Demystifying Page Load Performance with WProf

Xiao Sophia Wang, Aruna Balasubramanian, Arvind Krishnamurthy, and David Wetherall
University of Washington
{*wangxiao, arunab, arvind, djw*}*@cs.washington.edu*

Abstract

Web page load time (PLT) is a key performance metric that many techniques aim to reduce. Unfortunately, the bottlenecks of PLT are difficult to identify due to the complexity of the page load process. We abstract a dependency graph of the activities that make up a page load and develop a lightweight in-browser profiler, WProf, to produce this graph. Combined with critical path analysis, WProf reports suggest that computation is a significant factor that makes up as much as 35% of the critical path and that synchronous JavaScript plays a significant role by blocking HTML parsing. We plan to demonstrate WProf that produces and visualizes the dependency graph of any given Web page. [1]

Problem: Numerous techniques have been developed to reduce PLT. They range from caching and CDNs, to more recent innovations such as the SPDY protocol that replaces HTTP, and the mod_pagespeed server extension that enforces Web page best practices. Thus it is surprising to realize that the bottlenecks that limit PLT are still not well understood. Part of the culprit is the complexity of the page load process, which is further complicated by browser implementation strategies. The result is that, for example, we are unable to explain why a change in the way a page is written will help or harm PLT.

Previous measurement studies have measured Web performance in different settings, *e.g.*, cellular versus wired, and correlated PLT wth variables such as the number of resources and domains. However, these factors are only coarse indicators of performance and lack the power to explain why a page load proceeds as is.

Approach: Our position is that demystifying the composition of PLT is crucial to advance research in this area. The challenge here is that the activities that compose PLT are likely happen in parallel. This includes concurrently loading Web objects which can be further parallelized by evaluating JavaScript. Thus, extracting the dependencies imposed by a Web page is the key.

We extract four categories of dependencies that are caused by (i) the natural order that activities occur, (ii) the correctness of execution when multiple processes modify a shared resource, (iii) tradeoffs between data downloads and page load latencies, and (iv) limited computation power and network resources. Figure 1 shows an example of the dependency graph of a simple Web page below which, however, exhibits complex dependencies.

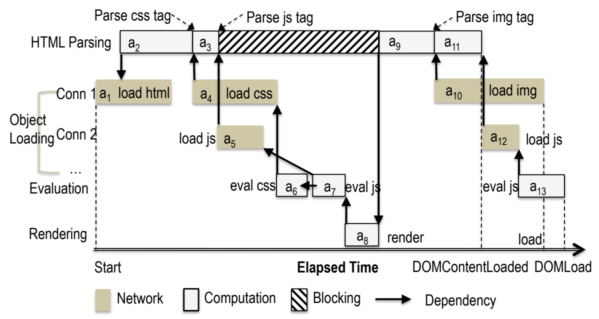

Figure 1: An example of the dependency graph.

```
<html>
  <head>
    <link rel="stylesheet" src="a.css">
    <script src="b.js" />
  </head>
  <body onload="..."><!--request a JS-->
    <img src="c.png" />
  </body>
</html>
```

Tool and demo: We have developed WProf, a lightweight in-browser profiler that automatically captures the dependency graph for a given Web page. By performing critical path analysis on the dependency graph, we find that computation is a significant factor the makes up as much as 35% of the critical path and that synchronous JavaScript plays a significant role by blocking HTML parsing. More results are included in our technical paper.

Our demo will include a WProf-instrumented Chrome browser and a visualization of dependency graphs similar to Figure 1. A typical procedure is below. First, people visit Web pages of their own choices using WProf-instrumented Chrome. Second, we upload WProf logs to our Web server to visualize the dependency graphs. A dependency graph includes activities (in blocks) and dependencies (in arrows). When a block is clicked, the visualization tool will explain what its corresponding activity means (*e.g.*, evaluating a JavaScript) and what its corresponding dependencies mean (*e.g.*, evaluating a JavaScript depends on evaluating a previously appeared CSS). The visualization tool will also highlight the critical path. Up-to-date tool and visualization of WProf are at `http://wprof.cs.washington.edu/`.

[1]Xiao Sophia Wang is a student. We plan to set up a demo. This work is accepted to NSDI'13 technical program.