Linaro connect
San Francisco 2017

SFO17-TR05: EAS Profiling On Hikey960

Leo Yan & Daniel Thompson
Linaro Solutions and Support Engineering

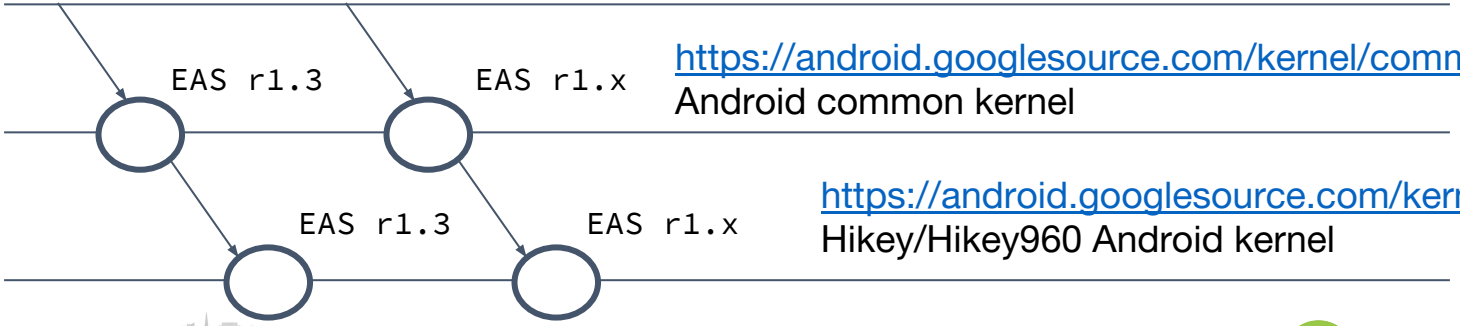
Overview

- Current status of EAS r1.3 on Hikey960
  - EAS in Android common kernel
  - Hikey960 power management introduction
  - Power modeling on Hikey960
- Testing methodology
- Optimization with CGroup
- Observed issues and proposed patches
- Conclusion

EAS in Android common kernel

- EAS r1.3 is merged into Android common kernel and Hikey kernel
  - Many optimizations have been merged in this version!
  - EAS r1.3 “has done a large refactoring of find_best_target” to rework wakeup, simplify the decision making and make heuristics clearer
  - EAS r1.3 makes ‘schedutil’ the recommended CPUFreq governor
  - EAS r1.3 will optimize small tasks by placing them on LITTLE CPUs most of the time

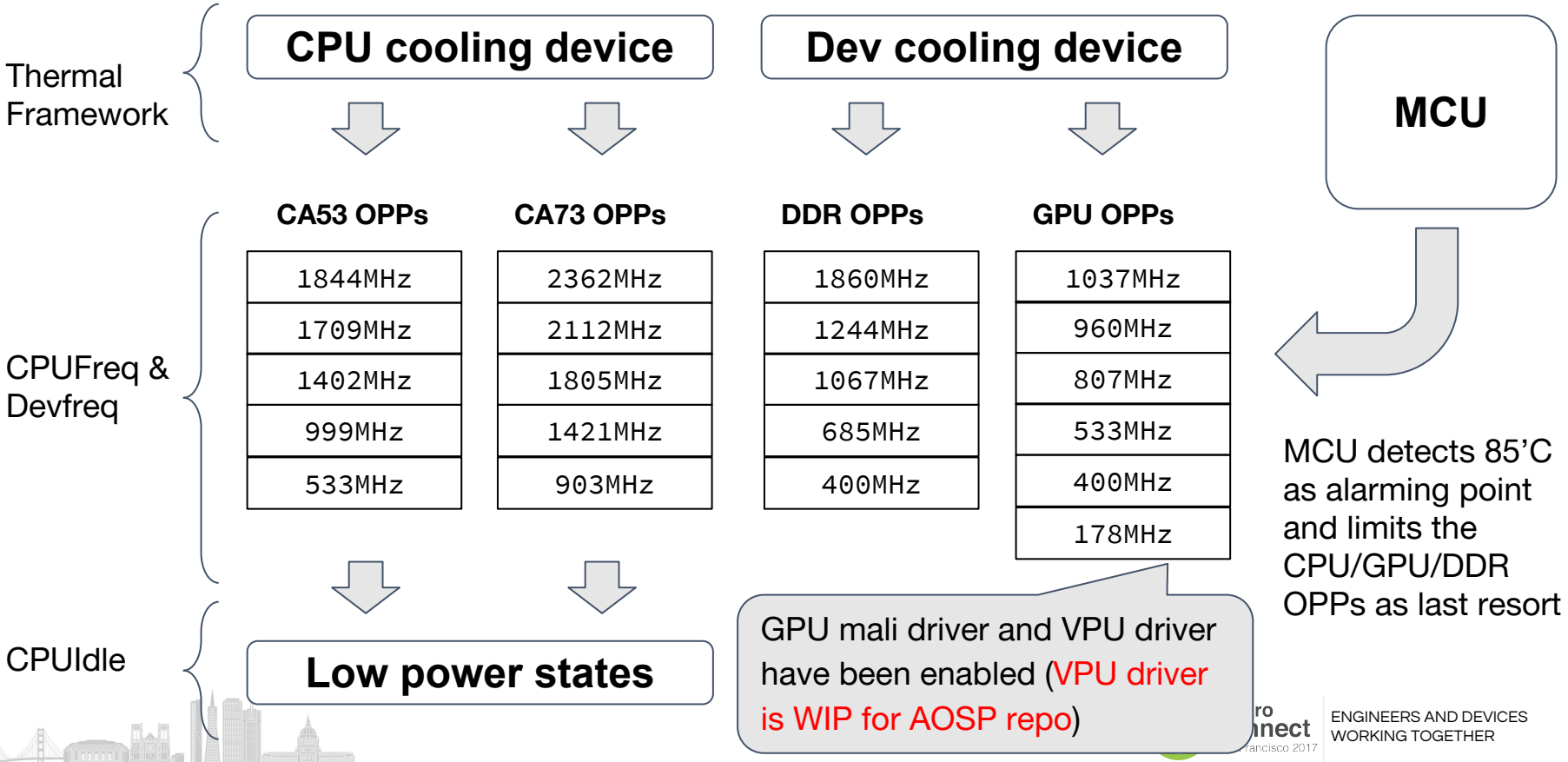
https://android-review.googlesource.com:
patches committing and merging



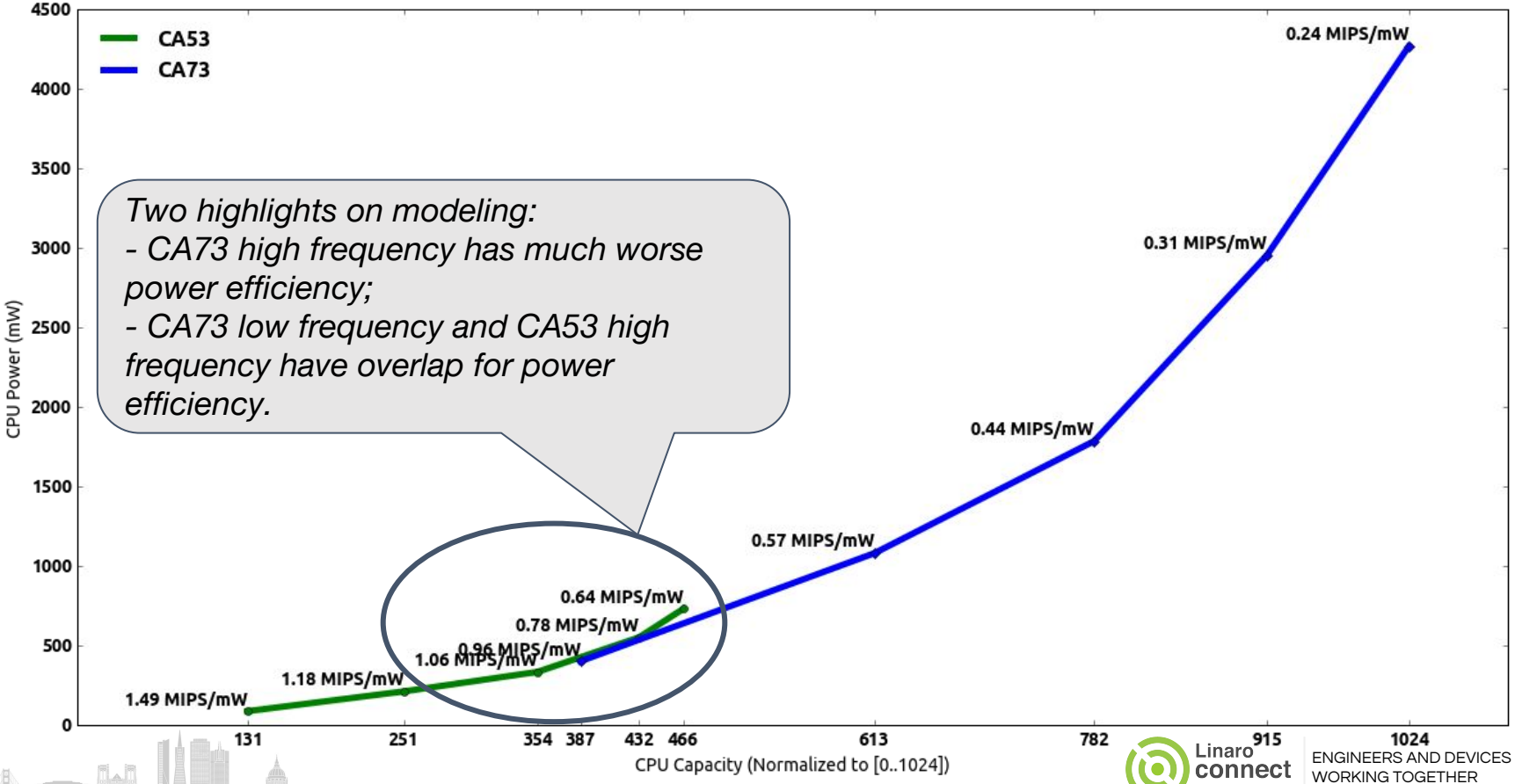
https://android.googlesource.com/kernel/common/:
Android common kernel

https://android.googlesource.com/kernel/hikey-linaro:
Hikey/Hikey960 Android kernel

Hikey960 Power Management Introduction



Power modeling on Hikey960



Linaro
connect
San Francisco 2017

ENGINEERS
AND DEVICES
WORKING
TOGETHER

Overview

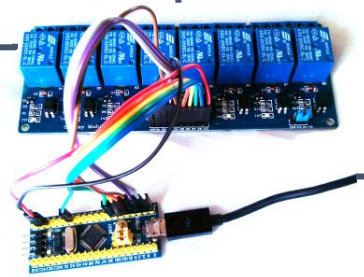
- Current status of EAS r1.3 on Hikey960
- Testing methodology
  - Automatic testing framework
  - Testing suites
  - Temperature is critical for fair game!
  - Measurement metrics
- Optimization with CGroup
- Observed issues and proposed patches
- Conclusion

Automatic testing framework

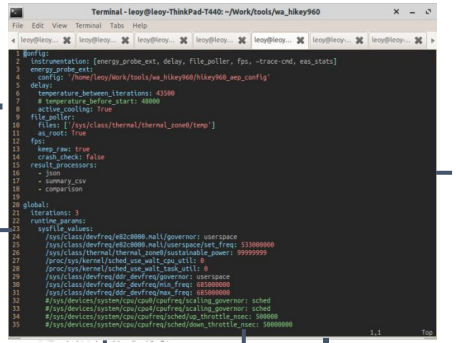
USB-Relay [1]

[1] https://github.com/daniel-thompson/usb-relay

Fan



Workload Automation

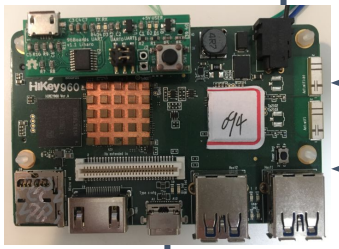


Charts

4. Output results

2. Cool down device with fan

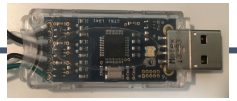
Hikey960



1. Burn images (mainly for kernel)

3. Run test cases

ARM Energy Probe



Workload automation

- “Workload Automation (WA) is a framework for executing workloads and collecting measurements on Android and Linux devices.” [1]
  - WA captures energy instrument data
  - WA captures performance metrics
  - WA captures ftrace log
  - WA captures kernel statistics (interrupt numbers during test case running)
- WA can run test cases for power scenarios/interactive performance/intensive performance testing for multiple iterations in single one testing configuration
- WA absents some features but we can easily extend it [2]
  - Support multiple kernel burning
  - Support kernel scheduler statistics
  - Support result comparison with different configurations
- Co-works with LISA for detailed scheduler analysis

[1] https://github.com/ARM-software/workload-automation
[2] https://github.com/Leo-Yan/workload-automation

Configuration for baseline testing

Android

Home screen: on

WIFI: off

Many peripherals and devices are powered on, so baseline power is high (~4w)

- *Fix DDR@685MHz and GPU@960MHz;*
- *Disable thermal capping by set high sustainable value (but MCU capping still works);*
- *use “sched-freq” governor and up threshold is 500us and down threshold is 50ms;*
- *use **PELT** signals due it’s more reliable for energy modeling*

Workload agenda

```
config:
  instrumentation: [energy_probe_ext, delay, file_poller, fps, ~trace-cmd, eas_stats]
  energy_probe_ext:
    config: '/home/leoy/Work/tools/wa_hikey960/hikey960_aep_config'
  delay:
    temperature_between_iterations: 43500
    active_cooling: True
  file_poller:
    files: ['/sys/class/thermal/thermal_zone0/temp']
    as_root: True
  fps:
    keep_raw: true
    crash_check: false
  result_processors:
    - json
    - summary_csv
    - comparison

global:
  iterations: 2
  runtime_params:
    sysfile_values:
      /sys/class/devfreq/e82c0000.mali/governor: userspace
      /sys/class/devfreq/e82c0000.mali/userspace/set_freq: 960000000
      /sys/class/thermal/thermal_zone0/sustainable_power: 99999999
      /proc/sys/kernel/sched_use_walt_cpu_util: 0
      /proc/sys/kernel/sched_use_walt_task_util: 0
      /sys/class/devfreq/ddr_devfreq/governor: userspace
      /sys/class/devfreq/ddr_devfreq/min_freq: 685000000
      /sys/class/devfreq/ddr_devfreq/max_freq: 685000000
      /sys/devices/system/cpu/cpu0/cpufreq/scaling_governor: sched
      /sys/devices/system/cpu/cpu4/cpufreq/scaling_governor: sched
      /sys/devices/system/cpu/cpufreq/sched/up_throttle_nsec: 500000
      /sys/devices/system/cpu/cpufreq/sched/down_throttle_nsec: 50000000
```

Temperature is critical for a fair game!

Ambient temperature:

26.5’C ~ 27.5’C

SoC start temperature: 43.5’C;

takes 2~5 minutes to reach start temperature

```
config:
  instrumentation: [energy_probe_ext, delay, file_poller, fps, ~trace-cmd, eas_stats]
  energy_probe_ext:
    config: '/home/leoy/Work/tools/wa_hikey960/hikey960_aep_config'
  delay:
    temperature_between_iterations: 43500
    active_cooling: True
  file_poller:
    files: ['/sys/class/thermal/thermal_zone0/temp']
    as_root: True
  fps:
    keep_raw: true
    crash_check: false
  result_processors:
    - json
    - summary_csv
    - comparison
```

The fan is used to accelerate cooling down time but it is stopped after cool down to specific temperature (e.g. 43.5'C). So we can avoid the fan to interfere the testing and give more penalty for power consumption when SoC temperature increases and has more close behavior with the real production.

Test suites

- Power saving cases
  - “idle” - sleep for 120s
  - “audio” - playback mp3 for 120s
  - “video” - playback video (1080p) for 120s with **hardware** codec
- Interactive cases
  - UiBench
    - “InflatingListActivity” - Scroll the screen for list widget
    - “InvalidateActivity” - Reverse color for small grids, invalidate and redraw view
    - “ActivityTransition” - Click screen to zoom in and zoom out picture
    - “TrivialAnimationActivity” - Animation for rendering whole screen color
  - “Galleryfling” - Launch gallery with hundreds pictures and fling pictures
  - “Recentfling” - Open recents screen and fling recent tasks list
  - “Emailfling” - Open email app and fling emails list
  - ~~“Browserfling” - Open browser icecat and fling the web page~~

Measurement metrics

#	Test cases		Scheduler Statistics	Power Metrics	Performance Metrics
1	idle		Wake up path statistics	mW = Energy (J) / Time (S)	n/a
2	audio		Wake up path statistics	mW	n/a
3	video		Wake up path statistics	mW	n/a
4	galleryfling		Wake up path statistics	mW	janks% = janks / total_frame
5	recentfling		Wake up path statistics	mW	janks%
6	emailfling		Wake up path statistics	mW	janks%
7	UiBench	InflatingListActivity	Wake up path statistics	mW	janks%
8		InvalidateActivity	Wake up path statistics	mW	janks%
9		ActivityTransition	Wake up path statistics	mW	janks%
10		TrivialAnimationActivity	Wake up path statistics	mW	janks%

Linaro
connect
San Francisco 2017

ENGINEERS
AND DEVICES
WORKING
TOGETHER

Overview

- Current status of EAS r1.3 on Hikey960
- Testing methodology
- Optimization with CGroup
  - Using CPUSET for background tasks
  - How to set schedtune's 'prefer_idle'?
  - Select boost margin for interactive cases
- Observed issues and proposed patches
- Conclusion

Control groups (CGroups) in Android system

- “Control Groups provide a mechanism for aggregating/partitioning sets of tasks, and all their future children, into hierarchical groups with specialized behaviour… mainly for **resource-tracking purposes**”: https://www.kernel.org/doc/Documentation/cgroup-v1/cgroups.txt
- A cgroup associates a set of tasks with a set of **parameters**
  - CPUCtl: restrict CPU time and keeps background tasks to only small portion bandwidth
    - **`cpu.shares`**
    - **`cpu.rt_runtime_us`**
    - **`cpu.rt_period_us`**
  - CPUSet: pin tasks to specified CPUs
    - **`cpus:`** CPU affinity for tasks
  - Schedtune: speedup the time-to-completion for a task activation
    - **`boost`**: inflate task and CPU utilization and impact OPP selection
    - **`prefer_idle`**: select idle CPUs for task to avoid scheduling latency
- Init scripts can statically set cgroups parameters
- Android PowerHAL can dynamically set cgroups parameters for different scenarios
- **PowerHAL is disabled in this slides and use WA script to statically set parameters**

Using CPUSET for background tasks affinity

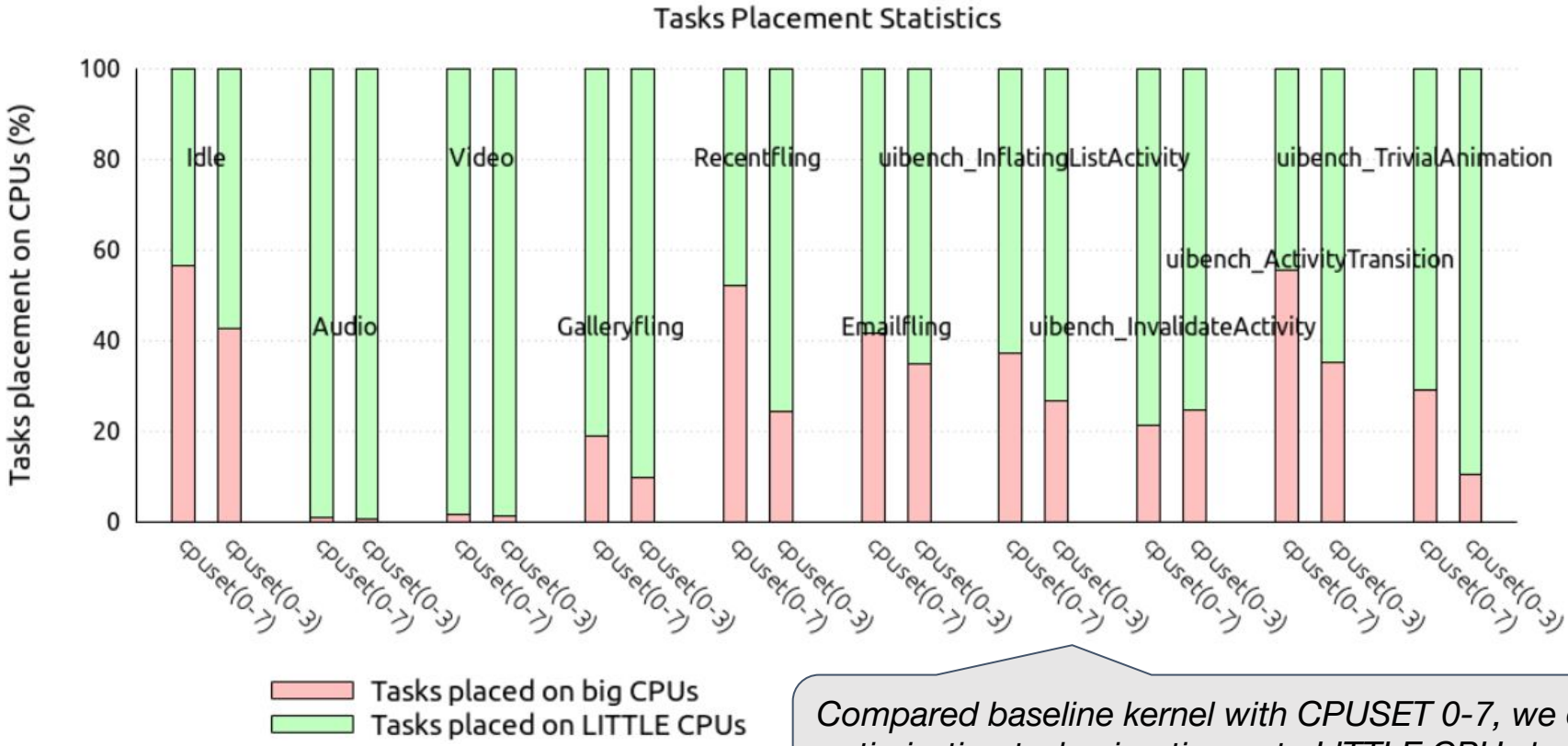
Using CPUSET to set cpus to '0-3' so can pin background tasks always run on LITTLE CPUs.

```
runtime_params:
  sysfile_values:
    /dev/cpuset/foreground/cpus: 0-7
    /dev/cpuset/top-app/cpus: 0-7
    /dev/cpuset/background/cpus: 0-3
    /dev/cpuset/system-background/cpus: 0-3
    /dev/stune/schedtune.boost: 0
    /dev/stune/schedtune.prefer_idle: 0
    /dev/stune/background/schedtune.boost: 0
    /dev/stune/background/schedtune.prefer_idle: 0
    /dev/stune/foreground/schedtune.boost: 0
    /dev/stune/foreground/schedtune.prefer_idle: 0
    /dev/stune/top-app/schedtune.boost: 0
    /dev/stune/top-app/schedtune.prefer_idle: 0
```

CPUSET knobs

Schedtune knobs

Task placement optimization with setting CPUSET 0~3 for background tasks



*Compared baseline kernel with CPUSET 0-7, we can optimization task migration onto LITTLE CPUs by pinning background tasks with **CPUSET 0-3**.*

Understanding ‘prefer_idle’

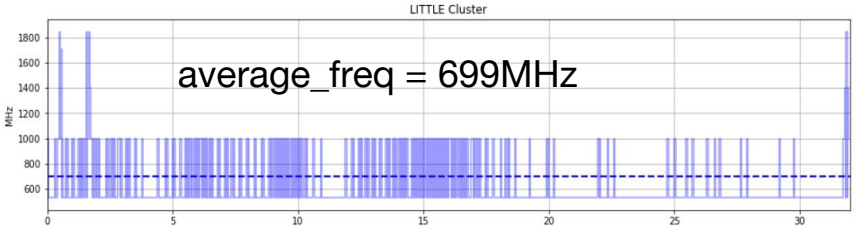
- ‘prefer_idle’ is indicator to ask scheduler to select idle CPU for ‘prefer_idle’ tasks as possible; it’s mainly used for multi-threading concurrency performance boosting
- Select idle CPUs with specific orders and **impacts power heavily**
  - When boost > 0, the big cluster is firstly iterated so ‘prefer_idle’ tasks have much more chance to be migrated on big CPUs
  - When boost = 0, tries to find IDLE CPU from LITTLE cluster firstly; ‘prefer_idle’ tasks can be well spreaded within LITTLE cluster, this results in power saving for middle workload scenario (e.g. video playback)
  - Iteration starts from low ID for CPU, so low ID CPU has higher priority
- Using CPUSET + ‘prefer_idle’ can guarantee latency for important tasks (top-app)

Caveats for ‘prefer_idle’

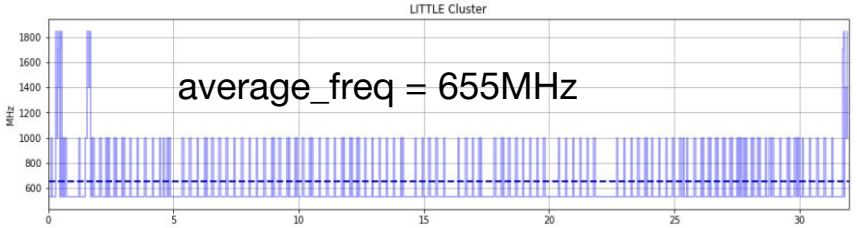
Assumption: ‘prefer_idle’ is considered to spread tasks and it doesn’t take account the energy when spread tasks to big CPUs; ‘prefer_idle’ enabling may introduce higher power by migration tasks to big CPUs. So we usually avoid to enable ‘prefer_idle’ for power sensitive scenarios.

In fact ‘prefer_idle’ co-works with `find_best_target()` underlying. `find_best_target()` tries to pack small tasks onto single CPU, and ‘prefer_idle’ can ease the packing when ‘prefer_idle’ can spread within LITTLE cluster and reduce CPU frequency; we can save power by ‘prefer_idle’ for some middle workload case.

```
/dev/stune/foreground/schedtune.prefer_idle: 0
/dev/stune/top-app/schedtune.prefer_idle: 1
```



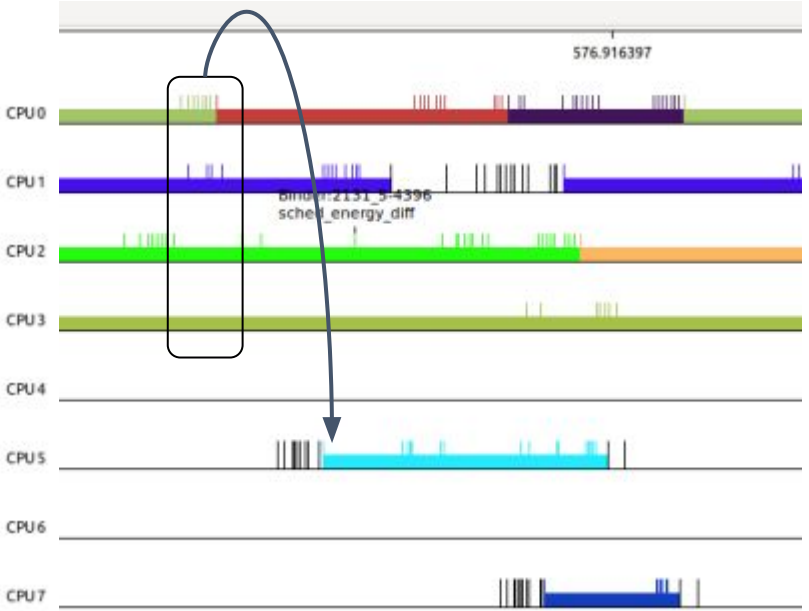

```
/dev/stune/foreground/schedtune.prefer_idle: 1
/dev/stune/top-app/schedtune.prefer_idle: 1
```



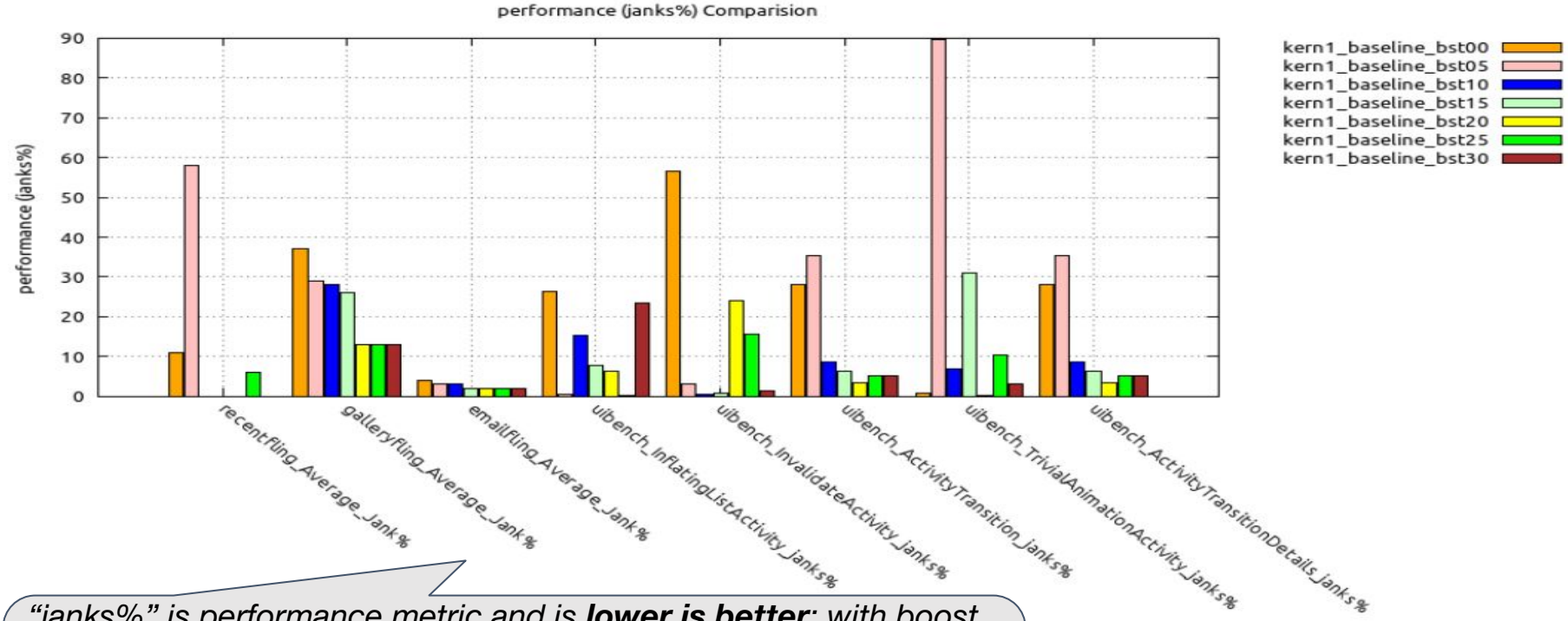
Caveats for ‘prefer_idle’ - cont.

Caveat #2:

Don’t use prefer_idle too much; otherwise it has more chance to contend IDLE CPUs with ‘prefer_idle’ tasks, and prefer_idle tasks also contend with non prefer-idle small tasks or even RT threads; as result there have more chance to migrate ‘prefer_idle’ tasks onto big CPUs, so at the end cannot save power.



Boost margin comparison for interactive cases



*"janks%" is performance metric and is **lower is better**; with boost margin = 15% or 20%, the janks% can be optimized to less than 10% for most cases; after setting boost = 25% or 30% the janks% is regressed by thermal throttling. Choosing **15%** boost margin in this slides for interactive cases.*

CPUSET and schedtune settings

Power hints can dynamically switch the settings between different scenarios.

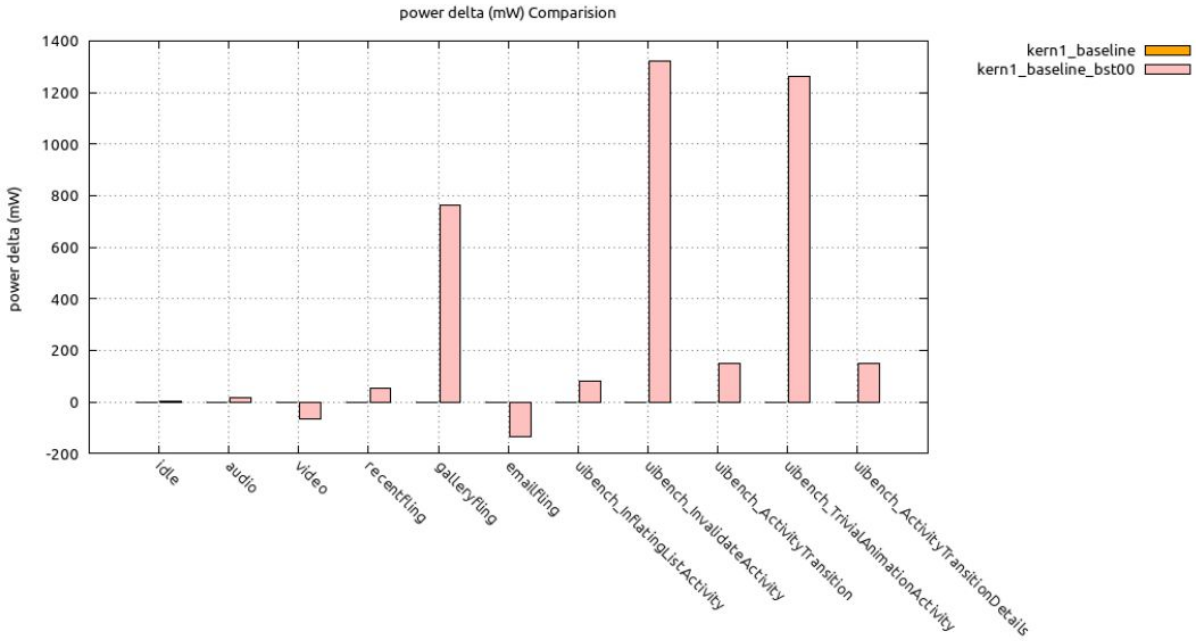
Used for power saving scenarios

```
runtime_params:
  sysfile_values:
    /dev/cpuset/foreground/cpus: 0-7
    /dev/cpuset/top-app/cpus: 0-7
    /dev/cpuset/background/cpus: 0-3
    /dev/cpuset/system-background/cpus: 0-3
    /dev/stune/schedtune.boost: 0
    /dev/stune/schedtune.prefer_idle: 0
    /dev/stune/background/schedtune.boost: 0
    /dev/stune/background/schedtune.prefer_idle: 0
    /dev/stune/foreground/schedtune.boost: 0
    /dev/stune/foreground/schedtune.prefer_idle: 1
    /dev/stune/top-app/schedtune.boost: 0
    /dev/stune/top-app/schedtune.prefer_idle: 1
```

Used for interactive scenarios

```
runtime_params:
  sysfile_values:
    /dev/cpuset/foreground/cpus: 0-7
    /dev/cpuset/top-app/cpus: 0-7
    /dev/cpuset/background/cpus: 0-3
    /dev/cpuset/system-background/cpus: 0-3
    /dev/stune/schedtune.boost: 0
    /dev/stune/schedtune.prefer_idle: 0
    /dev/stune/background/schedtune.boost: 0
    /dev/stune/background/schedtune.prefer_idle: 0
    /dev/stune/foreground/schedtune.boost: 15
    /dev/stune/foreground/schedtune.prefer_idle: 1
    /dev/stune/top-app/schedtune.boost: 15
    /dev/stune/top-app/schedtune.prefer_idle: 1
```

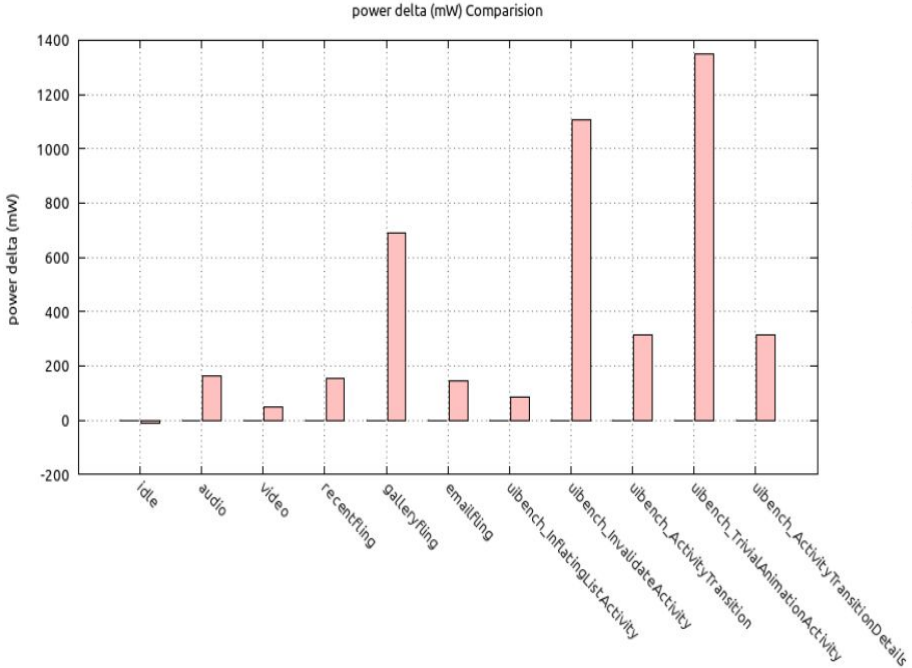
Power comparison (**boost%=0**)

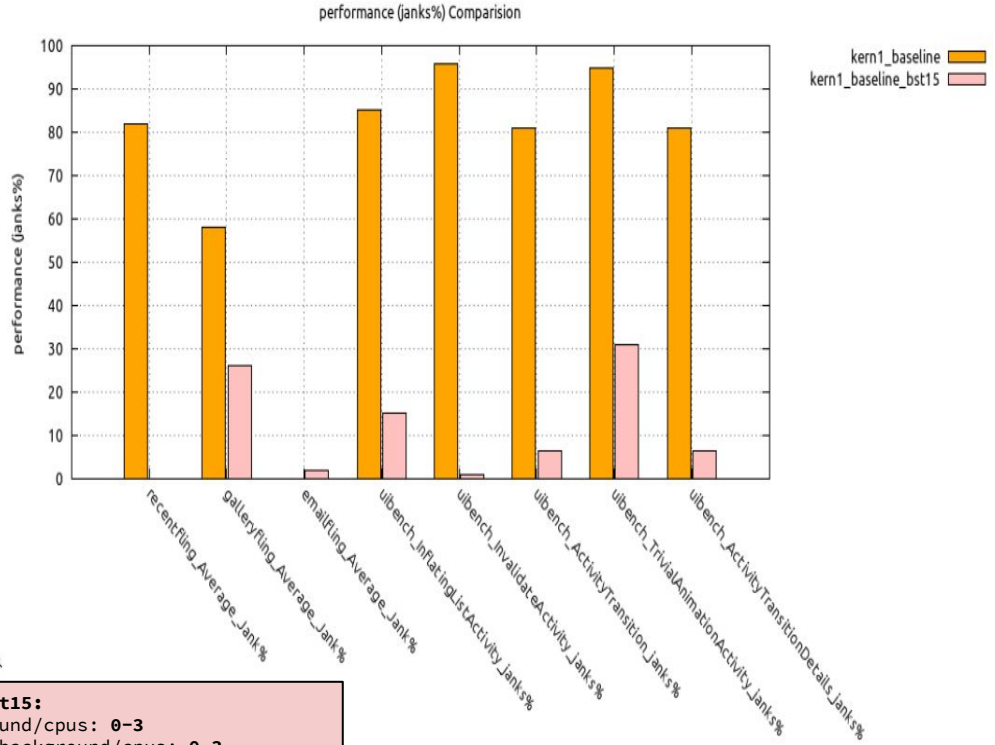


Kernel1_baseline:
/dev/cpuset/background/cpus: 0-7
/dev/cpuset/system-background/cpus: 0-7
/dev/stune/foreground/schedtune.boost: 0
/dev/stune/foreground/schedtune.prefer_idle: 0
/dev/stune/top-app/schedtune.boost: 0
/dev/stune/top-app/schedtune.prefer_idle: 0

Kernel1_baseline_bst00:
/dev/cpuset/background/cpus: **0-3**
/dev/cpuset/system-background/cpus: **0-3**
/dev/stune/foreground/schedtune.boost: 0
/dev/stune/foreground/schedtune.prefer_idle: **1**
/dev/stune/top-app/schedtune.boost: 0
/dev/stune/top-app/schedtune.prefer_idle: **1**

Power and performance comparison (**boost%=15**)





Kernel1_baseline:
/dev/cpuset/background/cpus: 0-7
/dev/cpuset/system-background/cpus: 0-7
/dev/stune/foreground/schedtune.boost: 0
/dev/stune/foreground/schedtune.prefer_idle: 0
/dev/stune/top-app/schedtune.boost: 0
/dev/stune/top-app/schedtune.prefer_idle: 0

Kernel1_baseline_bst15:
/dev/cpuset/background/cpus: **0-3**
/dev/cpuset/system-background/cpus: **0-3**
/dev/stune/foreground/schedtune.boost: **15**
/dev/stune/foreground/schedtune.prefer_idle: **1**
/dev/stune/top-app/schedtune.boost: **15**
/dev/stune/top-app/schedtune.prefer_idle: **1**

Linaro
connect
San Francisco 2017

ENGINEERS
AND DEVICES
WORKING
TOGETHER

Overview

- Current status of EAS r1.3 on Hikey960
- Testing methodology
- Optimization with CGroup
- Observed issues and proposed patches
  - Optimization methodology for EAS core algorithms
  - Optimization with active CPU and IDLE CPU frequency
  - Comparison energy with cpumask (**in hacking session**)
- Conclusion

Methodology for optimization in this chapter

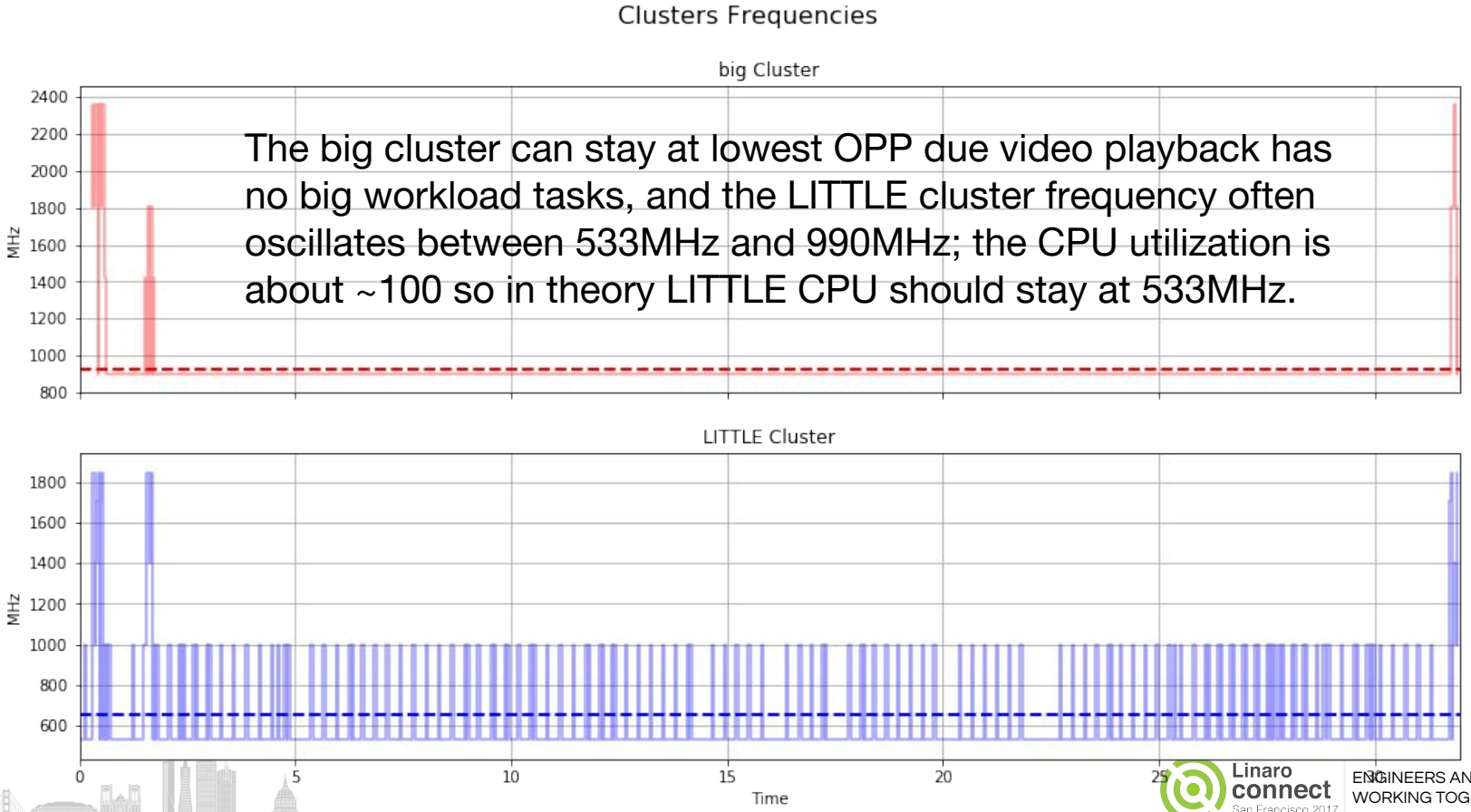
The first step:

We can start to analyze the CPU frequency selection for some power sensitive cases (e.g. video/audio playback etc); if there have some scenarios with CPU frequency is overrated, it's good start point for analysis.

The second step:

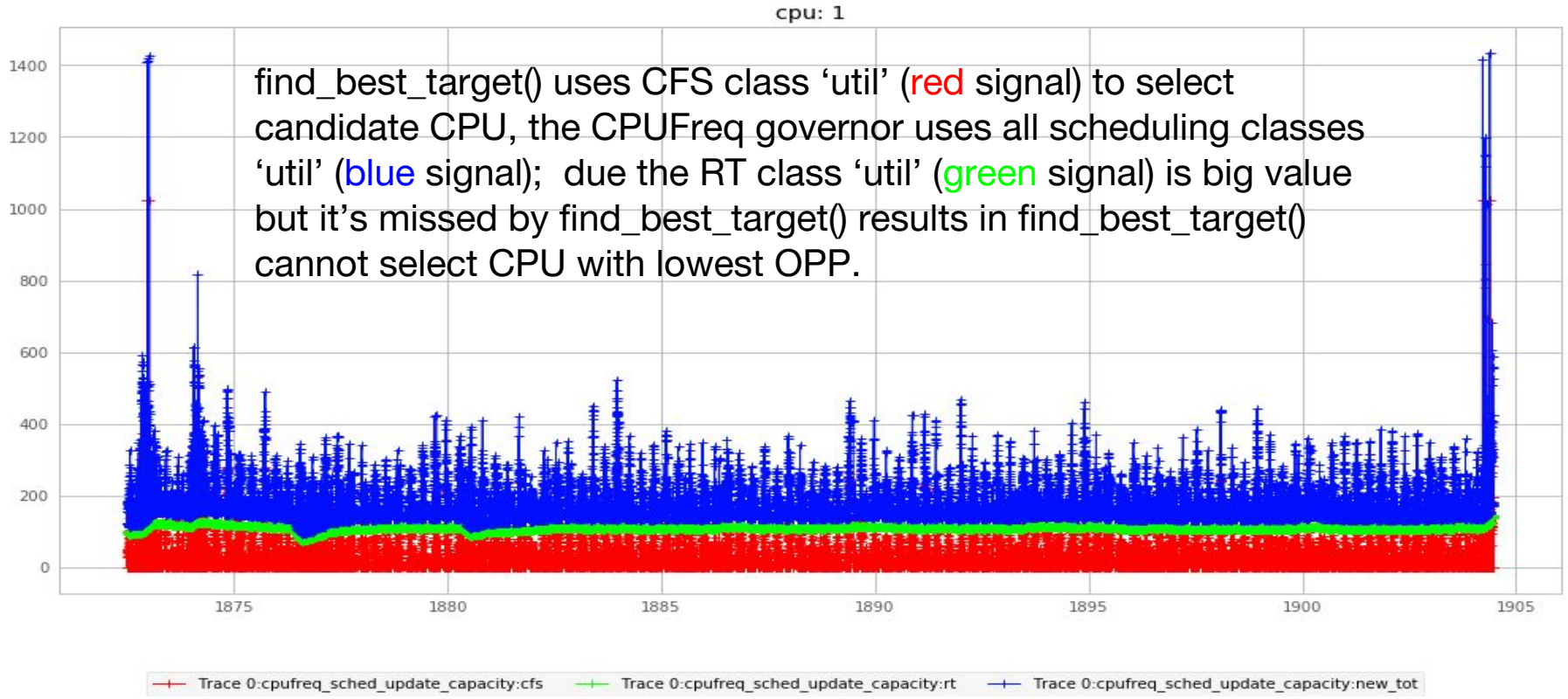
Based on the first step optimization, we can explore optimization for task placement crossing different clusters; this mean we don't merely optimize small tasks placed on LITTLE CPUs, but also need find better occasions to place big workload tasks to big CPUs.

Video playback (**1080p**) frequency analysis



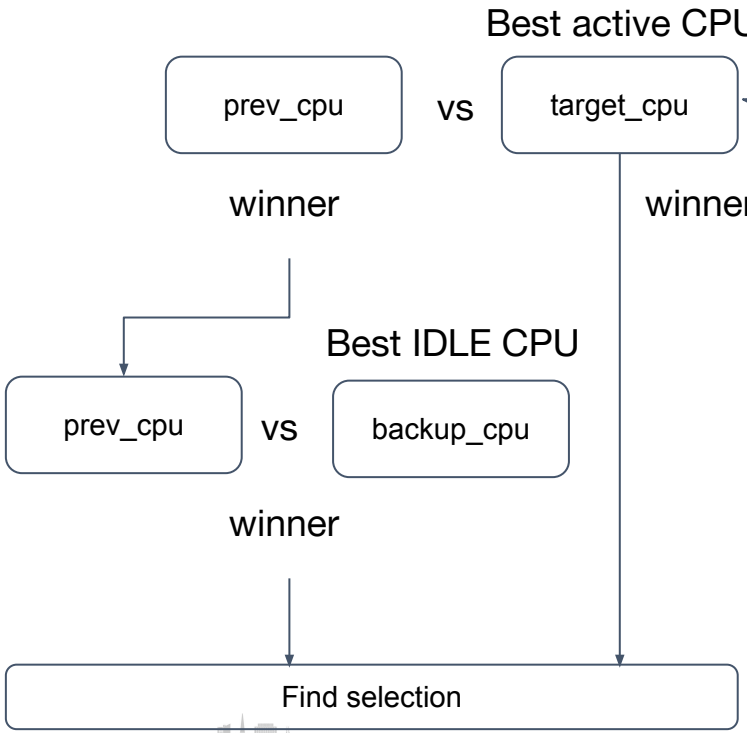
The big cluster can stay at lowest OPP due video playback has no big workload tasks, and the LITTLE cluster frequency often oscillates between 533MHz and 990MHz; the CPU utilization is about ~100 so in theory LITTLE CPU should stay at 533MHz.

RT threads util is high in video playback case



Optimization for CPU selection with lower frequency

EAS r1.3



Proposed optimization

Consistent CPU utilization estimation crosses find_best_target(), energy calculation and CPUFreq governor.

Change “target CPU” as CPU least util CPU (and **includes** IDLE CPU).

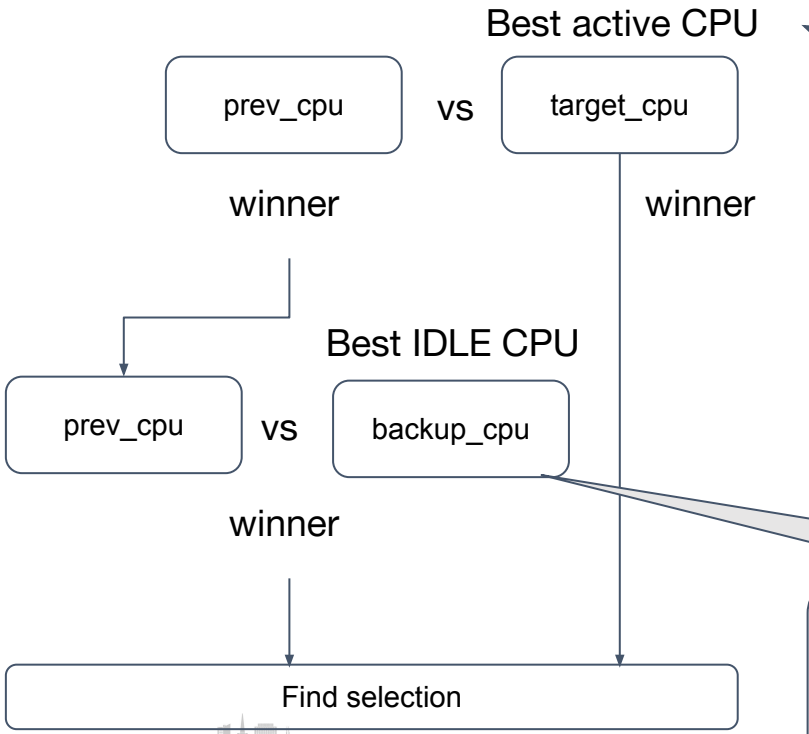
Function `calc_total_util()` introduced for total utilization estimation

```
static unsigned long calc_total_util(int cpu, struct task_struct *p, unsigned long cpu_util,
                                     unsigned long tsk_util)
{
        unsigned long cfs_util, rt_util, dl_util, total;
        int boost;
        int tsk_boost = p ? schedtune_task_boost(p) : 0;
        int cpu_boost = schedtune_cpu_boost(cpu);

        /*
         * Estimate cfs utilization with boost margin
         */
        boost = max(tsk_boost, cpu_boost);
        cfs_util = cpu_util + tsk_util;
        cfs_util += schedtune_margin(cfs_util, boost);

        /* Convert rt class utilization with CPU capacity */
        rt_util = get_rt_cpu_capacity(cpu) * capacity_orig_of(cpu);
        rt_util = rt_util / SCHED_CAPACITY_SCALE;

        /* Convert dl class utilization with CPU capacity */
        dl_util = get_dl_cpu_capacity(cpu) * capacity_orig_of(cpu);
        dl_util = dl_util / SCHED_CAPACITY_SCALE;

        total = cfs_util + rt_util + dl_util;
        return total;
}
```

```
Total util = CFS util
           += CFS boost margin
           += RT util
           += DL util
```

Optimization target CPU with lower OPP

EAS r1.3



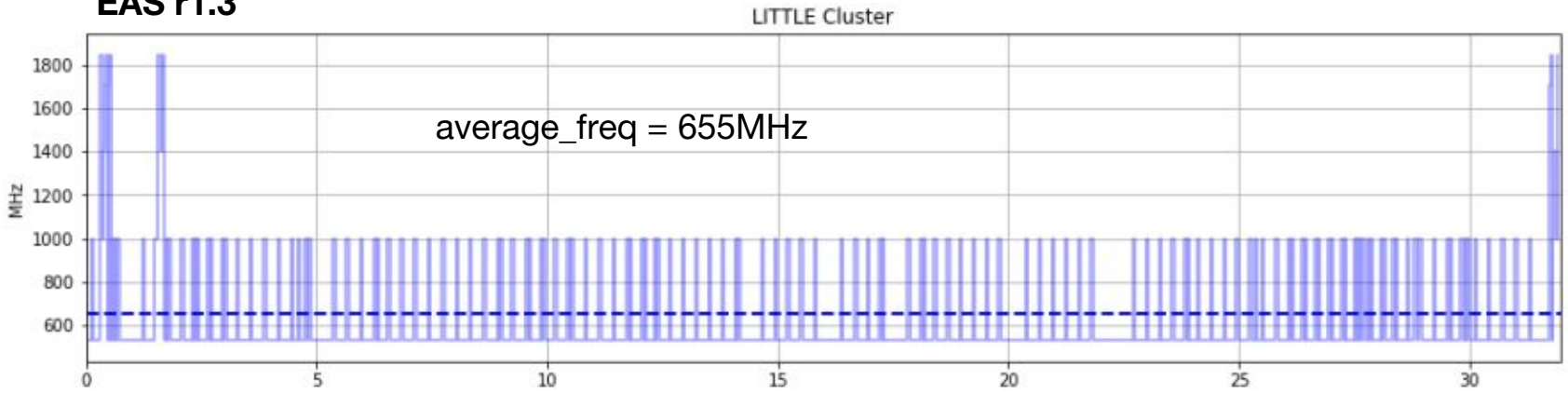
Find one target CPU with lowest util (**includes IDLE CPUs**)

```
if (idle_cpu(i)) {
     [...]

     break;
}

/* Favor CPUs with maximum spare capacity */
if ((capacity_orig - new_util) <
    target_max_spare_cap)
     continue;

target_max_spare_cap = capacity_orig - new_util;
target_cpu = i;
```

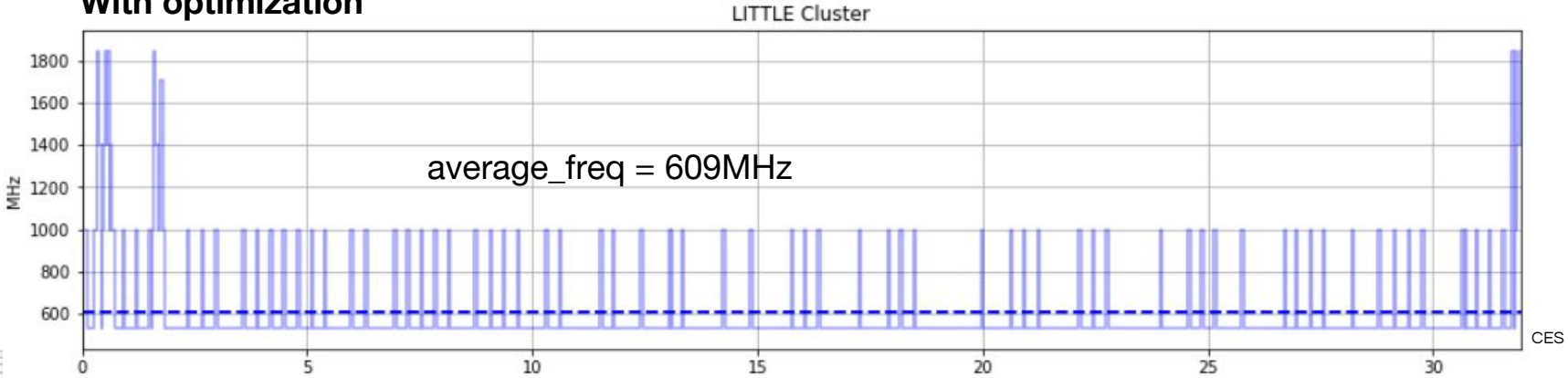
Based on Joonwoo Park's patch: "sched/fair: take CPU energy cost into consideration for placement", **this patch is in reviewing and not merged into AOSP**.

Optimized frequency for video playback (1080p)

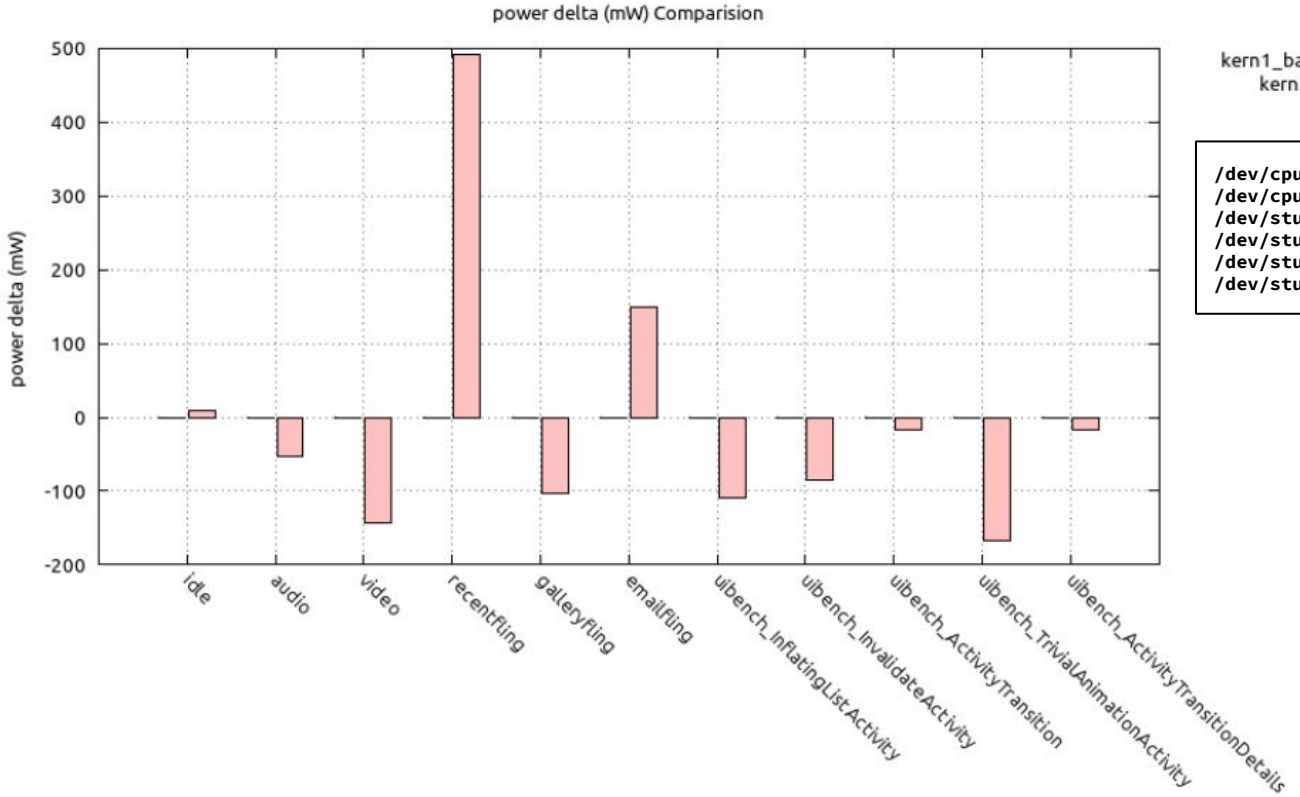
EAS r1.3



With optimization



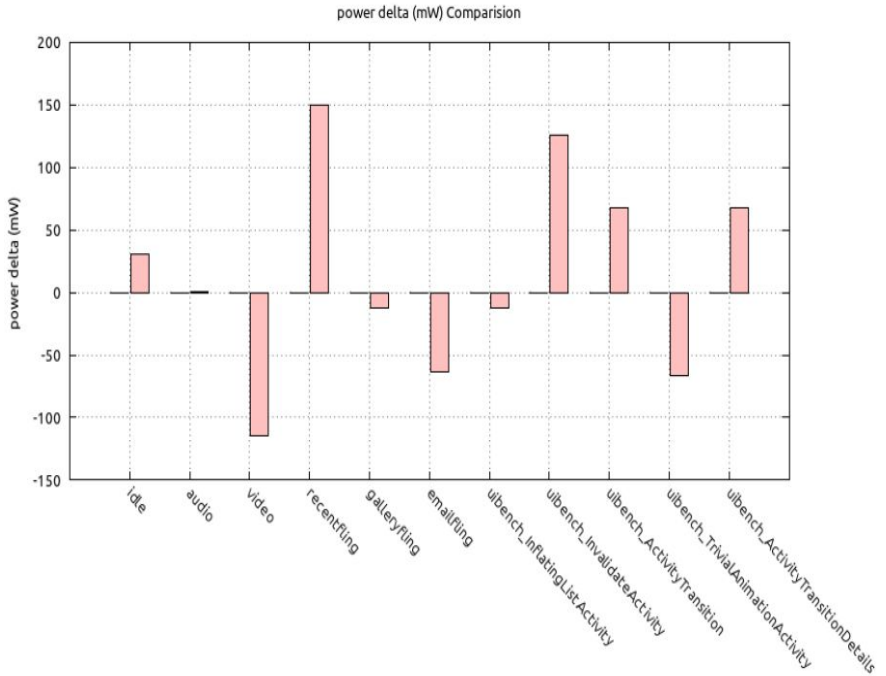
Power comparison (**boost%=0**)

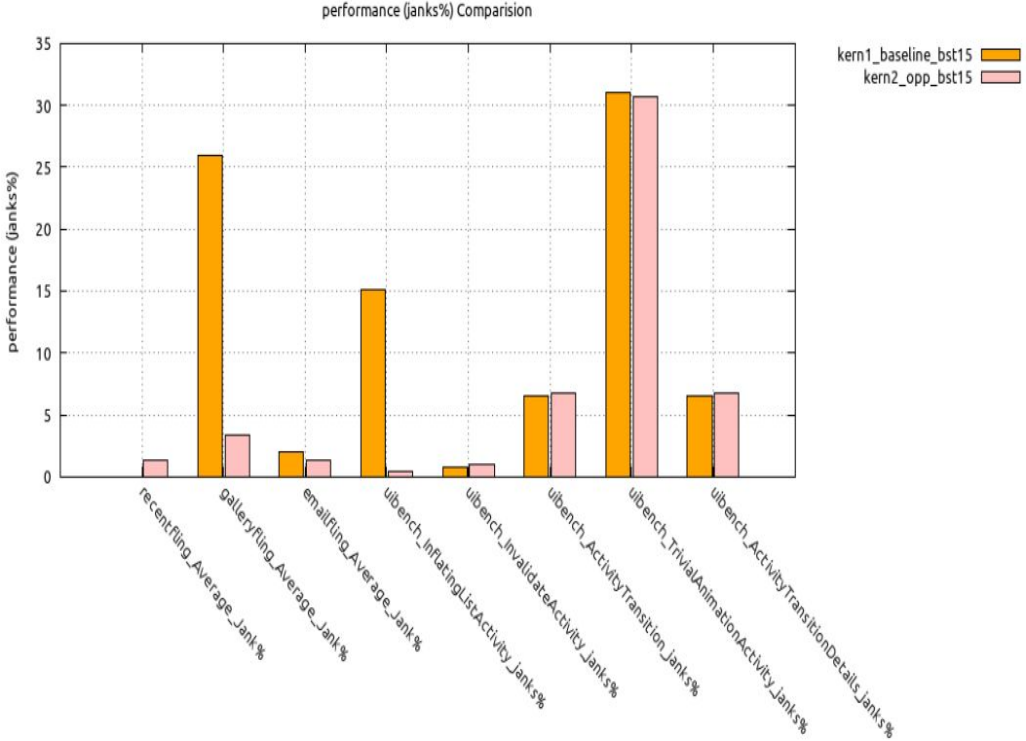



```
/dev/cpuset/background/cpus: 0-3
/dev/cpuset/system-background/cpus: 0-3
/dev/stune/foreground/schedtune.boost: 0
/dev/stune/foreground/schedtune.prefer_idle: 1
/dev/stune/top-app/schedtune.boost: 0
/dev/stune/top-app/schedtune.prefer_idle: 1
```

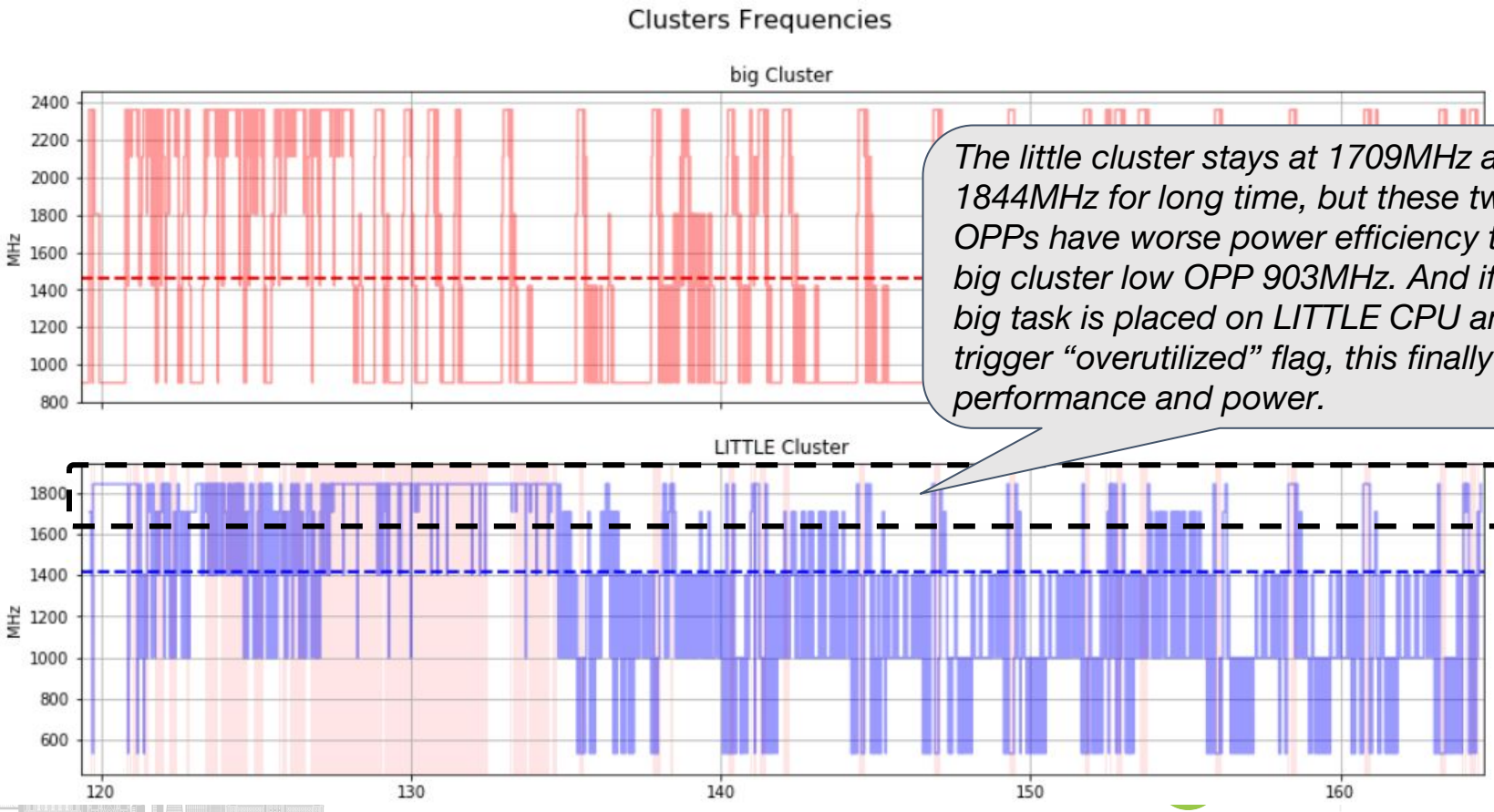
Power and performance comparison (**boost%=15**)

```
/dev/cpuset/background/cpus: 0-3
/dev/cpuset/system-background/cpus: 0-3
/dev/stune/foreground/schedtune.boost: 15
/dev/stune/foreground/schedtune.prefer_idle: 1
/dev/stune/top-app/schedtune.boost: 15
/dev/stune/top-app/schedtune.prefer_idle: 1
```





Problems for heavy workload (**will discuss in hacking session**)



Linaro
connect
San Francisco 2017

ENGINEERS
AND DEVICES
WORKING
TOGETHER

Overview

- Current status of EAS r1.3 on Hikey960
- Testing methodology
- Optimization with CGroup
- Observed issues and proposed patches
- Conclusion

Conclusion

- EAS on Hikey960 is working well for EAS development; suggest to use UEFI+ARM-TF firmwares and latest MCU firmware
- CPUSET/SchedTune is the recommended interface for simple EAS tuning by device manufacturers for quick optimization
- Investigation of EAS on Hikey960 has shown some good ideas for further improvements and some of these are being proposed on Google code review for a future AOSP common kernel

Conclusion - cont.

- Proposed optimization
  - Firstly we need consistent method for CPU utilization estimation which includes utilization contributed by RT and DL tasks
  - The power optimization can be achieved by optimization the active CPU frequency; due it has side effects for spreading tasks to more spare capacity CPU so also can benefit some interactive cases (**Optimize frequency for the IDLE CPU?**)
  - Using cpumask to give more chance for energy calculation, it can significantly save power for interactive cases and sustainable cases (**Will discuss in hacking session**)

Related materials

- Workload-automation scripts
  - Add support for Hikey960 board
  - Add support for flashing kernel/dtb images
  - Add support for fan as cooling device
  - Add workload Uibench test case
  - Add workload browserfling galleryfling emailfling
  - Result comparison and plot results for different testing configurations
- Kernel patches on gerrit but need to rebase on EAS r1.4
  - https://android-review.googlesource.com/#/c/kernel/common/+/453616/

Linaro connect
San Francisco 2017

Thank You

#SFO17

SFO17 keynotes and videos on: connect.linaro.org

For further information: www.linaro.org