arm

SCHED_{DEADLINE, RT} for capacity constrained systems

Morten Rasmussen <morten.rasmussen@arm.com>
Patrick Bellasi <patrick.bellasi@arm.com>

OSPM Summit 2019, 20-22 May, Pisa

SCHED_DEADLINE

Challenges for real use-cases on mobile platforms

Requires each task to specify bandwidth requirements, provides guarantee in return

- Determining bandwidth requirements for complex real workloads is difficult
  - Real-world use-cases have **complex dependencies** among **multiple tasks**
  - Tasks can have quite **variable demand**, e.g. frame composition time
  - **Worst-case** bandwidth reservations are too **expensive**
    - Conservative reservations restrict the number of possible reservations
    - High, mostly unused, bandwidth requirements are expensive in energy
- (Too) simple view on CPU compute capacity
  - Assumes CPUS with **symmetric capacity** and performance **never capped**, while many systems have:
    - **DVFS**: Some governors are not aware of bandwidth guarantees
    - **SMT**: Compute bandwidth of sibling threads is hard to predict
    - **Asymmetric CPU capacity**: Not all CPUs in the systems may be able to deliver the reserved bandwidth
    - **Performance capping**: For a range of reasons, including thermal, CPUs might not be able to guarantee delivery of high bandwidth

SCHED_DEADLINE

What level of guarantee should SCHED_DEADLINE provide?

Weak guarantees

- Pros
  - Admission control can be optimistic
    i.e. accept nearly full utilization of CPUs
  - No sustainable performance level info needed
- Cons
  - Guarantees might be violated
    by performance capping at OS and/or HW/FW level
  - Tasks need notifications
    when their reservations are not honoured
  - DL might not be useful for some use-cases
    e.g cases that don't handle broken guarantees well

Hard(er) guarantees

- Pros
  - Users can (mostly) trust guarantees
    Only break when there are bigger problems than bandwidth reservations not being honored
- Cons
  - Guaranteed performance level reported by HW
    is needed for admission control
  - Allowed CPU utilization is likely to be pessimistic
  - DL might not be useful for some use-cases
    e.g, cases requiring relatively high bandwidth

What about mixed guarantees?

hard guarantee tasks admitted based on sustainable capacities
weak guarantee DL tasks admitted on "exceeding capacity"

arm

SCHED_DEADLINE
Suggestions for improvements

- Capacity awareness: Admission control and task placement (RFC by Luca Abeni)
- Cpufreq policy/governor integration (exists already for sched_util)
- Guaranteed performance levels from HW/FW integrated with admission control and task placement.
- Energy-aware task placement

SCHED_RT
Outstanding issues for mobile systems

- **Priority based** scheduling class used for **latency sensitive** tasks
- Current limitations hindering RT use in mobile systems:
  - Running tasks at the highest CPU capacity is ***too expensive*** and ***not always required***
  - Assumes symmetric CPU capacities
  - Completely ignore CFS tasks

- Possible improvements for discussion:
  - Capacity-awareness to get more predictable performance on asymmetric capacity systems?
    - e.g. avoid an RT task to SYNC_WAKE a CFS task on a LITTLE CPU
      by adding support for RT entities PELT and/or UtilClamp constraints
  - Per task(_group) performance constraints to better guide cpufreq decisions, i.e. UtilClamp?
    - Make RT aware of already busy CPUs
      i.e. if possible, avoid to preempt CFS tasks?

Recap and action items

Focus and priorities?

DEADLINE Scheduler

- Capacity awareness
  - Partitioned CBS with enter/exit time tasks placement?
  - Guaranteed bandwidth allocation with standardized HW/FW integration?
  - Support mixed Hard and Soft DL guarantees with graceful degradation for soft DL tasks?
- Energy awareness
  - GRUB-PA improvements…?
  - Admission time tasks placement?
- Others awareness
  - Proxy execution and/or hierarchical sched?

RT Scheduler

- Capacity awareness
  - Per task(group) constraints aware placements?
  - Preferred vs Possible cpus affinity mask?
- Energy awareness
  - Energy sensitive vs non-energy sensitive RT tasks?
- Others awareness
  - CFS busy CPUs avoidance?
  - Make CFS aware of RT busy CPUs?

arm