Klotski: Reprioritizing Web Content to Improve User Experience on Mobile Devices

Michael Butkiewicz ♔, Daimeng Wang ♔, Zhe Wu ♖, Harsha V. Madhyastha ♕, Vyas Sekar ♘

♕ UC Riverside ♕ University of Michigan ♕ CMU
Motivation: Slow Mobile Web
Motivation: Slow Mobile Web
Motivation: Slow Mobile Web

Slow page loads → Less users, Lost business
Wide Range of Existing Solutions
Wide Range of Existing Solutions

Compression

Caching
Wide Range of Existing Solutions

- Compression
- Mobile format web pages
- SPDY
- Caching
Wide Range of Existing Solutions

- Compression
- Faster Networks
- Mobile format web pages
- SPDY
- Caching
- Better web browsers
Wide Range of Existing Solutions

- Compression
- Mobile format web pages
- Faster Networks
- Better web browsers

Common Focus: Reduce Page Load Times
Reducing Load Time is Not Enough

Rising Website Complexity
Reducing Load Time is Not Enough

Rising Website Complexity

Falling User Tolerance

Source: Akamai
Our Approach: Dynamic Reprioritization

- Forseeable Future: High load times norm, not exception
Our Approach:
Dynamic Reprioritization

- Forseeable Future: High load times norm, not exception
- Reformulate the problem:
Our Approach: Dynamic Reprioritization

- Forseeable Future: High load times norm, not exception
- Reformulate the problem:
  How to reduce page load time?
Our Approach: Dynamic Reprioritization

- Forseeable Future: High load times norm, not exception
- Reformulate the problem:
  How to reduce page load time?
  How to **Prioritize** the content most **important** to user?
Our Approach: Dynamic Reprioritization

- Forseeable Future: High load times norm, not exception
- Reformulate the problem:
  - How to reduce page load time?
  - How to **Prioritize** the content most **important** to user?
    - Typical tolerance limit of 2-4 seconds
    - Deliver “high utility” resources within time budget
Our Approach: Dynamic Reprioritization

- Forseeable Future: High load times norm, not exception
- Reformulate the problem:
  
  How to reduce page load time?
  
  How to **Prioritize** the content most *important* to user?
  
  - Typical tolerance limit of 2-4 seconds
  - Deliver “high utility” resources within time budget
- Our solution: Klotski proxy
  
  - No modifications to clients and web servers!
Klotski: Idealized View

[Diagram showing a face, Klotski Proxy, and Web Server connected with arrows indicating the flow of interaction.]

Webpage Resource
Klotski: Idealized View

Diagram:
- User
- Klotski Proxy
- Web Server
- Webpage Resource
Klotski: Idealized View

Klotski Proxy

Web Server

High Priority Resource

Low Priority Resource
Klotski: Idealized View
Challenges with Idealized View

C1: Dynamic content + dependencies
Challenges with Idealized View

C1: Dynamic content + dependencies
C2: Fast selection of subset to prioritize

High Priority Resource
Low Priority Resource
Challenges with Idealized View

C1: Dynamic content + dependencies
C2: Fast selection of subset to prioritize
C3: How long will a subset take?
Challenge 1: Dynamic Content and Dependencies

Robust Dependency Knowledge

Image A requires Javascript

Web Server

High Priority Resource
Low Priority Resource

time budget

Klotski Proxy
Intuition: Page structure is stable

- Prior work on static dependencies
  - E.g., WebProphet, WProf
Intuition: Page structure is stable

- Prior work on static dependencies
  - E.g., WebProphet, WProf

- Problem: Dependencies not reusable due to dynamic content
Intuition: Page structure is stable

- Prior work on static dependencies
  - E.g., WebProphet, WProf

- Problem: Dependencies not reusable due to dynamic content

- Our observation:
  - Nodes in DAG change
  - DAG structure largely stable
Intuition: Page structure is stable

- Prior work on static dependencies
  - E.g., WebProphet, WProf

- Problem: Dependencies not reusable due to dynamic content

- Our observation:
  - Nodes in DAG change
  - DAG structure largely stable

Load page repeatedly to generate fingerprint:
- DAG structure with a URL pattern at every node
- Pattern generalizes URL of dynamic resources
Learn URL Patterns

• Generalize known prior URLs of a dynamic resource

  foo.com/SG39HZ78/a.js  foo.com/SHFS2732/a.js  →  foo.com/*a.js
Learn URL Patterns

- Generalize known prior URLs of a dynamic resource

  foo.com/SG39HZ78/a.js
  foo.com/SHFS2732/a.js → foo.com/*/a.js

- 3 Cases = 90% of Replacements:
  - Single token in URL changes
  - Only URL argument changes: www.site.org/a.js?FOO=1...
  - CDN node name: {CDN2.bar.com/x.jpg, CDN5.bar.com/x.jpg}
Identify Resource Replacements

- Capture set of prior URLs:
  Track Replacement of Resource over multiple page loads
Identify Resource Replacements

- Capture set of prior URLs: Track Replacement of Resource over multiple page loads
Identify Resource Replacements

- Capture set of prior URLs: Track Replacement of Resource over multiple page loads

- Combination of techniques:
  - Match position in DAG dependency structure
  - Identical position on screen
  - Identical reference in source
Challenge 2: Optimal Prioritization Schedule

Which subset to prioritize and how?

High Priority Resource
Low Priority Resource

Web Server

High Priority Resource
Low Priority Resource

Klotski Proxy

time budget
Select Subset of High Utility Resources

Dependency structure in fingerprint
Select Subset of High Utility Resources

Dependency structure in fingerprint
Select Subset of High Utility Resources

- Reduces to knapsack with dependencies: NP-Hard
- Apply greedy heuristic

Dependency structure in fingerprint
Prioritizing High-Utility Resources

- **Static URLs**: Use *SPDY PUSH* to pre-emptively deliver as soon as main HTML is requested.

- **Dynamic URLs**: Prioritize delivery if *match with regular expression* of a selected resource.
Challenge 3: Estimating Load Times in the “Wild”

Will subset load within budget?

High Priority Resource
Low Priority Resource

Web Server

Klotski Proxy

time budget

High Priority Resource
Low Priority Resource
Seemingly Natural Non-Solutions

Model: bytes, #requests, ..?
Use Apriori Load Times?

Model: Too simplistic to capture browser effects
Apriori Loads: No longer valid with reprioritization
Cannot capture diversity in client conditions

Web Server
Klotski Proxy
time budget
Resource
Resource
Resource
Resource
Intuition Behind Klotski Estimator

Bottleneck = Client-Proxy Link (e.g., 4G)
Intuition Behind Klotski Estimator

Build Fluid Model Simulation
- Proxy as work conserving scheduler
- Priorities/Dependencies, fairly shared bw for concurrent transfers
- Some subtle issues: PUSH, client processing delays
Klotski: System Architecture

Web Servers

Klotski Service

BackEnd

FrontEnd

Fingerprint Generator

Resource Selection

Load Time Estimator

Clients
Klotski: Experimental Results

Data Set

Websites
● 50 Random From Alexa Top 200

Mobile Device
● Android Smartphone
● 4G Connection
● Google Chrome

Paper Only Results
● Desktop PC + Ethernet
● Smartphone + Full Website
● User study on utility preferences
● Resource churn over time
Klotski Improves User Experience

![Box plot showing the fraction of util loaded across different load time budgets.](image-url)
Klotski Improves User Experience

Within the first X seconds
Klotski Improves User Experience

% of site's high util resources loaded

Within the first X seconds
Klotski Improves User Experience

50 Websites
Klotski Improves User Experience

50 Websites

- 75% loaded within 2 seconds
- 50% loaded within 3 seconds
- 25% loaded within 4 seconds
Klotski Improves User Experience

Median site 25% ATF in 2 secs
Klotski Improves User Experience

Fraction of Utility Loaded

Load Time Budget

1 sec 2 sec 3 sec 4 sec

Original Klotski
Klotski Improves User Experience

35% utility gain
Klotski Improves User Experience

2x Util

25% ➝ 60%
High Utility Gain For Diverse Users

Relative Increase in Utility (%)
High Utility Gain For Diverse Users

Relative Increase in Utility (%)
High Utility Gain For Diverse Users

Utility Gained at 2 Seconds

User Preferences
High Utility Gain For Diverse Users
Conclusions

- Mobile web continues to be a pain point
  - Focus on load time alone is likely insufficient
- Instead we focus on dynamic reprioritization
- Key challenges we address:
  - dynamic dependency representation
  - fast resource selection
  - load time estimation
- Klotski greatly improves user experience
  - for diverse preferences