How do download accelerators work?

Asked 16 years, 3 months ago Modified 2 years, 9 months ago

Viewed 14k times

24

We require all requests for downloads to have a valid login (non-http) and we generate transaction tickets for each download. If you were to go to one of the download links and attempt to "replay" the transaction, we use HTTP codes to forward you to get a new transaction ticket. This works fine for a majority of users. There's a small subset, however, that are using Download Accelerators that simply try to replay the transaction ticket several times.

So, in order to determine whether we want to or even *can* support download accelerators or not, we are trying to understand how they work.

How does having a second, third or even fourth concurrent connection to the web server delivering a static file speed the download process?

What does the accelerator program do?

html download accelerator

Share

Improve this question

Follow

edited Sep 18, 2008 at 15:43

Mark Biek

150k ● 54 ● 158 ● 201

asked Sep 18, 2008 at 15:35

Josef

7,599 ● 3 ● 34 ● 33

8 Answers

Sorted by: Highest score (default)

22

You'll get a more comprehensive overview of Download Accelerators at wikipedia.

Acceleration is multi-faceted

First

A substantial benefit of managed/accelerated downloads is the tool in question remembers Start/Stop offsets transferred and uses "partial" and 'range' headers to request parts of the file instead of all of it.

This means if something dies mid transaction (ie: TCP Time-out) it just reconnects where it left off and you don't have to start from scratch.

Thus, if you have an intermittent connection, the aggregate transfer time is greatly lessened.

Second

Download accelerators like to break a single transfer into several smaller segments of equal size, using the same start-range-stop mechanics, and perform them in parallel, which greatly improves transfer time over slow networks.

There's this annoying thing called bandwidth-delay-product where the size of the TCP buffers at either end do some math thing in conjunction with ping time to get the actual experienced speed, and this in practice means large ping times will limit your speed regardless how many megabits/sec all the interim connections have.

However, this limitation appears to be "per connection", so multiple TCP connections to a single server can help mitigate the performance hit of the high latency ping time.

Hence, people who live near by are not so likely to need to do a segmented transfer, but people who live in far away locations are more likely to benefit from going crazy with their segmentation.

Thirdly

In some cases it is possible to find multiple servers that provide the same resource, sometimes a single DNS address round-robins to several IP addresses, or a server is part of a mirror network of some kind. And download managers/accelerators can detect this and apply the segmented transfer technique across multiple servers, allowing the downloader to get more collective bandwidth delivered to them.

Support

Supporting the first kind of acceleration is what I personally suggest as a "minimum" for support. Mostly, because it makes a users life easy, and it reduces the amount of aggregate data transfer you have to provide due to users not having to fetch the same content repeatedly.

And to facilitate this, its recommended you, compute how much they have transferred and don't expire the ticket till they look "finished" (while binding traffic to the first IP that used the ticket), or a given 'reasonable' time to download it has passed. ie: give them a window of grace before requiring they get a new ticket.

Supporting the second and third give you bonus points, and users generally desire it at least the second, mostly because international customers don't like being treated as second class customers simply because of the greater ping time, and it doesn't objectively consume more bandwidth in any sense that matters. The worst that happens is they might cause your total throughput to be undesirable for how your service operates.

It's reasonably straight forward to deliver the first kind of benefit without allowing the second simply by restricting the number of concurrent transfers from a single ticket.

Share Improve this answer Follow

edited Oct 8, 2014 at 20:18

answered Sep 18, 2008 at 15:43

Kent Fredric
57.3k ● 14 ● 111 ● 151

I think you are describing a *download manager* rather than a *download accelerator* – user585968 Oct 5, 2014 at 23:41

@User52784246 In your opinion, what is the difference? IMO, "managers" are download accelerators for >1 files. And Managers may offer no acceleration services – Kent Fredric Oct 8, 2014 at 14:56

1 Keep in mind, that multiplexed transfers fundamentally rely on the infrastructure that is established by the simple "resumable" design with partial/range headers. – Kent Fredric Oct 8, 2014 at 14:58

1 All *download accelerators* are download managers but not all *download managers* are necessarily accelerators – user585968 Oct 8, 2014 at 14:59

@User52784246 I've greatly expanded on my answer, hope that helps =) – Kent Fredric Oct 8, 2014 at 15:27

1

I believe the idea is that many servers limit or evenly distribute bandwidth across connections. By having multiple connections, you're cheating that system and getting more than your "fair" share of bandwidth.

answered Sep 18, 2008 at 15:39

Share Improve this answer Follow

Aaron Jensen
6,060 ● 1 ● 32 ● 40

1

It's all about Little's Law. Specifically each stream to the web server is seeing a certain amount of TCP latency and so will only carry so much data. Tricks like increasing the TCP window size and implementing selective acks help but are poorly implemented and generally cause more problems than they solve.

Having multiple streams means that the latency seen by each stream is less important as the global throughput increases overall.

Another key advantage with a download accelerator even when using a single thread is that it's generally better than using the web browsers built in download tool. For example if the web browser decides to die the download tool will continue. And the download tool may support functionality like pausing/resuming that the built-in brower doesn't.

Share Improve this answer Follow

answered Sep 18, 2008 at 15:40

Paul Hargreaves
1,797 ● 3 ● 23 ● 23

1

My understanding is that one method download accelerators use is by opening many parallel TCP connections - each TCP connection can only go so fast, and is often limited on the server side.

TCP is implemented such that if a timeout occurs, the timeout period is increased. This is very effective at preventing network overloads, at the cost of speed on individual TCP connections.

Download accelerators can get around this by opening dozens of TCP connections and dropping the ones that slow to below a certain threshold, then opening new ones to replace the slow connections.

While effective for a single user, I believe it is bad etiquette in general.

You're seeing the download accelerator trying to re-authenticate using the same transaction ticket - I'd recommend ignoring these requests.

Share Improve this answer Follow

answered Sep 18, 2008 at 16:26

Arc the daft
225 ● 1 ● 2

0

From: http://askville.amazon.com/download-accelerator-protocol-work-advantages-benefits-application-area-scope-plz-suggest-URLs/AnswerViewer.do?requestId=9337813

Quote: The most common way of accelerating downloads is to open up parllel downloads. Many servers limit the bandwith of one connection so opening more in parallel increases the rate. This works by specifying an offset a download should start which is supported for HTTP and FTP alike.

Of course this way of acceleration is quite "unsocial". The limitation of bandwith is implemented to be able to serve a higher number of clients so using this technique lowers the maximum number of peers that is able to download. That's the reason why many servers are limiting the number of parallel connection (recognized by IP), e.g. many FTP-servers do this so you run into problems if you download a file and try to continue browsing using your browser. Technically these are two parallel connections.

Another technique to increase the download-rate is a peer-to-peer-network where different sources e.g. limited by asynchron DSL on the upload-side are used for downloading.

Share Improve this answer Follow

answered Sep 18, 2008 at 15:39

Haabda
1,433 ● 2 ● 13 ● 17

I believe the pedant inside me wants to accuse you of missusing the word 'peer'. – Henry B Sep 18, 2008 at 15:45

0

Most download 'accelerators' really don't speed up anything at all. What they are good at doing is congesting network traffic, hammering your server, and breaking custom scripts like you've seen. Basically how it works is that instead of making one request and downloading the file from beginning to end, it makes say four requests...the first one downloads from 0-25%, the second from 25-50%, and so on, and it makes them all at the same time. The only particular case where this helps any, is if their ISP or firewall does some kind of traffic shaping such that an individual download speed is limited to less than their total download speed.

Personally, if it's causing you any trouble, I'd say just put a notice that download accelerators are not supported, and have the users download them normally, or only using a single thread.

answered Sep 18, 2008 at 15:40

davr
19.1k ● 17 ● 80 ● 99

0

They don't, generally.

To answer the substance of your question, the assumption is that the server is rate-limiting downloads on a per-connection basis, so simultaneously downloading multiple chunks will enable the user to make the most of the bandwidth available at their end.

answered Sep 18, 2008 at 15:41

moonshadow
88.9k ● 7 ● 86 ● 121

0

Typically download-accelerators depend on `partial content download - status code 206`. Just like the streaming media players, media players ask for a small chunk of the full file to the server and then download it and play. Now the catch is if a server restricts `partial-content-download` then the download accelerator won't work!. It's easy to configure a server like `Nginx` to restrict `partial-content-download`.

How to know if a file can be downloaded via ranges/partially?

Ans: check for a header value `Accept-Ranges:`. If it does exist then you are good to go.

How to implement a feature like this in any programming language?

Ans: well, it's pretty easy. Just spin up some threads/co-routines(choose threads/co-routines over processes in I/O or network bound system) to download the N-number of chunks in parallel. Save the partial files in the right position in the file. and you are technically done. Calculate the download speed by keeping a global variable `downloaded_till_now=0` and increment it as one thread completes downloading a chunk. don't forget about `mutex` as we are writing to a global resource from multiple thread so do a `thread.acquire()` and `thread.release()`. And also keep a unix-time counter. and do math like `speed_in_bytes_per_sec = downloaded_till_now/(current_unix_time-start_unix_time)`

Share Improve this answer Follow

answered Mar 23, 2022 at 7:24

Amin Pial

491 ● 9 ● 14