Computing over a high-latency network means you have to bulk up

devblogs.microsoft.com/oldnewthing/20060407-25

April 7, 2006

Raymond Chen

One of the big complaints about Explorer we've received from corporations is how often it accesses the network. If the computer you're accessing is in the next room, then accessing it a large number of times isn't too much of a problem since you get the response back rather quickly. But if the computer you're talking to is halfway around the world, then even if you can communicate at the theoretical maximum possible speed (namely, the speed of light), it'll take 66 milliseconds for your request to reach the other computer and another 66 milliseconds for the reply to come back. In practice, the signal takes longer than that to make its round trip. A latency of a half second is not unusual for global networks. A latency of one to two seconds is typical for satellite networks. Note that latency and bandwidth are independent metrics. Bandwidth is how fast you can shovel data, measured in data per unit time (e.g. bits per second); latency is how long it takes the data to reach its destination, measured in time (e.g. milliseconds). Even though these global networks have very high bandwidth, the high latency is what kills you. (If you're a physicist, you're going to see the units "data per unit time" and "time" and instinctively want to multiply them together to see what the resulting "data" unit means. Bandwidth times latency is known as the "pipe". When doing data transfer, you want your transfer window to be the size of your pipe.) High latency means that you should try to issue as few I/O requests as possible, although it's okay for each of those requests to be rather large if your bandwidth is also high. Significant work went into reducing the number of I/O requests issued by Explorer during common operations such as enumerating the contents of a folder. Enumerating the contents of a folder in Explorer is more than just getting the file names. The file system shell folder needs other file metadata such as the last-modification time and the file size in order to build up its `SHITEMID`, which is the unit of item identification in the shell namespace. One of the other pieces of information that the shell needs is the file's index, a 64-bit value that is different for each file on a volume. Now, this information is not returned by the "slow" `FindNextFile` function. As a result, the shell would have to perform three round-trip operations to retrieve this extra information:

- `CreateFile()`,

- `GetFileInformationByHandle()` (which returns the file index in the `BY_HANDLE_FILE_INFORMATION` structure), and finally
- `CloseHandle()`.

If you assume a 500ms network latency, then these three additional operations add a second and a half **for each file in the directory**. If a directory has even just forty files, that's a whole minute spent just obtaining the file indices. (As we saw last time, the `FindNextFile` does its own internal batching to avoid this problem when doing traditional file enumeration.) And that's where this "fast mode" came from. The "fast mode" query is another type of bulk query to the server which returns all the normal `FindNextFile` information as well as the file indices. As a result, the file index information is piggybacked on top of the existing `FindNextFile`-like query. That's what makes it fast. In "fast mode", enumerating 200 files from a directory would take just a few seconds (two "bulk queries" that return the `FindNextFile` information and the file indices at one go, plus some overhead for establishing and closing the connection). In "slow mode", getting the normal `FindNextFile` information takes a few seconds, but getting the file indices would add another 1.5 seconds for each file, for an additional 1.5 × 200 = 300 seconds, or five minutes. I think most people would agree that reducing the time it takes to obtain the `SHITEMID`s for all the files in a directory from five minutes to a few seconds is a big improvement. That's why the shell is so anxious to use this new "fast mode" query. If your program is going to be run by multinational corporations, you have to take high-latency networks into account. And this means bulking up. *Sidebar*: Some people have accused me of intentionally being misleading with the characterization of this bug. Any misleading on my part was unintentional. I didn't have all the facts when I wrote up that first article, and even now I still don't have all the facts. For example, `FindNextFile` using bulk queries? I didn't learn that until Tuesday night when I was investigating an earlier comment—time I should have been spending planning Wednesday night's dinner, mind you. (Yes, I'm a slacker and don't plan my meals out a week at a time like organized people do.) Note that the exercise is still valuable as a thought experiment. Suppose that `FindNextFile` didn't use bulk queries and that the problem really did manifest itself only after the 101st round-trip query. How would you fix it?

I should also point out that the bug in question is not my bug. I just saw it in the bug database and thought it would be an interesting springboard for discussion. By now, I'm kind of sick of it and will probably not bother checking back to see how things have settled out.

Raymond Chen

Follow