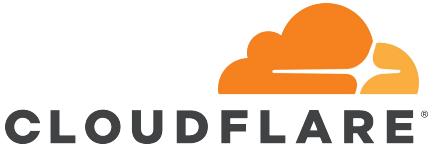


HTTP/2 Compression Dictionaries

Vlad Krasnov

In a nutshell

- Allow cross-stream compression in HTTP/2 by means of "dictionaries"
- Including a set(s?) of static dictionaries for initialization
  - Each dictionary targets a different MIME type
- Up to 256 dictionaries per connection
- Default dictionary size is 2^{17}
  - Defined by settings
- Server indicates if a stream might be used for compression in the future by sending a SET_DICTIONARY frame, before the data
  - Client keeps part of the data
- Server can use a previously defined dictionary with a USE_DICTIONARY frame

Rationale

- Yesterday
  - HTTP/1 with large assets
  - CPU time was expensive
  - Static assets compressed only with "gzip"
    - What about "brotli", "sdch", "sdch+gzip", "sdch+brotli"?
- Today
  - HTTP/2 with (ideally) smaller assets
  - CPU time significantly cheaper
    - Cloudflare uses gzip -8 for dynamic compression
    - Tomorrow: FPGAs
  - Store in gzip -> compress to other formats on demand
- Network
  - Gets cheaper, but slowly
  - Less data -> less packet loss

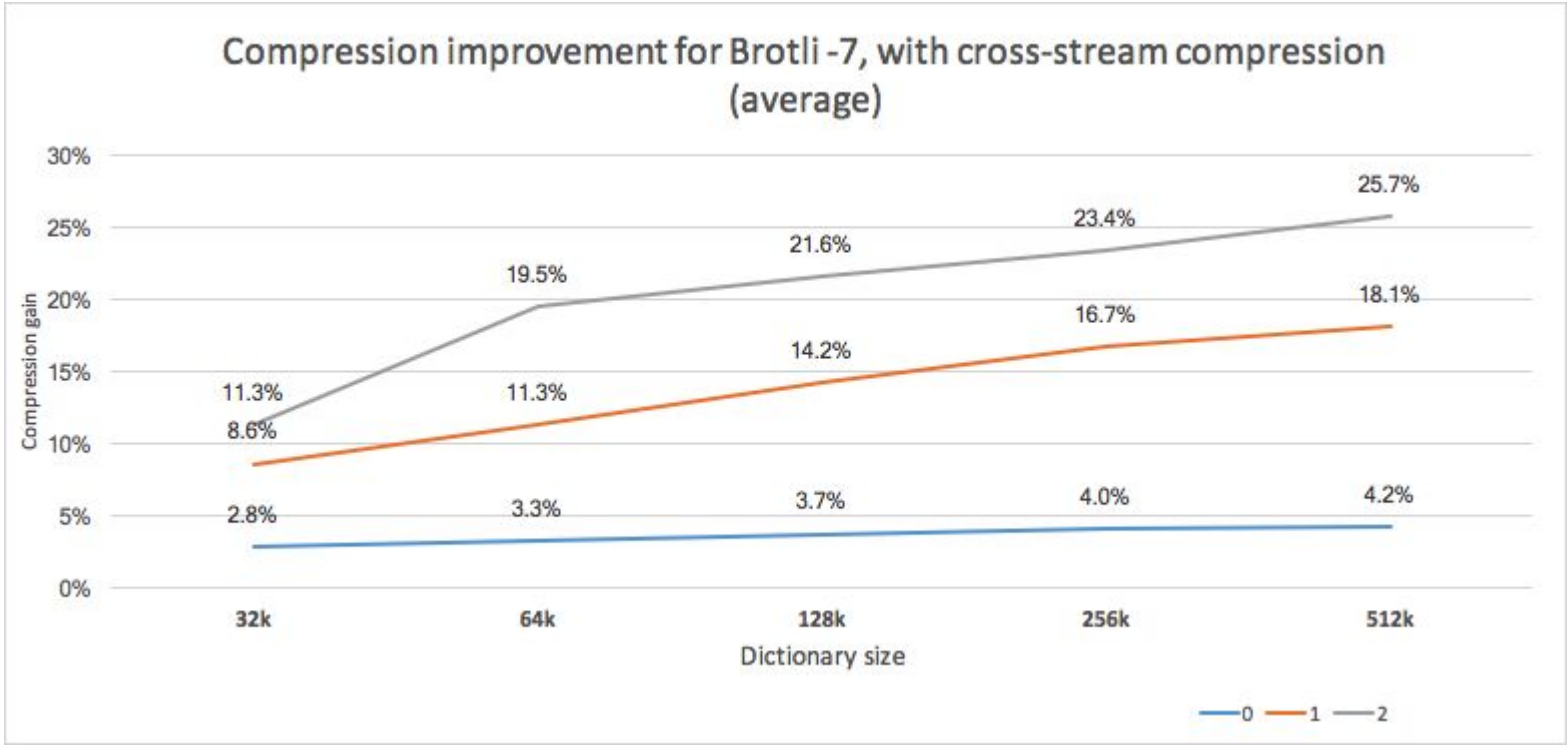
Benefits

- Client
  - Less bandwidth wasted
    - Reduced packet loss
    - Faster page loads
- Server
  - Less bandwidth wasted
  - Improved compression ratio almost for free
    - Alternatively: keep compression ratio, reduce CPU usage
  - Greater incentive to re-compress static content
- CDN
  - Highly efficient origin pulls
    - Almost free in many cases

Performance simulation

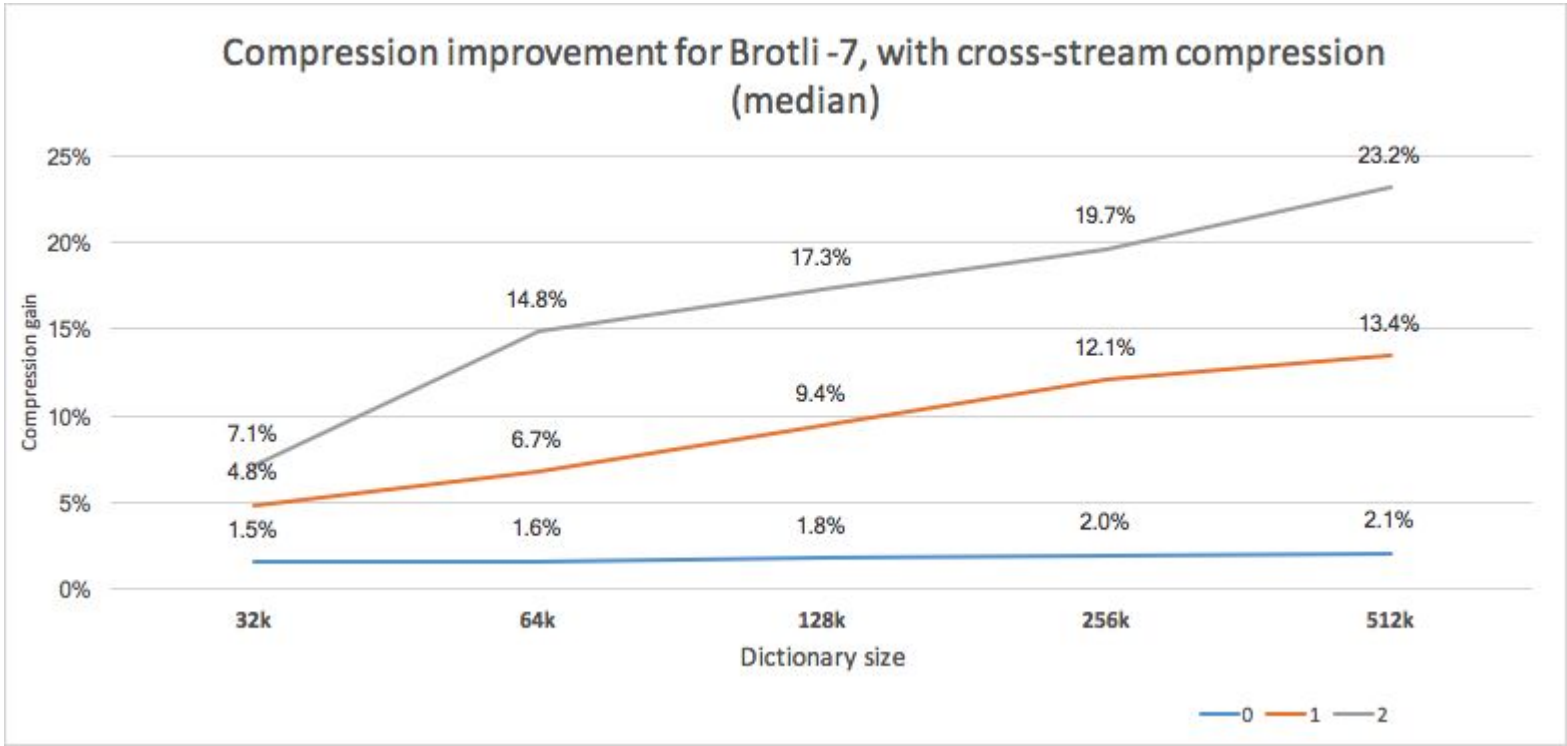
- Crawled over ~2000 Alexa top
- Used Chromedriver to load each page
- Simulated cross-stream compression with gzip and brotli
- Several compression strategies and dictionary sizes
- Best overall strategy:
  - If asset first of its type -> use static dictionary for type
  - Else use dynamic dictionary for type
  - Append asset to the dictionary for the type

Performance



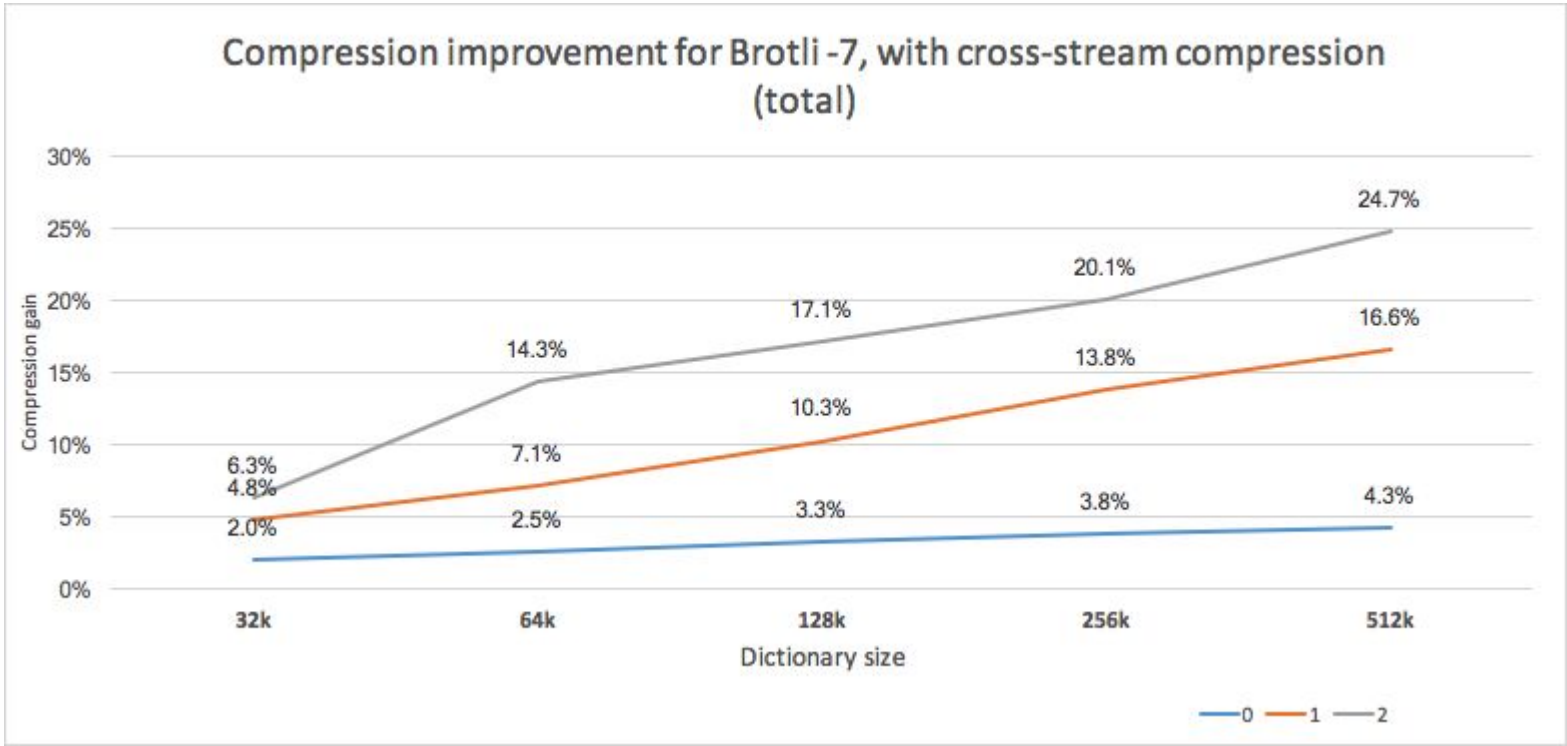
CLOUDFLARE

Performance



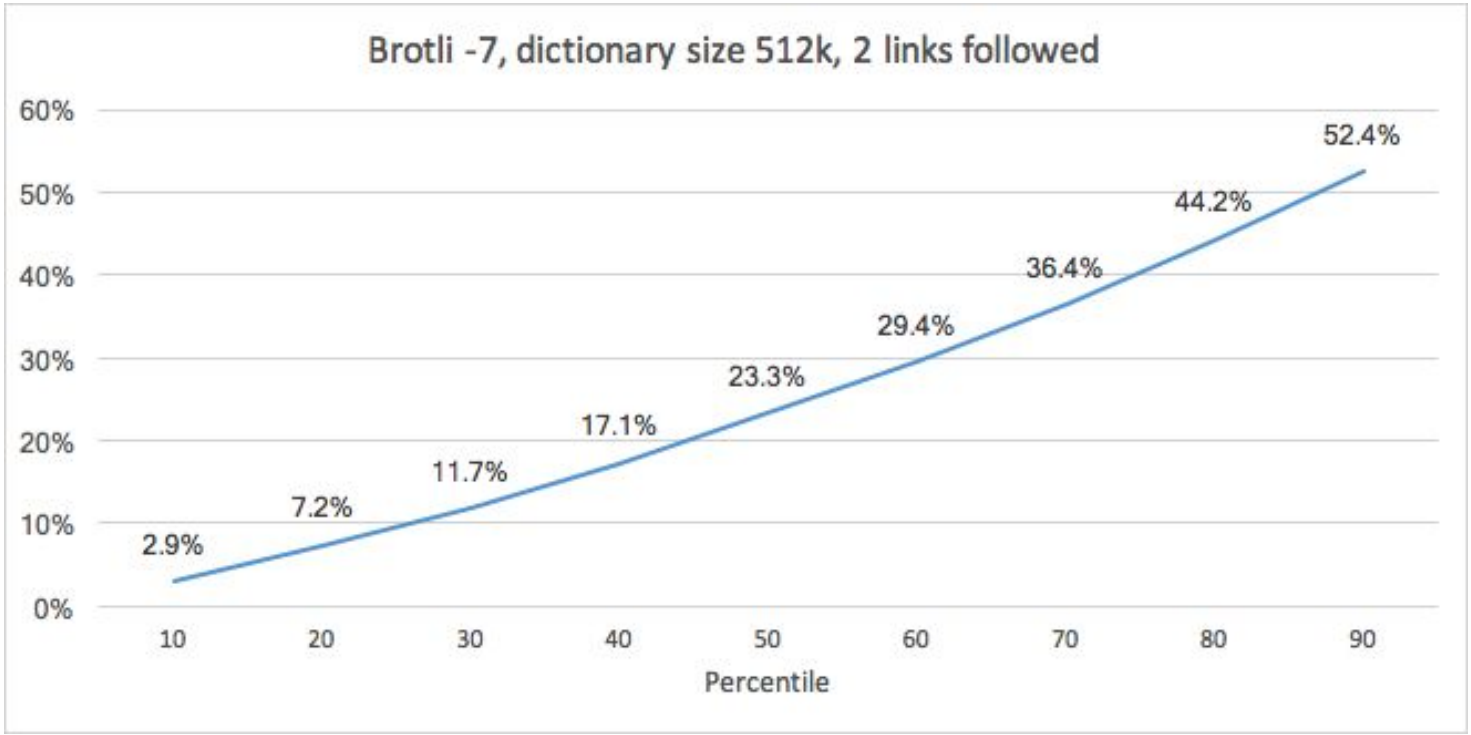
CLOUDFLARE

Performance



CLOUDFLARE

Performance



CLOUDFLARE

Performance

- Brotli -5 w. dict vs. Brotli -8 : 1.30X (2-3 times faster)
- Brotli -5 w. dict vs. Deflate -8: 1.46X (slightly slower)
- For reference: Deflate -8 vs Deflate -4: 1.04X (2 times slower)

Security

- Main concern: BREACH like attacks
  - Very likely that BREACH will be easier to execute
  - All BREACH mitigation techniques apply
    - Cross-referenced requests should not be compressed
    - Masking CSRF tokens etc.
  - In addition
    - Disabled by default, needs to be actively enabled (unlike HPACK)
    - Let client disable compression for a given stream (HEADER flag?)
      - Client has better knowledge who initiated a request
    - Intermediates are not allowed to use, unless instructed by origin
- Most websites are still non HTTPS
  - Improving HTTP/2 gives greater incentive to migrate
  - Most HTTPS websites aren't sensitive

How it compares to SDCH?

- This can definitely work with VCDIFF as well
- No need for a huge dictionary download
- Don't have to use the whole stream as dictionary
- Brotli has much higher compression ratio than SDCH+Brotli in cross-stream compression
- Appending streams, improves efficiency
- Fine-grained control in the protocol layer, improves efficiency, allows multiplexing
- Minimal overhead

CPU overhead

- No brainer for dynamic content
  - The overhead required to set a (large) dictionary, can be completely eliminated by slight modification of the compression API
- For static content: do you have to recompress the whole file?
  - You only need to recompress up to the sliding window size!