Android and EAS

Development

- Linux Kernel Mailing List is (of course) where upstream discussions happen
  - This is preferred path (upstream -> backport to AOSP)
- eas-dev mailing list is channel for “generic” EAS discussions (https://lists.linaro.org/pipermail/eas-dev/)
- Android targeted open development happens on AOSP gerrit:
  - Anyone interested in proposing improvements is welcome to post patches!
  - Use of hashtags/topics for groups of changes helps navigate changes, examples:
    - https://android-review.googlesource.com/#/q/hashtag:"eas_1.4_preview"
    - https://android-review.googlesource.com/#/q/topic:schedutil-walt
  - Relevant patches accepted upstream are backported, discussed on gerrit and merged in Android kernel/common as needed (one goal is to reduce delta with upstream)
  - Patches submitted must be tested for power/perf
    - Ideally the patch should specify what testing was done in comments
    - If not “standard” tests like hackbench add to the gerrit comments

Validation

- Interactivity:
  - UIBench (in AOSP)
  - SystemUI (in AOSP)
- Energy focused:
  - In particular loads that run for long time, for instance: YouTube, Camera
- Throughput:
  - Binder throughput test (in AOSP)
- Tools:
  - LISA (includes synthetic support with rt-app)
  - Scheduler workloads micro-conference
- Future validation
  - Jank tests for real world apps (Gmail, YouTube) ?
  - Running hackbench / unixbench on x86 to make sure patches upstream ready (?)
  - Boot time validation - using boottime validation tools

What did get merged upstream (w.r.t. last year) ?

- Capacity Awareness
  - "... performance on systems with asymmetric compute capacities ..."
  - https://marc.info/?l=linux-kernel&m=147645255724470
- DT bindings (capacity-dmips-mhz)
  - https://marc.info/?l=linux-kernel&m=147671927313798&w=2
- Refactoring of topology code
  - DT bindings parsing added quite a bit of arm/arm64 duplicated code, needed to fix that
  - https://marc.info/?l=linux-kernel&m=149625018223002&w=2
- PELT fixes (quite a lot :)
  - Propagation of signals across group levels
  - Task migrations across CPUs
- schedutil fixes/improvements
  - iowait_boost
  - remote callbacks
  - Timing reference for stale util contribution

What is in flight ?

- CPU/Frequency Invariance Engine
  - make use of capacity-dmips-mhz and actual clock frequency to scale task’s utilization
  - https://marc.info/?l=linux-kernel&m=150367155611291&w=2
- Add utilization clamping to the CPU controller (AKA Schedtune v4)
  - https://marc.info/?l=linux-kernel&m=150359816825787&w=2
- UTIL_EST
  - “improve some PELT behaviors to make it a better fit for the description of tasks which are common in embedded mobile use-cases”
  - https://marc.info/?l=linux-kernel&m=150365643406736&w=2
- Wakeup widening based on wake_q length
  - https://patchwork.kernel.org/patch/9895261/

What is in flight ? (cont.)

- SCHED_DEADLINE CPU/freq invariance and schedutil frequency selection
  - https://marc.info/?l=linux-kernel&m=149924523628277&w=2
- Wakeup & load balance fixes discovered while analysing benchmark response
  - Improve utilization on big.LITTLE: https://patchwork.kernel.org/patch/9885833/
  - Misc wakeup fixes:
    - find_idlest_group fixes https://lkml.org/lkml/2017/8/31/378
    - Fix for missing util sync: https://patchwork.kernel.org/patch/9876769/
- Sync flag proposal
  - Improve binder throughput test results by ~20%
  - Link: https://patchwork.kernel.org/patch/9923643/
- Skip cpufreq update on last DEQUEUE_SLEEP:
  - https://patchwork.kernel.org/patch/9910019/

Ideas on Future Improvements

- Already in AOSP
  - Energy model/awareness
  - Misfit tasks
  - No HZ signals updates (partially in AOSP)
  - Thermal capping Awareness in the scheduler
- Energy Model
  - Simplification by removing cluster/idle bits?
  - Awareness by the scheduler (use of energy_diff)
- Non-global overutilized
  - Is posted in eas-dev
- RT
  - PELT for RT-rq basis posted on LKML
  - Capacity awareness?

Align with mainline (from a kernel/common POV)

Goal is to keep AOSP kernel/common and Linux master (scheduler, cpufreq) as close as possible

- ✓ schedutil used by default
- ✓ capacity awareness backport
- ✓ PELT fixes/changes (partially)
- ✕ deprecate/remove schedfreq
- ✕ WALT (currently comparing it with PELT)
- ✕ Schedtune
- ✕ find_best_target()
- ✕ Energy model/awareness

Runtime/conf visible changes w.r.t. last year

- is_big_little is gone
- schedutil (instead of schedfreq)
- ...

Questions?

Backup Slides

- Schedtune v4
- UTIL_EST
- Find_best_target
- DEADLINE