Speeding up Web Page Loads with **Shandian**

Sophia Wang

University of Washington

10th USENIX Symposium on Networked Systems Design and Implementation

nsdi '13

APRIL 2–5, 2013
LOMBARD, IL
(NEAR CHICAGO)
Sponsored by USENIX in cooperation
with ACM SIGCOMM and ACM SIGOPS

usenix
THE ADVANCED
COMPUTING SYSTEMS
ASSOCIATION

Why is page load time (PLT) slow?


```
<html>
 <body onload="done();">
  <link src='1.css'>
  <script src='d3.js'></script>
  <script src='2.js'></script>
  <div id="content"></div>
 </body>
</html>
```


10th USENIX Symposium on Networked Systems Design and Implementation
nsdi '13
APRIL 2-5, 2013
LOMBARD, IL
(NEAR CHICAGO)
Sponsored by USENIX in cooperation
with ACM SIGCOMM and ACM SIGOPS
usenix
THE ADVANCED
COMPUTING SYSTEMS
ASSOCIATION

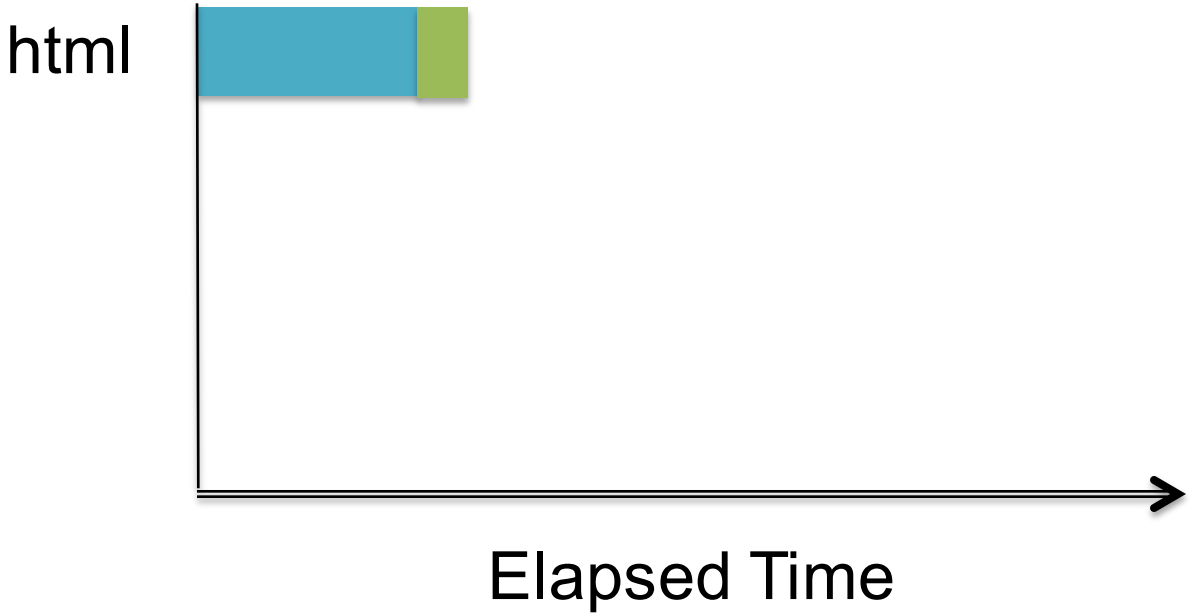

Elapsed Time

10th USENIX Symposium on Networked Systems Design and Implementation

nsdi '13

APRIL 2–5, 2013
LOMBARD, IL
(NEAR CHICAGO)
Sponsored by USENIX in cooperation with ACM SIGCOMM and ACM SIGOPS

usenix
THE ADVANCED COMPUTING SYSTEMS ASSOCIATION



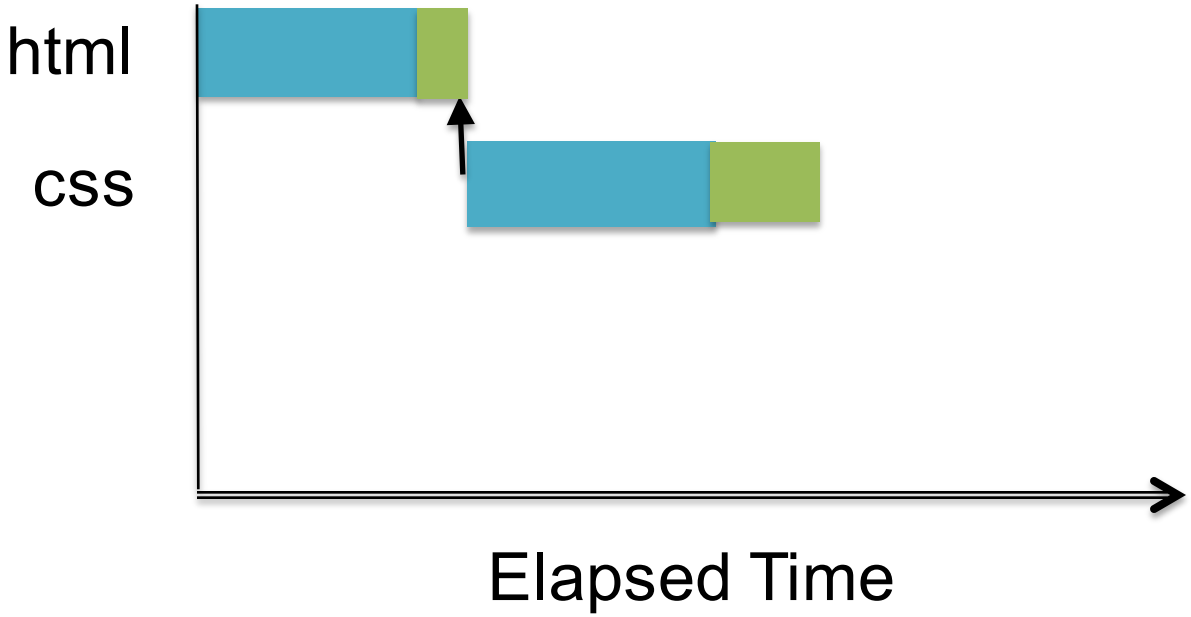
Network Computation

10th USENIX Symposium on Networked Systems Design and Implementation

nsdi '13

APRIL 2–5, 2013
LOMBARD, IL
(NEAR CHICAGO)
Sponsored by USENIX in cooperation with ACM SIGCOMM and ACM SIGOPS

usenix
THE ADVANCED COMPUTING SYSTEMS ASSOCIATION



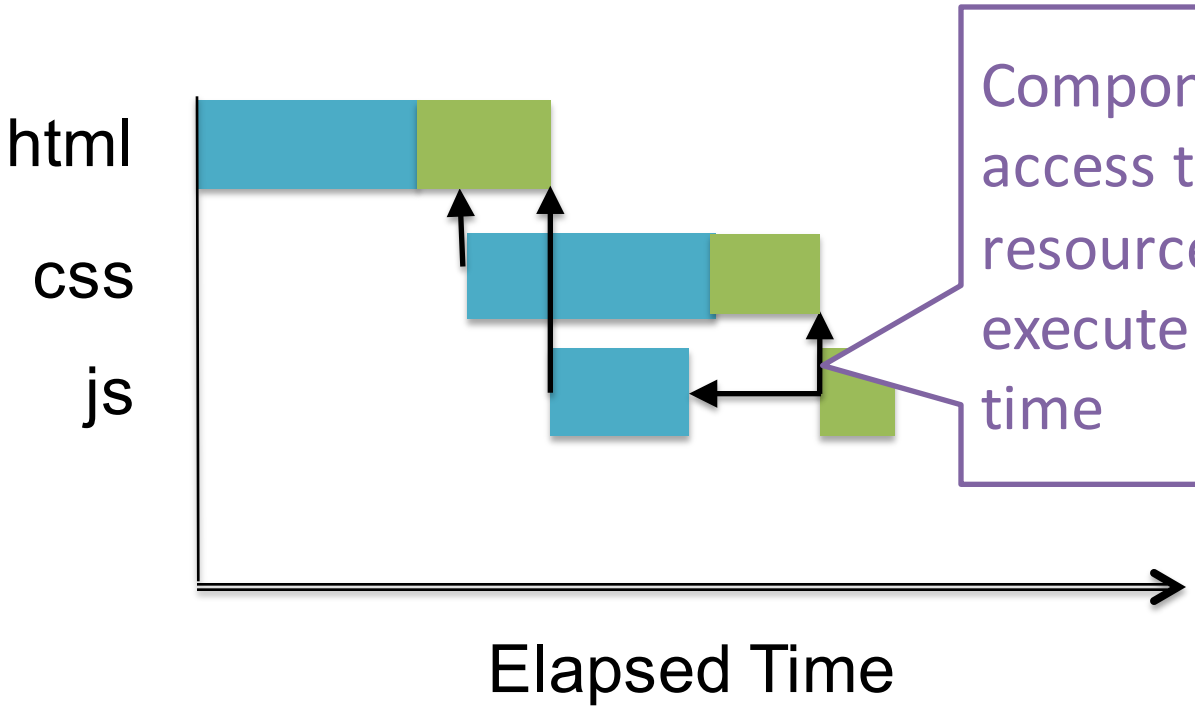
Network Computation → Dependency

10th USENIX Symposium on Networked Systems Design and Implementation

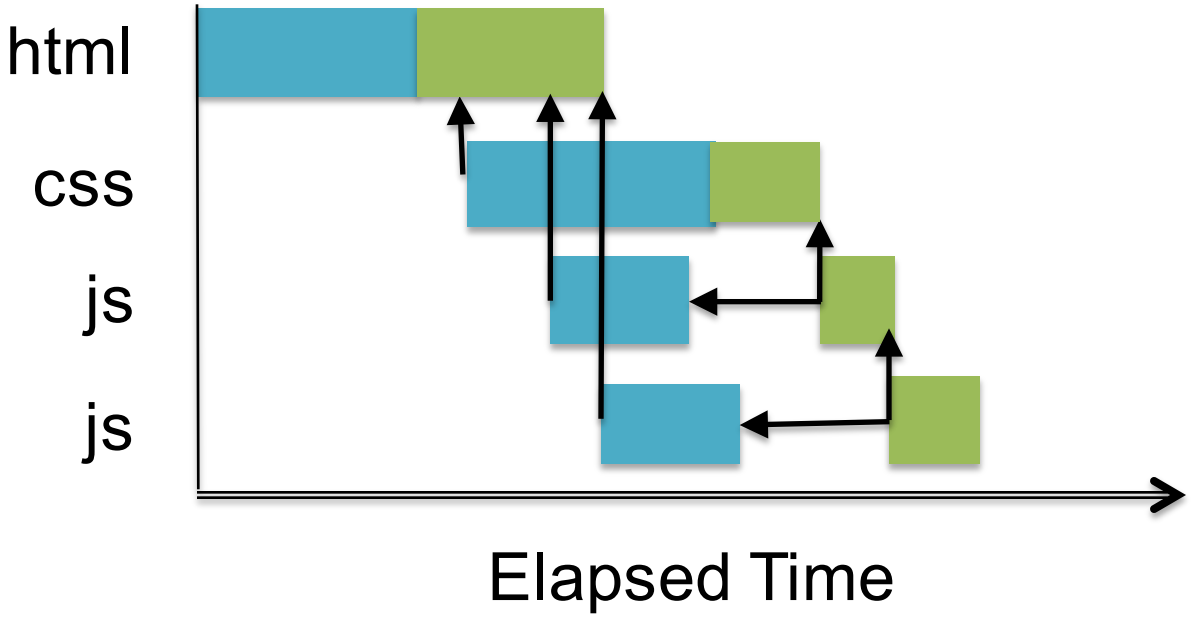
nsdi '13

APRIL 2–5, 2013
LOMBARD, IL
(NEAR CHICAGO)
Sponsored by USENIX in cooperation with ACM SIGCOMM and ACM SIGOPS

usenix
THE ADVANCED COMPUTING SYSTEMS ASSOCIATION



Network Computation → Dependency



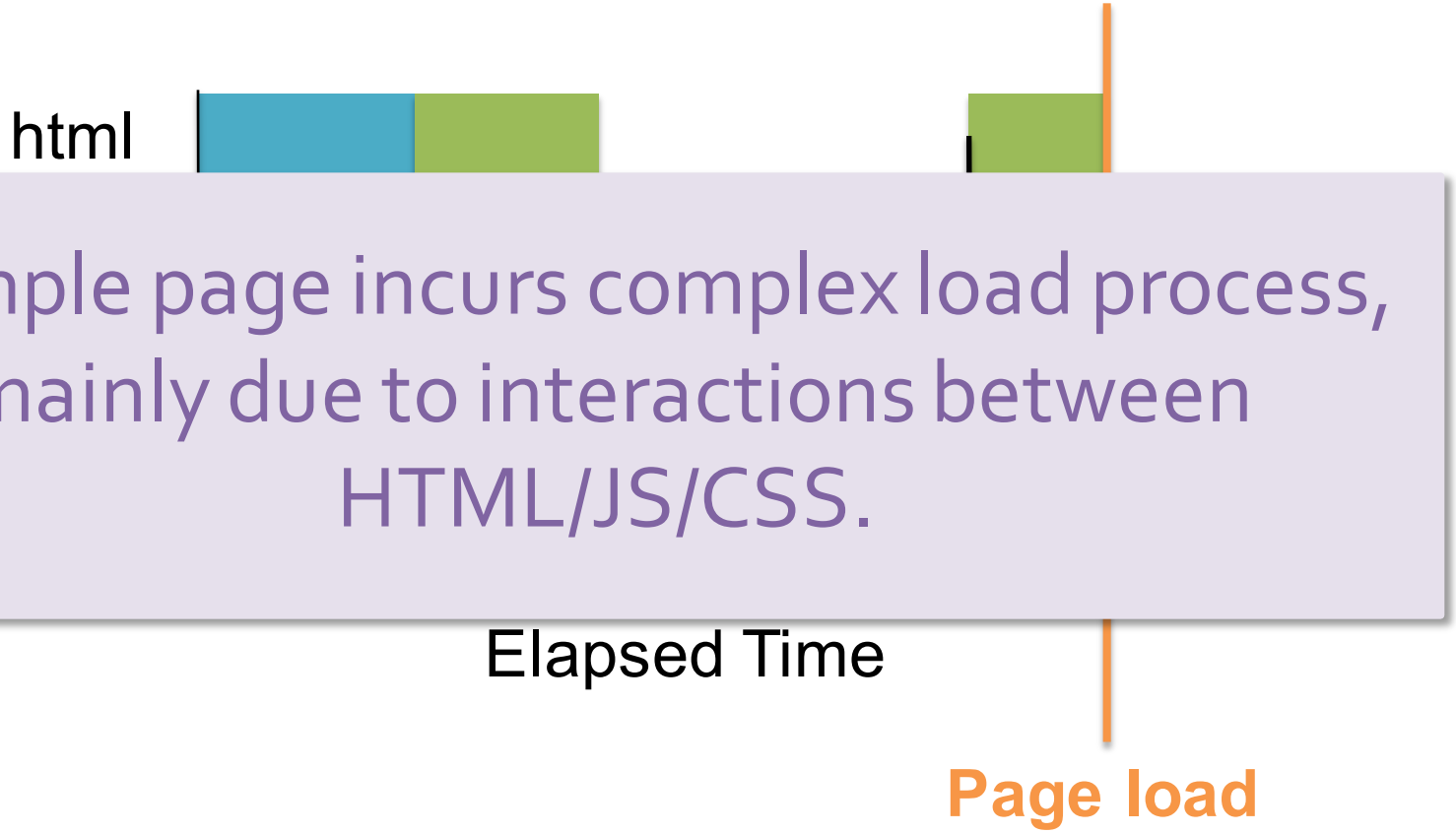
Network Computation → Dependency

10th USENIX Symposium on Networked Systems Design and Implementation

nsdi '13

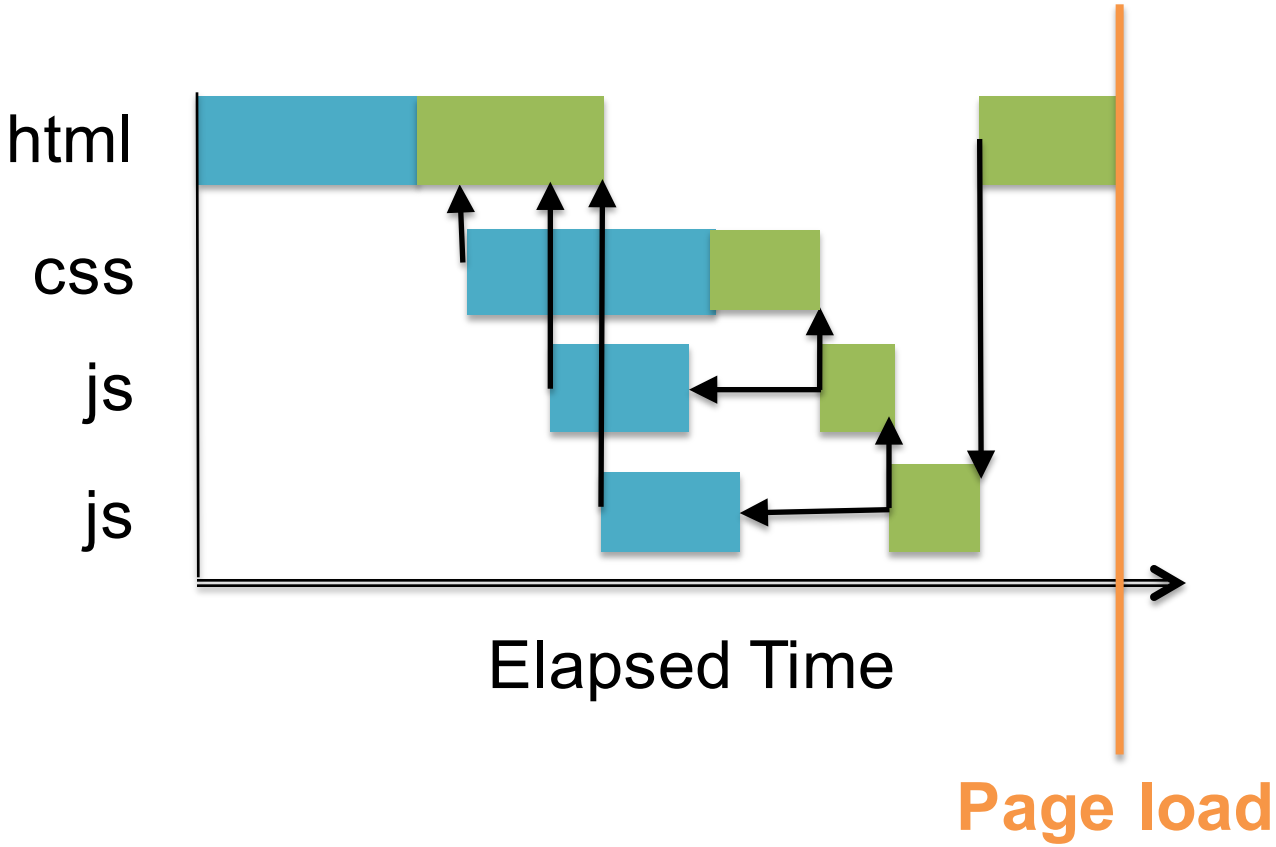
APRIL 2-5, 2013
LOMBARD, IL
(NEAR CHICAGO)
Sponsored by USENIX in cooperation with ACM SIGCOMM and ACM SIGOPS

usenix
THE ADVANCED COMPUTING SYSTEMS ASSOCIATION

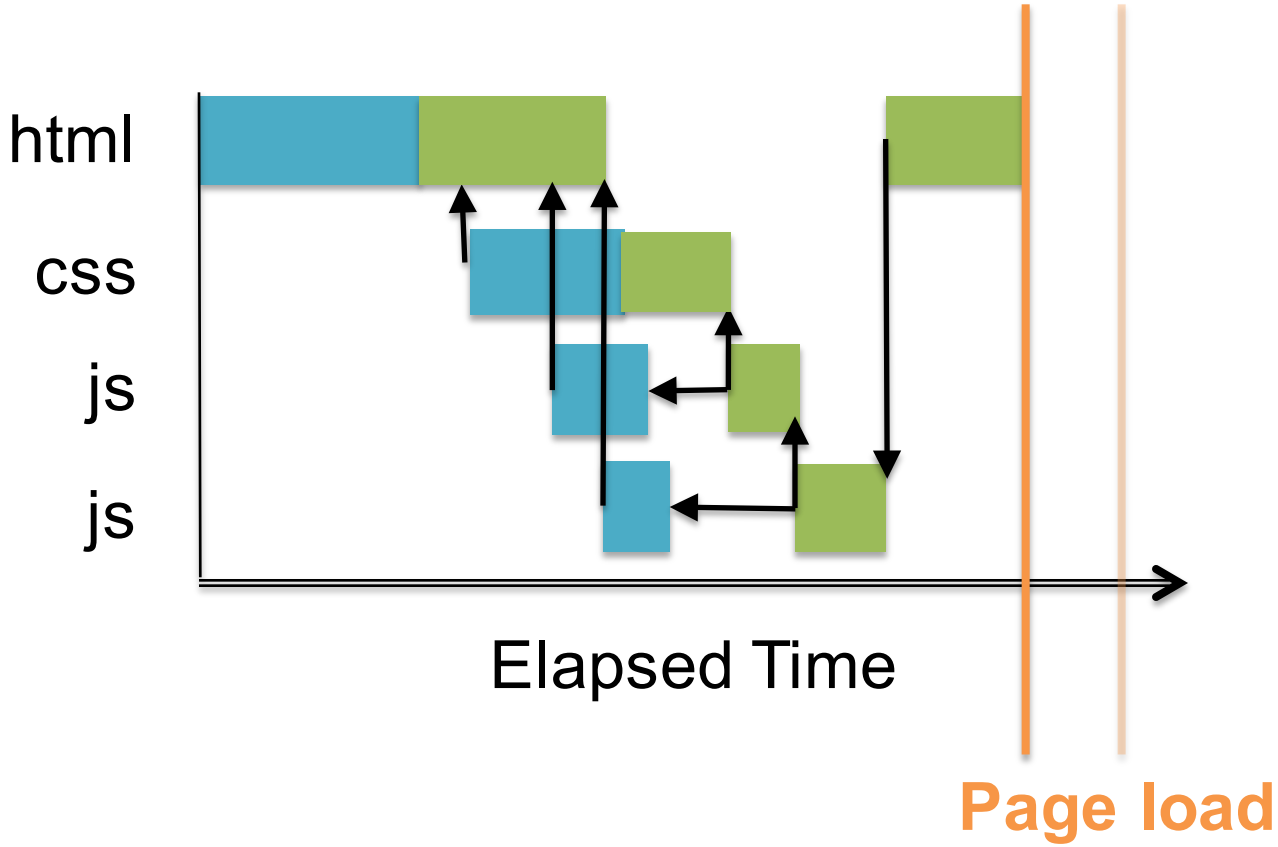


How much can SPDY help PLT?

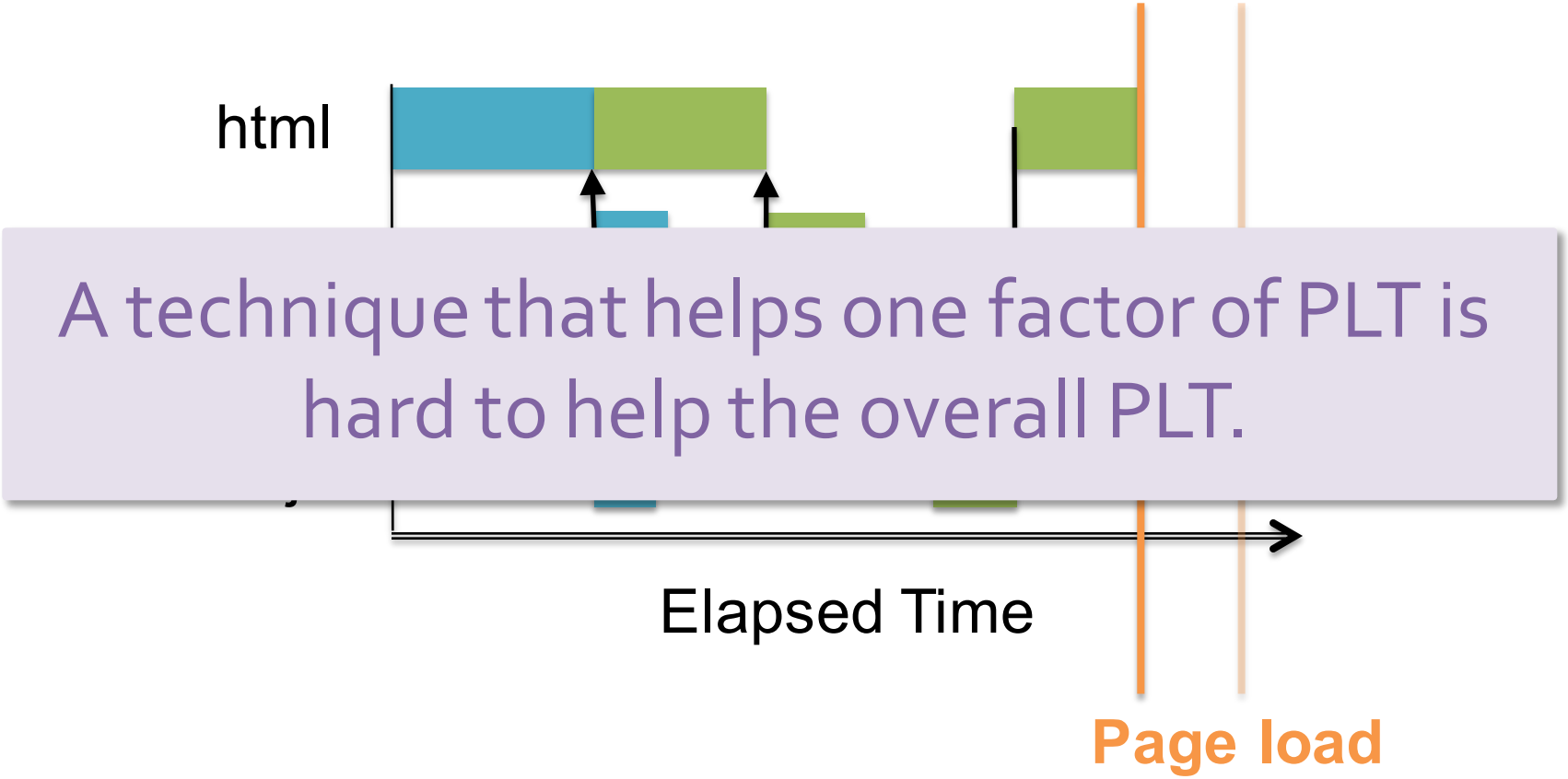
11th USENIX Symposium on Networked Systems Design and Implementation
nsdi '14
APRIL 2-4, 2014
SEATTLE, WA
Sponsored by USENIX in cooperation with ACM SIGCOMM and ACM SIGOPS
usenix THE ADVANCED COMPUTING SYSTEMS ASSOCIATION



11th USENIX Symposium on Networked Systems Design and Implementation
nsdi '14
APRIL 2-4, 2014
SEATTLE, WA
Sponsored by USENIX in cooperation with ACM SIGCOMM and ACM SIGOPS
usenix THE ADVANCED COMPUTING SYSTEMS ASSOCIATION

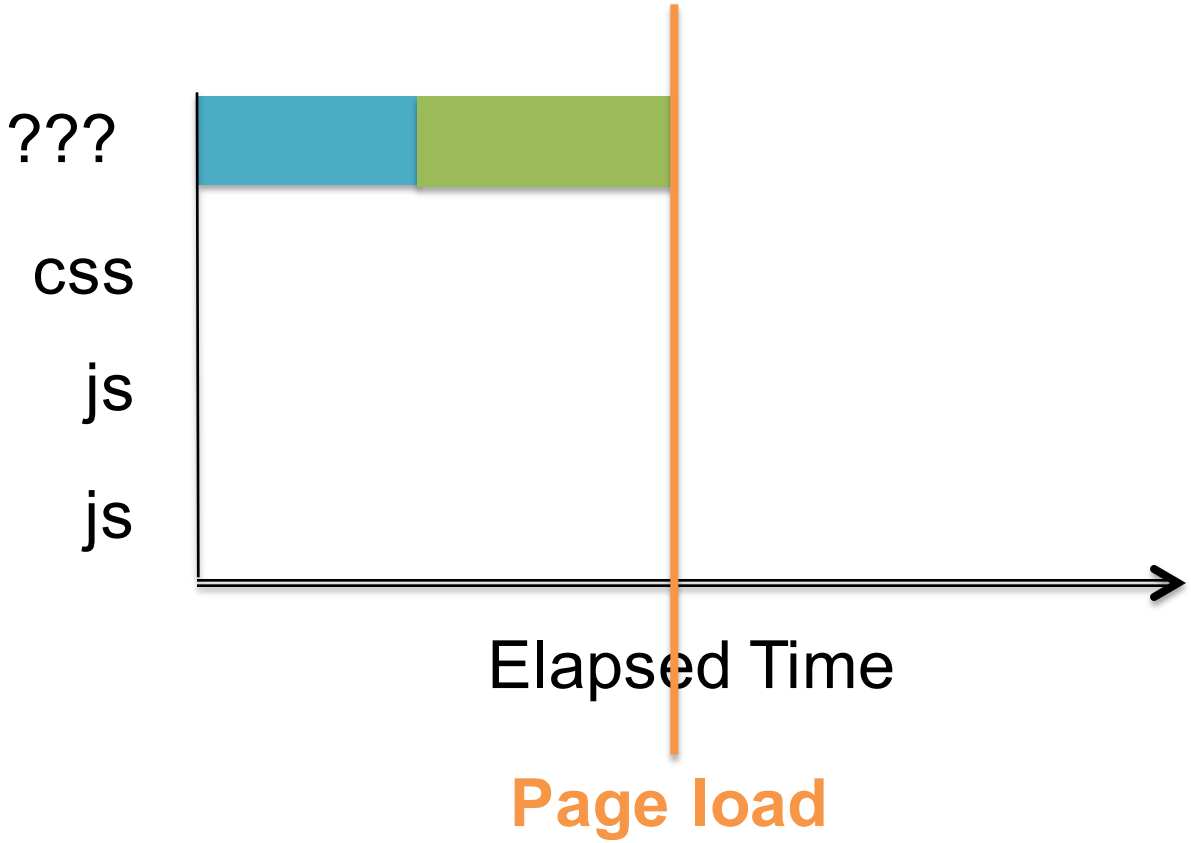


Network Computation → Dependency



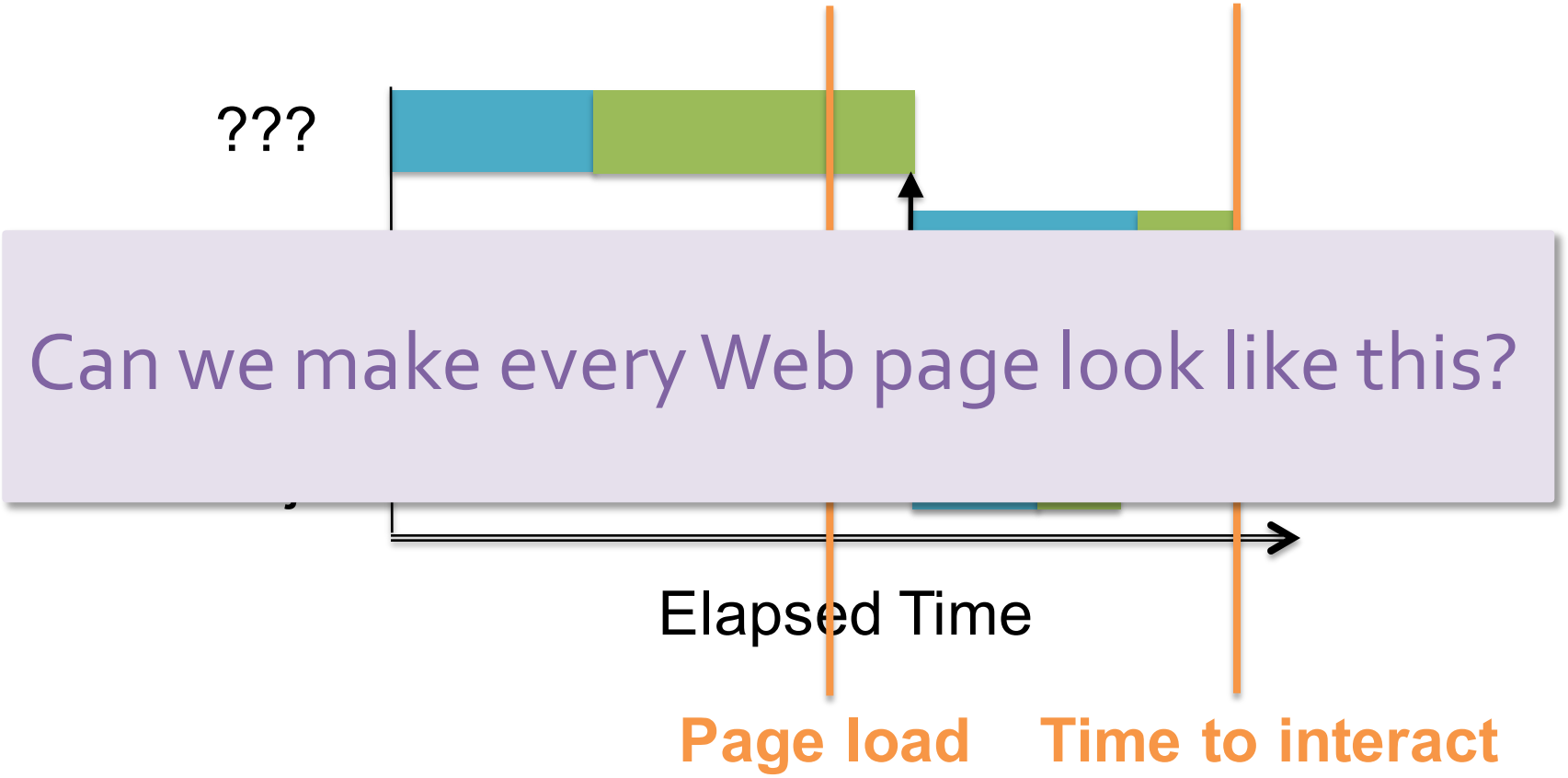
Network Computation → Dependency

What does the simplest dependency graph look like?



Network Computation → Dependency

nsdi '16
MARCH 16–18, 2016
SANTA CLARA, CA
SPONSORED BY USENIX IN COOPERATION WITH ACM SIGOPS
13th USENIX Symposium on Networked Systems Design and Implementation
usenix THE ADVANCED COMPUTING SYSTEMS ASSOCIATION



Can we make every Web page look like this?

Network Computation → Dependency

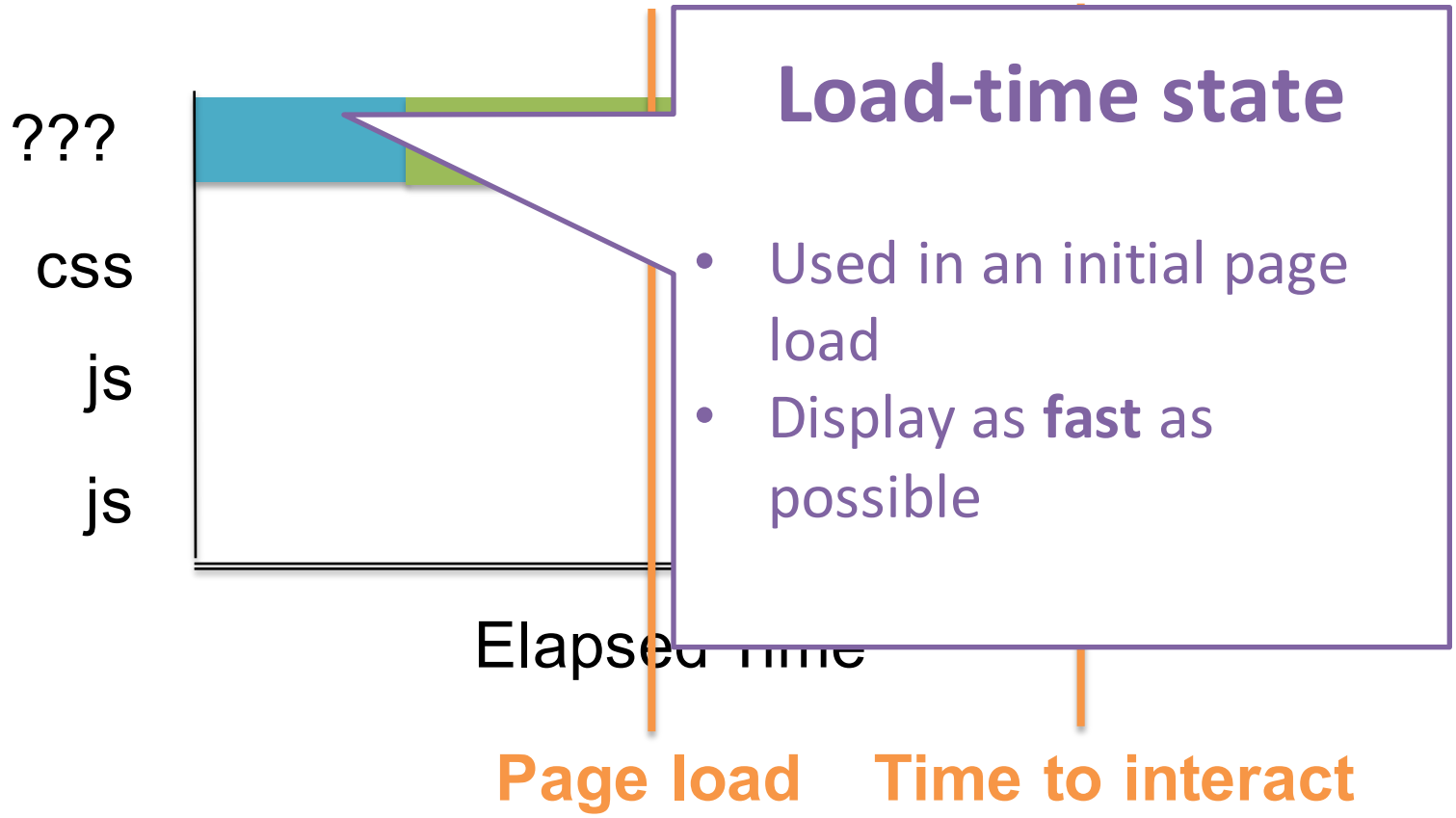
Yes, we want to make every page like this, automatically.

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

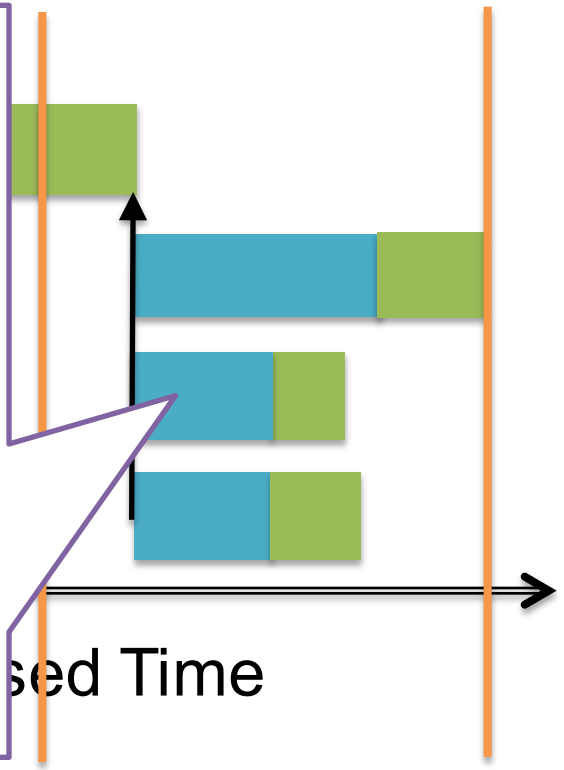


Approach: Split Browser

- 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



Page load **Time to interact**

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

```
{“loadTimeState”:{
  “css”:[“#main{font-size:12px;}"],
  “html”:{“children”: [{
   “tagNa
   “child
    “tagN
    “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]
 }]}]}}
```

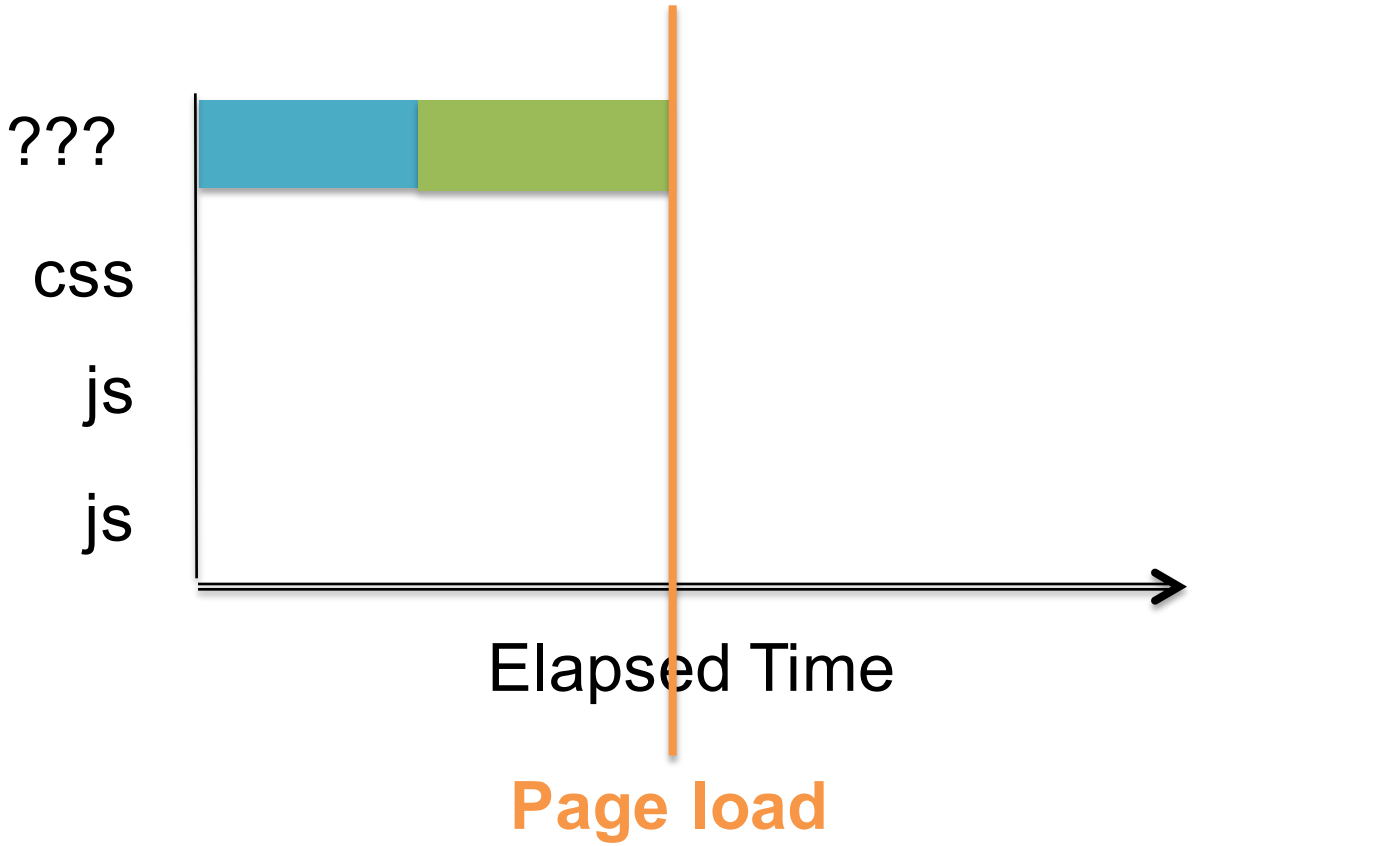
Visible HTML elements

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



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-

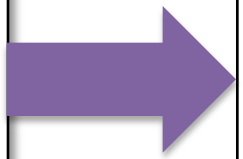
From here, how much can we improve?

What's equivalent to eval'ing this CSS?

```
#main {
  font-size:12px;
}
#main {
  font-size:12px;
}
#main {
  font-size:12px;
}
```

What's equivalent to eval'ing this CSS?

```
#main {
  font-size:12px;
}
#main {
  font-size:12px;
}
#main {
  font-size:12px;
}
```



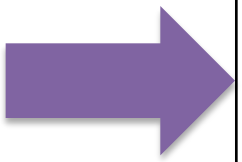
```
#main {
  font-size:12px;
}
```

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?

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

a += "hello world!\n"

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



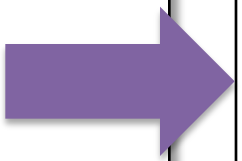
```
a += "hello world!\n"
   + "hello world!\n"
   + "hello world!\n"
```

What's equivalent to eval'ing this JS?

```
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?

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



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

Post-load state

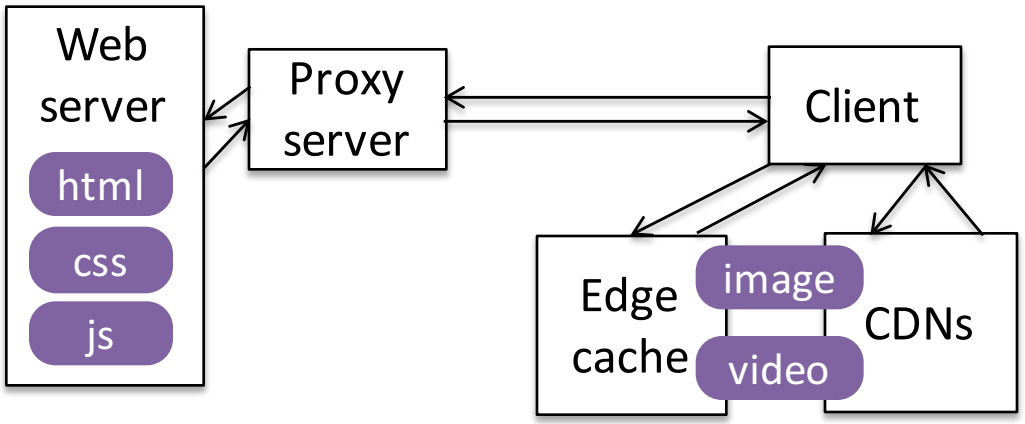
- Exploit the **idempotency** of evaluating CSS rules and JavaScript functions/statements
  - Eliminate redundant content that appeared in load-time state
  - Capture results of non-idempotent JS statements

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 3rd-party service: cloud servers



Implementation

- Server extension
  - Chrome's content_shell
  - Only handle HTML/JS/CSS
- Client browser
  - Chrome
  - JSON lexer, Blink, V8

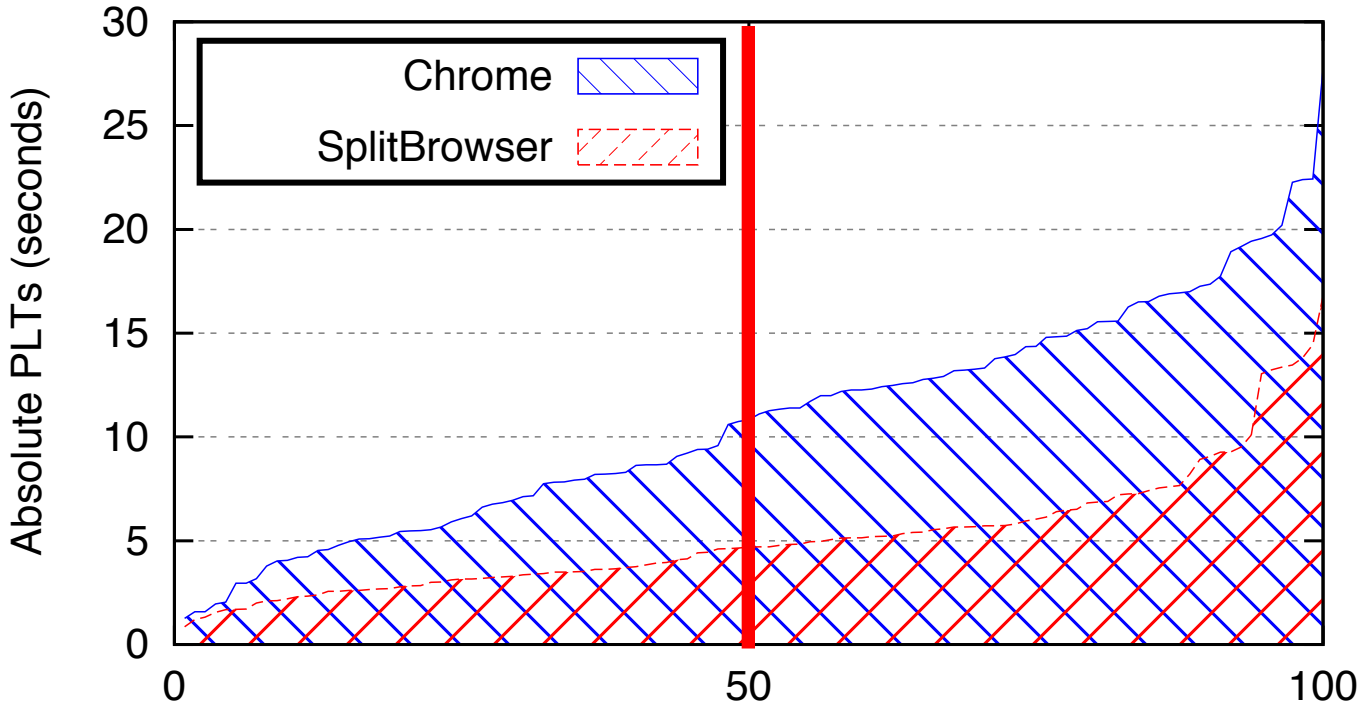
Outline

- 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

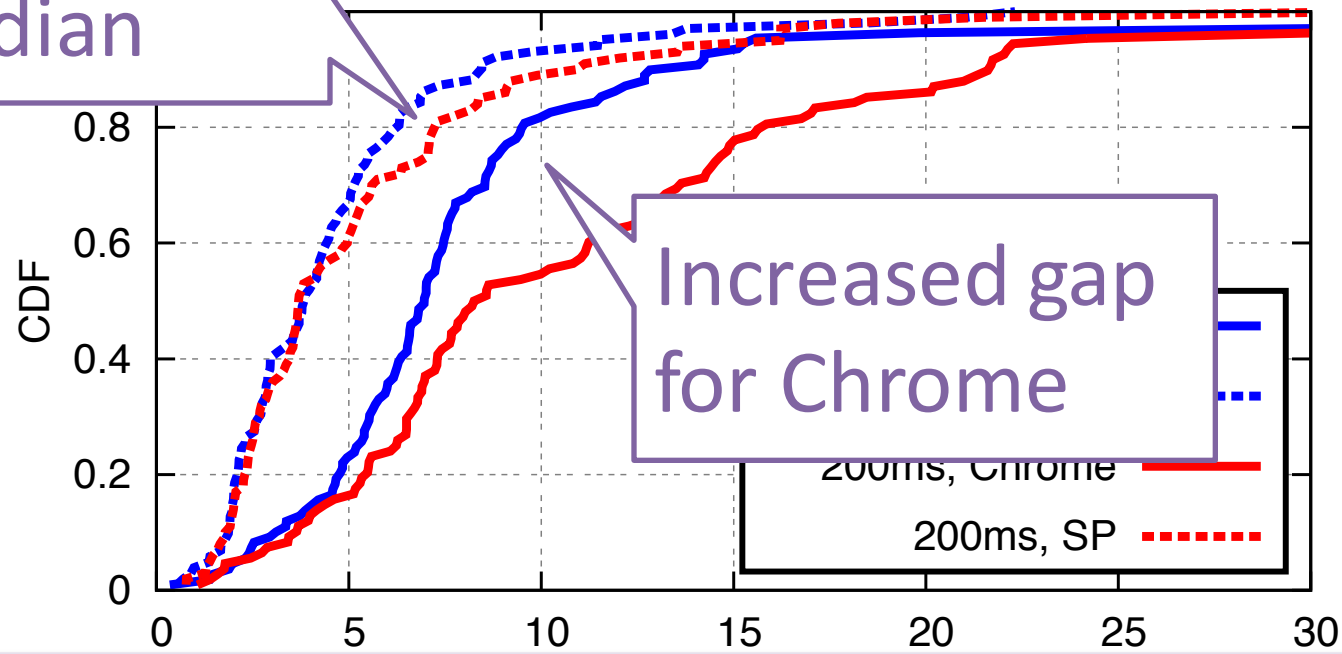
PLT on mobile



Shandian helps **60%** in the median case

PLT w/ varying RTT

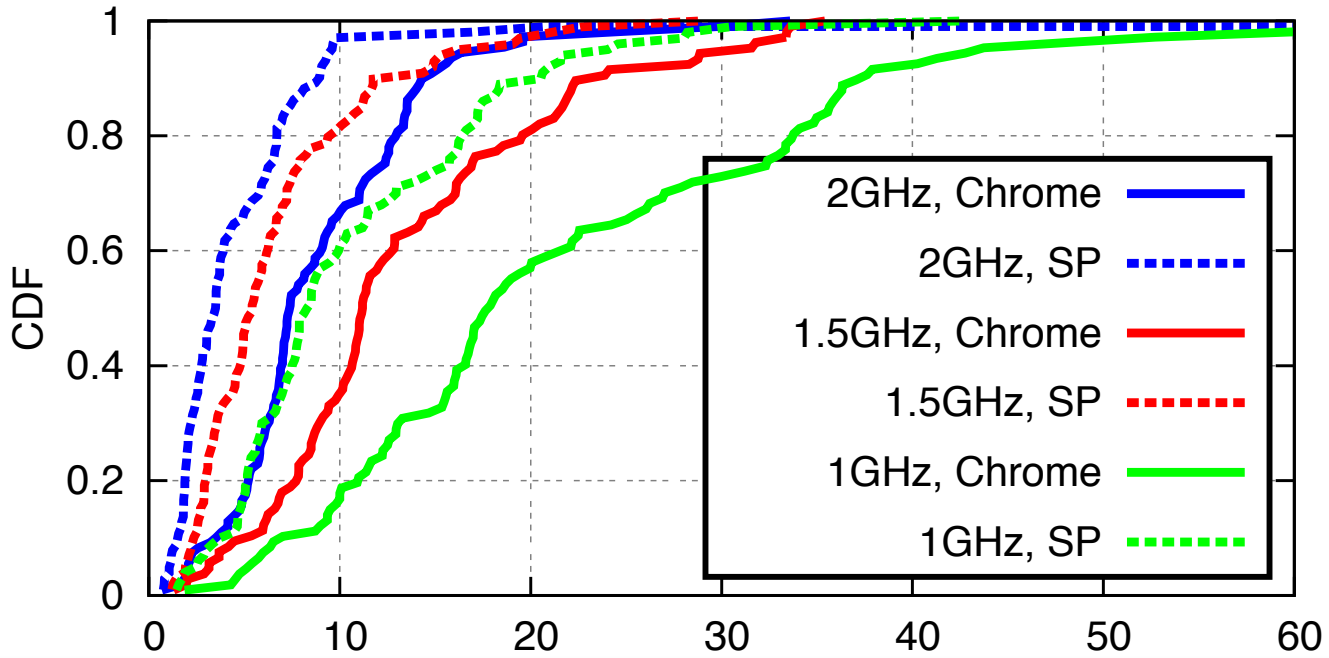
Small gap for Shandian



Increased gap for Chrome

Unlike Chrome, Shandian is not sensitive to RTT, due to simplified page load process

PLT w/ varying CPU



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