Survey of Windows update formats: The Quality update, which obsoletes all the others

devblogs.microsoft.com/oldnewthing/20200213-00

February 13, 2020

Raymond Chen

We've spent the past few days learning about Full updates, Delta updates, and Express updates. All that was just background information! Finally we can talk about the thing I actually wanted to talk about: The Quality update, which obsoletes them all.

The **Quality update** takes a different approach to patching: Not only does it patch a file forward to the latest version, it also patches the latest version of the file *backward* to the original version.

Update	Full file	Patch base					Reverse patch
		M0	M1	M2	M3	M4	
M1	M1	M0 to M1					M1 to M0
M2	M2	M0 to M2	M1 to M2				M2 to M0
M3							
M4	M4	M0 to M4	M1 to M4	M2 to M4			M4 to M0
M5	M5	M0 to M5	M1 to M5	M2 to M5		M4 to M5	M5 to M0

The Quality update includes only the two sets of patches, one to get from the initial version to the latest, and one to get from the latest version back to the original.

Quality update	Contents
M1	M0 to M1, M1 to M0
M2	M0 to M2, M2 to M0
M3	M0 to M2, M2 to M0

M4	M0 to M4, M4 to M0
M5	M0 to M5, M5 to M0

Note that the M3 Quality update is the same as the M2 Quality update since the file `F` did not change between M2 and M3.

The secret to the Quality update is that the client retains the patches necessary to bring its files back to the M0 version. At the release of M0, this is vacuous: The files are already at their M0 version, so no patches are needed. We'll see how this invariant is maintained at each subsequent update.

Applying a Quality update consists of downloading the update, and then for each file in the update, applying *two* sets of patches: First patch the current file *backward* to the original M0 version using the patch cached on the client. Second, patch the M0 file *forward* to the version targeted by the Quality update. The resulting fully-patched file goes onto the system, and the backward patch included in the Quality update is saved on the system in preparation for the next Quality update.

By analogy, it would be as if you wanted to meet with a bunch of friends, but instead of having to give different directions to each friend, you tell everybody, "Okay, start at the library, and then…" You trust that everybody knows how to get to the library, and you give one set of directions that tells how to get to the final destination from the library. You also give directions from the meeting place back to the library, so they are ready for the next time you need to meet somewhere. (Okay, so that's not really a good analogy, because your friends probably want to go home, not to the library.)

The total disk space required on the server is (eyeballs the graph in the blog post) roughly 250MB for the pair of patches. This is the smallest server footprint of all the patches we've been looking at this week.

Note that the download size of a Quality update is less than double the size of an Express update download. I suspect this is because the reverse patch can take advantage of the bytes in the M0 file that were calculated as part of applying the Quality update. For example, if the reverse patch would have said "Replace bytes 2000 through 3999 with these following 2000 bytes," the information downloaded from the server could say "Replace bytes 2000 through 3999 with bytes 5000 through 6999 of the M0 file you already have." This removes 2000 bytes from the download, and the client can get the 2000 bytes from the M0 file that it had temporarily created as part of applying the Quality update. In that way, what the client really downloads is not so much a reverse patch as it is a *template* for a reverse patch.

Feature summary of Quality updates:

- Quality updates can successfully update all customers, since every client knows how to roll back to M0, at which point they can apply the patch in the Quality update to move forward.
- Quality updates are about a third the size of a Full update.
- Quality updates require very little negotiation with the server. Every customer downloads the same update.
- Quality updates are cache-friendly, because every customer downloads the same update. Therefore, caching features like caching proxies, BranchCache, and peer-to-peer delivery are effective.
- Quality updates do not require significant server support. Once the package is negotiated, it is delivered in its entirety.

The blog article that announced the change to Quality updates reports a 40% improvement in memory usage on the client compared to Express updates, since the client doesn't need to do an inventory of all the files on the system.

Raymond Chen

Follow