is Kb,
 * DDR capacity is different for different platforms, may need to adjust
 */
trace-cmd record -e thermal -e thermal_power_allocator -b 102400
```

Ctrl+C can stop recording log, generate the trace.dat file in current directory, transform the format through below command:

```
trace-cmd report trace.dat > trace.txt
```

Then use adb to pull the file to PC, directly open to analyze or use lisa tool to analyze. Or you can pull the trace.dat file to PC, and use trace-cmd to convert it into trace.txt in PC.

Method 2: if there is no trace-cmd tool, use command can also acquire thermal control related log.

Enable thermal control related trace:

```
echo 1 > /sys/kernel/debug/tracing/events/thermal/enable
echo 1 > /sys/kernel/debug/tracing/events/thermal_power_allocator/enable
echo 1 > /sys/kernel/debug/tracing/tracing_on
```

Directly print out the trace data, and save as a file:

```
cat /sys/kernel/debug/tracing/trace
```

Or directly pull out the file through adb:

```
/*
 * after acquiring the data, directly open trace.txt to analyze,
 * or use lisa tool to analyze
 */
adb pull /sys/kernel/debug/tracing/trace ./trace.txt
```

Other operations:

```
echo 0 > /sys/kernel/debug/tracing/tracing_on  /* pause to capture data */
echo 0 > /sys/kernel/debug/tracing/trace       /* clean up the previous data */
```

4 User Interface Introduction

User interface is in the directory of /sys/class/thermal/, and the detailed contents correspond to the thermal zone node configuration in DTSI. Some platforms only have one sub node under the thermal zone node, so there is only thermal_zone0 sub directory under the directory of /sys/class/thermal/. Some platforms have two sub nodes, and correspondingly there will be thermal_zone0 and thermal_zone1 sub directories under

the directory of /sys/class/thermal/. Through user mode interface it is able to switch thermal control strategy, check current temperature and so on.

Take RK3399 as an example, the /sys/class/thermal/thermal_zone0/ directory includes below commonly used information:

```
temp                    /* current temperature */
available_policies      /* available thermal governors */
policy                  /* current thermal governor */
sustainable_power       /* power value under the desired highest temperature */
/*
 * The trigger condition of I in PID algorithm:
 * current temperature - desired highest temperature < integral_cutoff
 */
integral_cutoff
k_d                     /* the parameter used to compute D in PID algorithm */
k_i                     /* the parameter used to compute I in PID algorithm */
k_po                    /* the parameter used to compute P in PID algorithm */
k_pu                    /* the parameter used to compute P in PID algorithm */
/*
 * enabled: periodically acquire the temperature, judge if need to decrease the
 * frequency or not.
 * disabled: close the function
 */
mode
type                    /* type of current thermal zone */
/* different threshold temperatures, correspond to trips nodes */
trip_point_0_hyst
trip_point_0_temp
trip_point_0_type
trip_point_1_hyst
trip_point_1_temp
trip_point_1_type
trip_point_2_hyst
trip_point_2_temp
trip_point_2_type
/*
 * the statuses of cooling devices, correspond to cooling-maps nodes
 * cdev0 represent a cooling device, some platform may have cdev1, cdev2 and so on
 */
cdev0
    cur_state           /* current frequency of this cooling device */
    max_state           /* this cooling device has how many frequencies at most */
    type                /* type of this cooling device */
/* the multiply times used to compute the power of the cooling device */
cdev0_weight
```

Refer to the document “Documentation/thermal/sysfs-api.txt”.

5 Common Issues

5.1 Disable Thermal Control

Method 1: set the default thermal governor as user_space in menuconfig.

```
<*> Generic Thermal sysfs driver  --->
    --- Generic Thermal sysfs driver
    [*]   APIs to parse thermal data out of device tree
    [*]   Enable writable trip points
     /* chang governor from power_allocator to user_space */
        Default Thermal governor (user_space)  --->
```

Method 2: Use command to disable the thermal control after boot up.

Firstly, switch the thermal governor to user_space, that is, change the policy node in the user interface to user_space; Or set the mode as disabled; then, remove the frequency limitation, that is, set cur_state of all cdev in the user interface as 0.

Take RK3399 as an example, switch governor to user_space:

```
echo user_space > /sys/class/thermal/thermal_zone0/policy
```

Or set the mode as disabled:

```
echo disabled > /sys/class/thermal/thermal_zone0/mode
```

Remove the frequency limitation:

```
/* change below command according to the actual situation */
echo 0 > /sys/class/thermal/thermal_zone0/cdev0/cur_state
echo 0 > /sys/class/thermal/thermal_zone0/cdev1/cur_state
echo 0 > /sys/class/thermal/thermal_zone0/cdev2/cur_state
```

5.2 Acquire Current Temperature

Just look at the temp node in the directory of thermal_zone0 or thermal_zone1.

Take RK3399 as an example, input below command in debug console to acquire CPU temperature:

```
cat /sys/class/thermal/thermal_zone0/temp
```

Input below command in debug console to acquire GPU temperature:

```
cat /sys/class/thermal/thermal_zone1/temp
```