Schedutil and SCHED_DEADLINE

OSPM-summit 17

Juri Lelli - juri.lelli@arm.com
Claudio Scordino - claudio@evidence.eu.com

Frequency/CPU scaling

- Based on Luca’s bandwidth reclaiming (GRUB)
- Key idea: set CPU frequency based on rq’s active bandwidth (GRUB-PA)
- Reservation runtime needs scaling according to frequency and CPU max capacity
- for freq., use the ratio between max and current capacity to enlarge the runtime granted to a task at admission control time:

```
scaled_runtime = original_runtime * (max_cap / curr_cap)
```

- similarly for CPU, but using the ratio between biggest and current CPU capacity

Frequency scaling (example)

HiKey board has 5 Operating Performance Points (OOPs)

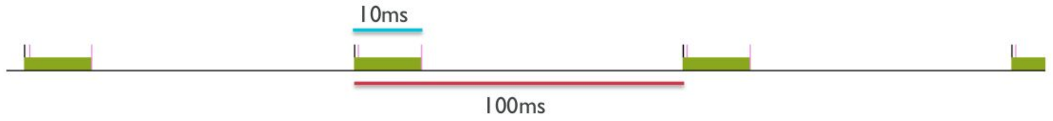
Frequency (MHz)	Capacity	% w.r.t. max
208	178	17
432	369	36
729	622	61
960	819	80
1200	1024	100

Running a task inside a 12ms/100ms reservation at min freq. means

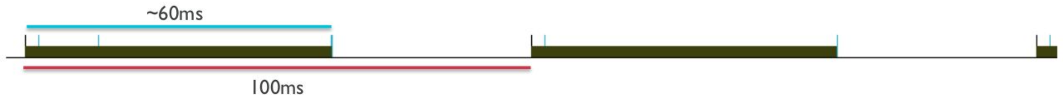
```
scaled_runtime = 12ms * (1024/178) ~= 69ms
```

Frequency scaling (example cont.)

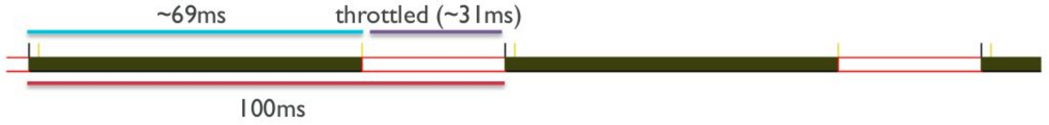
10ms/100ms task inside a 12ms/100ms reservation (at max freq)



10ms/100ms task inside a 12ms/100ms reservation (at min freq)



20ms/100ms (bad) task inside a 12ms/100ms reservation (at min freq)



Driving frequency selection

- scaling clock frequency, while meeting tasks' requirements (deadlines)
- scheduler driven CPU clock frequency selection
  - schedutil cpufreq governor
    SCHED_NORMAL - uses `util_avg` (PELT)
    SCHED_FIFO/RR and SCHED_DEADLINE - go to max!
- with Luca's bandwidth reclaiming
  - `rq->dl.running_bw` as SCHED_DEADLINE per-CPU util contribution (sum with others)
  - move CPU frequency selection triggering point (where `running_bw` actually changes)
- allow sugov kworker thread(s) to be SPECIAL (always preempt) - **for `!fast_switch_enabled` drivers**

Current design choices

- rq’s bandwidth used for freq. scaling:
  1. Active bandwidth (`running_bw`): ←
     - More aggressive
  2. Total bandwidth (`this_bw`):
     - It also accounts for inactive tasks (i.e. more conservative)
     - Could even work on current DL providing that we add rq’s bandwidth info (Luca’s patch 0007)

- freq. used for runtime accounting:
  1. Current value when calling `sched_class->update_curr()` (inaccurate) ←
  2. Notification mechanism to inform DL of frequency changes (overhead)
  3. Prevent CFS from changing freq. when there is DL load (inefficient)

SCHED_FLAG_SPECIAL, yuck!

- Bandwidth Inheritance on a busy I2C/SPI bus (mutexes)
- Make kworker go away - freq transitions in atomic context
- Some HW might work as “fire and forget”
  - what about HW that can’t ?
    set new voltage, wait for the voltage to settle down, set new clock freq. (might take a while)
- SW rework seems a daunting task :(
  - drivers use mutexes - easy to fix
  - clk framework uses mutexes - fixable/avoidable?
  - notifiers - some other subsys. rely on them, e.g. thermal
  - regulators use mutexes - hard to rework?