A whirlwind tour of Energy-aware Scheduling

5+ years of upstream development and discussions in 30 minutes

LEADING COLLABORATION IN THE ARM ECOSYSTEM

Amit Kucheria
Sr. Kernel Engineer, Linaro
Devconf.in 2018

About

- Me
  - Was Tech Lead: PMWG, 96Boards, now back to hacking
  - Have worked for Nokia, Canonical (Ubuntu), Linaro
  - Worked on wireless routers, mobile phones, tablets, laptops, embedded boards
- This Talk
  - Quick intro to the problem and where we are today
  - Not meant to capture all details of the project

What is EAS?

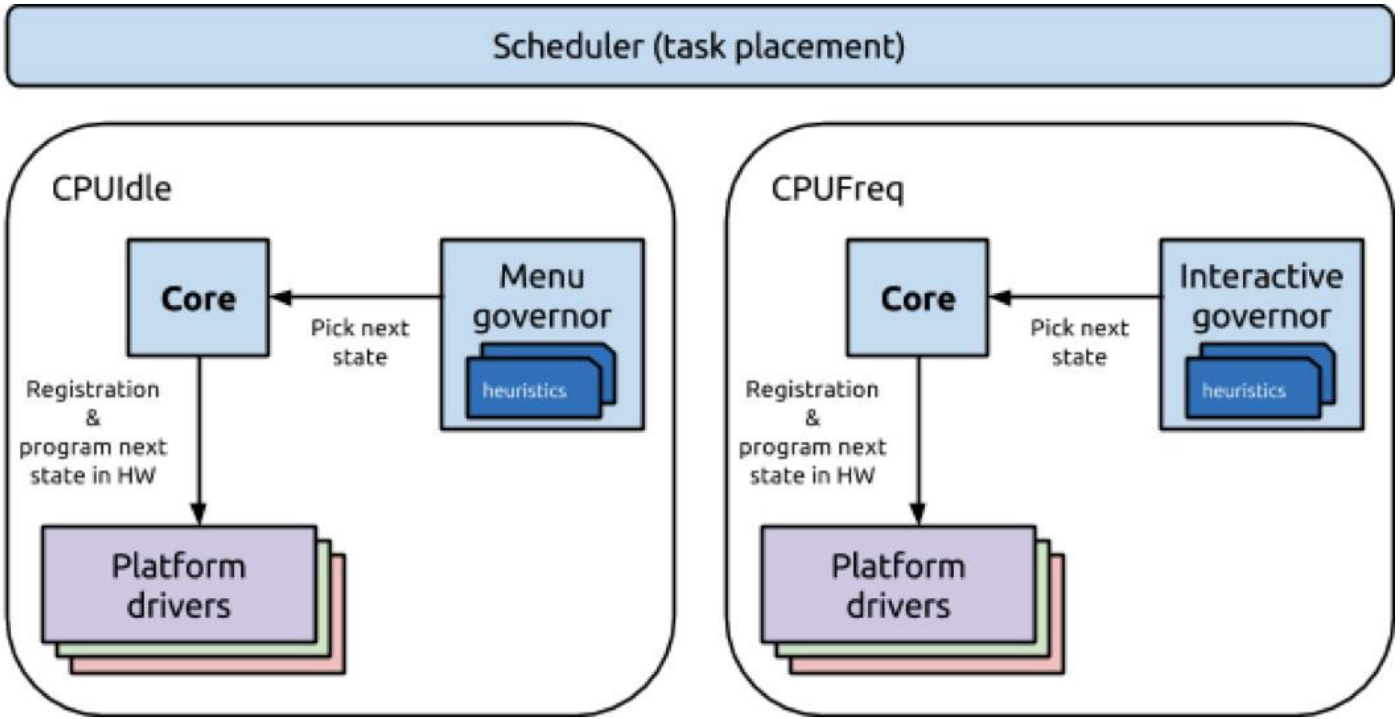
- Started with ARM’s big.LITTLE architecture
  - Not restricted to b.L though
  - Now it matters what cpu your task runs on
  - Throughput not the only metric, energy cost important too

History Lessons

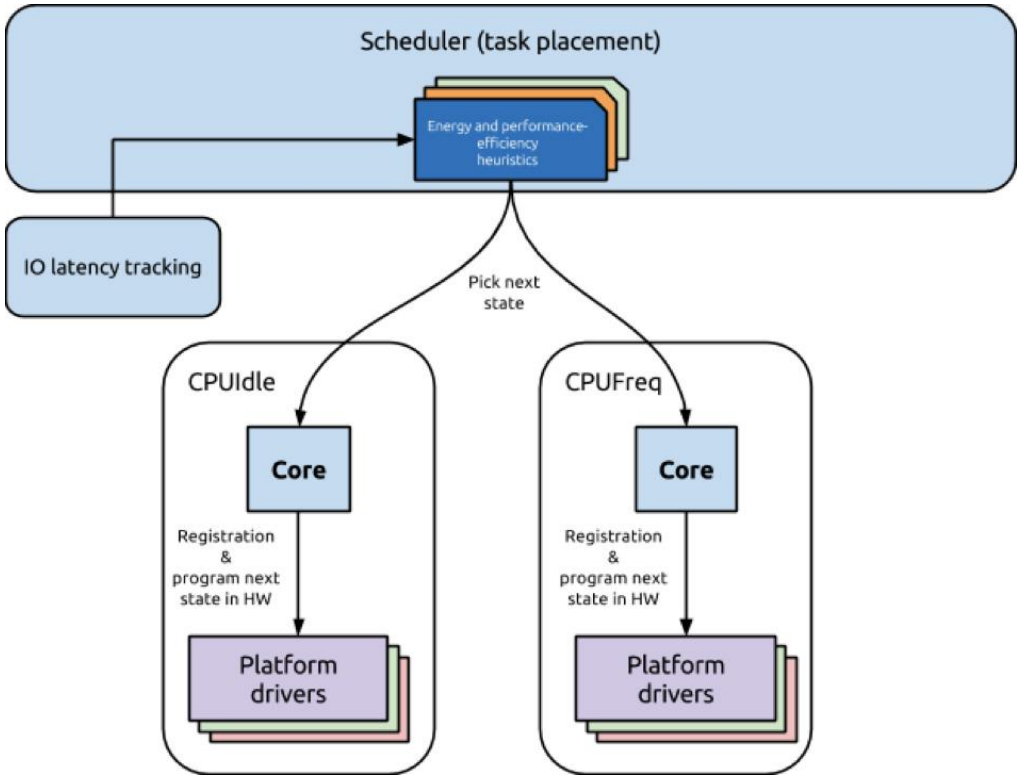
Design limitations of CPUFreq, CPUIdle

Do not coordinate their decisions with task scheduling decisions

History Lessons



Goal


Scheduler (task placement)
Energy and performance-efficiency heuristics
IO latency tracking
Pick next state
CPUIdle
Core
Registration & program next state in HW
Platform drivers
CPUFreq
Core
Registration & program next state in HW
Platform drivers

Tried piecemeal approaches

- In-kernel switcher (IKS)
- Small-task packing
- Separate power scheduler

The maintainers put their foot down

31st May 2013, Ingo Molnar on LKML:

"Today the power saving landscape is fragmented and sad: we just randomly interface scheduler task packing changes with some idle policy (and cpufreq policy), which might or might not combine correctly."

....

"_All_ policy, all metrics, all averaging should happen at the scheduler power saving level, in a single place, and then the scheduler should directly drive the new low level idle state driver mechanism."

...

"This is a "line in the sand", a 'must have' design property for any scheduler power saving patches to be acceptable - and I'm NAK-ing incomplete approaches that don't solve the root design cause of our power saving troubles..."

Problems solved along the way

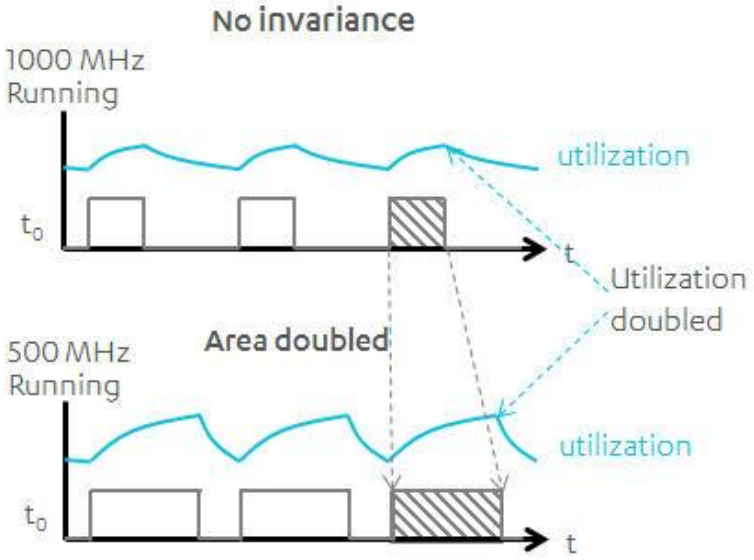
- Scheduler had no notion of power topology of the SoC [SD_SHARE_POWERDOMAIN]
- Linux assumed all cores are equal

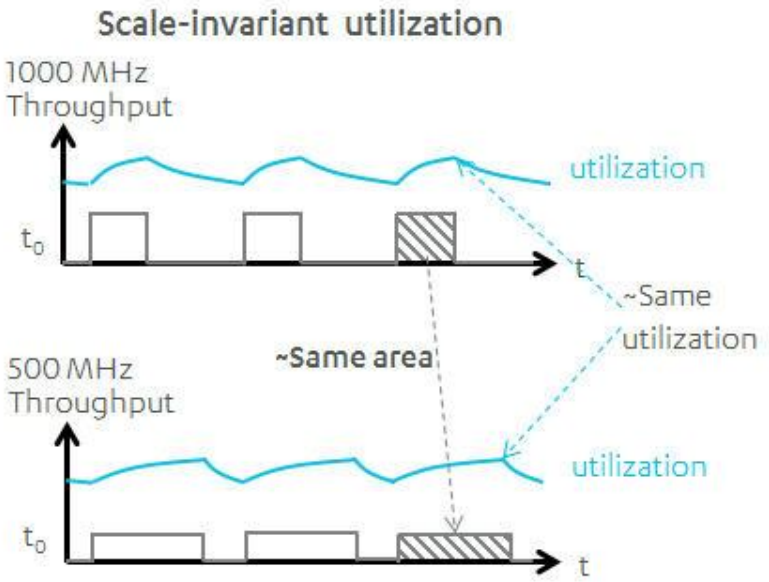
  [SD_ASYM_CPUCAPACITY, cpu capacity used in utilisation calculations now]
- Integration of the idle task into the scheduler
  - Scheduler knows about C-state and tries to pick one with the shallowest C-state

Problems solved along the way

- Scale invariance
  - 20% load at 500MHz becomes a 10% load at 1GHz, even though the actual work done is the same
- Scheduler-driven governor [schedutil]
  - Allow voltage scaling in sleeping context

Utilisation: Before and after





Note in the above, the utilization calculation is incorrect at 500MHz

Problems solved along the way

- Tools for workload simulation and repeated experiments
  - rtapp, workload-automation

Problems still being discussed

- Energy model for the SoC
  - Combine with thermal energy model used in power allocator governor and simplify
- Problems with PELT for interactive workloads
  - Merging the best of WALT into PELT
- Schedtune
  - A way to boost available capacity by cgroup

References

- LWN article on EAS
- Energy-Aware Scheduling (EAS) Project blog post, update
- EAS mainlining plans
- Energy model generation

Le Fin

Questions?

amit.kucheria@linaro.org