

HTTP1.0、HTTP1.1 和 HTTP2.0 的区别

一只好奇的茂 code小生 2017-07-25



作者 | 一只好奇的茂

地址 | <http://www.jianshu.com/p/be29d679cbff>

声明 | 本文是 一只好奇的茂 原创，已获授权发布，未经原作者允许请勿转载

一、HTTP的历史

早在 HTTP 建立之初，主要就是为了将超文本标记语言(HTML)文档从Web服务器传送到客户端的浏览器。也是说对于前端来说，我们所写的HTML页面将要放在我们的 web 服务器上，用户端通过浏览器访问url地址来获取网页的显示内容，但是到了 WEB2.0 以来，我们的页面变得复杂，不仅仅单纯的是一些简单的文字和图片，同时我们的 HTML 页面有了 CSS，Javascript，来丰富我们的页面展示，当 ajax 的出现，我们又多了一种向服务器端获取数据的方法，这些其实都是基于 HTTP 协议的。同样到了移动互联网时代，我们页面可以跑在手机端浏览器里面，但是和 PC 相比，手机端的网络情况更加复杂，这使得我们开始了不得不对 HTTP 进行深入理解并不断优化过程中。



二、HTTP的基本优化

影响一个 HTTP 网络请求的因素主要有两个：**带宽和延迟**。

- **带宽**：如果说我们还停留在拨号上网的阶段，带宽可能会成为一个比较严重影响请求的问题，但是现在网络基础建设已经使得带宽得到极大的提升，我们不再会担心由带宽而影响网速，那么就只剩下延迟了。
- **延迟**：
 - **浏览器阻塞（HOL blocking）**：浏览器会因为一些原因阻塞请求。浏览器对于同一个域名，同时只能有 4 个连接（这个根据浏览器内核不同可能会有所差异），超过浏览器最大连接数限制，后续请求就会被阻塞。
 - **DNS 查询（DNS Lookup）**：浏览器需要知道目标服务器的 IP 才能建立连接。将域名解析为 IP 的这个系统就是 DNS。这个通常可以利用DNS缓存结果来达到减少这个时间的目的。
 - **建立连接（Initial connection）**：HTTP 是基于 TCP 协议的，浏览器最快也要在第三次握手时才能捎带 HTTP 请求报文，达到真正的建立连接，但是这些连接无法复用会导致每次请求都经历三次握手和慢启动。三次握手在高延迟的场景下影响较明显，慢启动则对文件类大请求影响较大。

三、HTTP1.0和HTTP1.1的一些区别

HTTP1.0最早在网页中使用是在1996年，那个时候只是使用一些较为简单的网页上和网络请求上，而HTTP1.1则在1999年才开始广泛应用于现在的各大浏览器网络请求中，同时HTTP1.1也是当前使用最为广泛的HTTP协议。主要区别主要体现在：

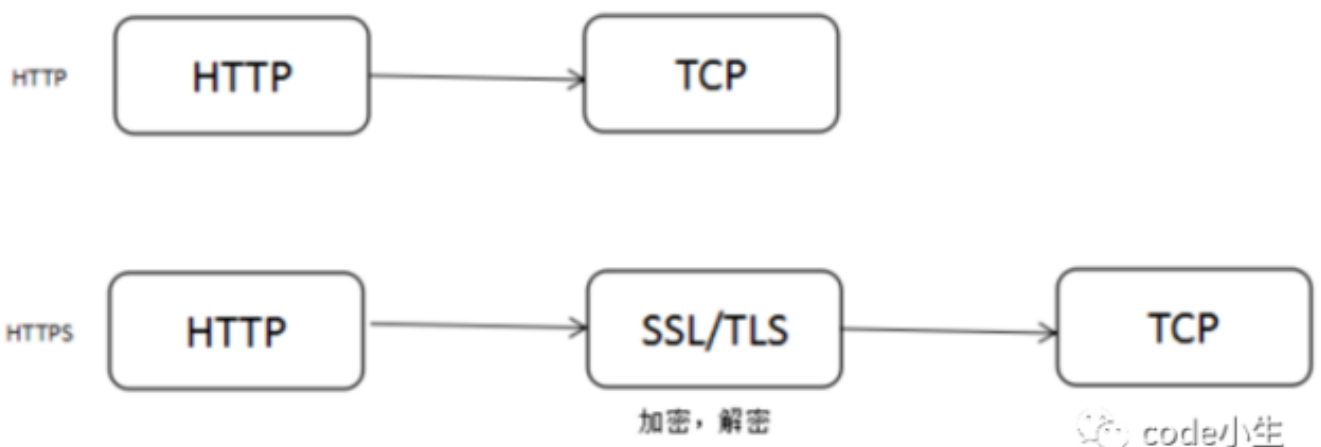
1. **缓存处理**，在HTTP1.0中主要使用header里的If-Modified-Since,Expires来做为缓存判断的标准，HTTP1.1则引入了更多的缓存控制策略例如Entity tag, If-Unmodified-Since, If-Match, If-None-Match等更多可供选择的缓存头来控制缓存策略。
2. **带宽优化及网络连接的使用**，HTTP1.0中，存在一些浪费带宽的现象，例如客户端只是需要某个对象的一部分，而服务器却将整个对象送过来了，并且不支持断点续传功能，HTTP1.1则在请求头引入了range头域，它允许只请求资源的某个部分，即返回码是

206 (Partial Content)，这样就方便了开发者自由的选择以便于充分利用带宽和连接。

3. **错误通知的管理**，在HTTP1.1中新增了24个错误状态响应码，如409 (Conflict) 表示请求的资源与资源的当前状态发生冲突；410 (Gone) 表示服务器上的某个资源被永久性的删除。
4. **Host头处理**，在HTTP1.0中认为每台服务器都绑定一个唯一的IP地址，因此，请求消息中的URL并没有传递主机名 (hostname)。但随着虚拟主机技术的发展，在一台物理服务器上可以存在多个虚拟主机 (Multi-homed Web Servers)，并且它们共享一个IP地址。HTTP1.1的请求消息和响应消息都应支持Host头域，且请求消息中如果没有Host头域会报告一个错误 (400 Bad Request)。
5. **长连接**，HTTP 1.1 支持长连接 (PersistentConnection) 和请求的流水线 (Pipelining) 处理，在一个TCP连接上可以传送多个HTTP请求和响应，减少了建立和关闭连接的消耗和延迟，在HTTP1.1中默认开启Connection: keep-alive，一定程度上弥补了HTTP1.0每次请求都要创建连接的缺点。

四、HTTPS与HTTP的一些区别

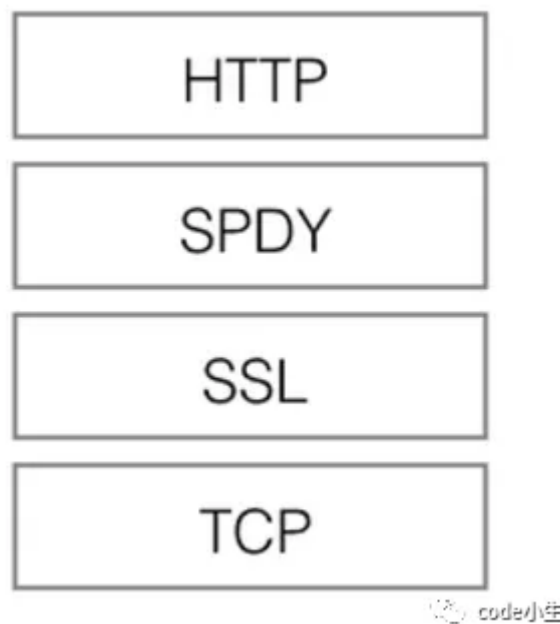
- HTTPS协议需要到CA申请证书，一般免费证书很少，需要交费。
- HTTP协议运行在TCP之上，所有传输的内容都是明文，HTTPS运行在SSL/TLS之上，SSL/TLS运行在TCP之上，所有传输的内容都经过加密的。
- HTTP和HTTPS使用的是完全不同的连接方式，用的端口也不一样，前者是80，后者是443。
- HTTPS可以有效的防止运营商劫持，解决了防劫持的一个大问题。



五、SPDY: HTTP1.x的优化

2012年google如一声惊雷提出了SPDY的方案，优化了HTTP1.X的请求延迟，解决了HTTP1.X的安全性，具体如下：

1. **降低延迟**，针对HTTP高延迟的问题，SPDY优雅的采取了多路复用（multiplexing）。多路复用通过多个请求stream共享一个tcp连接的方式，解决了HOL blocking的问题，降低了延迟同时提高了带宽的利用率。
2. **请求优先级**（request prioritization）。多路复用带来一个新的问题是，在连接共享的基础之上有可能导致关键请求被阻塞。SPDY允许给每个request设置优先级，这样重要的请求就会优先得到响应。比如浏览器加载首页，首页的html内容应该优先展示，之后才是各种静态资源文件，脚本文件等加载，这样可以保证用户能第一时间看到网页内容。
3. **header压缩**。前面提到HTTP1.x的header很多时候都是重复多余的。选择合适的压缩算法可以减小包的大小和数量。
4. **基于HTTPS的加密协议传输**，大大提高了传输数据的可靠性。
5. **服务端推送**（server push），采用了SPDY的网页，例如我的网页有一个style.css的请求，在客户端收到style.css数据的同时，服务端会将style.js的文件推送给客户端，当客户端再次尝试获取style.js时就可以直接从缓存中获取到，不用再发请求了。SPDY构成图：

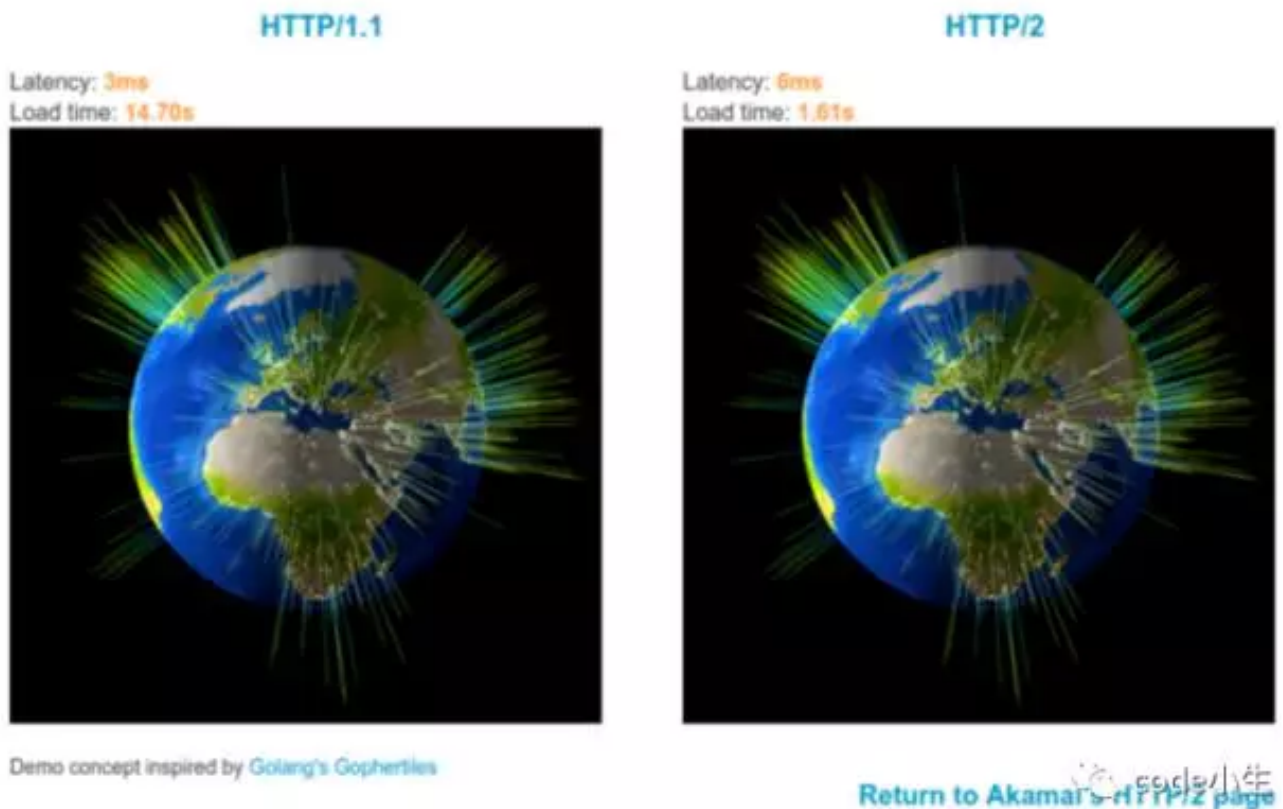


SPDY位于HTTP之下，TCP和SSL之上，这样可以轻松兼容老版本的HTTP协议(将HTTP1.x的内容封装成一种新的frame格式)，同时可以使用已有的SSL功能。

六、HTTP2.0性能惊人

HTTP/2: the Future of the Internet <https://link.zhihu.com/?target=https://http2.akamai.com/demo> 是 Akamai 公司建立的一个官方的演示，用以说明

HTTP/2 相比于之前的 HTTP/1.1 在性能上的大幅度提升。同时请求 379 张图片，从Load time 的对比可以看出 HTTP/2 在速度上的优势。



七、HTTP2.0: SPDY的升级版

HTTP2.0可以说是SPDY的升级版（其实原本也是基于SPDY设计的），但是，HTTP2.0 跟 SPDY 仍有不同的地方，如下：

HTTP2.0和SPDY的区别：

1. HTTP2.0 支持明文 HTTP 传输，而 SPDY 强制使用 HTTPS
2. HTTP2.0 消息头的压缩算法采用 **HPACK** <http://http2.github.io/http2-spec/compression.html>，而非 SPDY 采用的 **DEFLATE** <http://zh.wikipedia.org/wiki/DEFLATE>

八、HTTP2.0和HTTP1.X相比的新特性

- **新的二进制格式**（Binary Format），HTTP1.x的解析是基于文本。基于文本协议的格式解析存在天然缺陷，文本的表现形式有多样性，要做到健壮性考虑的场景必然很多，二进制则不同，只认0和1的组合。基于这种考虑HTTP2.0的协议解析决定采用二进制格式，实现方便且健壮。

- **多路复用** (MultiPlexing) , 即连接共享, 即每一个request都是是用作连接共享机制的。一个request对应一个id, 这样一个连接上可以有多个request, 每个连接的request可以随机的混杂在一起, 接收方可以根据request的 id将request再归属到各自不同的服务端请求里面。
- **header压缩**, 如上文中所言, 对前面提到过HTTP1.x的header带有大量信息, 而且每次都要重复发送, HTTP2.0使用encoder来减少需要传输的header大小, 通讯双方各自cache一份header fields表, 既避免了重复header的传输, 又减小了需要传输的大小。
- **服务端推送** (server push) , 同SPDY一样, HTTP2.0也具有server push功能。

九、HTTP2.0的升级改造

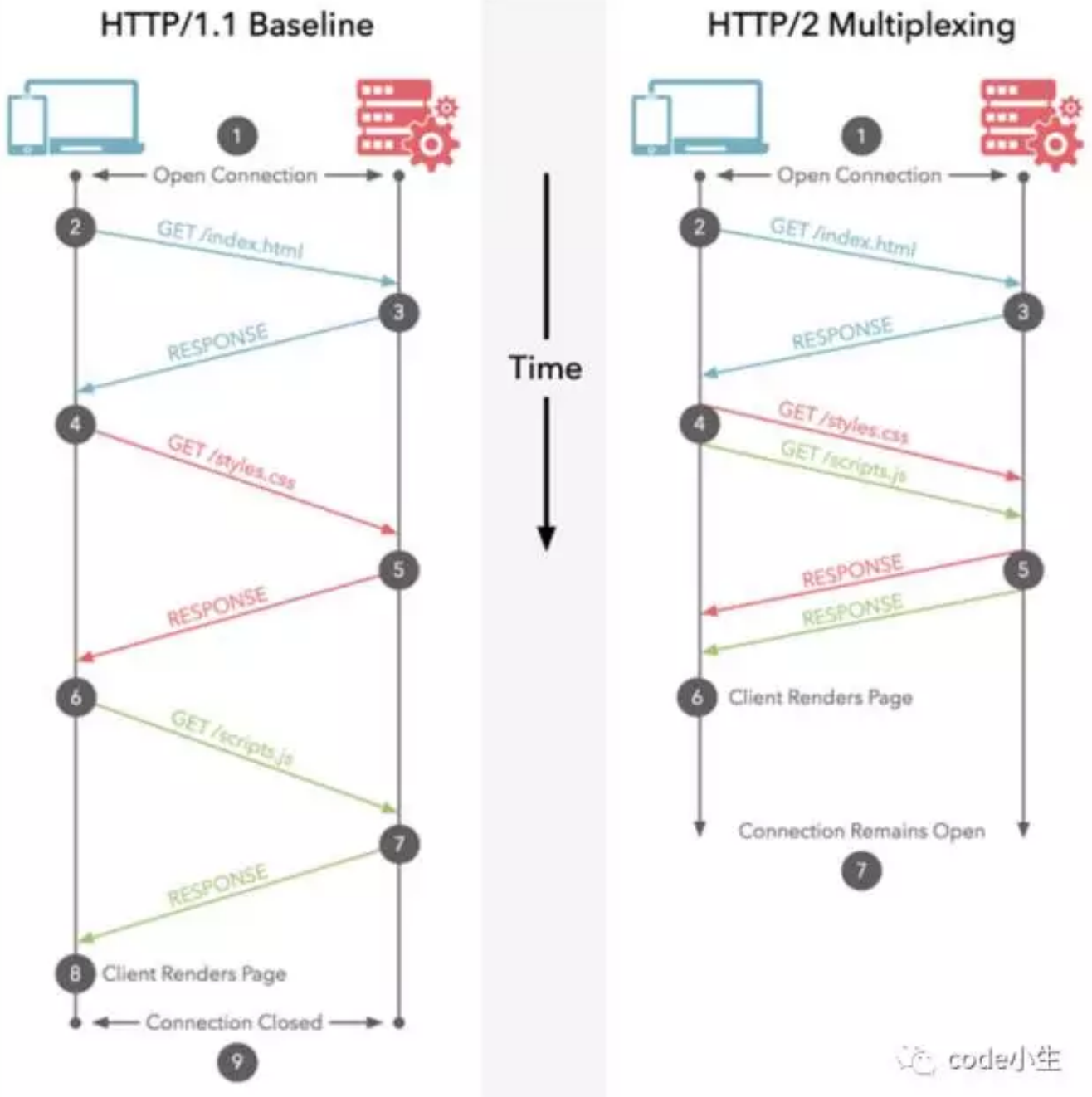
- 前文说了HTTP2.0其实可以支持非HTTPS的, 但是现在主流的浏览器像chrome, firefox表示还是只支持基于 TLS 部署的HTTP2.0协议, 所以要想升级成HTTP2.0还是先升级HTTPS为好。
- 当你的网站已经升级HTTPS之后, 那么升级HTTP2.0就简单很多, 如果你使用NGINX, 只要在配置文件中启动相应的协议就可以了, 可以参考**NGINX白皮书**, **NGINX配置HTTP2.0官方指南** <https://www.nginx.com/blog/nginx-1-9-5/>。
- 使用了HTTP2.0那么, 原本的HTTP1.x怎么办, 这个问题其实不用担心, HTTP2.0完全兼容HTTP1.x的语义, 对于不支持HTTP2.0的浏览器, NGINX会自动向下兼容的。

十、附注

HTTP2.0的多路复用和HTTP1.X中的长连接复用有什么区别?

- HTTP/1.* 一次请求-响应, 建立一个连接, 用完关闭; 每一个请求都要建立一个连接;
- HTTP/1.1 Pipeling解决方式为, 若干个请求排队串行化单线程处理, 后面的请求等待前面请求的返回才能获得执行机会, 一旦有某请求超时等, 后续请求只能被阻塞, 毫无办法, 也就是人们常说的线头阻塞;
- HTTP/2多个请求可同时在一个连接上并行执行。某个请求任务耗时严重, 不会影响到其它连接的正常执行;

具体如图:



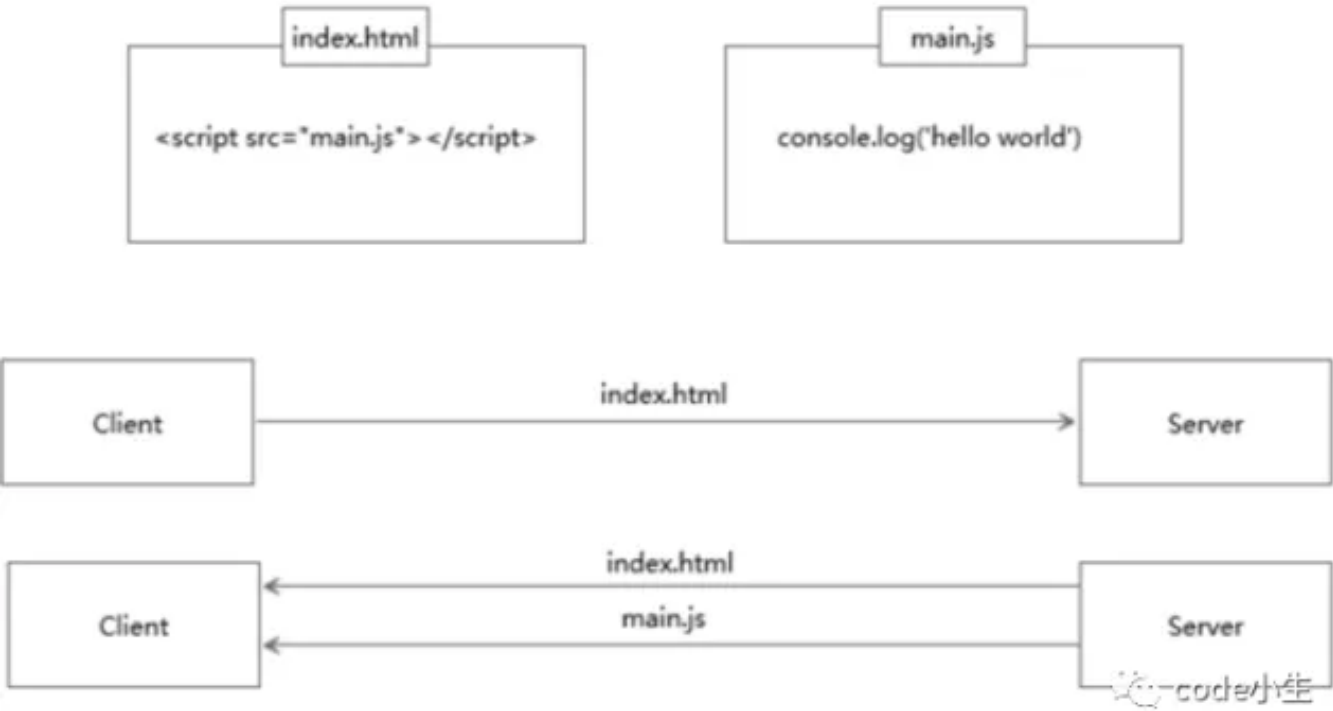
服务器推送到底是什么？

服务端推送能把客户端所需要的资源伴随着index.html一起发送到客户端，省去了客户端重复请求的步骤。正因为没有发起请求，建立连接等操作，所以静态资源通过服务端推送的方式可以极大地提升速度。具体如下：

- 普通的客户端请求过程：



- 服务端推送的过程：



为什么需要头部压缩？

假定一个页面有100个资源需要加载（这个数量对于今天的Web而言还是挺保守的），而每一次请求都有1kb的消息头（这同样也并不少见，因为Cookie和引用等东西的存在），则至少需要多消耗100kb来获取这些消息头。HTTP2.0可以维护一个字典，增量更新HTTP头部，大大降低因头部传输产生的流量。具体参考：HTTP/2 头部压缩技术介绍

HTTP2.0多路复用有多好？

HTTP 性能优化的关键并不在于高带宽，而是低延迟。TCP 连接会随着时间进行自我「调谐」，起初会限制连接的最大速度，如果数据成功传输，会随着时间的推移提高传输的速度。这种调谐则被称为 TCP 慢启动。由于这种原因，让原本就具有突发性和短时性的 HTTP 连接变的十分低效。HTTP/2 通过让所有数据流共用同一个连接，可以更有效地使用 TCP 连接，让高带宽也能真正的服务于 HTTP 的性能提升。

十一、参考

- HTTP/2.0 相比1.0有哪些重大改进？
- 深入研究：HTTP2 的真正性能到底如何
- HTTP/2 头部压缩技术介绍

公众号
「code小生」

日 更 精 彩

请系好安全带



Read more
