What-If Analysis of Page Load Time in Web Browsers Using Causal Profiling

Behnam Pourghassemi, Ardalan Amiri Sani, Aparna Chandramowlishwaran

University of California, Irvine

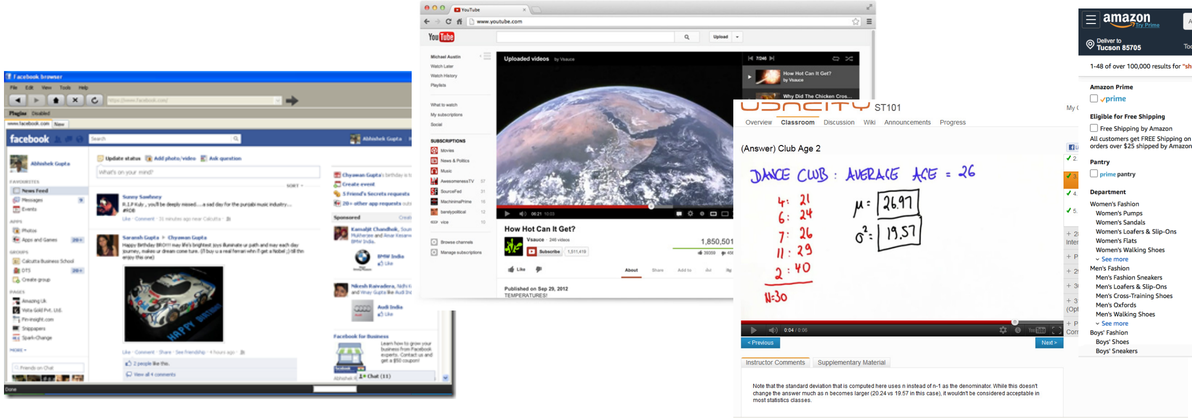
June 27, 2019 – SIGMETRICS'19

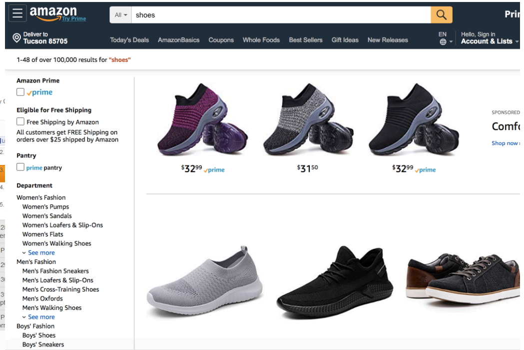
Outline

- Motivation
- Background
  - Browser architecture
  - Inter-dependency and critical path analysis
  - Chrome browser
  - Related work
- Methodology
  - Causal profiling
  - COZ+
- Experiments
  - Experiment setup
  - What-if analysis: Impact of computation stages on PLT
  - What-if analysis: Impact of PLT-variant factors on PLT
- Conclusion

Web browsers

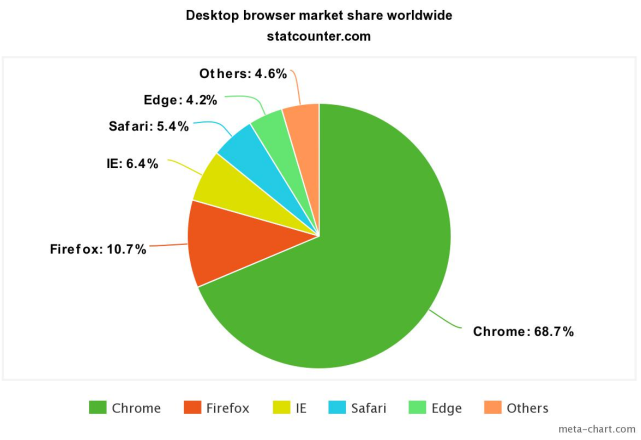
- Web browsers are one of the most frequently used applications for desktop and smartphones.





Browser performance

- Browser's usability and market share
- Webpage's business revenue
  - AliExpress reports 10.5% increase in orders by 36% reduction in the page load [1].
  - 53% of mobile site visitors leave a page that takes longer than 3 seconds to load [2].
- Web-app developers



Pixler.com

[1] https://edge.akamai.com/ec/us/highlights/keynote-speakers.jsp\#edge2016futureofcommercemodal

[2] https://www.thinkwithgoogle.com/marketing-resources/data-measurement/mobile-page-speed-new-industry-benchmarks

Challenges in improving performance

- Page load time (PLT): time from start of a user-initiated request to when the page content is loaded.

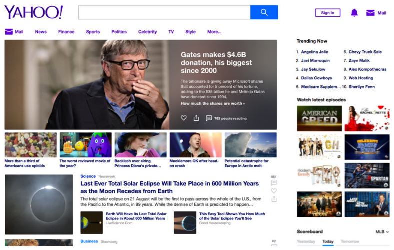
- Page load time potential bottlenecks
  - Network activities
  - Computation activities

- Challenges
  - Parallel and complex architecture of state-of-the-art browsers
  - Inter-dependency between activities and dynamic behavior of the page loading critical path

1995

2001

2005



2018

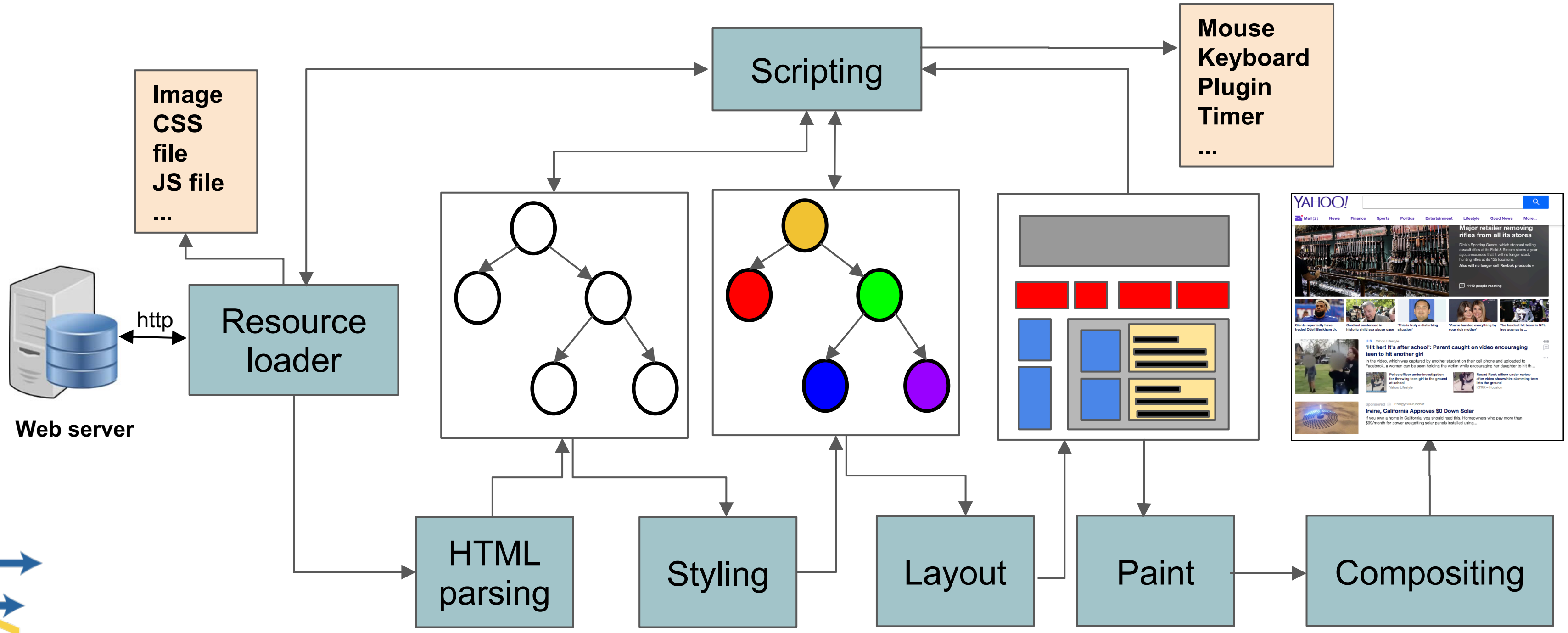
Research questions

1. **What are the critical computation activities in the page loading process?**
2. **How much performance improvement would we realistically achieve by reducing these bottlenecks?**

Outline

- Motivation
- **Background**
  - **Browser architecture**
  - **Inter-dependency and critical path analysis**
  - **Chrome browser**
  - **Related work**
- Methodology
  - Causal profiling
  - COZ+
- Experiments
  - Experiment setup
  - What-if analysis: Impact of computation stages on PLT
  - What-if analysis: Impact of PLT-variant factors on PLT
- Conclusion

Browser Architecture



Inter-dependency and critical path analysis

Flow	Loading an object → Parsing the tag that references the object
	Evaluating an object → Loading the object
	Rendering the DOM tree → Updating the DOM
	Loading an object referenced by a JavaScript or CSS → Evaluating the JavaScript or CSS
	Downloading/Evaluating an object → Listener triggers or timers
Output	Parsing the next tag → Completion of a previous JavaScript download and evaluation
	JavaScript evaluation → Completion of a previous CSS evaluation
	Parsing the next tag → Completion of a previous CSS download and evaluation
Lazy/Eager bindings	[Lazy]Loading an image appeared in a CSS → Parsing the tag decorated by the image
	[Lazy]Loading an image appeared in a CSS → Evaluation of any CSS that appears in front of the tag decorated by the image
	[Eager]Preloading embedded objects does not depend on the status of HTML parsing
Resource constraint	Number of objects fetched from different servers → Number of TCP connections allowed per domain
	Browsers may execute computational activities on the same thread, making dependencies b.w activities. This dependency is determined by the scheduling policy.

Table source: *Demystifying Page Load Performance with WProf* [NSDI'13]

Inter-dependency and critical path analysis

- Example:
  - <script> → block HTML parsing and execute JS
  - JS evaluation → evaluation of prior CSS

a.html

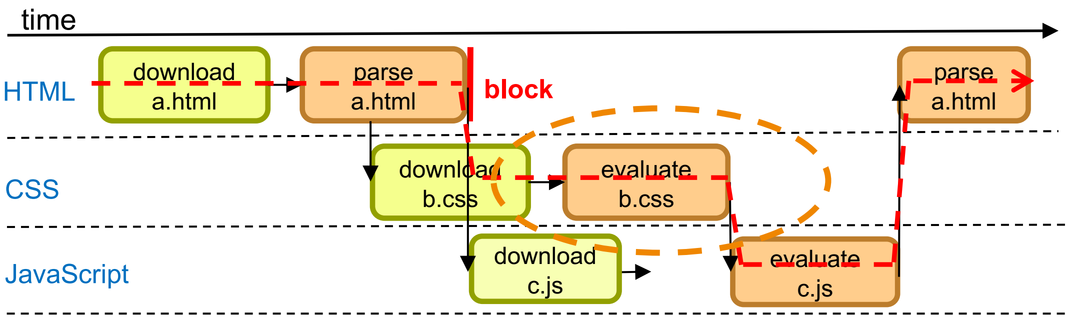
```
<html>
 <body>
  <p id="first_par"> old content </p>
  <link rel="stylesheet" href="b.css"></link>
  <script src="c.js"></script>
  ...
 </body>
</html>
```

b.css

```
#first_par{
 font-family:courier;
 text-align:center;
 ...
}
```

c.js

```
document.getElementById("first_par").innerHTML = "new content";
document.getElementById("first_par").style.color = "blue";
...
```



- Inter-dependency between activities generates the critical path for the page loading.

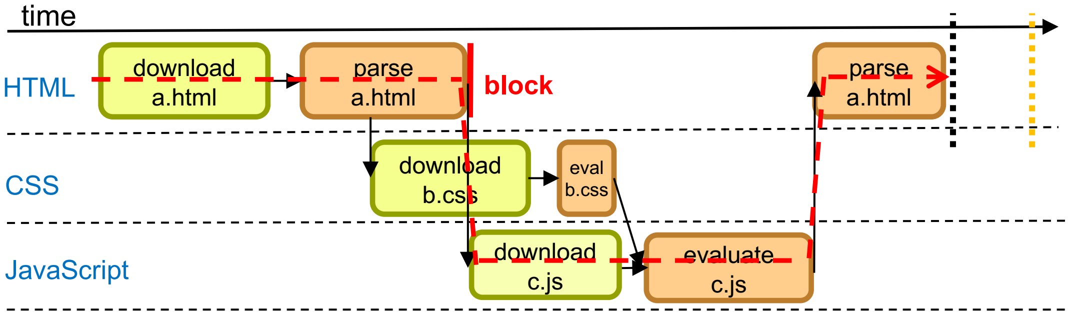
Inter-dependency and critical path analysis

- Example:
  - <script> → block HTML parsing and execute JS
  - JS evaluation → evaluation of prior CSS

a.html
```
<html>
 <body>
  <p id="first_par"> old content </p>
  <link rel="stylesheet" href="b.css"></link>
  <script src="c.js"></script>
  ...
 </body>
</html>
```

b.css
```
#first_par{
 font-family:courier;
 text-align:center;
 ...
}
```

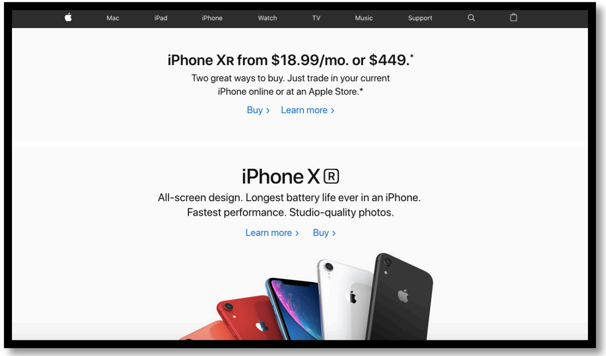
c.js
```
document.getElementById("first_par").innerHTML = "new content";
document.getElementById("first_par").style.color = "blue";
...
```



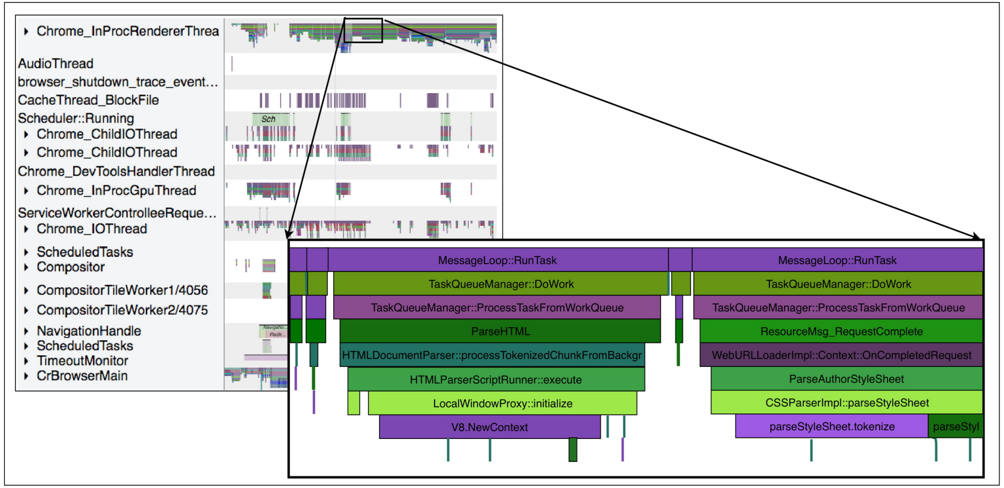
- Inter-dependency between activities generates the critical path for the page loading.

Chrome web browser

Example webpage: www.apple.com



Page load timeline for www.apple.com using chrome profiler



- Parallel and complex architecture of current browsers ⟶ difficult critical path analysis!

Related work

- Profiling and performance analysis tools
  - Dedicated browser profilers (e.g. Chrome DevTool)
  - General purpose profilers (e.g. gprof)
- Shortcoming
  - Do not provide quantitative and accurate what-if analysis
- Critical path analysis
  - webProphet [NSDI 2010]→ Dependency extraction through network perturbation.
  - Wprof [NSDI 2013]→ Dependency extraction based on predefined set of dependency policies and resource constraints.
  - Wprof-m [WWW 2016] and tempo [HotMobile 2011] → mobile browsers
- Shortcomings
  - Incomplete dependency extraction
  - Require exhaustive graph processing
  - Static analysis of the critical path

Outline

- Motivation
- Background
  - Browser architecture
  - Inter-dependency and critical path analysis
  - Chrome browser
  - Related work
- **Methodology**
  - **Causal profiling**
  - **COZ+**
- Experiments
  - Experiment setup
  - What-if analysis: Impact of computation stages on PLT
  - What-if analysis: Impact of PLT-variant factors on PLT
- Conclusion

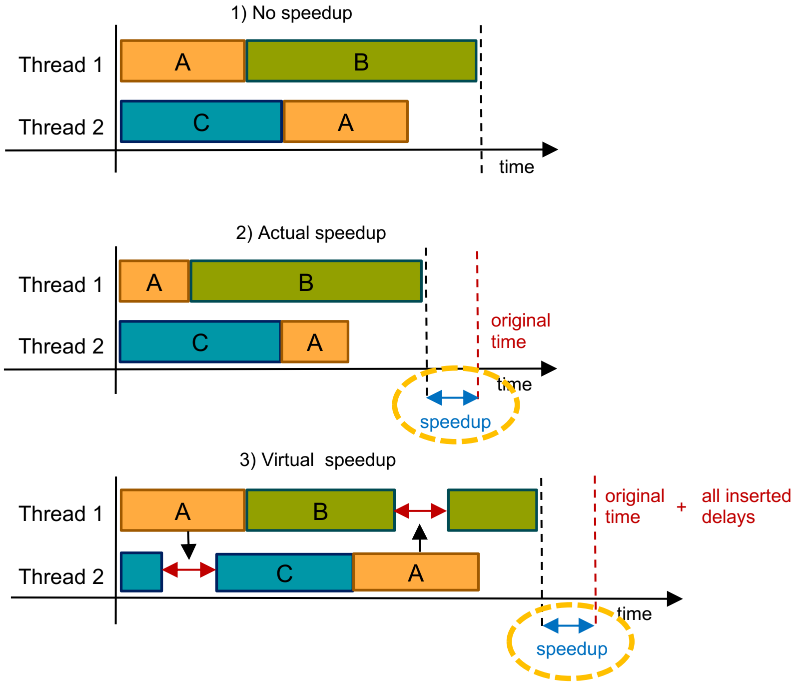
Methodology

- Use causal profiling [SOSP 2015] to detect performance bottlenecks and what-if analysis of main browser activities.
- Causal profiler can determine impact of optimization in a line of code on the total execution time.
- Causal profiler does not require explicit dependency graph generation and subsequent graph processing.
- Dependencies and impact of optimization are captured in runtime, hence, it considers dynamic behavior of the program.

Causal profiling

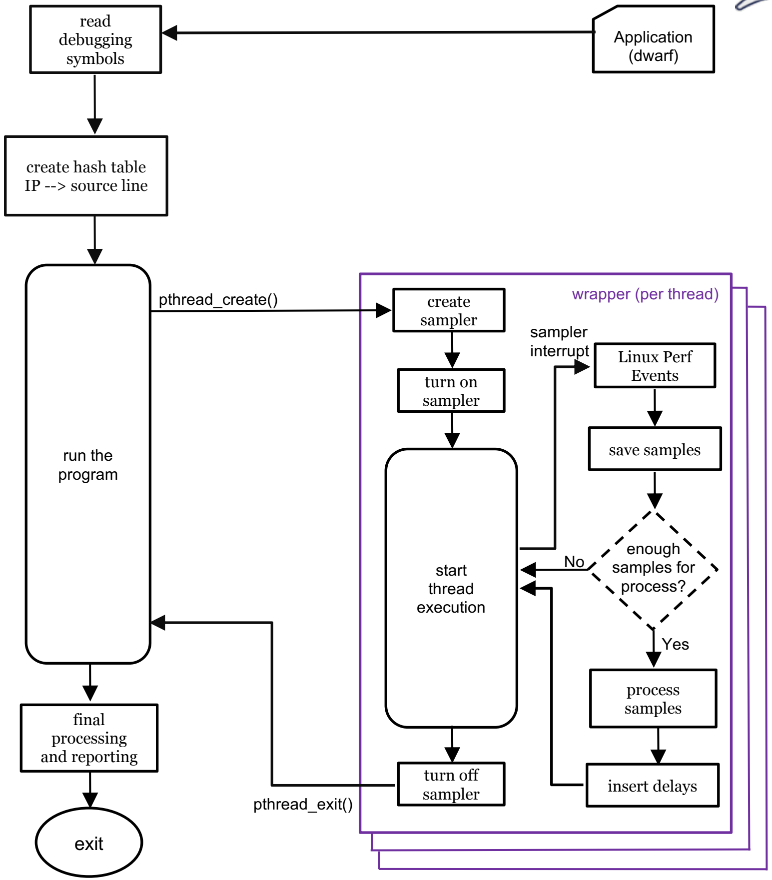
- Key idea: virtually speedup a selected code segment during the runtime.

- Virtual speedup
  - Run concurrent execution paths **slower** whenever the selected function is running.



COZ

- COZ: Implementation of causal profiler.
- Overview



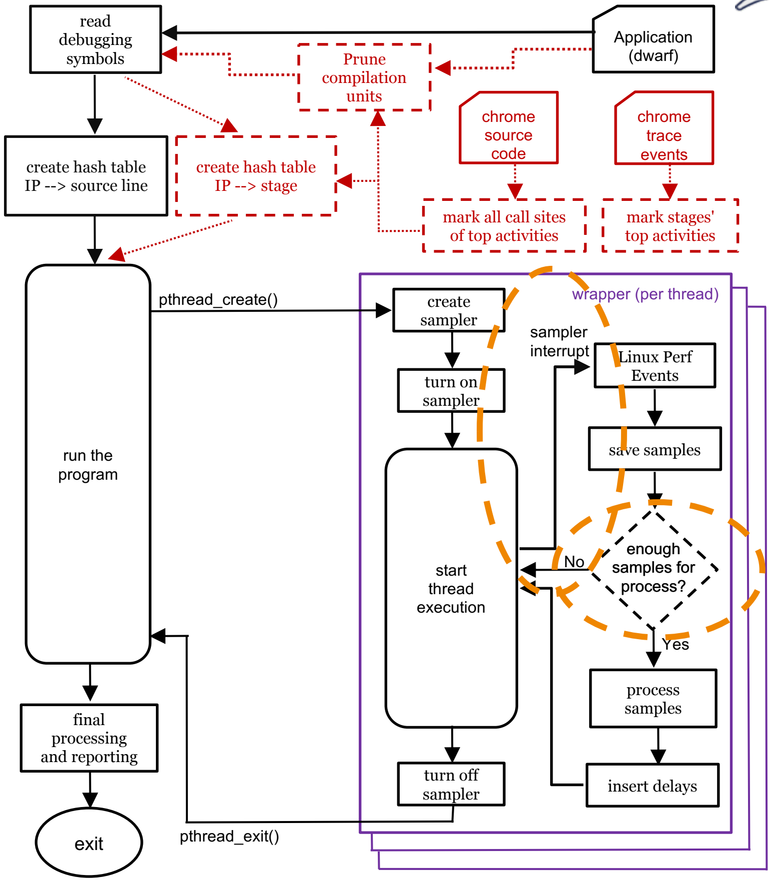
COZ limitations and COZ+

- Limitations:
  - Mainly tested on Parsec benchmarks (less than 5K LOC)
  - COZ crashes on large applications like Google Chrome (11 millions LOC!)
  - Design and implementation issues

- Develop **COZ+** on top of COZ
  - Fix implementation issues and redesign several modules
  - Make it scalable → low profiling overhead on large software systems
  - Optimize and customize for what-if analysis of Chromium browser
  - It is open-source! https://gitlab.com/coz-plus/coz-plus

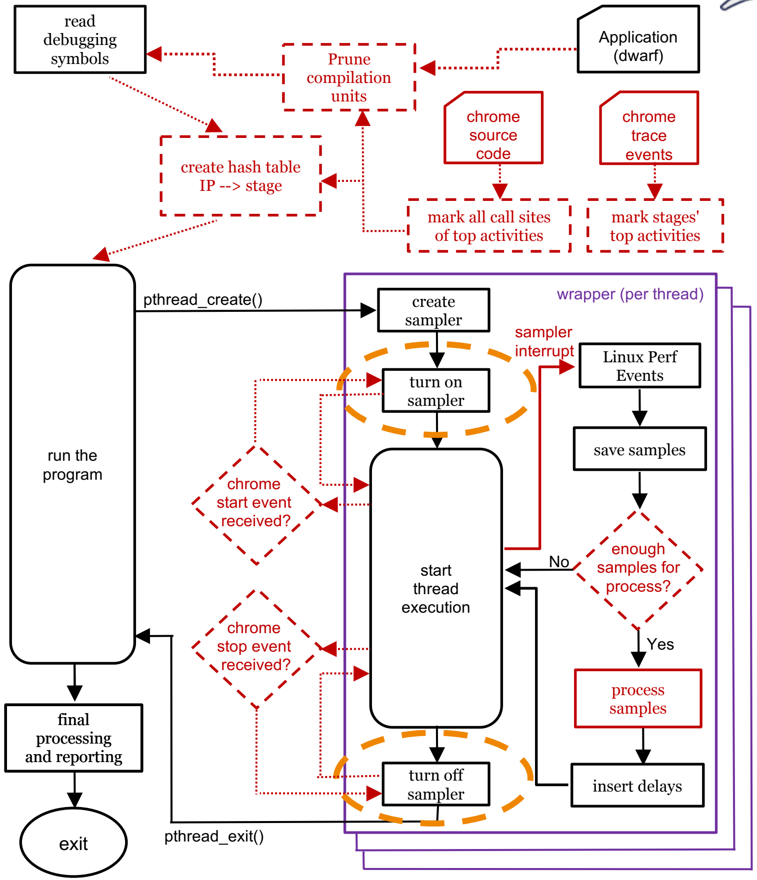
COZ+

- Optimizing symbol loading
  - Chromium:
    11 million C/C++ lines, 270K source files!
  - Take hours to read and process debugging symbols
  - New hash table is lighter
  - Take less than 1 minute to create a table!
- Flexible sampling
  - Frequency ↑ → Accuracy ↑ overhead ↑
  - batch size ↑ → Accuracy ↓ overhead ↓
  - Freq = 500Hz (2ms period)
  - batch size = 8 (PLT < 4s) and 10 (PLT > 4s)



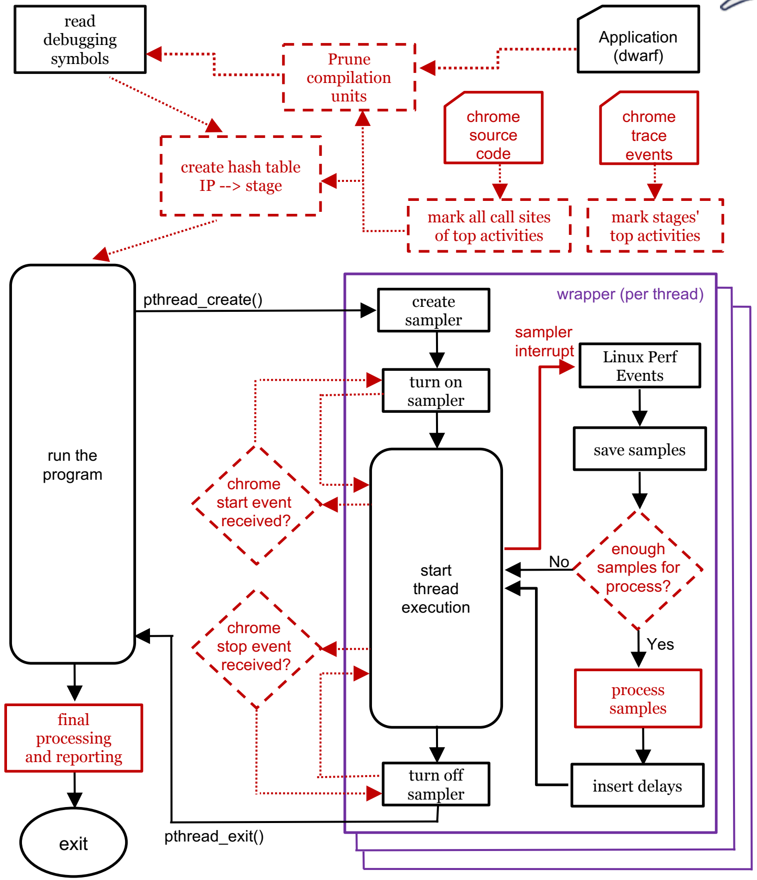
COZ+

- Metrics and reporting
  - COZ+ support multiple metrics for PLT
  - Example of start events:
    - navigationStart (enter URL)
    - onBeforeRequest (HTTP request is sent)
    - onHeadersReceived (the first byte is received)
  - Example of stop events:
    - DOMContentLoad (Dom is constructed)
    - loadFinish (Dom is loaded)
    - FP (first paint)
    - FMP (first meaningful paint)



COZ+

- Multi-process profiling
  - COZ profiler thread only sticks to application’s initial process
  - Challenge: debugging symbol table depends on the address space of the process
  - Solution: build debugging symbol table per child process → inefficient
  - Better solution: only debugging symbol table is sent to child process and add offset
  - COZ+ can profile child processes efficiently



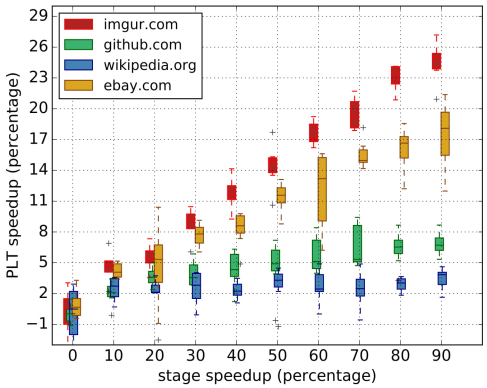
Outline

- Motivation
- Background
  - Browser architecture
  - Inter-dependency and critical path analysis
  - Chrome browser
  - Related work
- Methodology
  - Causal profiling
  - COZ+
- **Experiments**
  - **Experiment setup**
  - **What-if analysis: Impact of computation stages on PLT**
  - **What-if analysis: Impact of PLT-variant factors on PLT**
- Conclusion

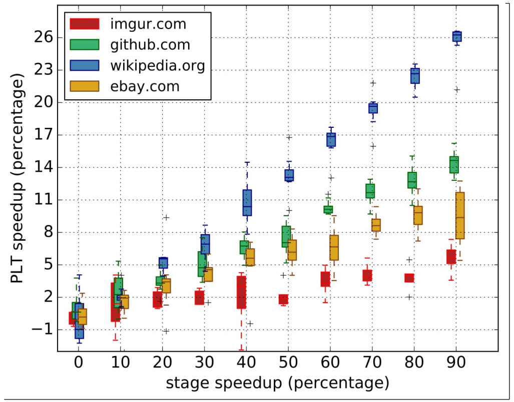
Experiment Setup

- System
  - MacBook Air (core i7, 4MB cache, 4GB RAM)
  - Zelda (Intel Xeon, 40MB cache, 64 GB RAM)
- Test suite
  - Top 100 webpages from Alexa
  - 10 runs for each configuration
- Network
  - 100Mbps Ethernet and 64Mbps WIFI
  - No local proxy
  - Use Linux Traffic Control (tc) to throttle network

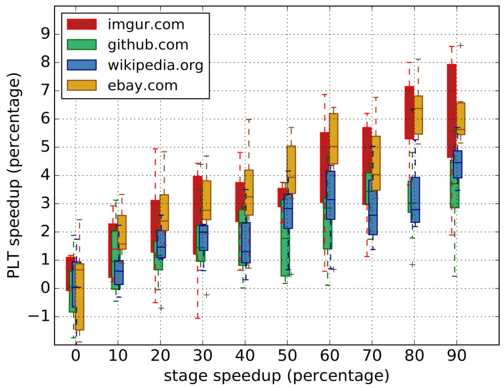
Impact of computation stages on PLT



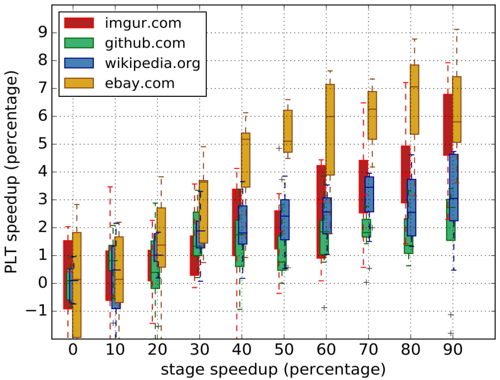
Scripting



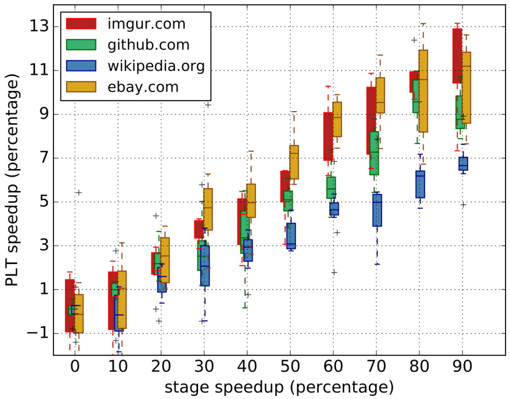
Layout



HTML



Painting

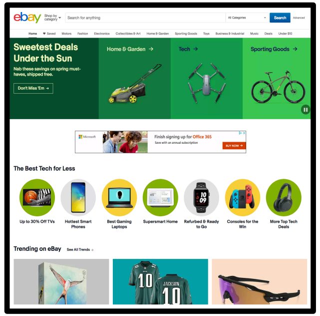


Styling

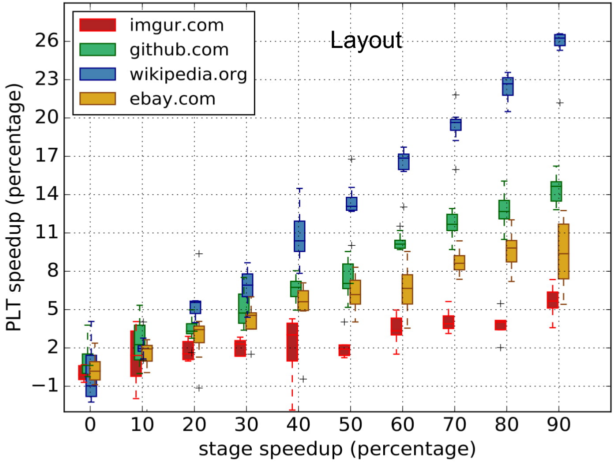
Impact of computation stages on PLT

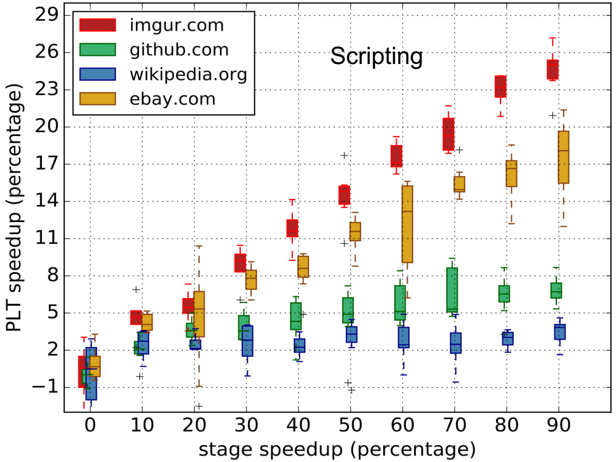
- Finding 1:
  - Mostly, a **linear** improvement in PLT → not enough concurrency between stages
- Finding 2:
  - Divergent patterns for webpages in different stages

wikipedia.com



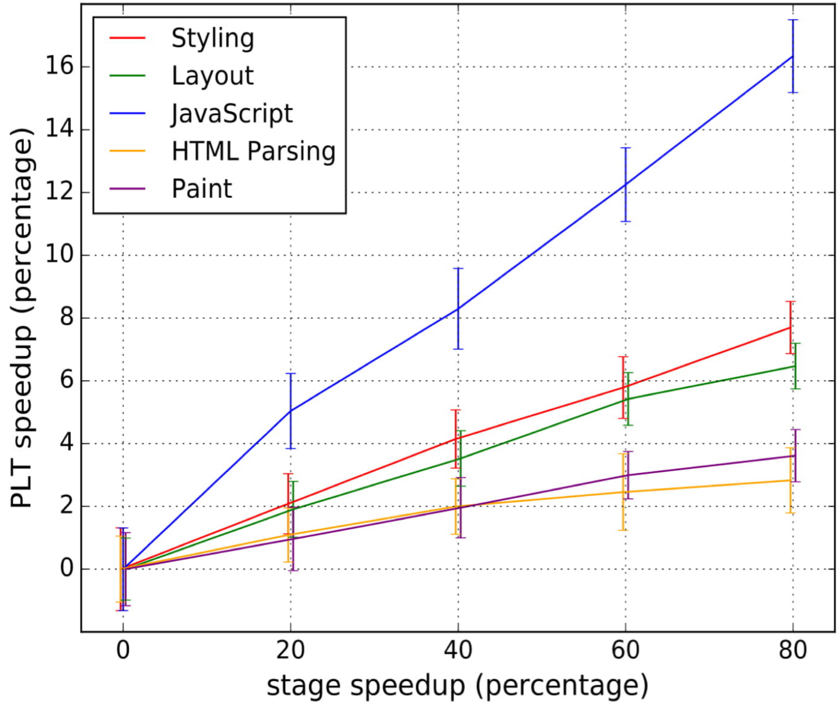
ebay.com





Impact of computation stages on PLT

- Finding 3:
  - JavaScript is the most influential stage compared to the other stages.
  - HTML parsing and Painting have insignificant impact on PLT.



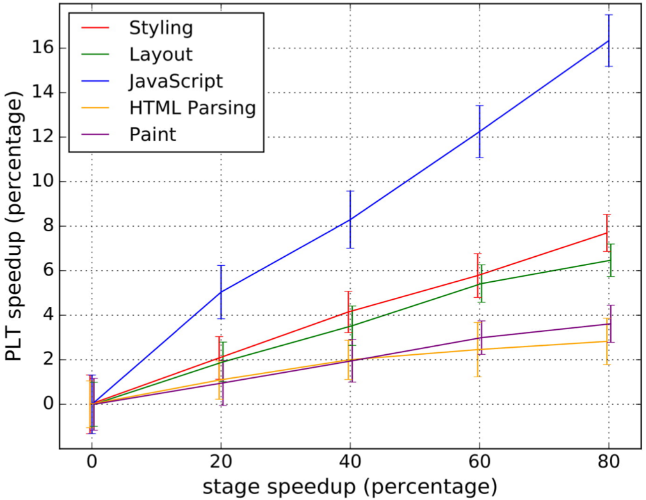
Impact of PLT-variant factors

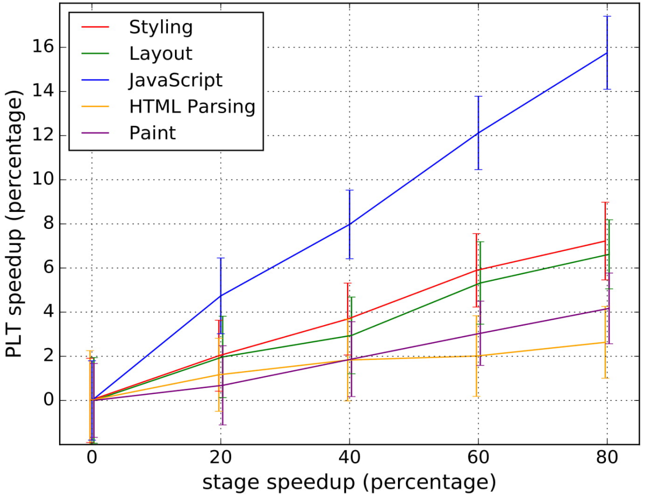
- How PLT-variant factors affect what-if analysis of computation stages?

- We examine effect of the following factors:
  - System hardware
  - Network
  - Browser caching

System hardware

- Finding 4:
    - Stage optimization payoff is fairly unrelated to the system hardware.

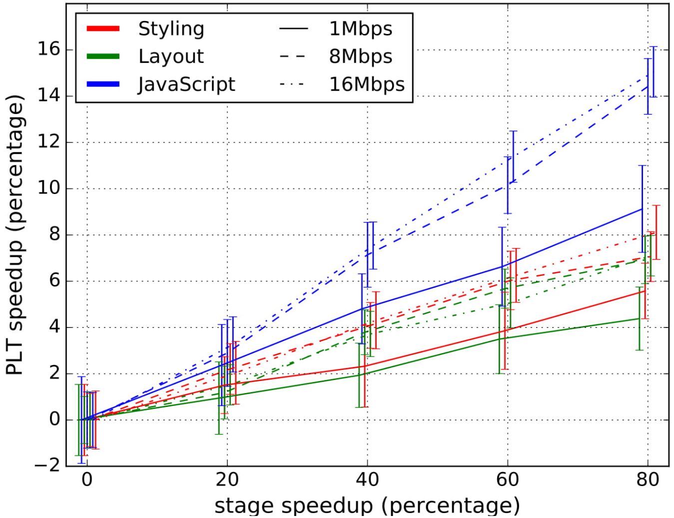


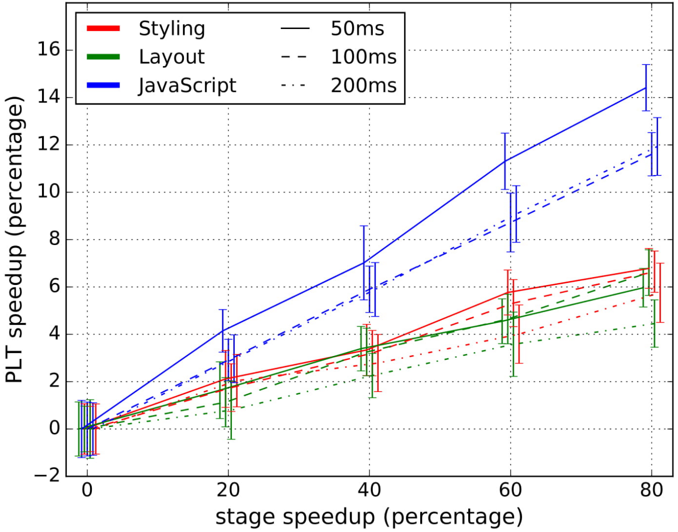


Top figure: what-if analysis on first system (MacBook air core i7).
Bottom Figure: what-if analysis on second system (Zelda Intel Xeon)

Network

- Finding 7:
  - Increasing network bandwidth and decreasing network delay increases the potential impact of computation stages on PLT.
  - Network bandwidth and delay does not change the pattern of what-if plots and the order of the stages in terms of effectiveness.
- Finding 8:
  - Increasing the network bandwidth has a trivial effect on what-if graph of stages in average and high-speed connections (i.e. above 8Mbps)





Browser caching

- Finding 11:
    - Browser caching doesn't affect stage's influence under high speed connection (e.g. 100Mbps)
    - PLT improvement doubles by enabling caching at 1Mbps network connection.

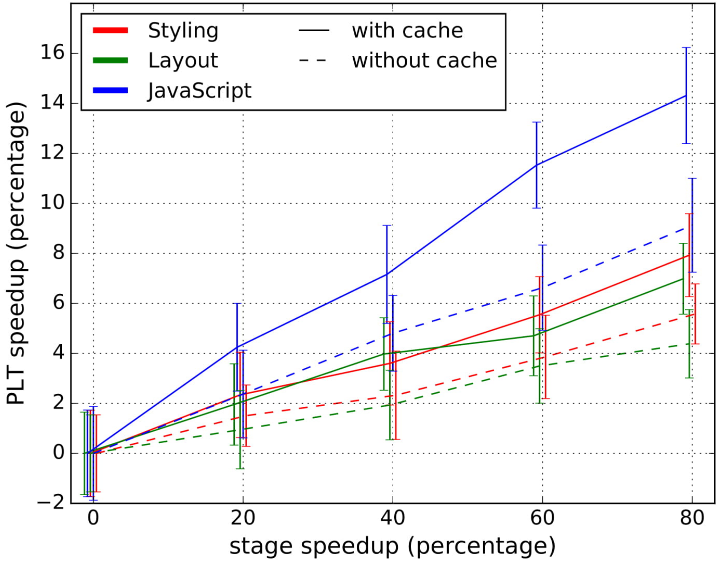


Figure: Impact of caching on what-if graphs under 1 Mbps connection

Outline

- ❑ Motivation
- ❑ Background
  - ❑ Browser architecture
  - ❑ Inter-dependency and critical path analysis
  - ❑ Chrome browser
  - ❑ Related work
- ❑ Methodology
  - ❑ Causal profiling
  - ❑ COZ+
- ❑ Experiments
  - ❑ Experiment setup
  - ❑ What-if analysis: Impact of computation stages on PLT
  - ❑ What-if analysis: Impact of PLT-variant factors on PLT
- ❑ **Conclusion**

Conclusion

- Investigate and prioritize the bottleneck activities in modern web browsers by using an **adaptive** approach (causal profiling).
- Develop **COZ+**, an overhaul of the COZ profiler, by adding multiple optimizations and redesigning several modules to make causal profiling practically feasible to large applications.
- Perform **comprehensive** and **quantitative** what-if analysis using COZ+ on the major browser stages.
- Examine impact of important PLT-variants on what-if analysis of computation stages.

Thanks for your attention!