Public document

CPU Reporting

Author: Utkarsh Goel <ugoel@akamai.com>
Contributors: Folks from Wikimedia, Google, MS, Shopify, Akamai, etc.
June 2019

At the recent WebPerf WG meeting, many folks expressed interest in browsers reporting the device's CPU information in HTTP requests. The motivation here is to estimate how slow or fast the device is and accordingly tune the website content to maintain fast page loads. The purpose of this document is to collect customer/developer use-cases and security/privacy concerns that may arise from browsers exposing such information. Some analysis of the exposed bits is available here. Additional comments and ideas can also be documented here.

Use-cases

The use-cases listed below are collected from interactions with developers. Please feel free to add your use-case below, if it's not listed already:

1. Block 3rd party scripts for pages loading on slow devices.
2. Use RUM to associate various web performance metrics with device CPU information, instead of or in addition to device identifiers present in the UA string.
    a. Estimate the fraction of requests that come from slow devices, which can help understand whether or not a developer needs to worry about slow devices accessing their websites.
    b. Understand whether Web performance improvements come from users upgrading to faster devices or from developers adopting new and faster Web technologies.
3. Avoid computationally heavy operations on slow devices
    a. Run JS tasks only when the CPU is not loaded
        i. Need a browser API to learn the current CPU load
    b. Remove animations from the page
    c. Replace image carrousels with static images
    d. Aggressive lazy loading of resources
    e. Run JS in the cloud, similarly to the Puffin browser
    f. Deliver lo-fi version of the page that does not require new JS features that need to be polyfilled
    g. Disable JS features entirely
    h. With respect to upload compression, use lower compression levels to utilize fewer CPU cycles
    i. Disable prefetching or prerendering of resources
4. Conditionally load CPU-intensive JS tasks only on high-end devices
5. Serve low-quality images and videos

    a. Since devices with slow CPU processors will likely also have other low-end hardware specifications, a developer may want to not send a high-quality media on phones.
    b. Some formats like WebP have a significantly higher decoding cost and would be best avoided on a device with limited processing power. This can allow better tradeoffs for these formats, which are useful in low bandwidth + sufficient spare processing power scenarios. Similarly, this proposal can additionally help identify other content types that are more costly than others on slow mobile phones.
    c. Decide whether to use AV1 (either natively, or polyfilled), which saves bandwidth, but costs more in CPU time. To support CDNs, this should be possible to detect server-side.
6. Select ads better suited for slower devices
    a. It may not be necessary to know upfront whether an ad consumes significant resources, such that it wouldn't perform well on slower devices. If CPU information could be used as a feature in machine learning models used to select ads.
    b. See this slide for similar usage of other WebPerf-related information to select ads.

What Info Could Browsers Report?

Device identifiers, such as the model name or number, in the UA strings, are currently used to approximate whether a requesting device is new (and presumably fast) or an old (presumably slow). However, device model numbers in the UA strings do not directly indicate how fast computationally-heavy operations will run on the device, which the developers need to detect when to tune webpage content. Moreover, UA strings reported by iPhones and iPads do not even include the device model names or numbers. As such, additional information, specifically about the device CPU, was discussed at the meeting as something that browsers could report in HTTP requests. Various options for what a browser may want to expose are listed below.

1. Available CPU processors on the device, including the number of cores and clock speed
    a. Not sure if the number of cores is useful to report, as JS operations are single-threaded. No matter how many cores there are on a given processor, webpages will only make use of one core.
    b. A device may have several CPU processors available and the battery-saving or power-saving modes on the device could switch between these processors. Therefore, the processor that is activated at the time of loading the page should be the one that is reported by browsers.
2. The number of CPU instructions performed by the device in the last few seconds.
    a. If this is measured actively, the downside would be that it would consume device battery and CPU resources that could have been made available to the browser for webpage loading.

    b. Alternatively, browsers may be able to estimate it passively as the pages load.
3. A browser could expose the average (or median) CPU utilization in the past 30 or so seconds (or a user-provided parameter) as an indication of whether or not the device is currently under heavy load. This information will also help prevent the general classification of devices as slow or fast and will instead rely on the device's current ability to run computationally heavy operations.

How to identify a slow or a fast phone from CPU Reporting?

A fast phone could become slow under certain conditions. For example, heated devices could throttle normal CPU processor speeds. The battery-saver or power-saving modes on devices can switch between faster and slower CPU processors. As such, it is important to realize that a fast phone with an active battery-saver mode can perform as slowly as a slow phone. A phone whose CPU clock speeds are lower than a certain threshold can be classified as "slow". Similarly, devices with lower Geekbench scores can also be classified as slow. Depending upon how a developer may want to tune website content, it may be up to the developer to decide how to classify devices as slow or fast based on their website.

Concerns

User Device Abuse

While CPU reporting could enable developers to tune webpage content and potentially speed websites on slow devices, it is important to protect a user's device resources. For example, a malicious website may learn that a user's device has a fast processor and may decide to run computationally heavy operations (for example, a crypto mining script), thus exhausting the CPU and draining the battery on the device. However, it is already possible for a malicious website to obtain CPU processor information from the device identifier in the UA string by using, for example, www.gsmarena.com and similar websites and perform the attack. While explicitly exposing CPU information in HTTP requests makes the job of a malicious website easier, it does not necessarily introduce/trigger new techniques to abuse the CPU resources on the device.

Fingerprinting

The data revealed needs to be coarse enough that a user can't be tracked based on this value. If the range of possible values is large, it needs to be fuzzed from one run to the next, and between different origins, if requested at the same time (to avoid cross-origin tracking).

Evolution

The scoring scheme needs to be able to keep up with technological improvements. I.e. you don't want a score that gets closer to "0" as devices get more powerful, as we might end up there sooner than we think, with all future devices beyond a certain processing power reporting the same score. Similarly, if you define buckets, they need to be able to evolve over time. The simplest solution is probably to use a scheme where the higher the score is, the more powerful the device. But then this has to be balanced with fingerprinting requirements. You don't want the fuzzing/noise to become ineffective as the gap between devices widens with technological improvements.

Related Research

Recent work has experimentally explored an approach that relies on information that browsers already expose in UA strings. The technique associates the mobile device model numbers obtained in UA strings with the model numbers (and the device CPU processor information) obtained from www.gsmarena.com. This approach helps build a mapping between device identifiers in the UA string and CPU processors on the device, which could then be used to perform webpage tuning when the request arrives on a web server.

A few downsides of this approach are: 1) it relies on the data collected from gsmarena.com, which may not have CPU processor information for all mobile devices connected to the Internet, and 2) for devices with multiple processors, there is still no way to know which processor was used on the device to make the HTTP request.