Agenda

- Deadline scheduling (SCHED_DEADLINE)
- Why is development now happening (out of the blue?)
- Bandwidth reclaiming
- Frequency/CPU scaling of reservation parameters
- Coupling with frequency selection
- Group scheduling
- Future
CHAPTER 1
What and Why
Agenda

- Deadline scheduling (SCHED_DEADLINE)
- Why is development now happening (out of the blue?)
- Bandwidth reclaiming
- Frequency/CPU scaling of reservation parameters
- Coupling with frequency selection
- Group scheduling
- Future
Deadline scheduling (previously on ...)

- mainline since v3.14
  30 March 2014 (~3y ago)
- it’s not only about deadlines
  - RT scheduling policy
  - explicit per-task latency constraints
  - avoids starvation
  - enriches scheduler’s knowledge about QoS requirements
- EDF + CBS
- resource reservation mechanism
- temporal isolation
- ELC16 presentation
  https://goo.gl/OVspul
Agenda

- Deadline scheduling (SCHED_DEADLINE)
- Why is development now happening (out of the blue?)
- Bandwidth reclaiming
- Frequency/CPU scaling of reservation parameters
- Coupling with frequency selection
- Group scheduling
- Future
Why is development now happening

- **Energy Aware Scheduling (EAS)**
  - extends the Linux kernel scheduler and power management to make it fully power/performance aware ([https://goo.gl/vQbUOu](https://goo.gl/vQbUOu))
  - scheduler modifications pertain to SCHED_NORMAL (so far)

- **Android Common Kernel**
  - EAS has been merged last year ([https://goo.gl/FXCdAX](https://goo.gl/FXCdAX))
  - performance usually means meeting latency requirements
  - considerable usage (and modifications) of SCHED_FIFO
  - SCHED_DEADLINE seems to be a better fit
    and mainline adoption of required changes should be less controversial

- Joint collaboration between ARM and Scuola Superiore Sant’Anna of Pisa
CHAPTER 2
Let’s reclaim!
Agenda

- Deadline scheduling (SCHED_DEADLINE)
- Why is development now happening (out of the blue?)
- Bandwidth reclaiming
- Frequency/CPU scaling of reservation parameters
- Coupling with frequency selection
- Group scheduling
- Future
Bandwidth Reclaiming

- **PROBLEM**
  - tasks’ bandwidth is fixed (can only be changed with `sched_setattr()`)
  - what if tasks occasionally need more bandwidth?
    e.g., occasional workload fluctuations (network traffic, rendering of particularly heavy frame, etc)

- **SOLUTION (proposed*)**
  - bandwidth reclaiming: allow tasks to consume more than allocated
    - up to a certain maximum fraction of CPU time
    - if this doesn’t break others’ guarantees

* [https://lkml.org/lkml/2016/12/30/107](https://lkml.org/lkml/2016/12/30/107)
Bandwidth Reclaiming (cont.)

- Greedy Reclamation of Unused Bandwidth (GRUB)\textsuperscript{1}
- 3 components\textsuperscript{2}
  - tracking of active utilization
  - modification of the accounting rule
  - multiprocessor support (original algorithm was designed for UP)

- for more details you are very welcome to attend the hacking session!

\textsuperscript{1} - Greedy reclamation of unused bandwidth in constant-bandwidth servers - Giuseppe Lipari, Sanjoy K. Baruah (https://goo.gl/xI4CUk)
\textsuperscript{2} - Greedy CPU reclaiming for SCHED DEADLINE - Luca Abeni, Juri Lelli, Claudio Scordino, Luigi Palopoli (https://goo.gl/e8EC8q)
Bandwidth Reclaiming (results)

- Task 1 (6ms, 20ms) constant execution time of 5ms
- Task 2 (45ms, 260ms) experiences occasional variances (35ms-52ms)
Bandwidth Reclaiming (results)

- Task1 (6ms, 20ms) constant execution time of 5ms
- Task2 (45ms, 260ms) experiences occasional variances (35ms-52ms)

- Cumulative Distribution Function (CDF) probability that Response time will be less or equal to x ms

![Graph showing CDF of response times for Task 1 and Task 2]
Bandwidth Reclaiming (results)

- Task 1 (6ms, 20ms) constant execution time of 5ms
- Task 2 (45ms, 260ms) experiences occasional variances (35ms-52ms)
- Plain CBS T2’s response time bigger than reservation period (~25%)
Bandwidth Reclaiming (results)

- Task 1 (6ms, 20ms) constant execution time of 5ms
- Task 2 (45ms, 260ms) experiences occasional variances (35ms-52ms)

- GRUB
  T2 always completes before reservation period (using bandwidth left by T1)
CHAPTER 3
Rock around the Clock (... and CPU)
Agenda

- Deadline scheduling (SCHED_DEADLINE)
- Why is development now happening (out of the blue?)
- Bandwidth reclaiming
- Frequency/CPU scaling of reservation parameters
- Coupling with frequency selection
- Group scheduling
- Future
Frequency/CPU scaling

- Reservation runtime needs scaling according to frequency and CPU max capacity
- for frequency, use the ratio between max and current capacity to enlarge the runtime granted to a task at admission control

\[ \text{scaled } \_\text{runtime} = \text{original } \_\text{runtime} \cdot \frac{\text{max } \_\text{capacity}}{\text{curr } \_\text{capacity}} \]

- similarly for CPU, but using the ratio between biggest and current CPU capacity
- nice example running on HiKey will be presented during the hacking session
Agenda

- Deadline scheduling (SCHED_DEADLINE)
- Why is development now happening (out of the blue?)
- Bandwidth reclaiming
- Frequency/CPU scaling of reservation parameters
- Coupling with frequency selection
- Group scheduling
- Future
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!
- once bandwidth reclaiming is in*
  - start using SCHED_DEADLINE’s per-CPU utilization contribution (sum to others)
  - move CPU frequency selection triggering points (where DL utilization changes)
  - allow sugov kworker thread(s) to always preempt SCHED_DEADLINE tasks (and lower priority) – for fast_switch_enabled drivers
- and again attend the hacking session for more details/results/fun

* Claudio Scordino (Evidence Srl) is helping with this.
CHAPTER 4

Groupies
Agenda

- Deadline scheduling (SCHED_DEADLINE)
- Why is development now happening (out of the blue?)
- Bandwidth reclaiming
- Frequency/CPU scaling of reservation parameters
- Coupling with frequency selection
- Group scheduling
- Future
Group scheduling

- Currently, one to one association between tasks and reservations
- Sometime it might be better/easier to group a set of tasks into the same reservation
  - virtual machine threads
  - rendering pipeline
  - legacy application (that for example needs forking)
  - high priority driver kthread(s)
- Hierarchical/Group scheduling\(^1,2,3\)
  - cgroups support
  - temporal isolation between groups (and single entities)

2 - Hierarchical Multiprocessor CPU Reservations for the Linux Kernel - F. Checconi, T. Cucinotta, D. Faggioli, G. Lipari (https://goo.gl/PlJaQe)
3 - The IRMOS real-time scheduler - T. Cucinotta, F. Checconi (https://lwn.net/Articles/398470/)
Group scheduling

- Hierarchical means
  - first level is EDF
  - second level is RT (FIFO/RR)

- Should eventually supplant RT-throttling
Group scheduling

- On multiprocessors

- One DEADLINE group entity per CPU
- Coexists with single DEADLINE entities
Group scheduling

- On multiprocessors

- One DEADLINE group entity per CPU
- Coexists with single DEADLINE entities
- Sub RT entities get migrated according to G-FP (push/pull)
CHAPTER 5
It IS bright!
Agenda

- Deadline scheduling (SCHED_DEADLINE)
- Why is development now happening (out of the blue?)
- Bandwidth reclaiming
- Frequency/CPU scaling of reservation parameters
- Coupling with frequency selection
- Group scheduling
- Future
Future

- **NEAR**
  - experimenting with Android
  - reclaiming by demotion towards lower priority class
  - capacity awareness (for heterogeneous systems)
  - energy awareness (Energy Aware Scheduling for DEADLINE)

- **NEAR(...ISH)**
  - support single CPU affinity
  - enhanced priority inheritance (M-BWI most probably)
  - dynamic feedback mechanism (adapt reservation parameters to task’ needs)
Get involved!

Shoot me an email <juri.lelli@arm.com>
Ask questions on LKML, linux-rt-users or eas-dev
Come join us @ OSPM-summit (https://goo.gl/ngTcgB)
... maybe remotely :-)

And don’t forget to collect your prizes!!!