The Clever Ways Chrome Hides Latency by Anticipating Your Every Need

Monday, June 18, 2012 at 9:15AM

Todd Hoff in Strategy

Ilya Grigorik has written another wonderful article lavishly detailing the extraordinary tactics Chrome employs to hide network latency from users: Chrome Networking: DNS Prefetch & TCP Preconnect. Ilya springs some surpising factoids on us, revealing how the web has slowed and super sized:

- The size of an average page has grown to 1059kB and is now composed of over 80 subresource requests.
- An average DNS lookup takes between 60 and 120ms. This creates a 100-200ms of latency before a request can be sent because of th full round-trip (RTT) to perform the TCP handshake.
- Slow mobile experiences are largely due to the much higher RTT's (200-1000ms) on wireless networks. Reducing the number of outbound connections and the total byte size of your pages is the single best optimization you can make for mobile today.

Chrome reduces apparent latency using a host of clever anticipatory mechanisms:

- Learns the network topology as you use it via a Predictor object that anticipates user behavior and future resources based on historical browsing data, heuristics, and many other hints from the browser to

anticipate the requests.
- Speculatively pre-resolves hostnames (DNS prefetching).
- Speculatively opens connections (TCP preconnect).
- Uses a pool of 8 threads to resolve DNS requests through the OS. A faster resolver is in the works.
- Resolves the first 10 most used URLs on startup.
- Specutively connects to the search engine when it looks like a search is being typed into the omnibar.
- Specutively connects to the most likely host when a URL is typed into the omnibar.
- Uses a speculative PreloadScanner that looks ahead in the document and queues up remote resources before the parser sees them.
- Learns the most visited resource domains for each hostname and preemptively resolves and preconnects to these resource hosts before the parser even sees them.
- Actions such as hovering over a link can kick off a prefetch.
- Adding a link element in the head of a document with *rel=dns-prefetch* hints to the browser to pre-resolve the DNS name of that host. A good and often overlooked reason for doing this is to pre-resolve redirects: if you know that a specific host request will return a 3XX to a different host, then you can also pre-resolve that via dns-prefetch.

These obviously sophisticated tactics may not apply directly to your application, but similar opportunities to hide latency could be hiding in your app. As Ilya says, "Small wins, but it all adds up!"

For more examples of good anticipatory design take a look at: 3 Secrets To Lightning Fast Mobile Design At Instagram.

---

http://highscalability.com/blog/2012/6/18/the-clever-ways-chrome-hides-latency-by-anticipating-your-ev.html