How are zlib, gzip and zip related? What do they have in common and how are they different?

Asked 10 years, 11 months ago Modified 1 year ago Viewed 495k times

1290

The compression algorithm used in *zlib* is essentially the same as that in *gzip* and *zip*. What are *gzip* and *zip*? How are they different and how are they same?

compression zip gzip zlib

Share

Improve this question

Follow

edited Sep 10, 2016 at 3:36

Francisco Corrales Morales

3,814 ● 1 ● 40 ● 64

asked Dec 24, 2013 at 13:48

Abhishek Jain

10.5k ● 5 ● 28 ● 40

3 Answers

Sorted by: Highest score (default)

3251

Short form:

`.zip` is an archive format using, usually, the Deflate compression method. The `.gz` gzip format is for single

files, also using the Deflate compression method. Often gzip is used in combination with tar to make a compressed archive format, `.tar.gz`. The zlib library provides Deflate compression and decompression code for use by zip, gzip, png (which uses the zlib wrapper on deflate data), and many other applications.

Long form:

The ZIP format was developed by Phil Katz as an open format with an open specification, where his implementation, PKZIP, was shareware. It is an archive format that stores files and their directory structure, where each file is individually compressed. The file type is `.zip`. The files, as well as the directory structure, can optionally be encrypted.

The ZIP format supports several compression methods:

```
    0 - The file is stored (no compression)
    1 - The file is Shrunk
    2 - The file is Reduced with compression
factor 1
    3 - The file is Reduced with compression
factor 2
    4 - The file is Reduced with compression
factor 3
    5 - The file is Reduced with compression
factor 4
    6 - The file is Imploded
    7 - Reserved for Tokenizing compression
algorithm
    8 - The file is Deflated
    9 - Enhanced Deflating using Deflate64(tm)
   10 - PKWARE Data Compression Library Imploding
(old IBM TERSE)
   11 - Reserved by PKWARE
```

```
    12 - File is compressed using BZIP2 algorithm
    13 - Reserved by PKWARE
    14 - LZMA
    15 - Reserved by PKWARE
    16 - IBM z/OS CMPSC Compression
    17 - Reserved by PKWARE
    18 - File is compressed using IBM TERSE (new)
    19 - IBM LZ77 z Architecture
    20 - deprecated (use method 93 for zstd)
    93 - Zstandard (zstd) Compression
    94 - MP3 Compression
    95 - XZ Compression
    96 - JPEG variant
    97 - WavPack compressed data
    98 - PPMd version I, Rev 1
    99 - AE-x encryption marker (see APPENDIX E)
```

Methods 1 to 7 are historical and are not in use. Methods 9 through 98 are relatively recent additions and are in varying, small amounts of use. The only method in truly widespread use in the ZIP format is method 8, Deflate, and to some smaller extent method 0, which is no compression at all. Virtually every `.zip` file that you will come across in the wild will use exclusively methods 8 and 0, likely just method 8. (Method 8 also has a means to effectively store the data with no compression and relatively little expansion, and Method 0 cannot be streamed whereas Method 8 can be.)

The ISO/IEC 21320-1:2015 standard for file containers is a restricted zip format, such as used in Java archive files (.jar), Office Open XML files (Microsoft Office .docx, .xlsx, .pptx), Office Document Format files (.odt, .ods, .odp), and EPUB files (.epub). That standard limits the

compression methods to 0 and 8, as well as other constraints such as no encryption or signatures.

Around 1990, the Info-ZIP group wrote portable, free, open-source implementations of `zip` and `unzip` utilities, supporting compression with the Deflate format, and decompression of that and the earlier formats. This greatly expanded the use of the `.zip` format.

In the early '90s, the gzip format was developed as a replacement for the Unix `compress` utility, derived from the Deflate code in the Info-ZIP utilities. Unix `compress` was designed to compress a single file or stream, appending a `.Z` to the file name. `compress` uses the LZW compression algorithm, which at the time was under patent and its free use was in dispute by the patent holders. Though some specific implementations of Deflate were patented by Phil Katz, the format was not, and so it was possible to write a Deflate implementation that did not infringe on any patents. That implementation has not been so challenged in the last 20+ years. The Unix `gzip` utility was intended as a drop-in replacement for `compress`, and in fact is able to decompress `compress`-compressed data (assuming that you were able to parse that sentence). `gzip` appends a `.gz` to the file name. `gzip` uses the Deflate compressed data format, which compresses quite a bit better than Unix `compress`, has very fast decompression, and adds a CRC-32 as an integrity check for the data. The header format also permits the storage of more information than

the `compress` format allowed, such as the original file name and the file modification time.

Though `compress` only compresses a single file, it was common to use the `tar` utility to create an archive of files, their attributes, and their directory structure into a single `.tar` file, and then compress it with `compress` to make a `.tar.Z` file. In fact, the `tar` utility had and still has the option to do the compression at the same time, instead of having to pipe the output of `tar` to `compress`. This all carried forward to the gzip format, and `tar` has an option to compress directly to the `.tar.gz` format. The `tar.gz` format compresses better than the `.zip` approach, since the compression of a `.tar` can take advantage of redundancy across files, especially many small files. `.tar.gz` is the most common archive format in use on Unix due to its very high portability, but there are more effective compression methods in use as well, so you will often see `.tar.bz2` and `.tar.xz` archives.

Unlike `.tar`, `.zip` has a central directory at the end, which provides a list of the contents. That and the separate compression provides random access to the individual entries in a `.zip` file. A `.tar` file would have to be decompressed and scanned from start to end in order to build a directory, which is how a `.tar` file is listed.

Shortly after the introduction of gzip, around the mid-1990s, the same patent dispute called into question the free use of the `.gif` image format, very widely used on bulletin boards and the World Wide Web (a new thing at

the time). So a small group created the PNG losslessly compressed image format, with file type `.png`, to replace `.gif`. That format also uses the Deflate format for compression, which is applied after filters on the image data expose more of the redundancy. In order to promote widespread usage of the PNG format, two free code libraries were created. libpng and zlib. libpng handled all of the features of the PNG format, and zlib provided the compression and decompression code for use by libpng, as well as for other applications. zlib was adapted from the `gzip` code.

All of the mentioned patents have since expired.

The zlib library supports Deflate compression and decompression, and three kinds of wrapping around the deflate streams. Those are no wrapping at all ("raw" deflate), zlib wrapping, which is used in the PNG format data blocks, and gzip wrapping, to provide gzip routines for the programmer. The main difference between zlib and gzip wrapping is that the zlib wrapping is more compact, six bytes vs. a minimum of 18 bytes for gzip, and the integrity check, Adler-32, runs faster than the CRC-32 that gzip uses. Raw deflate is used by programs that read and write the `.zip` format, which is another format that wraps around deflate compressed data.

zlib is now in wide use for data transmission and storage. For example, most HTTP transactions by servers and browsers compress and decompress the data using zlib, specifically HTTP header `Content-Encoding: deflate`

means deflate compression method wrapped inside the zlib data format.

Different implementations of deflate can result in different compressed output for the same input data, as evidenced by the existence of selectable compression levels that allow trading off compression effectiveness for CPU time. zlib and PKZIP are not the only implementations of deflate compression and decompression. Both the 7-Zip archiving utility and Google's zopfli library have the ability to use much more CPU time than zlib in order to squeeze out the last few bits possible when using the deflate format, reducing compressed sizes by a few percent as compared to zlib's highest compression level. The pigz utility, a parallel implementation of gzip, includes the option to use zlib (compression levels 1-9) or zopfli (compression level 11), and somewhat mitigates the time impact of using zopfli by splitting the compression of large files over multiple processors and cores.

Share Improve this answer Follow

edited Nov 27, 2023 at 21:06

answered Dec 24, 2013 at 18:03

Mark Adler

112k ● 15 ● 129 ● 173

190 This post is packed with so much history and information that I feel like some citations need be added incase people try to reference this post as an information source. Though if this information is reflected somewhere with citations like Wikipedia, a link to such similar cited work

would be appreciated. – ThorSummoner Oct 16, 2015 at 16:29

1866 I am the reference, having been part of all of that. This post could be cited in Wikipedia as an original source. – Mark Adler Oct 16, 2015 at 16:38

700 FYI: Mark Adler is an American software engineer, and has been heavily involved in space exploration. He is best known for his work in the field of data compression as the author of the Adler-32 checksum function, and a co-author of the zlib compression library and gzip. He has contributed to Info-ZIP, and has participated in developing the Portable Network Graphics (PNG) image format. Adler was also the Spirit Cruise Mission Manager for the Mars Exploration Rover mission. (wikipedia) – Isaac Hanson Dec 9, 2015 at 17:23

136 gzip was created to replace Unix compress. zip is *not* superior to tar + gzip on Unix, for several reasons. (When you see `.tar.gz` files, that's what they are.) First, tar + gzip compresses better than zip, since the compression of the next file can use history from the previous file (sometimes referred to as a "solid" archive). zip can only compress files individually. Second, tar preserves all of the Unix directory information, whereas zip was not designed to do that. (Later extensions to the zip format with Unix-specific extra blocks tries to remedy this problem.) – Mark Adler Feb 21, 2016 at 17:19

120 You seem to be confusing formats with implementation. The 7-Zip *implementation* of the deflate format can get something like your quoted 2% to 10% better compression than gzip with the very same deflate format (while taking much more CPU time to do so). The 7z LZMA2 *format* offers on the order of 40% better compression. – Mark Adler Mar 17, 2016 at 14:34

65

ZIP is a file format used for storing an arbitrary number of files and folders together with lossless compression. It makes no strict assumptions about the compression methods used, but is most frequently used with DEFLATE.

Gzip is both a compression algorithm based on DEFLATE but less encumbered with potential patents et al, and a file format for storing a single compressed file. It supports compressing an arbitrary number of files and folders when combined with tar. The resulting file has an extension of `.tgz` or `.tar.gz` and is commonly called a *tarball*.

zlib is a library of functions encapsulating DEFLATE in its most common LZ77 incarnation.

Share Improve this answer

Follow

answered Dec 24, 2013 at 13:55

Niels Keurentjes

41.9k ● 9 ● 100 ● 137

42

The most important difference is that gzip is only capable to compress a single file while zip compresses multiple files one by one and archives them into one single file afterwards. Thus, gzip comes along with tar most of the time (there are other possibilities, though). This comes along with some (dis)advantages.

If you have a big archive and you only need one single file out of it, you have to decompress the whole gzip file

to get to that file. This is not required if you have a zip file.

On the other hand, if you compress 10 similiar or even identical files, the zip archive will be much bigger because each file is compressed individually, whereas in gzip in combination with tar a single file is compressed which is much more effective if the files are similiar (equal).

Share Improve this answer

Follow

edited Sep 27, 2018 at 13:19

Konrad Rudolph

545k ● 139 ● 956 ● 1.2k

answered Dec 24, 2013 at 14:03

Tim Zimmermann

6,400 ● 3 ● 31 ● 36

12 You are overstating the point. If people wanted random-access compressed archives, they could create ".gz.tar" files instead of ".tar.gz" files. They don't, because most people aren't that interested in random access. There is a big community around the .warc.gz web archiving format, and they need random access, so they compress each web page separately. You use this format every time you look at a webpage in the Internet Archive Wayback Machine.
– Greg Lindahl Jun 18, 2016 at 14:57

12 ".gz.tar" does not offer random-access as the tar format is not capable to randomly access its entries. You need to go through all the entries from the beginning to get the one specific entry wanted, even worse: you need to go through all the entries until the end because the same file may be archived several times (in several versions) on several places in the same archive - and there is no means to figure

it out except to read the whole archive entry by entry.
– Min-Soo Pipefeet Jul 23, 2018 at 19:48

1 @Min-SooPipefeet right, because `tar` means "**tape** archive", and (unless you're playing with something like DECtape) tapes are nothing if not sequential! – RonJohn Aug 7, 2022 at 20:07