An update on Real-Time scheduling on Linux

Peter Zijlstra
Intel
Why deadline scheduling?

General Purpose OS

- Resource arbitration
- Resource isolation

FIFO/RR must be privileged because they violate all that.

- Misbehaving task affects other tasks
- Prio assignment is difficult and cannot easily be composed
Sporadic task model

(G)EDF scheduling provides arbitration
  – Easy composition of task sets

CBS provides isolation
  – Self suspending tasks
  – Constrained tasks must preserve density

SCHED_DEADLINE $\rightarrow$ (G)EDF + CBS
Accounting vs Enforcement

- Budget (q) is accounted in [ns]
  - Subject to platform clock resolution
- Budget depletion is tested on 'tick'
- Budget replenishment is 0-sum
'Global' EDF

- Per logical CPU runqueue
- Push on activation
- Pull on demote/idle
Hierarchy

• `pick_next_task()`
  - `class_stop`
  - `class_deadline` (DEADLINE)
  - `class_rt` (FIFO/RR)
  - `class_fair` (NORMAL)
  - `class_idle`
Inversion

Fixed Priority
- Priority inheritance

Dynamic Priority
- Deadline inheritance
- Bandwidth Inheritance
  - SMP?
Proxy Execution

Scheduling decision function invariant.

SMP tricky...
- Easy to end up executing the same task on multiple CPUs
Admission Control

(G)EDF: $U \leq m$

- Recoverable, avoids domino effect
  - Bounded tardiness
- Affinities are tricky

Proposed: $U_i = \sum_{t \cap i} \left( \frac{u_t}{w_t} \right) \leq 1$
Single CPU Affinity

- Often requested
- Expectation of UP-like behaviour
  - Mixed criticality
  - 'obvious' hierarchical EDF fails:
Mixed Criticality

EDF + LLF:

- At least 2 degrees of freedom in the model
- Laxity := \{d – e; for single CPU affine tasks, otherwise inf.
- If the EDF pick can run without the LLF pick turning 0, do so, otherwise run the LLF pick.
- Has similarities to EDZL

\[ t + e_{EDF} > d_{LLF} - e_{LLF} \]
Reclaim

- Soft-CBS
- Power Aware / Idle-reclaim
- GRUB (Greedy Reclaim of Unused Bandwidth)
  - Introduces active bw
    - \( U_{\text{act}} > 1 \) !!
  - \( dq = -U_{\text{act}} \, dt \)
  - Privileged; can consume lots of time
    - Per task / cgroup reclaim limits
Probabilistic

- Consider the per-task reclaim limit as an extension to the task model and interpret it as a measure of variance on the runtime.
- 0-sum overrun $\rightarrow$ avg, fairness
- Measurement based pWCET
Cgroups

- Cpuset → partitioning
  - AC vs partitioning broken

- Deadline
  - AC limits
  - Hierarchical CBS
Hierarchical scheduling

- CFS slack time scheduling
- FIFO servers
  - Minimal concurrency
  - Nested load-balancing
  - Arbitrary affinities are still a problem
Unprivileged

- Assume users are hostile
- Plug the BW (inheritance) hole
- DoS
  - Limits on the task model
Questions?