Browselite: A Private Data Saving Solution for the Web

Conor Kelton, Matteo Varvello, Andrius Aunicas, Ben Livshits
Stony Brook University, Nokia Bell Labs, Brave
How Expensive is Web Browsing?

Median Page ~ 2MB
Up to $0.25 on Limited Data plans!

Source: [1] Whatdoesmysitecost.com

~50% of this is images

Source: [2] HTTP Archive
Existing Solutions

1) Proxy Based Image Compression

2) URL Redirection + Server Side Pre-render

All fail in terms of user privacy, uncertain Web Compatibility!
Our Solution: Client Side

1) URL Instrumentation of server-side image infrastructure

2) Image Range Requests: Just Fetch Less!

https://example.com/gull.jpg?w=960
https://example.com/gull.jpg?w=376

https://example.com/gull.jpg?w=376
Range: bytes=0-9216
How is the User Affected?

URL Instrumentation:
Client knows image parameters, no visual loss!

https://example.com/gull.jpg?w=960

====>

https://example.com/gull.jpg?w=376

Blurry at native res...

Downsizing: not blurry on 411x731 Mobile!
~60% Savings!!
How is the User Affected?

1 - Baseline

2 - Reflection

![Graph showing visual completeness with and without reflection. The graph compares visual completion (%) against savings (%). Points 1 and 2 indicate distinct data points on the graph.](image-url)
How is the User Affected?

1 - Baseline  2 - Progressive

Diagram showing visual completeness for baseline and progressive methods.
Browselite: Saving Data for Web Media

1. Server responds with HTML

2. Puppeteer requests canvas

3. Regex Rules:
   - w=960 --> w=360
   - q=100 --> q=70
   - fmt=jpeg --> fmt=webp

4. Metadata + Image Request:
   - `example.com/img?q=70,w=360, fmt=webp; Range = 0-80000/160000`

5. Images resized and optimized
**Methodology and Parameters**

- **Dataset:** Crawl of 300 Landing, 800 Internal pages across 3 Page Rank Tiers (Top 100, 5k, 10k)

- **Range Request Parameter:** Pages ~90% Visually Complete with only 50% Image Data Requested

- **Comparison:** Non-private data saving method -- **Google Web Lite**
Results at Scale -- Data Savings

- **Browselite vs. MITM vs. Web Light**
  Median: 25% vs. 21% vs. 89%

- **MITM Proxy**: Compression/Resizing/Transcoding of all images

- **HTTP Proxy**: above, but no TLS (62% less availability)

- **Google Web Light**: upper bound, but act on page style, and compute, even remove contents (*webcompat!*)

![Graph showing CDF of savings across pages for different methods: MITM Proxy, HTTP Proxy, Web Light, Browselite. Median savings for each method are indicated with a horizontal line.](chart.png)
Results at Scale -- Performance

• Change in Speed Index:
  LTE 12Mbps/40ms RTT
  Wi-Fi 25 Mbps/20ms RTT

• 40% **speedup** by an average of 400ms

• 40% **slowdown** by <500ms, 20% **slowdown** by >500ms
If Range Requests uncachable can hurt performance on subsequent loads, or when switching off data restrictions!

**Experiments:**

1) 2 Range Requests for 0-20KB
2) After, requested 10-20KB
3) After 0-20KB, requested *full image without range*
4) Requested full image, then requested 0-20KB

**Caching Results:**

1) *Range requests are cacheable:*
   Second request pulled from cache
2) *Overlapping ranges are cacheable:*
   10-20 KB pulled from cache
3) *No data wasted by range request:*
   Browser transformed request to 20 KB - 600 KB
4) *Range requests cacheable if full image cached:*
   0-20 KB pulled from cache
Results -- User Studies

- Ratings generally acceptable, but room to improve...

<table>
<thead>
<tr>
<th>Method</th>
<th>Broken (1)</th>
<th>Poor (2)</th>
<th>Usable (3)</th>
<th>Good (4)</th>
<th>Very Good (5)</th>
</tr>
</thead>
<tbody>
<tr>
<td>BROWSELITE (Reflections)</td>
<td>0</td>
<td>8</td>
<td>21</td>
<td>8</td>
<td>3</td>
</tr>
<tr>
<td>BROWSELITE (Progressive)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>7</td>
</tr>
<tr>
<td>Web Light</td>
<td>0</td>
<td>2</td>
<td>11</td>
<td>21</td>
<td>7</td>
</tr>
</tbody>
</table>

- 200+ Crowdsourced User Study

- Future: Context Encoders / Saliency Prediction

CVPR ‘16
Conclusion + Questions

- **Browselite**: The Web Conference 2021
- Browser-controlled application implemented with Puppeteer
- 25% Data Saved at the median for 90% Visually complete pages
- Trade Offs: 80% imperceptible slowdowns, no effect on caching

- Next: DNN image interpolation
Results at Scale -- Performance (cont’d)

- 40% of Slowdown: Image Transformations
- 45% of Slowdown: Extra Range Requests
- 5% of Slowdown: DOM Search

Breakdown of performance decrease with fine grained timestamps

- Fix -- Tighter in-browser implementation
Web User Experience is Important!

**Speed!**
- 5.65 seconds
- 3.20 seconds

**Cost ($$$_{s}$$$_{s})!**
- 5.65 seconds → $100
- 3.20 seconds → $100

Studies show correlation between data savings and load time!