What happened if you tried to access a network file bigger than 2GB from MS-DOS?

devblogs.microsoft.com/oldnewthing/20191104-00

November 4, 2019

Raymond Chen

One of my friends is into retrocomputing, and he wondered what happened on MS-DOS if you asked it to access a file on a network share that was bigger than what FAT16 could express.

My friend was under the mistaken impression that when MS-DOS accessed a network resource, it was the sector access that was remoted. Under this model, MS-DOS would still open the boot sector, look for the FAT, parse it, then calculate where the directories were, read them directly from the network hard drive, and write raw data directly to the network hard drive.

This is not how it works.

For one thing, if it worked like that, then if two clients both accessed a network hard drive, they would corrupt each other. Each one has its own locally-cached copy of the FAT, and when it came time to allocate a new cluster, each one would pick a cluster (probably the same one), and assign that cluster to the new data.

What actually happens is that the file system operations themselves are sent remotely, rather than the low-level disk operations. You would send send requests to the server like "Please open a file called `AWESOME.TXT` in write mode" or "Please tell me how big the file `README.DOC` is." The remote server translated these requests into its own native file system, performed the operation, and sent the results back to the MS-DOS client.

The server need not be running a FAT file system. In practice, it was probably running Novell NetWare.

The next question was, "But what happens if the file is bigger than 2GB?"

A file bigger than 2GB? What planet are you from?

We're talking 1984 here. A 20 *megabyte* hard drive costed around $1800. To get to 2GB, you'd first have to invent RAID. Then create an array of 100 drives, which would put you at $180,000. Though you could probably get a bulk discount. And you'd have to be able to connect them and power them all. And then to create that file, you'd need to push 2GB of data over a T1 line, which would take about three hours.

My friend explained, "Well, let's say that the super-huge file is on a supercomputer somewhere. You're not downloading the file, but rather seeking to selected portions and reading little bits. How would that work for a file bigger than 2GB?"

The response to the request for file attributes had a 32-bit value for the file size. So if your file was 2GB, that would still fit.

Read requests took the form of a 32-bit file offset an a 16-bit size. If your file was bigger than 4GB, you would have no way to access any bytes beyond 4GB because it wouldn't fit in the 32-bit file offset.

Of course, if you're retrocomputing, then your poor 1984 MS-DOS system is going to be seeing wild and crazy things from the future, like hard drives bigger than 20MB and processors that can count to a billion in less than 55ms. Its brain might explode (or divide by zero). But that's part of what makes retrocomputing fun, I guess.

Raymond Chen

Follow