WALT Energy是如何计算的

1、代码路径：kernel/sched/energy.c

若干个关键的数据结构：

```
struct capacity_state {
    unsigned long cap;  /* compute capacity */
    unsigned long power;    /* power consumption at this compute capacity */
};

struct idle_state {
    unsigned long power;     /* power consumption in this idle state */
};
//capacity和power数据全部存放在此结构体中
struct sched_group_energy {
    unsigned int nr_idle_states;    /* number of idle states */
    struct idle_state *idle_states; /* ptr to idle state array */
    unsigned int nr_cap_states; /* number of capacity states */
    struct capacity_state *cap_states; /* ptr to capacity state array */
```

2、计算原理，贴出代码如下：

```
void init_sched_energy_costs(void)
{
    struct device_node *cn, *cp;
    struct capacity_state *cap_states;
    struct idle_state *idle_states;
    struct sched_group_energy *sge;
    const struct property *prop;
    int sd_level, i, nstates, cpu;
    const __be32 *val;
    unsigned long min_cap = ULONG_MAX;
    unsigned long capacity;
    //遍历系统所以的cpu
    for_each_possible_cpu(cpu) {
        cn = of_get_cpu_node(cpu, NULL);
        if (!cn) {
            pr_warn("CPU device node missing for CPU %d\n", cpu);
            return;
        }

        if (!of_find_property(cn, "sched-energy-costs", NULL)) {
            pr_warn("CPU device node has no sched-energy-costs\n");
            return;
        }

        for_each_possible_sd_level(sd_level) {
            cp = of_parse_phandle(cn, "sched-energy-costs", sd_level);
            if (!cp)
                break;

            prop = of_find_property(cp, "busy-cost-data", NULL);
            if (!prop || !prop->value) {
                pr_warn("No busy-cost data, skipping sched_energy init\n");
                goto out;
```

```
            }

            sge = kcalloc(1, sizeof(struct sched_group_energy),
                          GFP_NOWAIT);

            nstates = (prop->length / sizeof(u32)) / 2;
            cap_states = kcalloc(nstates,
                                 sizeof(struct capacity_state),
                                 GFP_NOWAIT);

            for (i = 0, val = prop->value; i < nstates; i++) {
                cap_states[i].cap = be32_to_cpup(val++);
                cap_states[i].power = be32_to_cpup(val++);
            }

            sge->nr_cap_states = nstates;
            sge->cap_states = cap_states;

            prop = of_find_property(cp, "idle-cost-data", NULL);
            if (!prop || !prop->value) {
                pr_warn("No idle-cost data, skipping sched_energy init\n");
                goto out;
            }

            nstates = (prop->length / sizeof(u32));
            idle_states = kcalloc(nstates,
                                  sizeof(struct idle_state),
                                  GFP_NOWAIT);

            for (i = 0, val = prop->value; i < nstates; i++)
                idle_states[i].power = be32_to_cpup(val++);

            sge->nr_idle_states = nstates;
            sge->idle_states = idle_states;
            //全部保存到这个全局二维数组中
            sge_array[cpu][sd_level] = sge;
        }

        /* find min_cap cpu masks */
        sge = sge_array[cpu][SD_LEVEL0];
        if (!sge)
            continue;
        capacity = sge->cap_states[sge->nr_cap_states - 1].cap;
        if (capacity < min_cap) {
            cpumask_clear(&min_cap_cpu_mask);
            min_cap = capacity;
        }

        if (capacity == min_cap)
            cpumask_set_cpu(cpu, &min_cap_cpu_mask);
    }
    pr_info("Energy-costs installed from DT, min_cap_cpu_mask: %*pbl\n",
            cpumask_pr_args(&min_cap_cpu_mask));
    return;
```

```
out:
    free_resources();
}
```

以sharkL3手机项目来解析cpu power/capacity数据

- of_get_cpu_node(cpu, NULL)，解析cpu节点信息，如下所示

```
CPU0: cpu@0 {
    device_type = "cpu";
    compatible = "arm,cortex-a55","arm,armv8";
    reg = <0x0 0x0>;
    enable-method = "psci";
    cpu-supply = <&fan53555_dcdc>;
    cpufreq-data-v1 = <&cpufreq_clus0>;
    cpu-idle-states = <&CORE_PD>;
    sched-energy-costs = <&CPU_COST_0 &CLUSTER_COST_0>;
```

继续解析sched-energy-costs字段，包括两个部分，

- 看如何解析sched-energy-costs，方法如下：

```
for_each_possible_sd_level(sd_level) {
    cp = of_parse_phandle(cn, "sched-energy-costs", sd_level);
```

能够看到sd_level表示字段sched-energy-costs包括几个level耗电，从cpu0字段可以看到有两个level，CPU_COST_0和CLUSTER_COST_0

- 分别解析CPU_COST_0和CLUSTER_COST_0包含的字段信息并保存到相应的结构体中：

```
energy-costs {
        CPU_COST_0: core-cost0 {
            busy-cost-data = <
                501 80  /* 768MHz */
                576 101 /* 884MHz */
                652 125 /* 1000MHz */
                717 151 /* 1100MHz */
                782 181 /* 1200MHz */
            >;
            idle-cost-data = <
                25      /* ACTIVE-IDLE */
                25      /* WFI */
                0       /* CORE_PD */
            >;
        };
        CPU_COST_1: core-cost1 {
            busy-cost-data = <
                501 110 /* 768MHz */
                685 160 /* 1050MHz */
                799 206 /* 1225MHz */
                913 258 /* 1400MHz */
                978 305 /* 1500MHz */
                1024    352 /* 1570MHz */
            >;
            idle-cost-data = <
                44      /* ACTIVE-IDLE */
                44      /* WFI */
                0       /* CORE_PD */
            >;
        };
```

```
        CLUSTER_COST_0: cluster-cost0 {
            busy-cost-data = <
                501 0   /* 768MHz */
                576 0   /* 884MHz */
                652 0   /* 1000MHz */
                717 0   /* 1100MHz */
                782 0   /* 1200MHz */
            >;
            idle-cost-data = <
                0       /* ACTIVE-IDLE */
                0       /* WFI */
                0       /* CORE_PD */
            >;
        };
        CLUSTER_COST_1: cluster-cost1 {
            busy-cost-data = <
                501 68  /* 768MHz */
                685 85  /* 1050MHz */
                799 106 /* 1225MHz */
                913 130 /* 1400MHz */
                978 153 /* 1500MHz */
                   1024 179 /* 1570MHz */
            >;
            idle-cost-data = <
                42      /* ACTIVE-IDLE */
                42      /* WFI */
                42      /* CORE_PD */
            >;
        };
    };
```

这样对于某个cpu的energy-costs每个字段的属性全部解析完毕，

- 遍历第二个cpu直到最后一个cpu为止。
- 最后的结果全部存放在sge_array结构体二维数组中，打印source code如下：

```
    unsigned int cpu_num = 0, i = 0;
    for(cpu_num = 0; cpu_num < NR_CPUS; cpu_num++) {
        printk("cpu%d",cpu_num);
        for(i = 0; i < 2; i++) {
            unsigned int temp = sge_array[cpu_num][i]->nr_cap_states;
            unsigned int k = 0;
            printk(" nr_cap_states =%d
",sge_array[cpu_num][i]->nr_cap_states);
            printk("cap:power: ");
            for(k = 0; k < temp; k++) {
                printk("%lu:",sge_array[cpu_num][i]->cap_states[k].cap);
                printk("%lu ",sge_array[cpu_num][i]->cap_states[k].power);
            }
            temp = sge_array[cpu_num][i]->nr_idle_states;
            printk("idle power:");
            for(k = 0; k < temp; k++) {
                printk("%lu ",sge_array[cpu_num][i]->idle_states[k].power);
            }
            printk("\n");
        }
    }
```

```
for_each_cpu(j, &min_cap_cpu_mask) {
        printk("j=%d\n",j);
    }
```

打印的结果如下：

```
[   52.304285] cpu0 nr_cap_states =5 cap:power: 501:80 576:101 652:125 717:151 782:181 idle power:25 25 0
[   52.304421] nr_cap_states =5 cap:power: 501:0 576:0 652:0 717:0 782:0 idle power:0 0 0
[   52.304562] cpu1 nr_cap_states =5 cap:power: 501:80 576:101 652:125 717:151 782:181 idle power:25 25 0
[   52.304719] nr_cap_states =5 cap:power: 501:0 576:0 652:0 717:0 782:0 idle power:0 0 0
[   52.304853] cpu2 nr_cap_states =5 cap:power: 501:80 576:101 652:125 717:151 782:181 idle power:25 25 0
[   52.305015] nr_cap_states =5 cap:power: 501:0 576:0 652:0 717:0 782:0 idle power:0 0 0
[   52.305155] cpu3 nr_cap_states =5 cap:power: 501:80 576:101 652:125 717:151 782:181 idle power:25 25 0
[   52.305310] nr_cap_states =5 cap:power: 501:0 576:0 652:0 717:0 782:0 idle power:0 0 0
[   52.305450] cpu4 nr_cap_states =6 cap:power: 501:110 685:160 799:206 913:258 978:305 1024:352 idle power:44 44 0
[   52.305623] nr_cap_states =6 cap:power: 501:68 685:85 799:106 913:130 978:153 1024:179 idle power:42 42 42
[   52.305781] cpu5 nr_cap_states =6 cap:power: 501:110 685:160 799:206 913:258 978:305 1024:352 idle power:44 44 0
[   52.305957] nr_cap_states =6 cap:power: 501:68 685:85 799:106 913:130 978:153 1024:179 idle power:42 42 42
[   52.306115] cpu6 nr_cap_states =6 cap:power: 501:110 685:160 799:206 913:258 978:305 1024:352 idle power:44 44 0
[   52.306288] nr_cap_states =6 cap:power: 501:68 685:85 799:106 913:130 978:153 1024:179 idle power:42 42 42
[   52.306441] cpu7 nr_cap_states =6 cap:power: 501:110 685:160 799:206 913:258 978:305 1024:352 idle power:44 44 0
[   52.306614] nr_cap_states =6 cap:power: 501:68 685:85 799:106 913:130 978:153 1024:179 idle power:42 42 42
[   52.306773] j=0
[   52.306790] j=1
[   52.306802] j=2
[   52.306828] j=3
```

能够很明显的看到，cpu0~3是一样的，也就是同一个cluster，而cpu4~7是同一个cluster的，与dts里面填写的数据是一样的

- 最后获取min_cap cpu mask，获取的就是capacity最小的一组cpu mask，从上面打印的数据也可以看到是一样的，是小core，cluster0。
- 调用是在arch/arm64/kernel/topology.c，这个文件也是读取dts获取cpu拓扑。另讲。

```
void __init init_cpu_topology(void)
{
    reset_cpu_topology();

    /*
     * Discard anything that was parsed if we hit an error so we
     * don't use partial information.
```

```
         */
        if (of_have_populated_dt() && parse_dt_topology())
            reset_cpu_topology();
        else
            set_sched_topology(arm64_topology);

    #ifdef CONFIG_DEFAULT_USE_ENERGY_AWARE
        init_sched_energy_costs();
    #endif

    #ifdef CONFIG_SCHED_COMPAT_LIMIT
        arch_cpu_mask_setup();
    #endif
    }
```