When I tell Windows to compress a file, the compression is far worse than I get if I ask WinZip to compress the file; why is that?

devblogs.microsoft.com/oldnewthing/20160718-00

July 18, 2016

Raymond Chen

A customer noted that when they took a very large (multiple gigabyte) file and went to the file's Properties and set "Compress contents to save disk space", the file shrunk by 25%. And then they needed to copy the file to a USB stick, so they used an old copy of WinZip to compress the file, and the result was half the size of the original.

Why is it that an old 10-year-old program can compress files so much better than Windows 2012's built-in disk compression? Is the NTFS compression team a bunch of lazy good-for-nothings?

Transparent file compression such as that used by NTFS has very different requirements from archival file compression such as that used by WinZip.

Programs like WinZip are not under time constraints; they can take a very long time to analyze the data in order to produce high compression ratios. Furthermore, the only operations typically offered by these programs are "Compress this entire file" and "Uncompress this entire file". If you want to read the last byte of a file, you have to uncompress the whole thing and throw away all but the last byte. If you want to update a byte in the middle of the file, you have to uncompress it, update the byte, then recompress the whole thing.

Transparent file compression, on the other hand, is under real-time pressure. Programs expect to be able to seek to a random position in a file and read a byte; they also expect to be able to seek to a random position in a file and write a byte, leaving the other bytes of the file unchanged. And these operations need to be $O(1)$, or close to it.

In practice, what this means is that the original file is broken up into chunks, and each chunk is compressed independently by an algorithm that strikes a balance between speed and compression. Compressing each chunk independently means that you can uncompress an arbitrary chunk of a file without having to uncompress any chunks that it is dependent upon.

However, since the chunks are independent, they cannot take advantage of redundancy that is present in another chunk. (For example, if two chunks are identical, they still need to be compressed separately; the second chunk cannot say “I’m a copy of that chunk over there.”)

All this means that transparent file compression must sacrifice compression for speed. That’s why its compression looks lousy when compared to an archival compression program, which is under no speed constraints.

Raymond Chen

Follow