

Compaz: Exploring the Potentials of Shared Dictionary Compression on the Web

Benjamin Wollmer^{1,3}, Wolfram Wingerath^{2,3}, Sophie Ferrlein³,
Felix Gessert³, and Norbert Ritter¹

¹ University of Hamburg, Germany
`dbis-research@uni-hamburg.de`

² University of Oldenburg, Germany
`data-science@uni-oldenburg.de`

³ Baqend, Hamburg, Germany
`research@baqend.com`

Abstract. In this demonstration, we present Compaz, an extensible benchmarking tool for web compression that enables evaluating approaches before they have been fully implemented and deployed. Compaz makes this possible by collecting all relevant data from user journeys on live websites first and then performing the benchmark analysis as a subsequent step with global knowledge of all transmitted resources. In our demonstration scenario, the audience can witness how current websites could improve their compression ratio and save bandwidth. They can choose from standard and widespread approaches such as *Brotli* or *gzip* and advanced approaches like *shared dictionary compression* that are currently not even supported by any browser.

Keywords: Delta Encoding · Caching · Dictionary Compression.

1 Introduction & State of the Art

Compression is one of the critical components to tune the performance of websites. However, we have not seen much movement in browser support of different compression algorithms over the last decades [10]. Since then, *gzip* has been the most widely used format for text-based compression, with *Brotli* as the only noteworthy contender [4]. Nevertheless, advanced approaches like *delta encoding* [6] or *shared dictionary compression* [5,3,7] could achieve better compression ratios using cached data as dictionaries. Since some of these advanced approaches are extremely complex, assessing their true potential for practical use cases remains challenging (especially for approaches that have not been implemented, yet). This paper introduces Compaz, an extensible compression analyzer that can benchmark context-reliant compression approaches without deploying them to an actual website [9]. Similar to tools like WebPageTest, it evaluates an existing website, but with a focus on possible payload improvements instead of the current performance.

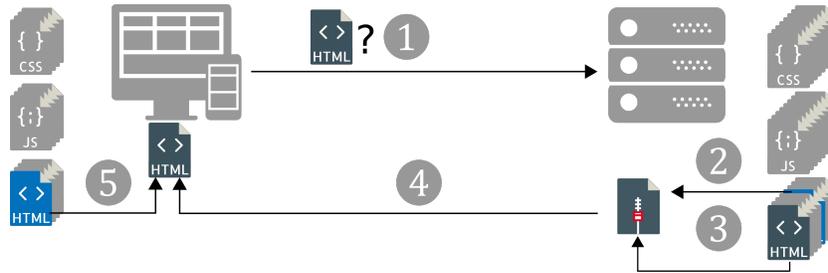


Fig. 1. Sharing the client’s context while requesting a file (1) and choosing the best dictionary (2) for compression (3) is a core challenge of shared dictionary approaches.

Motivating Example: Shared Dictionary Compression. We are particularly interested in the potential benefits of advanced shared dictionary compression as illustrated in Figure 1. A client request (1) shares its context, like the cache state, while requesting a file. The server could choose one of those files to be the dictionary (2) used for compressing the requested file (3) and send the compressed file to the client (4). The client uses its cached files (5) along with the compressed file to compute the initially requested file. This has some implications: First, the client needs to describe its cache state to the server. Since the cache can easily hold hundreds of files, sharing a list of every file in the cache is not feasible. Furthermore, the server would need to have every file of the client cache at hand, which can work for static content, but is especially difficult for dynamically rendered HTML. Moreover, the server would have to know which cache entry would be the best dictionary for this request. Still, this is a simplification of the problem, and for the best performance, this would need support for other advanced caching techniques [8] and CDNS. Since the compression result depends on the client cache, the results differ for each user, and therefore, the cache hit rate on the CDN level could drastically suffer from such an approach. This indicates that an implementation would be some approximation but not the optimal solution for data saving. Nevertheless, testing actual implementations is currently not easily done. While we had browser support for shared dictionary compression over HTTP (SDCH), it was removed due to a lack of traction [1], and as a result, browser support for custom dictionaries is currently nonexistent, and we have to fall back to a synthetic environment. Unlike static dictionary approaches like Brotli [2], user data is necessary to evaluate the impact of (dynamic) shared dictionary compression, since the compression result depends on the available dictionaries (e.g., previously visited HTML files) in the current client context and cannot be tested with a static set of files.

2 Compaz In a Nutshell

Generally, Compaz can be split into three parts: The collection of input data, the benchmarking of compression algorithms, and the data access.

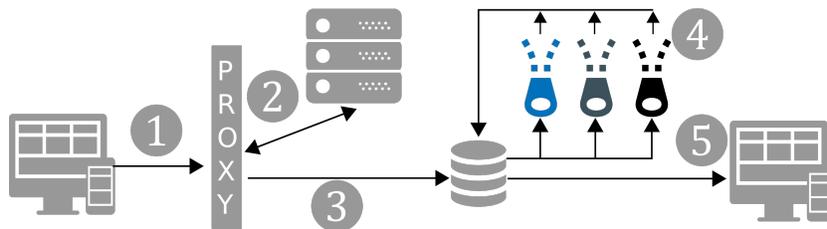


Fig. 2. Compaz utilizes a proxy to capture user journeys (1-2) and persists them in a database (3), so that the journey data can be used by different compressors (4) to compare the approaches (5).

Collecting Data. Compaz utilizes Selenium⁴ to navigate through user-defined journeys. Compaz currently supports three ways to define a journey. First, it can be defined as a static list of URLs, where Compaz will call the URLs one by one. Similarly, a list of regular expressions can be defined. These will be used to find links within the HTML so that Compaz will click them and proceed with the navigation. Compaz can also start a browser session that a user will manually handle as a third option. As Figure 2 shows, the browser connects through a proxy, from which we can copy each request and persist it in Compaz’s database. This makes sure that we see every request, but even more importantly, we only consider requests that were not served by the client’s cache. Additionally, we keep the chronological order and therefore can reconstruct which assets were available at which point in time for a shared dictionary approach.

Compressing Data. After a journey has been completed, it can be used by the compressors. For a shared dictionary approach, Compaz replays the requests and uses the decompressed assets to compress them with different approaches. Compaz provides the compressor instance with every asset for a shared dictionary approach, which could potentially be used as a dictionary to decompress (and therefore compress at the server side) the asset. It calculates every combination and keeps the best result. As mentioned in Section 1, this approach would not be feasible in practice but shows us the best potential of each compressor. The considered assets can be configured to, e.g., only consider those of the same MIME type or only those from a specific page in the journey. Furthermore, different compressors can be chained so that the output of one compressor is the input of another, which brings a significant improvement for some approaches. While we only persist the metrics for the compression results, we keep the original raw journey. This ensures that we can compare new approaches later against the same dataset, even if the approach did not exist while the journey was collected.

Comparing Data. The results are available via the API. We also created a frontend, which visualizes the results. Since the results are as low as on asset level, finding the best solution per asset is possible. This also shows which pages contain assets that could serve as dictionaries for shared dictionary approaches.

⁴ <https://www.selenium.dev/>

Prototyping New Compression Algorithms. Compaz was designed to be easily extendable, so new compression approaches can be benchmarked quickly without deploying them on live servers. As of now, Compaz has support for *gzip*, *open-vcdiff*, *Brotli*, and *zstandard*. Except for *gzip*, all other compressors can make use of a dictionary. To implement a new compressor, one must implement a configuration class with all necessary configuration parameters (e.g., compression level or window size) and the compressor itself. The compressor only needs two methods to be implemented: *compress* and *decompress*. While only the former is required for the calculation, the latter is to ensure integrity. Our unit test suite will automatically pick up each compressor without the need to write new tests. The logic to provide available dictionaries and chain different compressions is handled by Compaz and does not need any implementation considerations.

Similarly, new navigators can be implemented. We already provide an abstract class, which acts as a wrapper for Selenium. Each navigator can access the currently rendered DOM to decide the next action to be taken.

3 Conclusion

This paper has shown how Compaz can create repeatable compression evaluations using user journeys instead of arbitrary single files. Combined with Compaz’s extensibility, this ensures comparability of old results to new compression techniques. Compaz can be used to guide further research by evaluating a gold standard for currently not supported compression techniques, like delta encoding, before investing work in browser support or performance optimizations.

References

1. Intent to Unship: SDCH (2016), <https://groups.google.com/a/chromium.org/d/msg/blink-dev/nQ100RHy7sw/HNpR96sqAgAJ>, accessed: 2022-02-20
2. Alakuijala, J., Farruggia, A., Ferragina, P., Kliuchnikov, E., Obyrk, R., Szabadka, Z., Vandevenne, L.: Brotli: A general-purpose data compressor. *TOIS* **37**(1) (2018)
3. Chan, M.C., Woo, T.: Cache-Based Compaction: A New Technique for Optimizing Web transfer. In: *IEEE INFOCOM '99. Conference on Computer Communications*.
4. Fischbacher, T., Kliuchnikov, E., Comşa, I.: Web Almanac: Compression, <https://almanac.httparchive.org/en/2021/compression>, accessed: 2022-02-25
5. McQuade, B., Mixter, K., Lee, W.H., Butler, J.: A proposal for shared dictionary compression over http (2016)
6. Mogul, J.C., Douglis, F., Feldmann, A., Krishnamurthy, B.: Potential Benefits of Delta Encoding and Data Compression for HTTP. *SIGCOMM CCR* **27**(4) (1997)
7. Shapira, O.: Shared Dictionary Compression for HTTP at LinkedIn., <https://engineering.linkedin.com/shared-dictionary-compression-http-linkedin>
8. Wingerath, W., Gessert, F., Witt, E., Kuhlmann, H., Bücklers, F., Wollmer, B., Ritter, N.: Speed Kit: A Polyglot GDPR-Compliant Approach For Caching Personalized Content. In: *36th ICDE 2020, Dallas, Texas, April 20-24, 2020* (2020)
9. Wollmer, B., Wingerath, W., Ferrlein, S., Panse, F., Gessert, F., Ritter, N.: The Case for Cross-Entity Delta Encoding in Web Compression. In: *22th ICWE* (2022)
10. Wollmer, B., Wingerath, W., Ritter, N.: Context-Aware Encoding & Delivery in the Web. In: *20th ICWE 2020, Helsinki, Finland, June 9-12, 2020* (2020)