Why not network-side pre-fetching?

Lori MacVittie, 2009-14-04

The acceleration technique known as *pre-fetching* went the way of the do-do bird sometime around 2002. But perhaps it should be resurrected, just in a different place and with a slightly different focus.

A SHORT HISTORY OF ACCELERATION TECHNIQUES

Most modern acceleration techniques revolve around two things: decreasing the amount of data to be transferred (compression, optimization of the client-side cache) or twiddling with protocols (TCP, HTTP) and their associated behaviors to improve the overall speed at which a client and server communicate. Back in the early days of application acceleration most technologies were heavily focused on the client. Delta-based acceleration techniques were used to determine the difference between one page and another, and only changes were sent in order to minimize data transfer.

Another, now long forgotten technique, was ***pre-fetching***. This technique involved client-side logic that traversed all the links in a page and fetched the content, storing it in the cache, on the premise that the user was likely to visit one. When the user did visit one of the pre-fetched links the load-time was nearly instantaneous, resulting in what appeared to be almost supernatural acceleration. The downside was, and remains, that if the user ***doesn't*** deign to visit one of those links that the pre-fetching process has wasted time and bandwidth and artificially impacted the visitor metrics of sites it has pre-fetched.

The problem was, and remains, that pre-fetching technologies can't be prescient. They can't ***know*** with any certainty which sites a user will choose to visit. To guess accurately, even over time based on previous behavior, would require something akin to a neural network. That's just not feasible. Pre-fetching eventually fell by the way side, along with several other acceleration techniques as network speeds increased to the point that it didn't make sense anymore.

But the concept of pre-fetching is still a tantalizing one as a mechanism for improving the user experience as far as performance goes. It has the same appeal as pre-ordering dinner; it's just nice to have it waiting for you rather than the other way around. One of the reasons pre-fetching failed to be the "next big thing" was largely because it tried to predict which sites a user would visit. If it had ***known***, absolutely, then it certainly would have been a successful technique and perhaps would continue to exist as an option today. Perhaps we were just pre-fetching the wrong thing, and in the wrong place.

Pre-fetching on the client-side cast too wide a net for it to be useful. It was too hit and miss.

But what if we narrowed the focus of pre-fetching to only consider what we ***know*** is 95% certain to be loaded?

PRE-FETCHING ON THE NETWORK-SIDE

Consider that when a user accesses a site the first thing that is (necessarily) requested is a "page". That page contains the HTML and all dependent links within it. That page has to be loaded ***first***, by the client, before it can start making requests for all the objects that make up the page (images, CSS, scripts, etc…). That means that any application-aware intermediary has seen the entire page ***and*** all its composite object references.

The page is being loaded. The browser will almost certainly – barring a sudden desktop crash, Internet interruption, or a quick click on the "stop" button in the browser – make a request for all the composite objects. The browser just has to parse out the HTML and build its list of those links before it can request them. Even browsers that load progressively – which is most of them today – cannot make a request for an object until it hits a reference to it in the HTML.

Meanwhile, the intermediary is sitting around holding open a connection twiddling its thumbs like a teenage boy trying to figure out what to say next to the girl on the other end of the telephone line.

Yeah. Exactly. Why couldn't the intermediary spend a few cycles while it waits for data to be transferred to the client and the subsequent requests for more objects pre-fetching the composite objects? If the intermediary could pre-fetch, it could effectively reduce the response-time, as seen by the user, by the amount of time it takes to execute application logic specific to the requests. When implemented by a solution providing application security or other processing that introduces latency due to the nature of inspection, network-side pre-fetching could be a means of offsetting that latency. When the request came in, WHAM! It's immediately sent back with what appears to be amazing alacrity.

You might be thinking, "Hey, that sounds a lot like caching." And it is, but it isn't. Caching benefits multiple people by moving content closer to the edge so it can be retrieved quickly; network-side pre-fetching would be applicable only to a single user, and would not necessarily persist across multiple pages. I suppose it ***could***, or better yet it could take advantage of a caching solution to even further improve resource efficiency and response time.

Granted, network-side pre-fetching is only applicable to objects actually hosted on the same domain and can't be even partially generated by client-side scripting. And the technique would likely not be beneficial to AJAX-based requests. Still, speeding up requests for images, style-sheets, scripts, and other shared content such as headers and footers may provide a non-trivial increase in performance of applications and web sites.

Network-side pre-fetching also wouldn't reduce the burden on the servers – the same number of requests still have to be made – but it would potentially improve the response time and thus make users (corporate and otherwise) happier and thus more productive.

So why not?

Related articles and blogs:

- Architects need to better leverage virtualization
- Not all application requests are created equal
- True or False: Application acceleration solutions teach developers to write inefficient code
- 9 Ways to use network-side scripting to architect faster, scalable, more secure applications
- Understanding network-side scripting

F5 Networks, Inc. | 401 Elliot Avenue West, Seattle, WA 98119 | 888-882-4447 | f5.com

F5 Networks, Inc. Corporate Headquarters info@f5.com	F5 Networks Asia-Pacific apacinfo@f5.com	F5 Networks Ltd. Europe/Middle-East/Africa emeainfo@f5.com	F5 Networks Japan K.K. f5j-info@f5.com