December 04, 2017

Deterministic patch file generation for Deflate streams

Amin Hassani

Ryan Hitchman

Sen Jiang

Deterministic patch file generation for Deflate streams

ABSTRACT

This disclosure describes efficient techniques to generate small and deterministic patch files for Deflate compressed files. A transform is utilized to perform decompression of the Huffman part of the Deflate stream of the file while the structure of the LZ77 coding in the file is left unchanged. A corresponding reverse transform is utilized to deterministically obtain the original Deflate stream from the generated stream. Implementation of the techniques requires low amounts of memory and computation time. The techniques are compatible with many compression techniques. The techniques can be used to deliver software updates over a network.

KEYWORDS

software update; Deflate stream; patch file; file compression; mobile apps; operating system

BACKGROUND

Updates to operating system and other software, e.g., to deliver bug fixes, additional functionality, etc. are common. Patches are commonly used to deliver updates. With the use of a patch file, the size of the update is smaller than retransmitting the entire software. For example, a patch file includes data corresponding to the difference from an earlier version of a file (source file) that is already installed or available on a device and the newer version that the file is to be updated to (target file). Use of a patch file enables updates to be delivered faster and reduces network requirements.

For example, consider that a patch file is to be generated to enable obtaining a target file (file_B) using a source file (file_A). Two functions - “diff” and “patch” - are defined as follows:

```
patch_file = diff(file_A, file_B)
```

```
file_B = patch(file_A, patch_file)
```

"diff" generates a patch file between the two files A and B. "patch" obtains file B given file A and the patch file. Depending on the type of the files, and the type of diff function used, the size of patch file can vary substantially. *bsdiff* and *bspatch* are commonly used for the "diff" and "patch" operations for binary files.

Deflate is a lossless data compression algorithm and data format that is widely used in many file formats such as gzip, zlib, png, pdf, zip, etc. The Deflate format is a bit-aligned compression format where units of data are composed of bits of information with variable length. The Deflate compression format utilizes LZ77 compression, followed by a Huffman coding. Huffman coding reduces the size of the compressed data stream by using a smaller number of bits to represent symbols that occur with higher frequency in the data stream, and vice versa.

Due to the bit-aligned nature of the Deflate format, patch files created by *bsdiff* can be large. To reduce the size of the patch, *bsdiff* can be performed on uncompressed data instead of compressed data, e.g.,

```
patch_file = bsdiff(decompress(file_A), decompress(file_B))
```

However, this requires the use of recompression in the client. To obtain the target file, the recipient (e.g., client device) needs to perform the following operation:

```
file_B = compress(bspatch(decompress(file_A), patch_file))
```

This approach suffers from flaws. Most compression techniques (including the ones used in Deflate) are nondeterministic. When a file is compressed using two different versions of a compression program, the resultant compressed streams can be different. Consequently, if the file is recompressed, the resulting file (e.g., file_B) can be different from the original file. Further, full recompression is slow and resource-intensive.

DESCRIPTION

This disclosure describes techniques to reduce the size of patch file produced by *bsdiff* (or any other algorithm that computes difference) using Deflate compressed streams while also allowing the compressed file to be deterministically recreated. The techniques can be used to generated a patch-file between a source file and a target file where one or both the files contain Deflate streams. The generated patch file can be used in a reverse operation to generate the target file from the source file. For example, a patch file can be created at a server that distributes software updates and transmitted for use by client devices.

By using a transform, the source files and target files are converted into a format such that *bsdiff* can produce smaller sized patch files. The transform performs a decompression of the Huffman part of the Deflate stream while the structure of the LZ77 coding is left unchanged. A reverse transform can then deterministically convert the generated stream generated back to the original Deflate stream using the Huffman tables. The dynamic Huffman tables can be recreated uniquely from a code length array stored inside the stream.

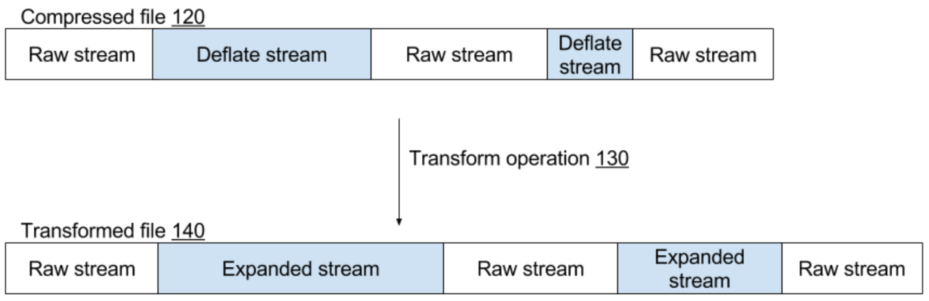

Fig. 1: The transform operation converts a compressed file to a transformed file

Fig. 1 illustrates the transformation of a compressed file (120) by applying a transform operation (130) to generate a transformed file (140). A compressed file can include multiple Deflate streams at different locations. When the transform operation is applied, the resultant stream contains the raw data that existed between the Deflate streams in the compressed file as well as the expanded stream that is derived from the Deflate stream. When the reverse transformation is applied, it uses stored information that is provided regarding the locations of sections of the expanded stream and the raw stream.

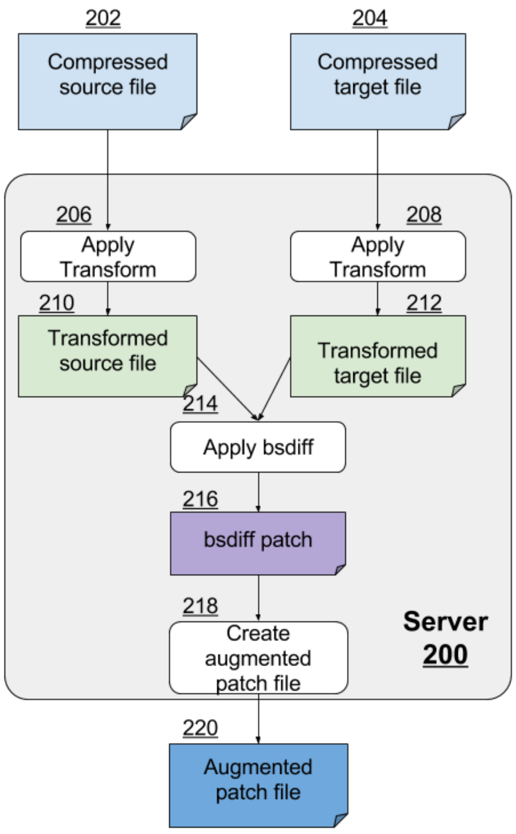

Fig. 2a: Server side operations for generation of an augmented patch file

Fig. 2a illustrates an example of use of the techniques of this disclosure by a server (200) to generate an augmented patch file. A compressed source file (202) and a compressed target file (204) are provided as inputs. A transform, as described earlier, is applied to each of the files (206, 208) to obtain the respective transformed source file (210) and transformed target file (212).

bsdiff is applied (214) to the transformed source file and transformed target file to obtain a bsdiff patch file (216). The bsdiff patch file (216) is augmented (218) with location information of sections of the expanded stream and the raw stream to obtain an augmented patch-file (220). The augmented patch file is provided to one or more client devices.

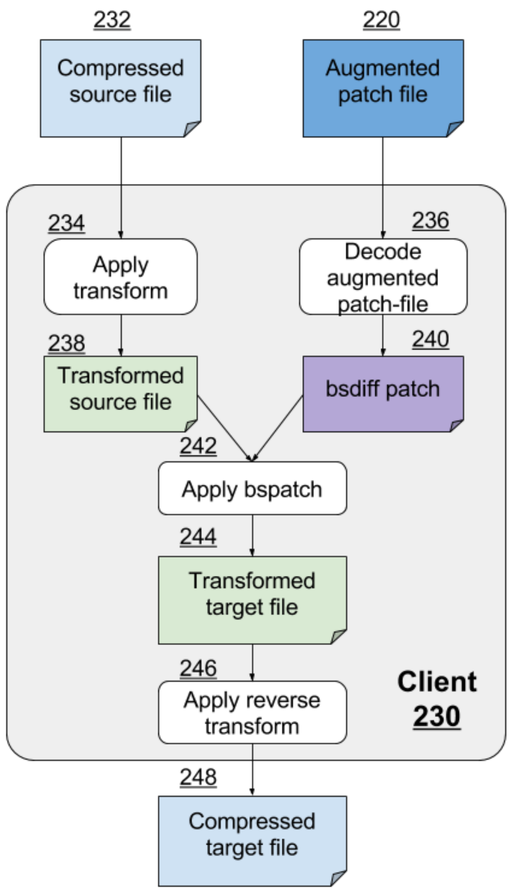

Fig. 2b: Client side operations for processing an augmented patch file

Fig. 2b illustrates application of a reverse operation at a client (230), to recover the compressed target file. The client copy of the compressed source file (232) and the augmented patch-file (220) received from the server are provided as inputs. A transform is applied (206) to the compressed source file (232) to obtain a transformed source file (238). The augmented patch-file is decoded (236) using the location data of the raw and Deflate streams to obtain a bsdiff patch (240). *bspatch* is applied (242) using the bsdiff patch (240) to obtain a transformed target file (244). A reverse transform is applied (246) to the transformed target file (244) to obtain the compressed target file (248).

In different implementations, the structure of the stream can be changed as needed. For example, different literals and copy length/ distance can be used. The format of the Huffman table in the stream can also modified.

Implementation of the techniques requires low amounts of memory and computation time. The techniques are compatible with different compression algorithms. The patch files produced by the described techniques are of small size, suitable for delivery of updates. The techniques are fast and deterministic. Any update system that updates different versions of files that are compressed in Deflate format can benefit from the described techniques. Compression formats similar to Deflate that combine a slow LZ77 step and a fast coding step can be transformed similarly. For example, LZMA compression is LZ77 followed by arithmetic coding, which can be transformed similar to Huffman coding.

The techniques can be used to deliver operating system and application software updates for computing devices, e.g., smartphones, tablet, wearable devices, laptops, etc. For example, the updates can be delivered by a software vendor, or an online store that provides software for such devices.

CONCLUSION

This disclosure describes efficient techniques to generate small and deterministic patch files for Deflate compressed files. A transform is utilized to perform decompression of the Huffman part of the Deflate stream of the file while the structure of the LZ77 coding in the file is left unchanged. A corresponding reverse transform is utilized to deterministically obtain the original Deflate stream from the generated stream. Implementation of the techniques requires low amounts of memory and computation time. The techniques are compatible with many compression techniques.