B0929030 周兆騰

HTTP（HyperText Transfer Protocol)是網際網路上應用最為廣泛的一種網路協議。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。HTTP 協議和 TCP 協議是不衝突的，HTTP 定義在七層協議中的應用層，TCP 解決的是傳輸層的邏輯。HTTP 使用 TCP 而不是 UDP 的原因在於（開啟）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。如 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。

HTTP 1.0 規定瀏覽器與伺服器只保持短暫的連線，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。但是，這也造成了一些效能上的缺陷，例如，一個包含有許多影象的網頁檔案中並沒有包含真正的影象資料內容，而只是指明瞭這些影象的 URL 地址，當 WEB 瀏覽器訪問這個網頁檔案時，瀏覽器首先要發出針對該網頁檔案的請求，當瀏覽器解析 WEB 伺服器返回的該網頁文件中的 HTML 內容時，發現其中的影象標籤後，瀏覽器將根據標籤中的 src 屬性所指定的 URL 地址再次向伺服器發出下載影象資料的請求。顯然，訪問一個包含有許多影象的網頁檔案的整個過程包含了多次請求和響應，每次請求和響應都需要建立一個單獨的連線，每次連線只是傳輸一個文件和影象，上一次和下一次請求完全分離。即使影象檔案都很小，但是客戶端和伺服器端每次建立和關閉連線卻是一個相對比較費時的過程，並且會嚴重影響客戶機和伺服器的效能。

為了克服 HTTP 1.0 的這個缺陷，HTTP 1.1 支援持久連線（HTTP/1.1 的預設模式使用帶流水線的持久連線），在一個 TCP 連線上可以傳送多個 HTTP 請求和響應，減少了建立和關閉連線的消耗和延遲。一個包含有許多影象的網頁檔案的多個請求和應答可以在一個連線中傳輸，但每個單獨的網頁檔案的請求和應答仍然需要使用各自的連線。HTTP 1.1 還允許客戶端不用等待上一次請求結果返回，就可以發出下一次請求，但伺服器端必須按照接收到客戶端請求的先後順序依次回送響應結果，以保證客戶端能夠區分出每次請求的響應內容，這樣也顯著地減少了整個下載過程所需要的時間。

HTTP2.0 比之前的協議在效能上有很大的提升。二進位制分幀在應用層(HTTP/2)和傳輸層(TCP or UDP)之間增加一個二進位制分幀層。在不改變 HTTP/1.x 的語義、方法、狀態碼、URI 以及首部欄位的情況下, 解決了 HTTP1.1 的效能限制，改進傳輸效能，實現低延遲和高吞吐量。在二進位制分幀層中，

HTTP/2 會將所有傳輸的資訊分割為更小的訊息和幀（frame）,並對它們採用二進位制格式的編碼 ，其中 HTTP1.x 的首部資訊會被封裝到 HEADER frame，而相應的 Request Body 則封裝到 DATA frame 裡面。首部壓縮（Header Compression）HTTP/1.1 並不支援 HTTP 首部壓縮，為此 SPDY 和 HTTP/2 應運而生， SPDY 使用的是通用的 DEFLATE 演算法，而 HTTP/2 則使用了專門為首部壓縮而設計的 HPACK 演算法。服務端推送（Server Push）是一種在客戶端請求之前傳送資料的機制。在 HTTP/2 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.x 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：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。