Worker Quality of Service - TPAC 2023 Breakout Session

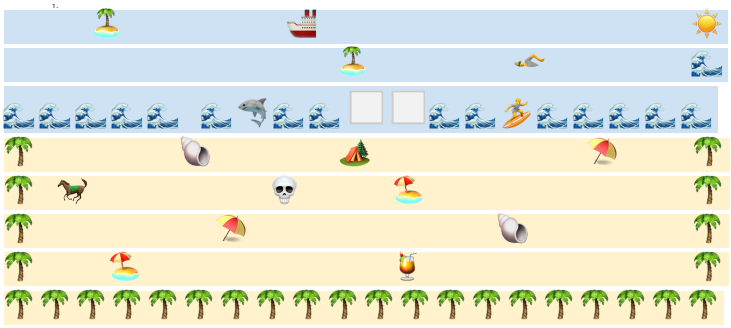
Present

- Ian Clelland (Google Chrome)
- Austin Sullivan (Google Chrome)
- Andrew Sutherland (Mozilla Firefox)
- Steven Rostedt (Google ChromeOS)
- Sangwhan Moon (TAG, Google, not Chrome)
- Paul Adenot (Mozilla Firefox)
- Zhibo Wang (Intel Software, online)
- Wojciech Filipek (eyeo)
- Youssef Esmat (Google ChromeOS, online)
- Hongchan Choi (Google Chrome)
- Randell Jesup (Mozilla Firefox)
- Lei Zhao(China Mobile, online)

Docs

- Explainer

🅿️ Cursor beach relaxing



Notes

- Riju presents slides

- THe main thread is overworked and underpaid, wherever possible we should move work off of it to workers
- With screen refresh rates higher and higher, should move all JS off; 16ms at 60fps; 144fps is much lower.
- Smoothness is a key metric of user happiness
- Workers allow for laborious processing
- Specific use cases: Zoom uses WASM for data processing
- Demo: Ray tracing
- Support on mainstream browsers for a long time. One issue. Cookie cutter approach doesn't distinguish between workloads. Depends on the OS.
- By default, treat all workloads as optimized for performance. Some workloads need low latency (audio), some do not. E.g. Background tasks. Some may not need performance, but need to be scheduled regularly
- Platform specific: Windows has SetThreadPriority API - several levels of QoS. Sometimes decided by the OS scheduler. QoS associated with thread.
- MacOS also supports this. GCD takes everything into account. QoS API linked in slides
- Linux and ChromeOS. Energy-aware scheduler in the Linux kernel. Predicts impact of its decisions on energy use. Relies on the energy model of the CPU. [company] also working on a new upstream scheduler
- Simple API sample
-

-
- (These are hints; don't guarantee core assignment)
- Cannot override OS scheduler decisions
- Did some Power+Perf analysis, ex. Video conference with background blur
-
- 13ms deadline for a frame; found out in this case that FPS was not affected. Laptops can perform render tasks at 30 FPS. Can get 21% power savings
- Not all good:

-
- Squoosh app; With QoS, 10% power savings, but execution time was ~2.7x.
- Not for everything, depends on workload
- Sometimes execution time can be 2.7x and only save 10% power
- There is a patch for Chromium. Objective of the session is for people to try it out. Do your own test and see if this matters. Mostly looking for more use cases. Did some experiments~~experimtes~~ with Zoom + Meet, and got some feedback. Want this to expand past RPC. Also looking for what kind of workloads could be used here. Someone suggested ZKP; solve a puzzle with encryption. Want to hear more about such workloads
- Open questions:
  -

-
- Conclusion: Simple API addition, can put workloads within a power budget. Haven't added priory

Question queue

Add your name / question here

- MM: Design decisions around metrics. Multiple objectives (power, latency). Appreciate the simplicity of three levels, but maybe we should set a target. Eg.g. return within 10ms. Just say that, let the system decide. Stay abstract, let hardware decide. Or state "minimize power consumption". What's motivating web app developers to use the lower power mode? What should the default be, considering compatibility? Probably low? Or maybe no target?
    - Riju: Initial thinking was that the default was set by the platform
    - MM: Problem is that it is hard to set globally. Not all workers are low; maybe you have a set of overall preference to optimize for.
    - What about the issue of setting power targets? E.g. A game needs a framerate. Worker has a time budget. E.g. 5ms. Want to set that as a target that the worker runs to. Have a feedback loop that the system can be adjusting to hit that target.
    - Riju: RIght. THis also needs some kind of priority so that the task is not waiting. [?]
    - MM: What about hardware acceleration. Eg. for AI. Makes a difference for power/
    - Riju: Workers run on CPU.
    - System may have a way of computing a target; gives more options for optimizing
- Randall Jesup: In terms of not setting defaults, letting the system do it; Guarantees that every web app has to set a value. By not setting one, they will get an unpredictable result.
    - Riju: Opinion is that the default should be low
    - RJ: How to specify the task and scheduler requirement for real-time operations is a research area. Setting deadlines is potentially sensible for RT applications. In this scenario, you may want something more thanthen ~~then~~ High/Low/Default linear

progression. Separate class for RT vs non-RT response may be a better initial way to categorize. Within those, you can set power-sensitive/not. For RT, you can set "how much processing do I want to get done per frame". "Frame" is different from one app to another. Lots of prior art
    - Riju: For RT, everyone uses audio worklets. Why would they try this out? Everything works fine with the audio pipeline
    - Not the only application. Games, eg. Any time an application where you really want to get work done in a certain amount of time. Need a feedback mechanism.
    - Riju: If you're doing video processing, the CPU is not the optimal place. For RT game video processing, GPU is better suited. Yes, there are more use cases
    - Should try to specify the problem, not the solution. Eg. not H/L/M mapping to OS levels, provide info about what you're doing, UA can adaptively choose. UA may change if targets are being missed.
- Steven Rostedt: Linux Kernel developer, RT scheduler maintainer.
    - RT QoS APIs are very complex. Kernel doesn't know what the application wants. Libraries give applications more control. One way to get the feedback loop - if you have a start and end, with a timeframe, you have a way to tell the kernel how to schedule you. If you can meet your deadlines at one level, you can stay, or the kernel can adjust. Could have a start and framerate. Because of caches, page faults, etc. CPU is not deterministic. Could we have something like this?
    - SM: RT aspect on web platform is a discussion. Because of possibility of abuse. For audio we have a carve out.
    - SR: Not hard HT.
- Youenn: Two objectives; Power efficiency and latency. On iOS, trying to tell the OS this is very difficult. Typically something the OS will do for you. Not sure it's valid. OTOH, latency, "I have 5 workers, if you have to sacrifice one, take this one". Some are user initiated, e.g. networking thread. When doing communication, can sacrifice [?]. Useful if you can lower the priority of certain tasks. In terms of API, this API is exposed for workers and shared workers. Main use cases are workers~~is workers~~.
    - Riju: Explainer talks about shared workers
    - YF: Already using work queues in webkit. Setting on creation. Not changing QoS.
    - R: Are you using H/L/Default?
    - YF: Yes, internally, not exposed
    - R: Extending enum?
    - YF: Might be hard to get consistency between browsers, even on the same~~on same~~ device.
- Paul: Wanted to add comments - there is a certain class of application that needs to use worker threads in addition to worklets. Pro digital audio app needs both. On native, create threads, set to same priority as RT threads, join to them. This has been requested by industry. Lock-free queues to match native threads to RT threads. In addition, deadline/timeslice scheduling helps tremendously with audio applications. Everything is intrinsically timeslice-based in audio. Also for abuse prevention. If you miss deadline, system can demote thread. Avoids abuse of RT threads. Also audio usually has a different scheduling class, which doesn't fit this model.

    - Riju: Customers want to use worklets and workers together?
    - Yes. Professional applications need this. Shared memory buffer to pass data back and forth.
    - Hongchan: Not audio specific - high priority threads for video processing;
    - Paul: Is it possible to tag threads based on use case and group together? (Newer MacOS can do this) Cooperating at the scheduler level. Associate multiple threads to a single audio stream, and things work nicely. Same use case.
- Harald: Agree that we need to set default. Even if you set "default" you have no idea. $100 Chromebook vs $1000 iPhone have different results. Two examples; one performed within deadline regardless; the other missed on low priority. What's the interface? How can you figure out whether your application is working or not? Can flip switches, but if you can't tell if it's working.
- In order to make priority useful, we need to match with some measurement interface so we can tell whether we're performing.
    - Riju: Not aware of HTML API within workers
    - YF: You have many tasks; some of them are performing within deadline. Need to reduce number of tasks.
    - HA: Sometimes you need to reduce workload, In other configurations you adjust priority. How can you tell?
- Mike:
    - What we really want is a way of estimating how well a workload performs to hit a user experience. Depends on several factors. E.g. Compute Pressure. We don't know how long the worker takes to get to you. Playing around with using Math benchmarks. Vector of numbers, compare to a vector of numbers generated on a target machine. Helps with one-shots.
    - Needs richer information/metadata for both the target and workload.
    - Leads to fingerprinting problem if metadata is given to the app. Decisions should not be visible to the app.
    - Riju: Compute pressure has privacy concerns. Cannot expose too much. Agree there should be some kind of marking or workloads.
    - Privacy is a sliding scale. Risk tradeoff. If we can pick something that minimizes~~minimises~~ exposure risk
- SM: I think it should be possible to change the priority after creation. Very important. If you want to scale down the priority. Shouldn't have to tear down and recreate. Implementation challenges. But we should be serving a better API to users and developers regardless of difficulty.
    - Riju: Is it doing more harm than good?
    - SM: Plenty of use cases
    -
-
- Riju: Are there other workloads that~~taht~~ we should consider?
    - MM: Demo based on bullet[?] ammo[?] Using WASM. Compute demo, multiple workers, using game physics.

- Austin: IO-intensive workloads the OPFS and Wasm (e.g. bring-your-own-database)
-