VROOM: Accelerating the Mobile Web with Server-Aided Dependency Resolution. **A review.**

Kevin Dackow

June 2018

1 Summary

VROOM attempts to address the problem of complex web dependencies slowing page load time (PLT), especially on mobile devices.[1] The authors begin by identifying two previously-implemented strategies for reducing PLT: 1) utilizing proxy servers to reduce client-side CPU usage and 2) client-side reprioritization of requests to reduce the time required to process more important site data. For both of these solutions the paper identifies problems: 1) the loss of security, especially regarding privacy for user-specific data (which would be transferred through a proxy rather than directly to the target server) and 2) the increased load on the client's CPU inherent with client-side scheduling. With these solutions identified as suboptimal, the authors present VROOM, which they claim will reduce PLT without compromising security or increasing the burden on the client's CPU.

VROOM achieves this by combining both server-side and client-side enhancements. In summary, VROOM calculates an efficient page load scheme on the server side and then uses simple scheduling logic on the client side to more efficiently load the page.

On the server side, VROOM utilizes HTTP/2 PUSH in conjuction with *dependency hints* to reduce PLT. HTTP/2 PUSH preemptively sends data to the client that is most likely going to be necessary for page load. However, the server push cannot send all required data to the client because of several reasons: 1) the data may be spread across many domains (e.g. JS libraries), 2) site data is often personalized (e.g. advertising), and 3) HTTPS would lose its end-to-end nature if the host server transferred 3rd party data to the client. With these flaws in mind, VROOM improves upon HTTP/2 PUSH through the use of dependency hints.

Sent as an additional HTTP header, dependency hints alert the client of dependencies early in the page load process in order to parallelize requests to web servers with the processing of the rest of the page. Internally, dependency hints are URLs that the client must request data from for page load to complete. By prioritizing the processing of dependency hints, the client is able to request all needed data before processing the page content. In doing so, as the client processes the HTML, CSS, and JS, the dependencies are downloading concurrently. In contrast to traditional page load, in which a client only downloads dependencies as they are first discovered in the page load process, dependency hints reduces the likelihood of a network bottleneck slowing down page load by parallelizing page processing with 3rd party downloads. In brief, VROOM uses dependency hints and HTTP/2 PUSH to reschedule client-side processing for increased efficiency.

In addition to parallelizing processing with downloading data, it is important to note that

[1] Note: PLT is not the only measure that VROOM attempts to reduce. Other measures such as Above-the-Fold-Time and Speed Index are also evaluated in the paper.

VROOM stipulates that the server calculates dependencies itself, thus removing the need for a proxy server to calculate dependencies. In doing so, and in sending dependency hints as URLs, the client is able to request information exclusively from the original host, thus preserving the end-to-end nature of HTTPS.

The creation of dependency hints relies on a server-side implementation of dependency resolution. For this, VROOM uses a combination of offline and online resource discovery. Through offline discovery, i.e. longer term analysis of which URLs remain constant in page load, VROOM is able to consistently send data that is known to be needed. However, this approach risks missing URLs, as dynamic content is increasingly frequent across sites. To combat this, online resource discovery loads the page server-side, then sends those dependencies to the client. The flaw with this approach is that it can lead to URLs unecessarily being requested by the client, when they are unneeded (e.g. an ad may be different per page load, so sending an unused ad would waste client resources). VROOM combine these two approaches in order to mitigate both of their isks. By performing both online and offline discovery, VROOM reduces the risk of sending a site to the client that is unecessary through the conservative offline discovery, while still increasing the number of correct URLs through online discovery.

After sending these dependency hints to the client, the actual processing must now be done. Through the use of a client-side scheduler, the client can use the dependency hints to dramatically decrease PLT. Specifically, the client gets all dependencies at the onset of site processing. It then begins downloading data, with priority given to data that needs processing, i.e. HTML, CSS, and JS files, while processing the other site data. In summary, VROOM decreases PLT by enabling the client to download dependencies while concurrently processing other site data. This theoretically maximizes the utilization of both bandwidth and CPU power on the client side.

Following the technical description of the projet, the paper continues to describe specific metrics and testing to demonstrate VROOM's reduction in PLT and other metrics, noting that median PLT was reduced by approxiamtely 2 seconds when uising VROOM.

2 Analysis

Though VROOM presents a uniquely powerful solution to slow PLTs, it relies on a few fundemental assumptions. The first and largest assumption is that the client possesses a large amount of processing power. Specifically, VROOM requires the client to process all site data (as usual) while concurrently downloading dependencies (rather than downloading as they are discovered). This is generally fine for high-end devices (e.g. Macbook Pro, Surface Pro, iPhone X), however, given the increase in browsing on mobile devices [1] (which tend to have lower computational power than computers), VROOM's assumption of high client compute power significantly reduces its deployability to the global market, especially in areas with lower-end mobile devices as the primary means of internet access. Though the paper acknowledges burden on the client as an issue, it only measures its impact as a decrease on PLT. That is, the paper does not factor in an increase in CPU usage in its metrics. Additionally, most of the testing for VROOM was done on a Nexus 6 smart phone, which has by default a 2.7GHz quad-core processor with 3GB of RAM, which is likely high-end enough to avoid these concurrency problems. This testing and design oversight would likely produce a CPU bottleneck on lower end devices. In future implementations of VROOM, it could be beneficial for proxy servers to be used to reduce the load on the client, despite potential data security vulnerabilities.

Another assumption made in the paper is that data that requires processing should be fetched first. Specifically in VROOM's calculation of dependency hints, it prioritizes data that needs processing on the client side, e.g. HTML, CSS, and JS. This assumption is valid for increasing PLT as a whole, however it does not prioritize user experience. Though PLT and AFT are useful measurements that VROOM does improve, VESPER [2] details how these measures are ineffective for measuring User Experience. Because VROOM prioritizes fetching resources that need

processing, things that users value, e.g. images and videos, get fetched later, which can reduce user experience. In future work, it would be beneficial for VROOM to reconsider its dependency hint weighting so that data that the user values is fetched first.

Despite these issues, for dominant global markets where high compute-power devices are common, VROOM is a powerful, effective, and secure platform that would significantly benefit user experience through the reduction of PLT and AFT.

References

[1] Gibbs, S., *Mobile Web Browsing Overtakes Desktop for the First Time*, in The Guardian, 2016.

[2] Netravali, R., Nathan, V., Mickens, J, and Balakrishnan, H, *Vesper: Measuring Time-to-Interactivity for Web Pages*, in NSDI '18.