Automatic and Dynamic Configuration of Data Compression for Web Servers

Eyal Zohar

Yuval Cassuto

November 13, 2014

LISA’14
HTTP Compression

• Most web servers compress content for lower bandwidth and faster response

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Sites supporting “gzip” format</td>
<td>92%</td>
</tr>
<tr>
<td>Sites supporting “deflate” format</td>
<td>9%</td>
</tr>
<tr>
<td>Average download size (compressed only)</td>
<td>22,526</td>
</tr>
<tr>
<td>Average uncompressed file</td>
<td>101,786</td>
</tr>
<tr>
<td>Compression ratio (best-median-worst)</td>
<td>11%-25%-53%</td>
</tr>
</tbody>
</table>
Compression Knobs

1. Enable

2. Set compression level

1=fast

6=default

9=best
Which Level to Use?

- Top 500 sites – most pick the default (6)
Which Level **Should** Use?

- Higher level ➔ better compression
Compression Cost

- Higher level → lower capacity (server)
  → higher latency (user)
Content Matters

• Compression time depends on content
• All 500 sites, est. capacity (locally):
Performance vs. Effort

- **mSec/MB** – processing time to save 1MB

![Graph showing the relationship between compression ratio and time/save (mSec/MB). The graph illustrates that lower effort levels result in higher compression gain, while higher effort levels lead to lower gain. The data is based on 500 sites at level 6.]
Compression under Dynamic State

• Demand (popularity/time-of-day/attack)
• Hosting costs
• Hardware availability
• Content size
• Content compressibility
• Background processes (CPU)
• I/O performance
• ...
Solution – 1) Auto Adjust

• Elastic level – according to load/latency

*Yevgeni Sabin, Alexander Chigirintsev (“Project A” 2012)

**Code available here: http://eyalzo.com
Solution – 2) Mix

• Implementing non-integer levels
  – 1.2 means (80% * level 1) + (20% * level 2)
Experiment 1 – Serve More

- Serving x1.5 requests
Experiment 2 – DoS Attack

a) Requests

b) Levels

c) CPU

d) Latency
Thank YOU!