How far can client-only solutions go for mobile browser speed?

Zhen Wang, Felix Xiaozhu Lin, Lin Zhong, Mansoor Chishtie
Mobile browsers are slow

• Average browser delay: 5 seconds

• Commercial impact
  – Google: 500 ms => 20% traffic loss
  – Amazon: 100 ms => 1% sales loss

Why is mobile browser slow?

Elapsed time (ms)

Resource Group

- Resource Loading
- Parsing
- Scripting
- Style
- Layout
- Painting
Why is mobile browser slow?

Elapsed time (ms)

Request
Why is mobile browser slow?

Elapsed time (ms)

Resource Group

Redirections

Resource Loading  Parsing  Scripting  Style  Layout  Painting
Why is mobile browser slow?

Elapsed time (ms)

Resource Group

Data packets

Resource Loading  Parsing  Scripting
Style  Layout  Painting
Why is mobile browser slow?

Elapsed time (ms)

Resource Group

0              1000            2000            3000            4000

1

2

3

4

5

6

7

8

Resource Loading

Parsing

Scripting

Style

Layout

Painting
Why is mobile browser slow?

Elapsed time (ms)

Resource Group

Resource Loading  Parsing  Scripting  Style  Layout  Painting

Subresources
Why is mobile browser slow?

- Resource Loading
- Parsing
- Scripting
- Style
- Layout
- Painting

Rendering
Why is mobile browser slow?

This is part of resource loading. It is mainly spent on network.
Computation does not matter

Elapsed time (ms)

Resource Group

Speed up layout operation
Shrink green boxes...
Computation does not matter
Computation does not matter

Elapsed time (ms)

Resource Group

0 1000 1500 2000 2500 3000 3500 4000

1 2 3 4

5 6 7 8

Resource Loading Parsing Scripting Style Layout Painting

They will not move

Shrink

Shift to the left
Computation does not matter
Computation does not matter

Resource Group

Elapsed time (ms)

0 1000 1500 2000 2500 3000 3500 4000

1
2
3
4
5
6
7
8

After shrink and shift

New

Original

Resource Loading
Parsing
Scripting
Style
Layout
Painting
Resource loading matters

Long network round trip time (RTT)
Resource loading matters

So many resources are needed ...
- JavaScript, CSS, image files
Existing solutions

• Infrastructure support
  – Difficult to deploy
  – Depend on server or proxy capability
  – Violate end-to-end security
Client-only approaches

— Caching

— Prefetching
Client-only approaches

- Caching
- Prefetching

How effective are they?
Understanding mobile users

• 24 iPhone 3GS users

• Feb. 2010 to Feb. 2011

• Timestamp and URL
Understanding mobile users

- Top 10 websites account for over 80% of visits
- 75% of the webpage visits are first-time
- 65% of the subresource visits are re-visits
Client-only approaches

- Caching
- Prefetching

How effective are they?
Caching is not effective

• Theory: *Cache helps by reducing network activity*

• Reality:
  70% of resource requests incur network activity
  – Half of them are due to cache revalidation
  – Cache revalidation saves bandwidth, not RTT
Web prefetching

Predict and download the right webpage while the user is still reading the left webpage.
Web prefetching is not effective

- Browser delay reduction: 1%
- Unnecessary data usage: 84%

- Reason
  - 75% of webpage visits are first-time
  - > hard to predict the next visit
Existing client-only solutions

• Caching is not effective
  – Resources quickly expire and need revalidation

• Web prefetching is even harmful
  – User behavior is not very predictable
Existing client-only solutions

• Caching is not effective
  – Resources quickly expire and need revalidation

• Web prefetching is even harmful
  – User behavior is not very predictable

Any new approaches?
Our insight

Intra-group dependency
Our insight

Inter-group dependency
Our insight

Single network connection for over 2 seconds!
Our insight

How many concurrent network connections are supported by browsers?

Single network connection for over 2 seconds!

**Figure:**
- Resource Loading
- Parsing
- Scripting
- Style
- Layout
- Painting

**Table:**
- Resource Group
  - 0
  - 1
  - 2
  - 3
  - 4
  - 5
  - 6
  - 7
  - 8

**Graph:**
- Elapsed time (ms)
  - 0
  - 1000
  - 1500
  - 2000
  - 2500
  - 3000
  - 3500
  - 4000
Our insight

Single network connection for over 2 seconds!

New approach: speculative loading
Speculative loading

Elapsed time (ms)

Resource Group

0              1000            1500            2000            2500            3000            3500            4000

1 2 3 4 5

1 2 3 4 5 6 7 8

Speculative loading

- Resource Loading
- Parsing
- Scripting
- Style
- Layout
- Painting
Speculative loading

Elapsed time (ms)

0 1000 1500 2000 2500 3000 3500 4000

Resource Group

1 2 3 4 5 6 7 8

Scripting

Resource Loading

Parsing

Style

Layout

Painting
Speculative loading

Elapsed time (ms)

Resource Group

0 1000 1500 2000 2500 3000 3500 4000

Resource Loading Parsing Scripting Style Layout Painting

Browser Delay Reduction
Speculative loading

How to predict the subresources accurately?

Browser Delay Reduction

<table>
<thead>
<tr>
<th>Resource Group</th>
<th>Elapsed time (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1000</td>
</tr>
<tr>
<td>1</td>
<td>1500</td>
</tr>
<tr>
<td>2</td>
<td>2000</td>
</tr>
<tr>
<td>3</td>
<td>2500</td>
</tr>
<tr>
<td>4</td>
<td>3000</td>
</tr>
<tr>
<td>5</td>
<td>3500</td>
</tr>
<tr>
<td>6</td>
<td>4000</td>
</tr>
</tbody>
</table>

Resource Loading | Parsing | Scripting | Style | Layout | Painting |

- Red: Resource Loading
- Black: Parsing
- Purple: Scripting
- Light Blue: Style
- Green: Layout
- Orange: Painting
How is a website structured?

- Website Node: rice.edu
- Subdomain Node: a.rice.edu, b.rice.edu
- Webpage Node: a.html, b.html, c.html, d.html
- Subresource Node: a.jpg, b.css, c.jpg, d.js, e.css, f.jpg

shared by multiple webpages
How is a website structured?

- **Website Node**: rice.edu
- **Subdomain Node**: a.rice.edu, b.rice.edu
- **Webpage Node**: a.html, b.html, c.html, d.html
- **Subresource Node**: a.jpg, b.css, c.jpg, d.js, e.css, f.jpg

*shared by multiple webpages*

*More sharing -> Easier subresource prediction*
How is a website structured?

76% of the subresources of a webpage are shared by other webpages in the same website.
Web prefetching    vs.    Speculative loading

Predict webpages    a user will visit

Predict subresources    a webpage will need

User behavior prediction    Server behavior prediction
Web prefetching vs. Speculative loading

Predict webpages a user will visit
User behavior prediction

Predict subresources a webpage will need
Server behavior prediction

Much easier!
First-time visits

- 75% Webpage
- 35% Subresource

Web prefetching vs. Speculative loading

User behavior prediction

Server behavior prediction

Much easier!
How much can speculative loading speed up a mobile browser?

– Upper bound through simulation

– Real world measurement
Upper bound of improvement

- **Legacy**: 3.2 s, 6.3 s, 6.7 s
- **Speculative**: 3.2 s, 4.3 s, 4.6 s

Legend:
- Blue: Fresh Cache
- Red: Expired Cache
- Green: Empty Cache
Upper bound of improvement

- Legacy:
  - Fresh Cache: 3.2 s
  - Expired Cache: 6.3 s
  - Empty Cache: 6.7 s

- Speculative:
  - Fresh Cache: 3.2 s
  - Expired Cache: 4.3 s
  - Empty Cache: 4.6 s
Upper bound of improvement

<table>
<thead>
<tr>
<th>Legacy</th>
<th>Speculative</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Fresh Cache</td>
<td>Fresh Cache</td>
</tr>
<tr>
<td>3.2 s</td>
<td>3.2 s</td>
</tr>
<tr>
<td>Expired</td>
<td>Expired Cache</td>
</tr>
<tr>
<td>6.3 s</td>
<td>4.3 s</td>
</tr>
<tr>
<td>Cache</td>
<td>Cache</td>
</tr>
<tr>
<td>6.7 s</td>
<td>4.6 s</td>
</tr>
<tr>
<td>Empty Cache</td>
<td>Empty Cache</td>
</tr>
</tbody>
</table>

33% improvement
Upper bound of improvement

Realistic Cache
1.4 s (22%)
Tempo

A speculative mobile browser
**Tempo architecture**

- Metadata Repository
- Speculative Loader
- Update Service
- Temporary Cache

Tempo Enhancement

Rest of the Android mobile browser

WebKit
Cache
**Tempo architecture**

- **Metadata Repository**: Store each website’s resource graph
- **Speculative Loader**: Predict and load subresources speculatively
- **Update Service**: Update and trim metadata repository
- **Temporary Cache**: Store resources with “no-cache” header
### Metadata Repository

<table>
<thead>
<tr>
<th>Website</th>
<th>Resource graph</th>
</tr>
</thead>
<tbody>
<tr>
<td>google.com</td>
<td>Google resource graph</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>cnn.com</td>
<td>CNN resource graph</td>
</tr>
</tbody>
</table>

**Website node**

**Type:** Website  
**URL:** cnn.com  
**Last visit time**  
**Children:** ...

**Subdomain nodes**

**Type:** Subdomain  
**URL:** m.cnn.com  
**Last visit time**  
**Children:** ...

**Webpage nodes**

**Type:** Webpage  
**URL:** http://m.cnn.com/...  
**Last visit time**  
**Children:** ...

**Sub-resource nodes** (JavaScript, CCS, Image)

**Type:** Subresource  
**URL:** ...  
**Last visit time**  
**Parents:** ...  
**Number of visits**
• **Input:** URL of the webpage
• **Output:** predicted subresources
• **Input**: URL of the webpage
• **Output**: predicted subresources

• **Revisit**
• **Input:** URL of the webpage
• **Output:** predicted subresources

**Revisit**
• **Input:** URL of the webpage
• **Output:** predicted subresources

**Revisit**
• Input: URL of the webpage
• Output: predicted subresources

• **First-time visit**
• Input: URL of the webpage
• Output: predicted subresources

• **First-time visit**
• Input: URL of the webpage
• Output: predicted subresources

• **First-time visit**

![Diagram of node structures]
• Input: URL of the webpage
• Output: predicted subresources

• **First-time visit**

![Diagram showing the relationship between Website node, Subdomain nodes, Webpage nodes, and Sub-resource nodes (JavaScript, CCS, Image).]
• Input: URL of the webpage
• Output: predicted subresources

• **First-time visit**
Rank subresources

- Number of parents (large > small)
- File types (JS > CSS > image)
- Number of visits (large > small)
- URL length (short > long)
Evaluation of *Tempo*

– Subresource prediction

– Browser delay reduction

– Overhead
Subresource prediction

Requested

Predicted
Subresource prediction

Requested

Predicted

Requested & Predicted
High prediction accuracy

79%

High browser delay reduction

Requested & Predicted
High prediction accuracy

Requested

Predicted

70%

Low additional data usage

Requested & Predicted
Browser delay reduction

• Lab experiments
  – Webpage revisits
  – First-time visits

• Field trial
Webpage revisits

<table>
<thead>
<tr>
<th></th>
<th>Legacy</th>
<th>Tempo</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fresh Cache</td>
<td>4.4 s</td>
<td>4.3 s</td>
</tr>
<tr>
<td>Expired Cache</td>
<td>6 s</td>
<td>4.6 s</td>
</tr>
<tr>
<td>Empty Cache</td>
<td>6.7 s</td>
<td>5.2 s</td>
</tr>
</tbody>
</table>

Legend:
- Fresh Cache
- Expired Cache
- Empty Cache
Webpage revisits

### Legacy
- Fresh Cache: 4.4 s
- Expired Cache: 6 s
- Empty Cache: 6.7 s

### Tempo
- Fresh Cache: 4.3 s
- Expired Cache: 4.6 s
- Empty Cache: 5.2 s
Webpage revisits

For Legacy:
- Fresh Cache: 4.4 s
- Expired Cache: 6 s
- Empty Cache: 6.7 s

For Tempo:
- Fresh Cache: 4.3 s
- Expired Cache: 4.6 s
- Empty Cache: 5.2 s

24% decrease in load time with Tempo.
Webpage revisits

Legacy: 6 s, 6.7 s, 4.6 s, 6 s
Tempo: 4.4 s, 4.3 s, 5.2 s

Realistic Cache: 1 s (20%)
First-time visits

Legacy

- Fresh Cache: 4.5 s
- Expired Cache: 4.6 s
- Empty Cache: 6.9 s

Tempo

- Fresh Cache: 4.3 s
- Expired Cache: 3.6 s
- Empty Cache: 5.8 s

19% reduction
Field trial

• First week
  – Start with empty cache
  – Enable speculative loading

• Second week
  – Keep the warm cache
  – Disable speculative loading

• The benefit we attribute to speculative loading will be unlikely from the caching effect
Field trial

1st Week

With speculative loading: 3.9 s

2nd Week

4.3 s

Without speculative loading
Field trial

• Speculative loading needs some time to construct the resource graph...

• How about 5 days’ training?
Field trial

Day 6 & 7 of 1st Week

With speculative loading

2.9 s

2nd Week

Without speculative loading

4.3 s
Low overhead

• Prediction Error
  – Occupy TCP connection (still > 1s reduction)
  – Additional data usage (< 1MB per week)

• Storing metadata repository
  – Additional storage (< 200KB after a year)
Conclusion

• Client-only solutions can reduce mobile browser delay by 1.4 second at most

• **Tempo**, a speculative mobile browser, can reduce mobile browser delay by 1 second (≈20%)
  - Google: increase 40% traffic
  - Amazon: increase 10% sales
Conclusion

• Client-only solutions can reduce mobile browser delay by 1.4 second at most

• **Tempo**, a speculative mobile browser, can reduce mobile browser delay by 1 second (~20%)
  – Google: increase 40% traffic
  – Amazon: increase 10% sales

Web usage data available for download:
http://www.owlnet.rice.edu/~zw3/tempo.html