DL Server

RT and Scheduling MC - LPC 2023

Daniel Bristot de Oliveira, Ph.D.
Senior Principal Software Engineer
The real-time throttling mechanism is a safeguard for misbehaving real-time tasks
- \[ \text{kernel.sched\_rt\_runtime\_us} / \text{kernel.sched\_rt\_period\_us} = 950000 / 1000000 \]
- It throttles the rt_rq

- It causes the system to go idle
- It does not work for fine-grained runtime
  - Many people deactivate it - though this is a workload problem
- It does not solve the starvation from SCHED\_DEADLINE
The RT Throttling problem

**Failed attempts Temp fixes and stalls**

- **The RT_RUNTIME_GREED options**
  - If there are no starving tasks, ignore throttling and keep running

- **stalld**
  - User-space tool that monitors runqueue: If a task is not scheduled within a "timeout."
    - Boost with SCHED_DEADLINE

- **SCHED_DEADLINE Servers**
  - Back in 2017? The DL server was proposed by Peter
  - But back then, we stalled on the way not to break RT

---

![Runtime Diagram]

- DEADLINE
- FIFO/RR
- NORMAL
- DL Server
The RT Throttling problem

- What we wanted:

- What we had:

- What we thought it would be the best solution:
The fixed priority scheduler has properties that many people rely on:
  • The highest priority task runs with minimum latency
  • The EDF does not have this priority because the highest priority task changes as time goes

Using the DL server is the best way to get out of sched fair starvation
  • It boosts the entire rq
  • It fixes all the other problems we had

But activating it when not needed caused us the problem on the first bullet
  • We need the DL server only if the fair scheduler is starving

We’ve learned from stalld that waiting for starvation to be imminent was a good thing!
DL Server with deferred activation

- Anytime a fair task is active the DL Server is started:
  - **period and runtime are set** if deadline already not set in the future
  - The server **starts throttled**
    - The **replenishment timer is set to deadline - runtime** (zero laxity time of the task alone at starting time)
  - Anytime the **fair scheduler runs**, the **dl_server has its runtime consumed**
    - Even if the dl_server is not actually running
  - **If the server had enough runtime before the replenishment time:**
    - Reset runtime & period, reschedule the replenishment timer
  - **Otherwise:** replenish the dl server accordingly to the CBS rule and run as a DL task
DL Server with deferred activation

- In other words:
  - The fair dl_server is always armed
  - If fair dl_server had enough runtime, it is postponed
  - otherwise, the fair scheduler becomes a SCHED_DEADLINE task
  - In a properly loaded RT system, the DL server should not be activated!

- Interface:
  - `/debug/sched/fair_server/cpu{ID}`:
    - runtime : 50ms
    - period : 1s
    - defer : 1

- The rt throttling is confined on RT_GROUP_SCHED
  - The sysctls (sched...rt_runtime_us...) are still there to limit sched deadline bandwidth

- It works! :D
Thank you

Red Hat is the world’s leading provider of enterprise open source software solutions. Award-winning support, training, and consulting services make Red Hat a trusted adviser to the Fortune 500.

linkedin.com/company/red-hat
youtube.com/user/RedHatVideos
facebook.com/redhatinc
twitter.com/RedHat