rtoy 6:14 PM

Oh. Hmm. I thought I was running nightly, but maybe not. It's Firefox Developer Edition.

rtoy 8:53 PM

Firefox Nightly runs the test just fine. Duh!

sebpiq 2:39 PM

Heya! Haven't been here for ages ... I have a little question! I am trying to benchmark some code from within an AudioWorklet.
In order to time the execution of my function I would normally use `performance.now()` but it doesn't seem to be available there ...
Any thoughts on how to benchmark arbitrary code from within an AudioWorklet ? (edited)

it's a problem, I've been meaning to provide a solution

in the meantime I can tell you how to do it in Firefox, and I know chrome has something

padenot 2:53 PM

I'm pretty sure I can do something in js, but spectre gets in the way 😟

5 replies

Last reply today at 3:02 PMView thread

@padenot you tell me how to do it in ff and Chrome?

🙏

attila 2:56 PM

as a sidenote, how do you access `currentTime` inside a worklet processor?

padenot 2:56 PM

now it's possible to disable spectre mitigation, if it's for local benchmarking

`this` being the `AudioWorkletGlobalScope`

attila 2:58 PM

but when you have a webassembly bridge, i guess you have to copy it in or pass?

@sebpiq for Chrome, `chrome://tracing` has something

padenot 2:59 PM

for Firefox, run it with `MOZ_DISABLE_CONTENT_SANDBOX=1 MOZ_LOG=AudioCallbackTracing:5,raw,sync MOZ_LOG_FILE=tracing.log ./mach run`, and this will output tracing files you can load in `chrome://tracing` , or you can use https://padenot.github.io/msg-load-analyzer/ to have useful statistics (edited)

I'm going to move all this to our profiler soon, it will be nicer

@attila yes, for this you can do it yourself

sebpiq 3:01 PM

Hmm ok ... so I can't get the times with code right ? I have to open the profiler ?

it works with `SharedArrayBuffer` and a web worker

padenot 3:04 PM

yw, I'll try my idea hopefully soon but I have lots of things to do to finish audioworklet properly in Firefox

sebpiq 3:06 PM

Another question actually ... I am benchmarking DSP code for now from standard workers (and wanted to move the benchmark to audioworklets to test in real conditions).
Is there any reason to expect very different results between audioworklet and web worker ? (edited)

what impacts the performance ? (edited)

padenot 3:06 PM

regular workers are on regular thread, audioworklets are on real-time threads

regular worker can process a bunch of audio at once (even if you're doing it by 128-sample frames buffers, CPU caches are going to be hot unless you're trying to make things more real), audio worklets are only doing some numbers of 128-sample frame buffers, and then yielding, and then it's called again, but caches are cold

sebpiq 3:08 PM

so it should be more predictable in audioworklets, and potentially faster I guess ? But then in terms of comparison of the performance between different implementation of the same DSP process it should be the same ratios no ?

sebpiq 3:09 PM

So I mean that it doesn't invalidate completely

benchmarking done in normal workers does it ?

padenot 3:09 PM

it depends on how you measure, if you know your web worker thread isn't going to yield, it's going to be more or less accurate, maybe the mean will be similar

padenot 3:10 PM

I think that comparing two versions of the same program in the same environment, you can make conclusion, and I think those conclusions will translate to the other environment well

padenot 3:10 PM

yield as in the thread will yield to the scheduler, because web workers aren't real time, so their execution can be stopped

and because it yields in between your two `performance.now` call, then the measurement will be off (edited)

sebpiq 3:11 PM

yeah right ... but there's not much I can do prevent that I guess, apart from starting the browser with special priorities? (edited)

but if you're doing enough iterations, the measurement will converge

like maybe 10 or 20s, depending on if you code JITs well, etc.

sebpiq 3:13 PM

Yeah I'm doing lots of them. I have pretty flat curves if I do enough

padenot 3:14 PM

also regular native profiling advices apply: try to not change the load of the machine (high and low load matter, but changing in the middle makes measuring harder), try to disable frequency stepping of the CPU

and try to only change a single thing at once

(never trust a web browser developer when they say something will be available soon)