QoS Hinting APIs

If we had them, what would they actually do?!

John Stultz <jstultz@google.com>

LINUX PLUMBERS CONFERENCE Vienna, Austria / Sept. 18-20, 2024

QoS: Problem Overview

Linux runs lots of different workloads, so the scheduler needs to be a general purpose scheduler.

- Provides classes to distinguish between some load types
- SCHED_IDLE, SCHED_FIFO/RR, SCHED_DEADLINE allow for different behaviors, but all pretty special case.
- SCHED_NORMAL is most widely used (no privs needed)

Can't know what's ideal for a every workload

- Often, system designers utilize the **global tuning knobs** to tune for a specific workload
- Taken to extremes, you get sched-ext, where you have a specific scheduler for a system's workload
- But this **isn't composable**! You can't combine multiple workloads on a system and make all of them happy!

See also:

- Youssef Esmat's OSPM24 talk on tuning EEVDF for CrOS
- Gautham Shenoy's OSPM24 talk on EEVDF tuning for Servers
- David Vernet's OSPM23 talk on sched-ext

QoS: High level approach

If applications provided hints as to their needs, the scheduler could better order things

- Similar to sched classes SCHED_RT or SCHED_DEADLINE
- But less prescriptive (and less dangerous?)

Many OSes have QoS interfaces

- Mac/iOS
- Windows
- Multiple approaches in Android
  - ADPF
  - QAPE
  - Samsung Galaxy Performance API

Generally these APIs are **trading throughput vs latency**

- Nice sometimes used as a slider here, but it's better for throughput boosting
- Latency can **be a separate axis** (ie: latency sensitive and throughput sensitive!)
- **Latency is imprecise** (task wakeup latency vs web page rendering latency)

See also:

- Len Brown's LPC2022 talk

QoS: Past discussions

At OSPM this was a common topic, and point of discussion

- Consensus that **we need "some" API** to provide this
- No consensus on what the API might look like
- Number of previous/current attempts:
  - Latency_nice
  - Qais's sched-qos sched_attr additions

"**Hint**" is a important detail: Kernel isn't going to make any contract promises here. Kernel maintainers want flexibility to do the best thing.

- Causes some vagueness in discussion

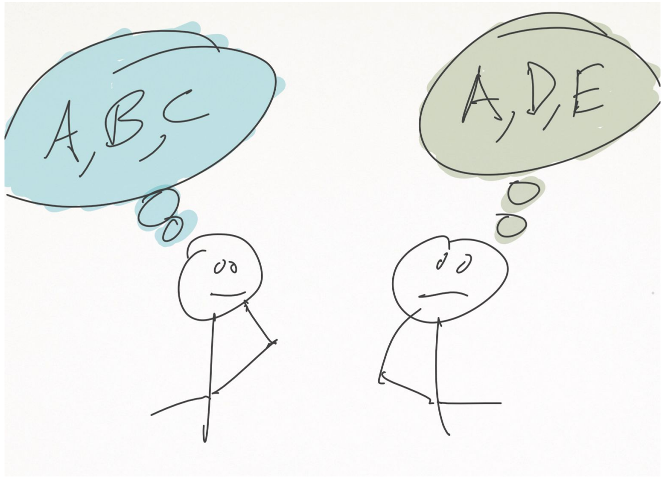
See also:

- Vincent Guittot's OSPM24 talk on latency hints
- Rafael J. Wysocki and Lukasz Luba's OSPM24 QoS talk

QoS: Complex conversation

Problem: If we wanted to reduce latency for a process, there's a lot of different things we might do

- Wakeup preemption (switch to it immediately)
- "Prefer idle" placement on wakeup
- Increase the cpufreq faster
- Up-migrate to bigger cpus faster
- Bump placement in the runqueue, so it will run sooner
- Let task run for longer so it can finish its work sooner
- …



I suspect part of the issue is we're not all really talking about the same thing!

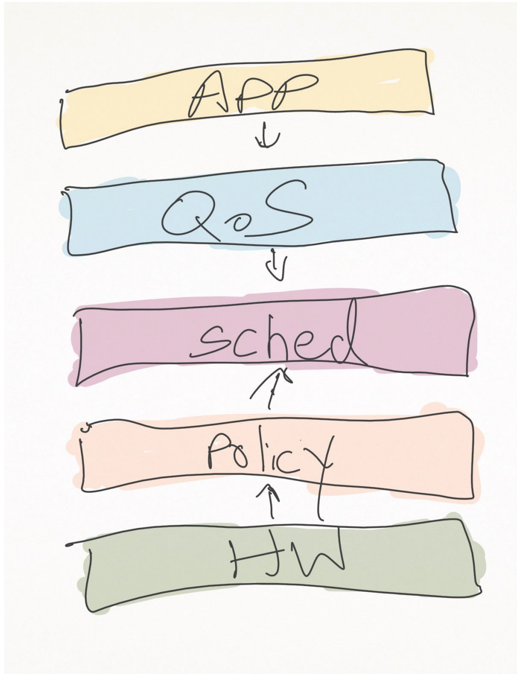
QoS: Lots of choices

Further those choices might depend on things:

- preempting a another latency sensitive task
- idle cpu is in deep sleep, and could take awhile to wake up
- idle cpu is a small cpu. We might start running sooner, but might not have capacity we need

The best choice here may depend on the system we're on.

- The scheduler needs to know these details in order to make the right choices
- Userland can hint needs, but scheduler must be aware of hardware capabilities and constraints.
- How do we inform the kernel of these system specific values?
- How do we develop policy of which actions to take where?



QoS: Discussion

Enumerate kernel actions

- What actions am I missing?

How do we create policy to map the right actions for the hardware for the QoS hints?

- What details do we need from the hardware?
- cpu idle wake up latencies
- migration latencies

Are the action choices different enough that we should have separate hints from userland? (ie: finer grained hinting then just “latency sensitive”)

When	Decision	Policy Inputs
Wakeup	Placement	Cpu-wakeup-latencies, Available capacity (including thermal caps)
Wakeup	Preemption	Relative priorities of running vs waking
On runqueue	RQ Order	Relative priority
On runqueue	Slice Length	How much work to do?
Once running	Cpufreq/L3/Bus Rampup	Uclamp, mem access pattern profile
Once running	Migration Margins	Migration latency
Load-change User-Hint	Re-evaluate/ calculate utilization	Rate-limiting interval?
<More here?>	?	?

Thank you!

John Stultz <jstultz@google.com>

Linux
Plumbers
Conference

Vienna, Austria | September 18-20, 2024