Effective Compression for the Web: Exploiting Document Linkages

Raymond Wan, Alistair Moffat

The Short Document Problem

- Adaptive sliding window compressors have poor initial performance.
- Once some input has been processed and the Huffman trie has data, they become reasonable.
- This initial ramp-up is costly, especially for short documents.

Example: Fiala-Green A1 “hownowbrowncow”

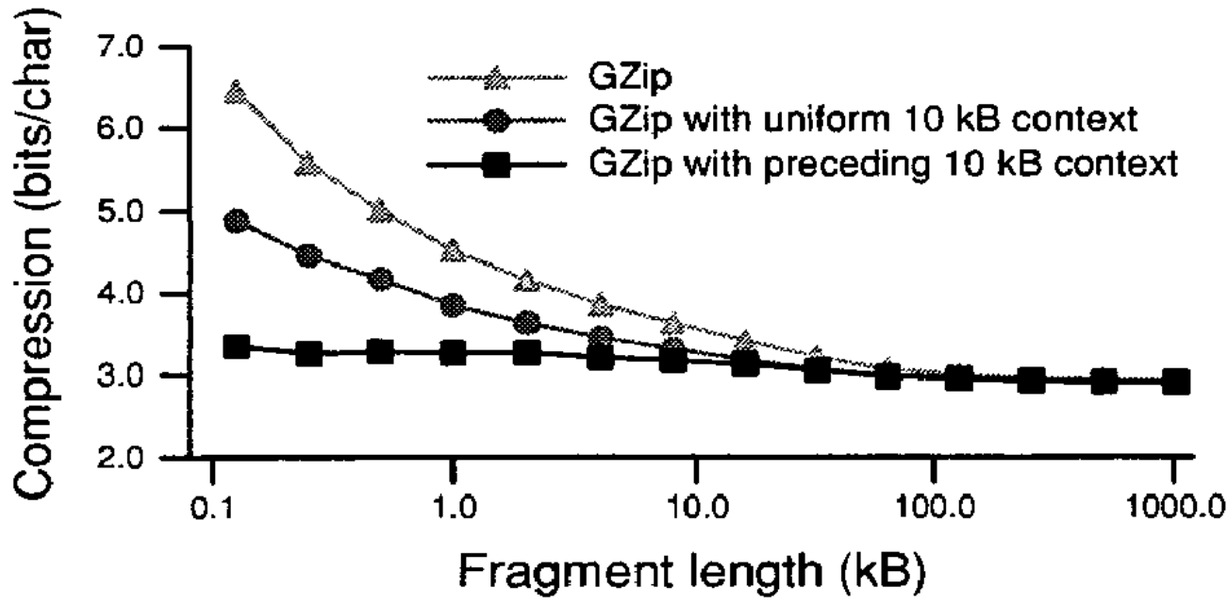
- Start in literal mode. We only exit literal mode if there is a length of match 3, so all of “hownowbr” is transmitted literally, for no savings.
- The next three characters, “own”, are a match, so go back into match mode and transmit pointer (length 3, displacement -7).
- When we read the next character, “c”, there is no match, so we must go back into literal mode, where we remain since there are no more length 3 matches.

Priming, A Solution

- Giving the compressor some "priming" text, prepending other input to the document to it, lets it build the window.

- gzip performance begins to stabilize at document sizes of around 10kB.

- Giving gzip 10kB of priming text lets it reach this level earlier (by making sure that every document is at least 10kB!)



Characteristics of Web Traffic

- Web server logs for a CS department's website over a 7-day period were analyzed.
- 443.1MB of HTML documents where transmitted to external users, with an average document size of 9.4kB.
- 70% of external page requests were part of a browser session in which more than one document was requested.

Exploiting Traffic Characteristics

- Independent: Just compress each page in isolation.
- Common: Priming text made from static analysis of all documents in site. Priming text is free.
- Transmitted: Common, plus cost to transmit priming text.
- Prior: Preceding page(s) in session used as priming text.
- Available: Common + Prior.
- Lossy: HTML normalization (comments, whitespace, case-folding.)

Method	Network bandwidth (% uncompressed)
NO-COMPRESSION	100.0
INDEPENDENT-GZIP	27.9
COMMON-PRIMING, 1 kB	27.1
COMMON-PRIMING, 10 kB	27.2
TRANSMITTED-PRIMING, 1 kB	31.5
TRANSMITTED-PRIMING, 10 kB	71.2
ONE-PRIOR	24.5
ALL-PRIOR	23.9
ALL-AVAILABLE, 10 kB	23.5
ALL-AVAILABLE, LOSSY, 10 kB	22.5
ALL-AVAILABLE, LOSSY, SPACES, 10 kB	22.1

State of Current Implementations

- INDEPENDENT-GZIP is in common use today.
  - Clients can say "Accept-Encoding: gzip". Support for this is widespread.
- Something similar to ALL-PRIOR could work with HTTP/1.1.
  - "application/http" media type can encapsulate multiple messages.
  - Pipelining allows a client to batch requests.
    - However, Firefox disables this by default because it "is not well-supported by some servers and proxies."
  - Google Web Accelerator does "prefetching" of linked documents.
  - A client could batch-request linked documents, and the server could send a single gzip-compressed application/http response.

Similar Issues

- The UNIX .tar.gz compression mechanism, where a group of files to be compressed are first combined into a single archive file, outperforms .zip, which compresses files one at a time.

- Nokia has a special compression format for SVG vector graphics. “CVG performs 2-4x better than general-purpose compression (such as GZIP) for small (less than 30K) SVG content.”

Method	Size
.tar.gz	934,226
.zip	1,319,629
.tar.zip	934,346

Size in bytes of my /etc directory compressed with various methods

Tradeoffs

- Compressed versions of pages can be cached to trade space for time.
  - Web server administrators typically don't, as most web servers aren't CPU-bound, and gzip over webpage-sized documents is extremely cheap.
- Compressing all possible versions of pages under the ALL-PRIOR scheme is probably prohibitively expensive.
- I don't know of anyone using bzip2 for web pages, it's too slow to be worthwhile.
- ONE-PRIOR and ALL-PRIOR are worthwhile, but are they worthwhile enough to be worth implementing and making sure that it is compatible with existing web standards and doesn't break legacy clients and servers? If no web servers implement it, is it worthwhile for web client authors?

Conclusions

- Adaptive sliding window methods benefit from having context available.
- This context is not available when compressing small documents.
- Multiple web pages in a session can be treated as a single larger document so that context is available.
- However, this is not usually worth the implementation complexity given modern bandwidth availability.