B0929045 叶子晴 資工二甲

Q: 研究並比較 HTTP 1.0, HTTP 1.1, HTTP 2.0 的差別

首先，HTTP（超文字傳輸協議，HyperText Transfer Protocol）是一個簡單的請求-響應協議，它通常基於TCP進行連線。所有的WWW檔案都必須遵守這個標準。設計HTTP最初的目的是為了提供一種釋出和接收HTML頁面的方法。是用於從WWW伺服器傳輸超文字到本地瀏覽器的傳輸協議。預設使用80埠，HTTP客戶端發起一個請求，建立一個到伺服器指定埠（預設是80埠）的TCP連線。請求和響應訊息的頭以ASCII碼形式給出，而訊息內容則具有一個類似MIME的格式。Http基於TCP連線而不是UDP的原因**在於一個網頁必須傳送很多資料**，而TCP協議提供傳輸控制，按順序組織資料，和錯誤糾正。HTTP協議的瓶頸及其優化技巧都是基於TCP協議本身的特性。TCP建立連線時三次握手有1.5個RTT（Round-Trip Time）的延遲，為了避免每次請求的都經歷握手帶來的延遲，應用層會選擇不同策略的http長連結方案。又如TCP在建立連線的初期有慢啟動（Slow Start）的特性，所以連線的重用總是比新建連線效能要好。

目前HTTP連線使用的是請求及響應的方式，在請求時需要先建立連線，在需要客戶端時就向伺服器發出請求，伺服器端才能回覆資料。而Http標準現狀分為三種情況，HTTP 1.0、 HTTP 1.1和 HTTP 2.0。

HTTP 1.0是第一個在通訊中指定版本號的HTTP 協議版本，至今仍被廣泛採用，特別是在代理伺服器中。但為了提高系統的效率，HTTP 1.0規定瀏覽器與伺服器**只會保持短暫的連線**，瀏覽器的每次請求都需要與伺服器建立一個TCP連線，伺服器完成請求處理後立即斷開TCP連線，伺服器不跟蹤每個客戶也不記錄過去的請求。**第一個缺點，是由於每次客戶端向伺服器請求時都要重新建立TCP連線**，當網頁中包含有外部的靜態資源的載入，例如圖片、CSS和JS檔案等等，客戶端還是要根據這些資源的URL重新與伺服器進行TCP的連線。不管檔案的大小，客戶端和伺服器端每次建立和關閉連線都是一個相對比較費時的過程，且會嚴重影響客戶機和伺服器的效能。**第二個缺點，就是連線無法複用**，客戶端是依據域名來向伺服器建立連線，一般PC端瀏覽器會針對單個域名的伺服器同時建立6～8個連線，手機端的連線數則一般控制在4～6個，且連線數並不是越多越好，資源開銷和整體延遲都會隨之增大。**會出現一個重大問題，則是 Head-Of-Line-Blocking(隊頭阻塞)，會導致頻寬無法被充分利用，以及後續健康請求被阻塞**。例如當有5個請求同時發出，在第一個請求沒有收到回覆之前，後續從應用層發出的請求只能排隊，請求2、3、4、5只能等請求1的響應回來之後才能逐個發出。雖然當網路通暢的時候影響不大，若請求1的Request沒有抵達伺服器，或者伺服器響應因為網路阻塞沒有及時返回，會影響的就是之後所有後續請求。由此就有了Http 1.1及Http 2.0來解決以上的問題。

**HTTP 1.1、HTTP 2.0與HTTP 1.0的差別在於連線無法複用問題及Head-Of-Line-Blocking問題的解決方法。**

**第一，連線無法複用問題，原本在HTTP 1.0裡可以設定Connection: Keep-Alive**，在Http首部裡設定Keep-Alive可以在一定時間內複用連線，具體複用時間的長短可以由伺服器控制，一般在15s左右。而一段時間內的連線複用對PC端瀏覽器的體驗幫助很大，大部分的請求在集中在一小段時間以內。但對移動app來說，成效不大，app端的請求比較分散且時間跨度相對較大。**而HTTP 1.1和HTTP 2.0則是原本Connection的預設值就是Keep-Alive，如果要關閉連線複用需要顯式的設定Connection: Close**。**第一種情況**是當客戶使用HTTP 1.1或HTTP 2.0時，但不希望使用長連結，則需要在Header中指明Connection的值為Close。**第二種情況**當服務方也不想支援長連結，則在Response中也需要明確說明Connection的值為Close。不論Request還是Response的Header中包含了值為Close的Connection，都需要表明當前正在使用的TCP連結且在當天請求處理完畢後會被斷掉，以後客戶再進行新的請求時就必須建立新的TCP連結。

**第二，Head Of Line Blocking問題，是**HTTP 1.1和HTTP 2.0的最大優勢。首先，HTTP1.1的解決方案是**Http Pipelining**，請求2，3不用等請求1的Response返回之後才發出，而是**幾乎在同一時間把Request發向了伺服器**，2，3及所有後續共用該連線的請求節約了等待的時間，**極大的降低了整體延遲**。只不過其缺點在於Pipelining只能適用於HTTP 1.1，由此Head-Of-Line -Blocking並沒有完全得到解決，Server的Response還是要求依次返回，遵循FIFO(First-In -First-Out)原則。絕大部分的http代理伺服器不支援pipelining，可能會導致新的Front-Of -Queue-Blocking問題，由此出現了HTTP 2.0的多路複用方法。

HTTP 2.0主要分為三步，**多路複用（Multiplexing）與請求優先順序（Request Prioritization）**、 **首部壓縮（Header Packet）、伺服器推送（Server Push）。**

**第一步，多路複用（multiplexing）**是允許同時通過單一的 HTTP 2.0 連線發起多重的請求及響應訊息。在 HTTP 1.1中瀏覽器客戶端在同一時間，針對同一域名下的請求有一定數量限制，超過限制數目的請求會被阻塞，一些站點會有多個靜態資源 CDN 域名的原因。而 HTTP 2.0 的多路複用(Multiplexing) 則允許同時通過單一的 HTTP 2.0 連線發起多重的請求及響應訊息，因此 HTTP 2.0 可以很容易的去實現**多流並行**而不用依賴建立多個 TCP 連線，把 HTTP 協議通訊的基本單位縮小為一個一個幀，這些幀對應著邏輯流中的訊息，並行地在同一個 TCP 連線上雙向交換訊息。

**第二步，首部壓縮（Header Compression），**在HTTP 1.1並不支援 HTTP 首部壓縮，為此 SPDY 和 HTTP 2.0 出現了， SPDY 使用的是通用的DEFLATE 演算法，而 HTTP 2.0 則使用了專門為首部壓縮而設計的 HPACK 演算法。

**第三步，服務端推送（Server Push）**伺服器推送是一種在客戶端請求之前傳送資料的機制。在 HTTP 2.0 中，伺服器可以對客戶端的一個請求傳送多個響應。Server Push 讓 HTTP1.0及HTTP 1.1 時代使用內嵌資源的優化手段變得沒有意義。如果一個請求是由你的主頁發起的，伺服器很可能會響應主頁內容、Logo 以及樣式表，因為它知道客戶端會用到這些東西，這相當於在一個 HTML 文件內集合了所有的資源。不過與之相比，伺服器推送還有一個很大的優勢，就是可以快取，也讓在遵循同源的情況下，不同頁面之間可以共享快取資源成為可能。