

Re: downloading a single file using multiple threads

Source:

<http://www.tech-archive.net/Archive/DotNet/microsoft.public.dotnet.languages.csharp/2007-03/msg04413.html>

- *From:* "Peter Duniho" <NpOeStPeAdM@xxxxxxxxxxxxxxxxxxxx>
 - *Date:* Wed, 28 Mar 2007 10:36:08 -0700
-

On Wed, 28 Mar 2007 09:59:21 -0700, Willy Denoyette [MVP] <willy.denoyette@xxxxxxxxxxx> wrote:

I know that HTTP 1.1 supports range requests, but this is not exactly my point. The multi part requests in HTTP1.1 are meant to request (for very specific application purposes) a single part or multiple parts in a single request, but you can't (AFAIK) request multiple parts in parallel from multiple client threads.

Well, I've never actually tried it myself. So I will refrain from claiming that I know firsthand that this generally works. However, the so-called "download managers" all claim to work with HTTP servers. I have no reason to doubt them, and based on what I know about HTTP (which is an extremely simple protocol) it seems likely.

There is no theoretical reason why it wouldn't work. There's nothing about HTTP that requires servers to restrict their communications to a given client to a single connection, and there's nothing about HTTP that stipulates that an HTTP server needs to coordinate communications on independent connections. If on one connection the client asks for the first megabyte and on a second connection the same client asks for the second megabyte, then if the server is capable of servicing both requests at the same time, there's no reason the client can't wind up receiving both the first and second megabytes in parallel.

Jon has already outlined the scenarios in which doing so would be helpful. You are absolutely correct that in many cases, the HTTP server is already providing data as fast as it can, and introducing multiple connections to the same server will only slow things down.

Likewise, if the HTTP server does throttle each connection, but your Internet connection is so slow that it's not even as fast as the throttled connection, multiple connections to the same server will again slow things down.

And of course if the HTTP server is configured to throttle the transfer for each connection, it is often also configured to disallow multiple connections from the same client IP address.

There are lots of situations in which a "download manager" won't help at all. It's one of the reasons I don't bother with them...their ability to improve things is greatly overstated IMHO.

But it is true that there are scenarios in which multiple connections retrieving the same file, either from the same HTTP server or from multiple mirror servers, can indeed improve throughput. These scenarios may not

Re: downloading a single file using multiple threads

be very common, but for some people they occur often enough to make it worthwhile using software that takes advantage of them.

Note that there's nothing to stop a download manager from retrieving different parts of the same file from multiple servers as well. Assuming an identical file stored on various mirrors, it doesn't matter which mirror a given part of the file comes from.

That's true, but these will use dedicated protocols don't they? The clients also should have multiple NIC's installed connected over segmented LAN's and/or routers to take some speed advantage of the parallelism.

It just depends. One well-known example of a dedicated protocol to do this sort of thing is BitTorrent. And you're right in suggesting that when this technique is used, it's not always via HTTP. But as you can tell from the success of BitTorrent, you don't actually need multiple NICs installed to take advantage of the technique, nor any special hardware at all.

The goal is to saturate your inbound network connection. If a server is throttling i/o (or is otherwise restricted) to a level below your inbound network connection (something that is becoming more and more common as broadband connections get faster and faster) then having that server send data on multiple connections (assuming it's not smart enough to detect and prevent that condition), or having multiple servers each sending different portions of the same data to the same client, can help saturate that inbound network connection, minimizing the time it takes to download 100% of the data requested.

Pete

.