Bound by the Speed of Light

There's only so much you can do to optimize NFS over a WAN.

Dear KV,

I've been asked to optimize our NFS (network file system) set up for a global network, but NFS doesn't work the same over a long link as it does over a LAN. Management keeps yelling that we have a multigigabit link between our remote sites but what our users experience when they try to access their files over the WAN link is truly frustrating. Is this just an impossible task?

Feeling Stretched Across the Sea

Dear Stretched,

The number of people who continue to confuse bandwidth with latency, and who don't seem to understand the limitations of the speed of light, does not seem to be decreasing, even though I kindly pointed this out in another context some time ago [Latency and Livelocks. *ACM Queue*, March/April 2008; <http://queue.acm.org/detail.cfm?id=1365494>]. I would have thought that by now word would have gotten out that it doesn't matter how fat your pipe is, over a long distance latency is your biggest challenge. I suspect that this kind of problem is going to continue to come up, because since the tech bubble collapse in 2001, the number of cheap, long-distance fibers has not decreased. The world is awash in a sea of cheap bandwidth. Latency, on the other hand, is another story.

To understand why latency is killing your performance, it pays to understand how the NFS protocol works, at least at a basic level. NFS is a client/server protocol where the user's machine, the client, is trying to get files from the server. When the user wants a file the client makes several requests to the server to get the data. Very simply speaking, the client has to look up the file—that is, tell the server which file it wants to read—and then it has to ask for each block of the file. The NFS protocol tries to read blocks from the file in 32-KB chunks, and it has to ask for each block in succession.

What does this have to do with latency? Many of the operations in NFS require that a previous operation has completed. Obviously the client cannot issue a READ request before it has looked up the file, and just as obviously it cannot issue a READ request for the next block in a file until it has received the previous one. Reading a file across NFS then looks like the following list of operations:

1. Look up file.
2. Read block 1.
3. Read block 2.
4. ...
5. Read block N.
6. Done.

Between each of these steps the client has to wait for an answer from the server. The farther away the server is, the longer that response is going to take. NFS was originally designed to work in a

LAN setting: one where computers were connected by a fast, 10-megabit (yes, you read that right, 10-megabit) network. The Ethernet LANs on which NFS was first deployed had round-trip latencies in the 5- to 10-millisecond range. During this same period computers had CPUs that were measured in the tens of megahertz (yes, again, go back and read that, tens of megahertz). The best thing you could say about this arrangement was that the network was far faster than the CPU, so users didn't mind waiting on the file server because they were used to waiting. This was so long ago that people still smoked cigarettes, and processing a long file generally meant it was time for a smoke break.

In the local area, speeds have continued to improve, both in bandwidth and latency. Most users now have 1-gigabit links to their networks, and LAN latencies are in the sub-millisecond range. Unfortunately, the speed of light gets involved when you start creating networks over global distances. It's typical for a transpacific network link to have a 120-ms round-trip time. Latencies across the Atlantic and North America are lower, but by no means are they fast, and they're not going to get much faster unless someone finds a way to violate some important parts of Einstein's theories. Every physicist wants to violate Einstein, but thus far the great man has remained pretty chaste.

Look at it this way: for every mile between the client and the server, a message cannot get to the server and back to the client in less than 10 microseconds, because light travels one mile in 5.4 microseconds in a vacuum. In a fiber-optic network, or in a copper cable, the signal travels considerably slower. If your server is 1,000 miles from your client, then the best round-trip time you could possibly achieve is 10 milliseconds.

Let's pretend for a moment that you happen to have an ultra-high-tech, light-in-a-vacuum network, and that your round trips are always 10 ms. Let's also pretend that bandwidth is not a problem. How long will it take to read a 1-MB file over that perfect link? If each request is for 32 KB, 32 requests will be sent, which works out to 320 milliseconds. Not so bad, you think, but people notice computer lags of just 200 ms. Whenever your users open a file, they're going to experience this lag, and they're going to be standing in your doorway, if you have a doorway, bitching about how your expensive network link is just too slow. They're not going to like the best answer, which is, "Do not use NFS over long distances," but that truly is the best answer.

There is one protocol that has been endlessly optimized over the past 30 years to deal with remote files over distances of more than a mile, and that's TCP. "Wait! I use NFS over TCP!", I hear you cry. That may be, but once you layer NFS on top of TCP, you have already lost; because of the block nature of NFS just described, you will never be able to use the underlying TCP connection efficiently. Only if NFS were able to get whole files from the server in one request would it be able to start using the underlying protocol efficiently.

There are things that can be done to improve your situation. While it's unlikely you'll be able to do much to tune NFS to do the right things, you can tune your underlying TCP settings; this is normally done on a system-by-system basis, however, which means that you might sacrifice some local performance to improve the user's remote experience. Search the Web for information on "tuning TCP for high bandwidth/delay product networks" and apply their suggestions.

Remember to test what you try, instead of just blindly applying the numbers you are given. By tuning TCP, it's quite easy to make things worse than they were in the default case. I suggest using a program such as `scp` to copy a file that you're also trying to copy across NFS and compare the times. I know that `scp` has some cryptography overhead, but suggesting that people use `rcp` is like suggesting that they learn to juggle by starting with scissors.

I've included a link to a decent bandwidth/delay calculator, just to get you started: <http://www.speedguide.net/bdp.php>.

KV

KODE VICIOUS, known to mere mortals as George V. Neville-Neil, works on networking and operating system code for fun and profit. He also teaches courses on various subjects related to programming. His areas of interest are code spelunking, operating systems, and rewriting your bad code (OK, maybe not that last one). He earned his bachelor's degree in computer science at Northeastern University in Boston, Massachusetts, and is a member of ACM, the Usenix Association, and IEEE. He is an avid bicyclist and traveler who currently lives in New York City.

© 2010 ACM 1542-7730/10/1200 \$10.00