EAS: sched-DVFS and SchedTune
Status and next steps

Juri Lelli
Patrick Bellasi
Agenda

Sched-DVFS
- Brief intro
- Current status
- Open questions/next-steps

SchedTune
- Brief intro
- Current status
- Open questions/next-steps

Can’t cover all of the above here
- Follow on deep dives in the hacking rooms
Based on CFS’ per-entity load tracking (PELT)
Plus optimizations: jump to max OPP
Key trigger points
- enqueue_task_fair()
- dequeue_task_fair()
- task_tick_fair()
sched-DVFS: status

- Mike's RFCv3 was the last posting on this topic
- In our EAS RFCv5 posting we added additional optimisations
- Want to work with Mike to plan next steps
- At LPC the landscape for sched-DVFS has changed to some extent (next slide)

A desired outcome from this discussion:
  - Who is working on what?
  - What general time scales can they commit to?
sched-DVFS: Open questions/next-steps

Policy implementation
- Do we have broad agreement on the current design?
- Do we need to have a scheduler policy governor interface (similar to but different from cpufreq)?
- Who's working on this?

Locking implementation
- PeterZ wants RCU locking APIs to be used in hot paths shared with cpufreq core
- Who should be working on this?

sched-DVFS for other scheduler classes
- Reports that RT threads aren't getting serviced (unlike interactive governor)
- Who should be working on this?
SchedTune: Intro

Aims to service the long standing maintainer request for a simple, central tunable for

- **Power reduction vs performance boosting**

Extends Sched-DVFS for **OPP selection** and EAS for **task placement**

- provides sched-DVFS with behaviours similar to other governors
e.g. Interactive, performance
- supports EAS to trade-off power reduction for performance boosting

Fosters the collection of sensible information from user-space

- To support better task scheduling decisions
- Provides a simple, effective API to middleware
e.g. Android, ChromeOS
SchedTune: status

Posted RFCv1 on LKML [1]

- Extends sched-DVFS: biases OPP selection
- Provides global and per-task tunable (based on cgroups)
- Testing on ARM TC2, ARM Juno, partner silicon
- Used an extended version of rt-app to report a “performance index”

Has triggered an interesting discussion with lots of valuable feedback

Evaluate the benefits of the per-task approach

- Is it worth it?
- Is cgroups the best user space API for this? Alternatives (syscall, prctl)?

Integration with EAS to bias task placement

- Definition of power-performance space
- Energy_diff() filtering
- Initial prototype already available [1]

Evaluation wrt existing cpufreq governors

- What use-cases should we focus on for comparison?
- How to setup a common workflow and shareable experimental data?

[1] https://chromium-review.googlesource.com/#/c/296883/4