What is WofCompressedData? Does WOF mean that Windows is a dog?

devblogs.microsoft.com/oldnewthing/20190618-00

June 18, 2019

Raymond Chen

A customer doing performance analysis of their program discovered that there were reads from an alternate data stream called WofCompressedData. On the Internet, if you search for "WofCompressedData", you mostly see people wondering what it is. Some people suspect that it's malware, and others suspect (or even state confidently) that it's an artifact of anti-malware software and can be deleted.

What is WofCompressedData?

The documentation for `wofapi.h` says merely "This header is used by Data Access and Storage." For more information, it refers you to another web page that contains no additional information.

WOF stands for Windows Overlay Filter, which is a nice name that doesn't really tell you much about what it does or what it's for.

First, let's look at how Windows was installed before the introduction of the Windows Overlay Filter.

The Windows installation begins with a `install.wim` file that contains basically all of Windows. A WIM file is a container file, similar in spirit to other container files, like ZIP and Cabinet. Traditionally, the WIM file is copied to the recovery partition for use during emergencies, such as push-button reset. The contents of the WIM file are then uncompressed, and corresponding files are created on your boot volume, and it is these uncompressed files that are used when you run Windows. The WIM file sits in your recovery partition, ignored, but waiting for its opportunity to spring into action should the need arise.

This traditional layout means that every Windows system file is present twice: A compressed copy is in the WIM file on the recovery partition, and an uncompressed copy in the live Windows installation.

Windows 8.1 introduced a feature known as Windows Image File Boot (WIMBoot): A system manufacturer can set up a system so that the recovery partition contains the `install.wim` file as well as a `custom.wim` file which contains the OEM customizations, such as drivers for any special hardware. But instead of uncompressing the files and putting them into the live Windows installation, WIMBoot creates tiny little stub files in the live Windows installation that say, "Hey, um, I'm just a stub. If you want to see the contents, you want to uncompress *those bytes over there*." WIMBoot therefore avoids the duplication by allowing the live Windows installation to share the disk storage with the WIM file on the recovery partition.

Furthermore, since the file contents in the WIM are compressed, this reduces disk I/O, though naturally at a cost of higher CPU usage in order to perform the decompression.

The way this magic works is that the live Windows files are formally sparse NTFS files, so that when you ask for the file size, you get the correct number, even though there is no actual data in them. When you open the file, the Windows Overlay Filter steps in and generates the data by decompressing the data in the WIM file on demand.

Unlike native NTFS file compression, the Windows Overlay Filter supports only read operations. This means that it doesn't need to sector-align each compressed chunk,[1] so the compressed data can be packed more tightly together. If you open the file for writing,[2] the Windows Overlay Filter just decompresses the entire file, turning it back into a plain file.[3] At the time WIMBoot was released, there was also a guidance document warning you not to run around opening files for writing unnecessarily. Not opening files for writing unnecessarily is good advice in general, but it's particular important for WIMBoot in order to prevent unnecessary conversion.

The Windows Overlay Filter can take advantage of newer compression algorithms developed over the past 20 years, algorithms which produce better compression ratios, can be run in parallel on multiple cores, and which require less CPU and memory for decompression. It can also use algorithms tailored to the scenario: For example, it can choose algorithms where compression is expensive but decompression is cheap.

Changing the native NTFS file compression would be a disk format breaking change, which is not something taken lightly. Doing it as a filter provides much more flexibility. The downside is that if you mount the volume on a system that doesn't support the Windows Overlay Filter, all you see is an empty file. Fortunately, WOF is used only for system-installed files, and if you are mounting the volume onto another system, it's probably for data recovery purposes, so you're interested in user data, not system files.

It's called the "Windows Overlay Filter" because it "overlays" virtual files into a directory that also contains normal physical files.

When you read through the above description, you may have realized something: Whenever Windows Update updates a file, that file is converted from a virtual file to a plain uncompressed physical file because the file's backing data is no longer in the WIM file. This means that over time, the Windows system files occupy more and more disk space as more of them no longer match the copy in the WIM and revert from their compressed form to their uncompressed form.

Windows 10 introduced a feature known as Compact OS, which takes a different approach. With Compact OS, the Windows Overlay Filter gains the ability to recompress files: Based on a hardware performance check, the system may decide to take the updated files, recompress them, store the compressed data in the WofCompressedData alternate data stream, and free the original uncompressed data using the same "sparse file" trick to make the file appear as if it were a normal file.

If you open one of these recompressed files, the file is decompressed on the fly based on data in the WofCompressedData alternate data stream. And as before, if you open one of these files for writing, then the file reverts to its uncompressed form.

Bonus chatter: You can use the `WofShouldCompressBinaries` function to determine whether the system is using WOF to compress system files. From the command line, you can use the `compact.exe` program to inspect the compression state of a file, or of the system.

Oh, and going back to the customer's original question: the system's choice to use Windows Overlay Filter compression spends a small amount of parallel computation in order to save a small amount of I/O. It's theoretically possible that you stumbled across a hardware configuration where the system's automatic evaluation suggested using the Windows Overlay Filter even though it was a net performance loss. I guess that would happen if you had a really fast storage device attached to a low-end CPU, and it somehow managed to trick the the automatic evaluation into thinking that compression was a good idea. In practice, it is rather unusual to have a hardware configuration consisting of fast storage and a slow CPU.

Many thanks to Malcolm Smith for his assistance with this article.

[1] The sector alignment was necessary to permit data to be rewritten into the middle of the file. But since the Windows Overlay Filter doesn't support writing, it doesn't need to enforce sector alignments.

[2] Since these files are Windows system files, opening them for writing requires administrator access. Normal usage therefore would not trigger a full decompression.

[3] This "decompress on write" behavior merely describes the current behavior and is not contractual.

Raymond Chen

Follow