http 1.0, http 1.1, http 2.0 的差別

HTTP基本介紹:

Hyper Text Transfer Protocol: 一種用戶端瀏覽器和伺服端伺服器之間溝通的標準協定，他是屬於OSI[1]七層模型中的應用層。

基本上，HTTP是一種Client/Server的應用，Client端透過網址、超連結向Server下達HTTP請求(Request)，請求Web server的虛擬目錄的資源(html、image or backend的執行結果)，處理完畢後，使用MIME格式[2]回應(respond)回Client端。

目前主要版本有HTTP/1.0、HTTP/1.1、HTTP/2.0。

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[3]，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。造成一些效能上的缺陷，

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性能。

HTTP/1.1

為了克服 HTTP1.0 的這個缺陷，HTTP1.1 支援持久連線，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。

---

[1] OSI 模型是一種制定網路標準都會參考的概念性架構，依據網路運作方式，OSI 模型共切分成 7 個不同的層級，每級按照網路傳輸的模式，定義所屬的規範及標準。

[2] 多用途互聯網郵件擴展類型。是設定某種擴展名的文件用一種應用程序來打開的方式類型，當該擴展名文件被訪問的時候，瀏覽器會自動使用指定應用程序來打開。

[3] 負責在網際網路將資料從設備傳輸到伺服器

HTTP1.1 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。

HTTP/2.0

2000 年以後，網頁技術飛躍性地進步，像是 AJAX、jQuery、CSS3、HTML 5 等等，意味著 client 跟 server 端會容易有更多的互動，網頁所需要的數據大小或是請求數不斷上升。

HTTP/1.x 仍有一些基礎上的限制，像是純文字傳輸、缺少 server 主動推送資料的能力等等，讓我們沒辦法進一步提高效能，於是就有了HTTP/2.0

HTTP/2 比 HTTP/1.x 好在哪？

- 與 HTTP/1.x 的許多標準相容（method, status code, etc.）
- Header Compression（頭部壓縮）

  HTTP/2 利用 HPACK 進行壓縮，其中包含 1. 靜態字典 2. 動態字典 3. 霍夫曼編碼三種方式，將大量重複的 Header key: value 壓縮成 1 ~ 2 bytes。

  在許多 modern web，初始頁面渲染常常要 80 甚至 100 個 requests，Header Compression 在節省流量上有顯著的效益。
- Binary Protocol（二進位協定）

  HTTP/2 在傳輸時是以 Binary 為主，這大大減輕了實作上的負擔，而且 Binary 的 Parsing 遠遠比 Text 還要來得有效率。

  在 HTTP/1.1，為了正確處理純文字的 parsing（換行、空行、空白等等都是需要特別處理），定義了四種方法，而在 HTTP/2 中只需要一種處理方式。
- HTTP/2 Server Push（伺服器推送）

以一般的 web 為例，傳統 HTTP/1.x 要等到 browser 收到 HTML 之後，再根據上面指定的網址去索取對應的 CSS 跟 JavaScript。

而透過 Server Push，server 可以在 client 提出後續請求之前，主動將這些已知會被接著請求的檔案推送給 client。

- TCP Multiplexing（多路複用）

  HTTP/1.x 一直以來都有 Head-of-Line blocking[4] 的問題，而 HTTP/2 允許 client 透過同一個 TCP connection 同步發送多個 requests 給 server，而 server 也能用同一個 TCP connection 同步回傳，進而減少額外的 RTT (round trip time)

[4] 當一個 request 下載的內容過大時，會阻塞其他 request