# Linux Sched QoS API

Len Brown & Vaibhav Shankar - Intel

Linux Plumbers Conference Dublin, Ireland Sept 2022

# Opportunity

Multi-OS apps (eg Chromium Browser) categorize threads by QOS type.

Can we improve how those apps run on Linux?

# Linux needs a Scheduler QOS API (and it isn't nice(2))

Task placement decisions and hardware operating points impact application performance and energy efficiency.

The Linux scheduler and the hardware export low level knobs that allow an expert to influence these settings.  But that expert needs to know details about the hardware, about the Linux scheduler, and about every task that is running on the system.

This is not a reasonable ask of multi-platform applications.  Here we look at what, say Chromium, must to do run on Linux, Windows, and MacOS; and how we can improve the Linux piece of the puzzle.

# Acknowledgement: Chromium contributors

- Vaibhav Shankar
- Zheda Chen
- Zhibo Wang
- Jianlin Qui
- Richard Winterton

# Some Chromium Context

- Newly proposed thread_type API (had been thread_priority)

- Browser run by session manager under ChromeOS -- elevated privileges
- Browser run as regular user under Linux/Ubuntu etc. -- default privileges

# Linux nice(2)

Relative to every other task in the system

permission required to increase priority from default

# Linux sched: Real Time Classes

Complexity hurdle

Portability challenge

Permission (and trust) challenge

# Preemption

Helps w/ latency w/o complexity and trust burden of RT

How best to expose to apps?

# Hybrid CPU HW

fair.c (ITMT and EAS) maximize perf and efficiency

...but assume all tasks are equal (unless special util/clamping)

# Linux SCHED_IDLE

Who uses this, anyway?

(and if we are optimizing for idle, can we coalesce instead of spread?)

# Per-Task HW Hints

Efficient vs Performance Core task placement

Performance vs Efficiency HW frequency hints

Other HW hints, with tie-breakers opaque to SW...

# Windows

- The Quality of Service (QoS) associated with a thread is used to indicate the desired performance and power efficiency– QOS assigned at thread level

- Scheduling priority remains the main metric by which the system determines which thread to schedule next, QoS can influence core selection and processor power management (QoS and Thread priorities are separate)

- Commonly used QoS levels by Chromium on Windows:  High, Medium, Low, Eco

# Windows QOS

| QoS level | Description | Performance and power |
|---|---|---|
| High | Windowed applications that are in the foreground and in focus, or audible, and explicitly tag processes with SetProcessInformation or threads with SetThreadInformation | high performance |
| Medium | Windowed applications that may be visible to the end user but are not in focus. | between High and Low |
| Low | Windowed applications that are not visible or audible to the end user. | DC: selects most efficient CPU frequency and schedules to efficient core |

**Credits:**
Microsoft-Documentation

# Windows QOS (cont..)

| QoS level | Description | Performance and power |
|---|---|---|
| Utility | Background services | DC: selects most efficient CPU frequency and schedules to efficient cores |
| Eco | Applications that explicitly tag processes with SetProcessInformation or threads with SetThreadInformation. | AC+DC: selects most efficient CPU frequency and schedules to efficient cores. |

**Credits:**
Microsoft-Documentation

# Windows QOS (cont..)

| QoS level | Description | Performance and power |
|---|---|---|
| Media | Threads explicitly tagged by the Multimedia Class Scheduler Service to denote multimedia batch buffering | CPU frequency reduced for efficient batch processing |
| Deadline | Threads explicitly tagged by Multimedia Class Scheduler Service to denote that audio threads require performance to meet deadlines | High performance to meet media deadlines |

**Credits:**
Microsoft-Documentation

# MacOS

- The system uses QoS information to adjust priorities such as scheduling, CPU and I/O throughput, and timer latency
- Apple schedules most of the threads/Process initially in E-cores and as thread/process need more compute, move to P-core
- Chromium on M1 uses the following QOS classes- Background, User initiated, and User interactive

# MacOS - QOS

| QoS Class | Type of work and focus of QoS | Power and Perf | Duration of work to be performed |
|---|---|---|---|
| User interactive | Work that is interacting with the user, such as operating on the main thread, refreshing the user interface, or performing animations. | Responsiveness and performance | Work is virtually instantaneous |
| User initiated | Work that the user has initiated and requires immediate results, such as opening a saved document or performing an action when the user clicks something in the user interface. | Responsiveness and performance | Work is nearly instantaneous, such as a few seconds or less |

**Credits:**
Apple-developer-documentation

# MacOS - QOS

| QoS Class | Type of work and focus of QoS | Power and Perf | Duration of work to be performed |
|---|---|---|---|
| Utility | Work that may take some time to complete and doesn't require an immediate result, such as downloading or importing data. Utility tasks typically have a progress bar that is visible to the user. | Providing a balance between responsiveness , performance, and energy efficiency | Work takes a few seconds to a few minutes |
| Background | Work that operates in the background and isn't visible to the user, such as indexing, synchronizing, and backups. | Energy efficiency | Work takes significant time, such as minutes or hours |

**Credits:**
Apple-developer-documentation

# pthread_attr_set_qos_class_np(3)

_np = non-portable

Apple's qos.h API definition published under APSL

# Example Chromium Linux Scheduler Ask

- Ability to designate periodic media tasks such that:
  - Run fast enough to meet deadline
  - while running at efficient operating point
    - eg. Ecores w/ low frequency

Example: 30fps Video conferencing:

Audio 10ms

Camera Encode/Decode , Composition- 33ms

Webrtc n/w ~1ms or less

Mojo IPC (Inter Process Communication) ~1ms or less

# What if: add sched_setattr(2) QoS, plumb to pthreads?

| sched_setattr(2) *QOS_* | Preempt | Class | Pri | Per-task QOS hint (HW and OP/placement) |
|---|---|---|---|---|
| MEDIA | Enabled | NORMAL | 0 | max performance |
| INTERACTIVE | | NORMAL | 0 | performance balance |
| DEFAULT | | NORMAL | +5 | balance |
| UTILITY | | NORMAL | +10 | power balance |
| BACKGROUND | | NORMAL (IDLE-class) | +20 | max efficiency |

# Discussion