Speeding up Web Page Loads with Shandian

Sophia Wang
University of Washington
Why is page load time (PLT) slow?
html

Elapsed Time

Network  Computation
Components that access to the same resource can’t execute at the same time.
html

css

js

js

Elapsed Time

Network

Computation → Dependency
A simple page incurs complex load process, mainly due to interactions between HTML/JS/CSS.
How much can SPDY help PLT?
3/27/16

html

css

js

js

Elapsed Time

Page load

3/27/16
A technique that helps one factor of PLT is hard to help the overall PLT.
What does the simplest dependency graph look like?
Can we make every Web page look like this?
Yes, we want to make every page like this, automatically.
Approach: Split Browser

• Preprocess Web pages on a proxy server to simplify the client-side page load process
Approach: Split Browser

• Preprocess Web pages on a proxy server according to whether they are used initially

<table>
<thead>
<tr>
<th>Load-time state</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Used in an initial page load</td>
</tr>
<tr>
<td>• Display as <strong>fast</strong> as possible</td>
</tr>
</tbody>
</table>

**Chart:**

- ???
- css
- js
- js

**Axes:**

- Elapsed time
- Page load
- Time to interact

3/27/16
Approach: Split Browser

• Preprocess Web pages on a proxy server according to whether they are used initially

**Post-load state**

• Not used in an initial page load
• Ensure **correctness** of future interactions, and **compatibility** with existing technologies
Outline

• Load-time state
• Post-load state
• Deployment and implementation
• Evaluation
Outline

• Load-time state
• Post-load state
• Deployment and implementation
• Evaluation
Load-time State

• Goal
  – Display as fast as possible

• Approach
  – Eliminate both contents and computation of JS and CSS on the client as many as possible
Loading load-time state

```
{“loadTimeState”:{
 “css”:[“#main{font-size:12px;}"],
 “html”:{“children”: [{
 “tagName”:“body”, ...
 “children”: [...., {
 “tagName”:“div”,“id”:”main”,
 “css”:[0]
 }]}]}}
```
Loading load-time state

```json
{
"loadTimeState": {
  "css": ["#main{font-size:12px;}"],
  "html": {
    "children": [
      {
        "tagName": "body",
        "children": [
          {
            "tagName": "div",
            "id": "main",
            "css": [0]
          }
        ]
      }
    ]
  }
}

A list of matched CSS rules
```
Loading load-time state

```
{"loadTimeState":{
  "css":["#main{font-size:12px;}"]},
  "html":{"children": [{
  "tagName":"body", ...
  "children": [..., {
  "tagName":"div","id":"main",
  "css":[0]
  }]]}
}}
```

Visible HTML elements
Loading load-time state

{“loadTimeState”:{
   “css”:[“#main{font-size:12px;}”],
   “html”:{“children”: [{
   “tagName”:”body”, ...
   “children”: [..., {
   “tagName”:”div”,“id”:”main”,
   “css”:[0]
   }}]}}}

Which HTML element matches which CSS rules
Loading load-time state

Elapsed Time

Network  Computation  Dependency

Page load

3/27/16
Outline

• Load-time state
• Post-load state
• Deployment and implementation
• Evaluation
Post-load state

• Goals
  – Correctness of future interactions
    • Requirement: Post-load and load-time state contain full state of a Web page
  – Compatibility
    • Requirement: Post-load state contains unmodified JS/CSS snippets
Vanilla post-load state

• The entire Web page itself

• Pros
  – Easy to ensure correctness of interactions and compatibility with caching/CDN

• Cons
  – Redundant contents and computation from load-time state

From here, how much can we improve?
What’s equivalent to eval’ing this CSS?

```css
#main {
  font-size:12px;
}
#main {
  font-size:12px;
}
#main {
  font-size:12px;
}
#main {
  font-size:12px;
}
```
What’s equivalent to eval’ing this CSS?

```css
#main {
  font-size: 12px;
}
#main {
  font-size: 12px;
}
#main {
  font-size: 12px;
}
```
What’s equivalent to eval’ing this JS?

a += "hello world!\n"
a += "hello world!\n"
a += "hello world!\n"
What’s equivalent to eval’ing this JS?

```javascript
a += "hello world!\n"

a += "hello world!\n"

a += "hello world!\n"
```

```javascript
a += "hello world!\n" + "hello world!\n" + "hello world!\n"
```
What’s equivalent to eval’ing this JS?

```javascript
function add(a, b) {
    return a + b;
}
function add(a, b) {
    return a + b;
}
function add(a, b) {
    return a + b;
}
```
What’s equivalent to eval’ing this JS?

```javascript
function add(a, b) {
    return a + b;
}
```

```javascript
function add(a, b) {
    return a + b;
}
```

```javascript
function add(a, b) {
    return a + b;
}
```

```javascript
function add(a, b) {
    return a + b;
}
```
Post-load state

• Exploit the **idempotency** of evaluating CSS rules and JavaScript functions/statements
  – Eliminate redundant content that appeared in load-time state
  – Capture results of non-idempotent JS statements
Outline

• Load-time state
• Post-load state
• Deployment and implementation
• Evaluation
Deployment

• How to fast load on the proxy server?
  – Use a beefy server
  – Co-locate with Web front ends
    • As part of the website: reverse proxy
    • As a 3\textsuperscript{rd}-party service: cloud servers
Implementation

• Server extension
  – Chrome’s content_shell
  – Only handle HTML/JS/CSS

• Client browser
  – Chrome
  – JSON lexer, Blink, V8
Outline

• Load-time state
• Post-load state
• Deployment and implementation
• Evaluation
Experimental setup

• Server: 2.4GHz 16 core CPU, 16GB memory

• Clients
  - Mobile: Nexus S, 1GHz Cortex-A8 CPU, 512MB RAM
  - Desktop: Linux VM, 2GHz CPU, 1GB memory

• Top 100 Web pages
PLT on mobile

Shandian helps 60% in the median case
Unlike Chrome, Shandian is not sensitive to RTT, due to simplified page load process.
CPU has the same amount of impact for both Chrome and Shandian
More results

• PLT breakdowns
  – Time spent on proxy server is negligible
  – Most time is spent on client

• Page size
  – Shandian increases page size by 1% after applying gzip compression
Difference from related work

• Amazon Silk, Opera mini
  – Our client can run JavaScript
  – We place proxy servers near Web servers

• Prioritizing resources (server push, Klotski)
  – We remove page load dependencies on the client
Summary

• Split the page state according to whether they are used for an initial page load
• The dependency graph until the page is loaded is fairly simple
• Improve PLT by more than half consistently for various settings
• Is compatible with caching/CDNs