

## 33 | 我应该迁移到HTTP/2吗？

2019-08-12 Chrono

《透视HTTP协议》

课程介绍 >



讲述：Chrono

时长 10:38 大小 12.18M



这一讲是“飞翔篇”的最后一讲，而 HTTP 的所有知识也差不多快学完了。

前面你已经看到了新的 HTTP/2 和 HTTP/3 协议，了解了它们的特点和工作原理，如果再联系上前几天“安全篇”的 HTTPS，你可能又会发出疑问：

“刚费了好大的力气升级到 HTTPS，这又出了一个 HTTP/2，还有再次升级的必要吗？”

与各大浏览器“强推”HTTPS 的待遇不一样，HTTP/2 的公布可谓是“波澜不惊”。虽然它是 HTTP 协议的一个重大升级，但 Apple、Google 等科技巨头并没有像 HTTPS 那样给予大量资源的支持。

领资料



直到今天，HTTP/2 在互联网上还是处于“不温不火”的状态，虽然已经有了不少的网站改造升级到了 HTTP/2，但普及的速度远不及 HTTPS。

所以，你有这样的疑问也是很自然的，升级到 HTTP/2 究竟能给我们带来多少好处呢？到底“值不值”呢？

## HTTP/2 的优点

前面的几讲主要关注了 HTTP/2 的内部实现，今天我们就来看看它有哪些优点和缺点。

首先要说的是，HTTP/2 最大的一个优点是**完全保持了与 HTTP/1 的兼容**，在语义上没有任何变化，之前在 HTTP 上的所有投入都不会浪费。

因为兼容 HTTP/1，所以 HTTP/2 也具有 HTTP/1 的所有优点，并且“基本”解决了 HTTP/1 的所有缺点，安全与性能兼顾，可以认为是“**更安全的 HTTP、更快的 HTTPS**”。

在安全上，HTTP/2 对 HTTPS 在各方面都做了强化。下层的 TLS 至少是 1.2，而且只能使用前向安全的密码套件（即 ECDHE），这同时也就默认实现了“TLS False Start”，支持 1-RTT 握手，所以不需要再加额外的配置就可以自动实现 HTTPS 加速。

安全有了保障，再来看 HTTP/2 在性能方面的改进。

你应该知道，影响网络速度的两个关键因素是“**带宽**”和“**延迟**”，HTTP/2 的头部压缩、多路复用、流优先级、服务器推送等手段其实都是针对这两个要点。

所谓的“带宽”就是网络的传输速度。从最早的 56K/s，到如今的 100M/s，虽然网速已经是“今非昔比”，比从前快了几十倍、几百倍，但仍然是“稀缺资源”，图片、视频这样的多媒体数据很容易会把带宽用尽。

节约带宽的基本手段就是压缩，在 HTTP/1 里只能压缩 body，而 HTTP/2 则可以用 HPACK 算法压缩 header，这对高流量的网站非常有价值，有数据表明能节省大概 5%~10% 的流量，这是实实在在的“真金白银”。

与 HTTP/1“并发多个连接”不同，HTTP/2 的“多路复用”特性要求对**一个域名（或者 IP）只用一个 TCP 连接**，所有的数据都在这一个连接上传输，这样不仅节约了客户端、服务器和网络资源，还可以把带宽跑满，让 TCP 充分“吃饱”。

这是为什么呢？

领资料



我们来看一下在 HTTP/1 里的长连接，虽然是双向通信，但任意一个时间点实际上还是单向的：上行请求时下行空闲，下行响应时上行空闲，再加上“队头阻塞”，实际的带宽打了个“对折”还不止（可参考 [🔗 第 17 讲](#)）。

而在 HTTP/2 里，“多路复用”则让 TCP 开足了马力，“全速狂奔”，多个请求响应并发，每时每刻上下行方向上都有流在传输数据，没有空闲的时候，带宽的利用率能够接近 100%。所以，HTTP/2 只使用一个连接，就能抵得过 HTTP/1 里的五六个连接。

不过流也可能会有依赖关系，可能会存在等待导致的阻塞，这就是“延迟”，所以 HTTP/2 的其他特性就派上了用场。

“优先级”可以让客户端告诉服务器，哪个文件更重要，更需要优先传输，服务器就可以调高流的优先级，合理地分配有限的带宽资源，让高优先级的 HTML、图片更快地到达客户端，尽早加载显示。

“服务器推送”也是降低延迟的有效手段，它不需要客户端预先请求，服务器直接就发给客户端，这就省去了客户端解析 HTML 再请求的时间。

## HTTP/2 的缺点

说了一大堆 HTTP/2 的优点，再来看看它有什么缺点吧。

听过上一讲 HTTP/3 的介绍，你就知道 HTTP/2 在 TCP 级别还是存在“队头阻塞”的问题。所以，如果网络连接质量差，发生丢包，那么 TCP 会等待重传，传输速度就会降低。

另外，在移动网络中发生 IP 地址切换的时候，下层的 TCP 必须重新建连，要再次“握手”，经历“慢启动”，而且之前连接里积累的 HPACK 字典也都消失了，必须重头开始计算，导致带宽浪费和时延。

刚才也说了，HTTP/2 对一个域名只开一个连接，所以一旦这个连接出问题，那么整个网站的体验也就变差了。

而这些情况下 HTTP/1 反而不会受到影响，因为它“本来就慢”，而且还会对一个域名开 6~8 个连接，顶多其中的一两个连接会“更慢”，其他的连接不会受到影响。



## 应该迁移到 HTTP/2 吗？

说到这里，你对迁移到 HTTP/2 是否已经有了自己的判断呢？

在我看来，HTTP/2 处于一个略“尴尬”的位置，前面有“老前辈”HTTP/1，后面有“新来者”HTTP/3，即有“老前辈”的“打压”，又有“新来者”的“追赶”，也就难怪没有获得市场的大力“吹捧”了。

但这绝不是说 HTTP/2“一无是处”，实际上 HTTP/2 的性能改进效果是非常明显的，Top 1000 的网站中已经有超过 40% 运行在了 HTTP/2 上，包括知名的 Apple、Facebook、Google、Twitter 等等。仅用了四年的时间，HTTP/2 就拥有了这么大的市场份额和巨头的认可，足以证明它的价值。

因为 HTTP/2 的侧重点是“性能”，所以“是否迁移”就需要在这方面进行评估。如果网站的流量很大，那么 HTTP/2 就可以带来可观的收益；反之，如果网站流量比较小，那么升级到 HTTP/2 就没有太多必要了，只要利用现有的 HTTP 再优化就足矣。

不过如果你是新建网站，我觉得完全可以跳过 HTTP/1、HTTPS，直接“一步到位”，上 HTTP/2，这样不仅可以获得性能提升，还免去了老旧的“历史包袱”，日后也不会有迁移的烦恼。

顺便再多嘴一句，HTTP/2 毕竟是“下一代”HTTP 协议，它的很多特性也延续到了 HTTP/3，提早升级到 HTTP/2 还可以让你在 HTTP/3 到来时有更多的技术积累和储备，不至于落后于时代。

## 配置 HTTP/2

假设你已经决定要使用 HTTP/2，应该如何搭建服务呢？

因为 HTTP/2“事实上”是加密的，所以如果你已经在“安全篇”里成功迁移到了 HTTPS，那么在 Nginx 里启用 HTTP/2 简直可以说是“不费吹灰之力”，只需要在 server 配置里再加一个参数就可以搞定了。

领资料



复制代码

```
1 server {  
2     listen      443 ssl http2;  
3 }
```

```
4
5     server_name    www.xxx.net;
6
7
8     ssl_certificate      xxx.crt;
9     ssl_certificate_key  xxx.key;
```

注意“listen”指令，在“ssl”后面多了一个“http2”，这就表示在 443 端口上开启了 SSL 加密，然后再启用 HTTP/2。

配置服务器推送特性可以使用指令“http2\_push”和“http2\_push\_preload”：

```
1 http2_push          /style/xxx.css;
2 http2_push_preload on;
```

 复制代码

不过如何合理地配置推送是个难题，如果推送给浏览器不需要的资源，反而浪费了带宽。

这方面暂时没有一般性的原则指导，你必须根据自己网站的实际情况去“猜测”客户端最需要的数据。

优化方面，HTTPS 的一些策略依然适用，比如精简密码套件、ECC 证书、会话复用、HSTS 减少重定向跳转等等。

但还有一些优化手段在 HTTP/2 里是不适用的，而且还会有反效果，比如说常见的精灵图（Spriting）、资源内联（inlining）、域名分片（Sharding）等，至于原因是什么，我把它留给你自己去思考（提示，与缓存有关）。

还要注意一点，HTTP/2 默认启用 header 压缩（HPACK），但并没有默认启用 body 压缩，所以不要忘了在 Nginx 配置文件里加上“gzip”指令，压缩 HTML、JS 等文本数据。

## 应用层协议协商（ALPN）

最后说一下 HTTP/2 的“服务发现”吧。

 领资料





你有没有想过，在 URI 里用的都是 HTTPS 协议名，没有版本标记，浏览器怎么知道服务器支持 HTTP/2 呢？为什么上来就能用 HTTP/2，而不是用 HTTP/1 通信呢？

答案在 TLS 的扩展里，有一个叫“**ALPN**”（Application Layer Protocol Negotiation）的东西，用来与服务器就 TLS 上跑的应用协议进行“协商”。

客户端在发起“Client Hello”握手的时候，后面会带上一个“ALPN”扩展，里面按照优先顺序列出客户端支持的应用协议。

就像下图这样，最优先的是“h2”，其次是“http/1.1”，以前还有“spdy”，以后还可能会有“h3”。

```
> Extension: session_ticket (len=0)
✓ Extension: application_layer_protocol_negotiation (len=14)
  Type: application_layer_protocol_negotiation (16)
  Length: 14
  ALPN Extension Length: 12
  ✓ ALPN Protocol
    ALPN string length: 2
    ALPN Next Protocol: h2
    ALPN string length: 8
    ALPN Next Protocol: http/1.1
> Extension: status_request (len=5)
```

服务器看到 ALPN 扩展以后就可以从列表里选择一种应用协议，在“Server Hello”里也带上“ALPN”扩展，告诉客户端服务器决定使用的是哪一种。因为我们在 Nginx 配置里使用了 HTTP/2 协议，所以在这里它选择的的就是“h2”。

```
✓ Extension: application_layer_protocol_negotiation (len=5)
  Type: application_layer_protocol_negotiation (16)
  Length: 5
  ALPN Extension Length: 3
  ✓ ALPN Protocol
    ALPN string length: 2
    ALPN Next Protocol: h2
```

领资料



这样在 TLS 握手结束后，客户端和服务端就通过“ALPN”完成了应用层的协议协商，后面就可以使用 HTTP/2 通信了。

## 小结

今天我们讨论了是否应该迁移到 HTTP/2，还有应该如何迁移到 HTTP/2。

1. HTTP/2 完全兼容 HTTP/1，是“更安全的 HTTP、更快的 HTTPS”，头部压缩、多路复用等技术可以充分利用带宽，降低延迟，从而大幅度提高上网体验；
2. TCP 协议存在“队头阻塞”，所以 HTTP/2 在弱网或者移动网络下的性能表现会不如 HTTP/1；
3. 迁移到 HTTP/2 肯定会有性能提升，但高流量网站效果会更显著；
4. 如果已经升级到了 HTTPS，那么再升级到 HTTP/2 会很简单；
5. TLS 协议提供“ALPN”扩展，让客户端和服务端协商使用的应用层协议，“发现”HTTP/2 服务。

## 课下作业

1. 和“安全篇”的第 29 讲类似，结合自己的实际情况，分析一下是否应该迁移到 HTTP/2，有没有难点？
2. 精灵图（Spriting）、资源内联（inlining）、域名分片（Sharding）这些手段为什么会对 HTTP/2 的性能优化造成反效果呢？

欢迎你把自己的学习体会写在留言区，与我和其他同学一起讨论。如果你觉得有所收获，也欢迎把文章分享给你的朋友。



领资料



## 课外小贴士

01 Nginx 也支持明文形式的 HTTP/2 (即 “h2c”),

在配置“listen”指令时不添加“ssl”参数即可，但无法使用 Chrome 等浏览器直接测试，因为浏览器只支持“h2”。


- 02 HTTP/2 的优先级只使用一个字节，优先级最低是 0，最高是 255，一些过时的书刊和网上资料中把 HTTP/2 的优先级写成了  $2^{31}$ ，是非常错误的。
- 03 ALPN 的前身是 Google 的 NPN (Next Protocol Negotiation)，它与 ALPN 的协商过程刚好相反，服务器提供支持的协议列表，由客户端决定最终使用的协议。
- 04 明文的 HTTP/2 (“h2c”) 不使用 TLS，也就无法使用 ALPN 进行“协议协商”，所以需要使⽤头字段“Connection: Upgrade”升级到 HTTP/2，服务器返回状态码 101 切换协议。
- 05 目前国内已经有不少大网站迁移到了 HTTP/2，比如 [www.qq.com](http://www.qq.com)、[www.tmall.com](http://www.tmall.com)，你可以用 Chrome 的开发者工具检查它们的 Protocol。



分享给需要的人，Ta订阅超级会员，你将得 50 元

Ta单独购买本课程，你将得 20 元

 生成海报并分享

 赞 11  提建议

© 版权归极客邦科技所有，未经许可不得传播售卖。页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

上一篇 32 | 未来之路：HTTP/3展望

下一篇 34 | Nginx：高性能的Web服务器

## 学习推荐

# JVM + NIO + Spring

各大厂面试题及知识点详解

限时免费 



领资料

## 精选留言 (17)

 写留言



我行我素

2019-08-12

2.因为HTTP/2中使用小颗粒化的资源，优化了缓存，而使用精灵图就相当于传输大文件，但

是大文件会延迟客户端的处理执行，并且缓存失效的开销很昂贵，很少数量的数据更新就会使整个精灵图失效，需要重新下载(http1中使用精灵图是为了减少请求)；

HTTP1中使用内联资源也是为了减少请求，内联资源没有办法独立缓存，破坏了HTTP/2的多路复用和优先级策略；

域名分片在HTTP1中是为了突破浏览器每个域名下同时连接数，但是这在HTTP/2中使用多路复用解决了这个问题，如果使用域名分片反而会限制HTTP2的自由发挥

作者回复: 回答的非常好。



👍 41



Fstar

2019-08-13

课下作业的第二题的个人理解。

问：精灵图（Spriting）、资源内联（inlining）、域名分片（Sharding）这些手段为什么会对 HTTP/2 的性能优化造成反效果呢？

答：主要是缓存和请求速度的原因。使用 HTTP/2 后，请求就可以做到乱序收发、多路复用。

1. 图片就算很多，在 HTTP/2 也可以做到“并发”。使用了精灵图的话，首先文件变大了，在 HTTP/2 中相比分开请求要更慢，而且不利于缓存（比如修改了其中几个图片）。

2. 资源内联，是指将一个资源作为另一个资源的一部分，使二者作为一个整体的资源来请求，比如 HTML 文件里嵌入 base64 的图片，该方案是为了减少 HTTP/1 下的请求数，加快网页响应时间。HTTP/2 不存在网页加载变慢的情况，而且不内联的话，能更好地发挥缓存的优势（比如图片是固定的，但 HTML 是动态的）。

3. 域名分片。这个不是很懂，老师在本文也说到：“HTTP/2 对一个域名只开一个连接，所以一旦这个连接出问题，那么整个网站的体验也就变差了”。这么说来，域名分片建立多个连接，貌似就可以解决这个问题？如果不考虑这个问题的话，因为多路复用的原因，就算多了一个连接，也只是变成了两个多路复用，并没有提高多少效率，倒不是很有必要，而且代码实现也会比较麻烦。

作者回复: 回答的很好。

对于第三点，域名分片对于http/2会增加连接成本、HPack字典、慢启动等多个不利因素，所以应该少用。

最后的综合篇还会再讲一下。

领资料





12



、景

2019-08-13

老师这么理解有错吗？

http1 最好把多个请求合成一个请求（比如精灵图，资源内联，域名分片等），原因是 http1 存在 http 的队头阻塞，每次发送新的请求都又需要带上没有压缩的请求头，DNS及时有缓存也需要查找，而且有时候会被清空。http2 某些情况比如精灵图，资源内联就不需要合成一个请求，因为 http2 是基于流和帧的，没有了 http1 的队头阻塞，可以并发多个请求，而且 http2 的请求头使用了 hpage 压缩算法，下一次请求时请求头的长度会非常短（比如 65 一个数字就能表示以前的一长串 UA）。而且还加入了服务器推送，服务器会预先把可能需要的资源先推送给你。如果还是使用一个请求，可能会因为只更新一点点资源而更新整个缓存（比如更新精灵图或资源内联其中的一小部分）。

作者回复：基本正确。

http/1的问题一个是队头阻塞，另一个是报文头数据冗余，多个小请求会浪费带宽，所以资源合并、内联就很有必要。



8



许童童

2019-08-12

分析一下是否应该迁移到 HTTP/2，有没有难点？

从慢、贵、难三个角度来分析

速度快，免费，部署简单，具体分析过程就不写了，我觉得应该立即迁移到 HTTP/2。

精灵图（Spriting）、资源内联（inlining）、域名分片（Sharding）这些手段为什么会对 HTTP/2 的性能优化造成反效果呢？

精灵图、资源内联可以减少HTTP请求数但加大单个请求的大小、域名分片为了突破与同一域名最多建立6个连接的限制，HTTP/2使用流并发请求响应多个资源，完全不需要此优化，相反还会多建立TCP连接浪费资源。

作者回复：迁移到http/2还需要考虑运维、部署等成本，技术上的难度倒是不大。



4

领资料



阿锋

2019-08-12

HTTP/1 里实现了长连接，为啥还会对一个域名开 6~8 个连接，是不是http为了解决http自身

的对头阻塞，不要让http等待完响应之后，才发出下一个请求，而打开多个连接？这些打开了的连接会得到重复利用，就是一段时间内不会关闭？这样才会比HTTP（短连接）高效？可以解析一下内联资源，域名分片，什么意思？

作者回复:

1.可以参考一下第17讲，里面解释了域名分片。

2.资源内联，就是把比较小的图片、js等资源编码成base64，以文本的形式嵌入进html，这样通过一次请求就可以下载到更多的数据。



👍 3



看，有只猪

2019-10-28

老师，我还有一个问题，既然通过ALPN协商了通信采用的协议，那建立好TLS连接后，后续操作直接采用HTTP2即可，为什么还需要发送Magic呢

作者回复: 是的，这个magic的确是显得有点“多余”，但协议标准里就是这么规定的，只能遵守。

在rfc里，对这个的说明是对协议的最终确认，相当于是一个简单的校验机制，也可以理解成是正式收发数据前的“握手”，确认是http/2而不是其他的协议。



👍 2



明月

2020-04-08

精灵图、资源内联在http2效果不好 终极原因也还是http2只能建立一个连接 这样一个大文件传输会占用这唯一的连接 缓存失效后又要更新占用这个连接更新资源 我理解的对吗

作者回复: 不完全对。http/2传输大文件不是问题，问题在于这一个大文件里包含了很多的小文件，缓存失效后重传会浪费带宽，不划算。

这个与连接没有关系，http/2在一个连接上可以开多个流，不是独占传输数据的。

共 2 条评论 >

👍 1



看，有只猪

2019-10-27

老师，请问我对服务发现的理解对吗？

针对加密版本的HTTP/2，客户端需要`Client Hello`的`ALPN`中添加支持的协议，由服务器选择，服务器也通过`ALPN`告知客户端后续通信采用的协议。

领资料



非加密版本的HTTP/1中，客户端建立TCP连接后，将发送Magic。服务器识别到后，后续通信采用HTTP/2。

作者回复: 加密版本你理解的是正确的。

对于非加密版本，http/2要求使用upgrade机制升级到http/2，步骤比alpn要复杂一些，可以参考rfc。



1



Luke

2019-09-03

TCP 协议存在“队头阻塞”，所以 HTTP/2 在弱网或者移动网络下的性能表现会不如 HTTP/1；

老师，HTTP/1，HTTP/2都是基于TCP协议的，在同样“队头阻塞”的情况下，为什么HTTP/1的性能要优于HTTP/2？

作者回复: http/1开多个连接，而http/2只有一个连接，弱网下多个连接显然要比一个连接的传输效果好。

共 2 条评论 >

1



小二

2019-08-23

有一点我不明白，http2只有一条连接，那在多线程并发时，应该需要锁吧，也就是同一瞬间只能一个写入，多几条连接，不是能并发吗？

还是说网卡层同一瞬间也只能处理一个写的请求，所以并不并发都没关系。不过在tcp层面上的队头阻塞还是有关系的吧，多开一个连接，就不会都阻塞住了

如果有多个网卡呢？

作者回复:

1.在http/2协议里没有操作系统线程的概念，应用发送数据需要自己处理socket的并发读写，但数据写入socket后就会由http/2自己分帧管理，实现多路复用。

2.http/2用一个连接能够实现最佳性能，可以省去连接、慢启动等成本，当然因为tcp层有队头阻塞，所以在网络质量差时会有性能损失。

3.多个网卡就是多个ip地址，就可以开多个连接，但每个http/2连接就会重建hpack动态表、慢启动，效率会低一点。

领资料







1

**fmouse**

2022-01-22

老师好，对于 HTTP/2 的“服务发现”用 TLS 的扩展里“ALPN”。

在《30 | 时代之风（上）：HTTP/2特性概览》提到“在 HTTP/2 标准制定的时候（2015 年）已经发现了很多 SSL/TLS 的弱点，而新的 TLS1.3 还未发布。”

TLS1.2+的时候已经支持扩展特性了是吗。

作者回复: 是的，可以去看rfc。

**路漫漫**

2021-11-25

老师，http2 多开几个连接，岂不是很厉害吗，为啥要限制一个，之前还6-8个呢？

作者回复: 因为一个http&#47;2连接的传输效率已经很高了，再多的连接反而会浪费资源，虽然理论上说会增加传输能力，但综合性价比不高。

**Unknown element**

2021-08-13

老师课外小贴士第四条connection: upgrade应该是明文的http 1携带的字段吧 升级到HTTP2 之前客户端和服务端不应该是通过HTTP1通信吗

作者回复: 是的，明文的http/2就只能从http/1升级。

**hao**

2021-06-05

精灵图（Spriting）、资源内联（inlining）、域名分片（Sharding）这些手段为什么会对HTTP/2的性能优化造成反效果呢？

**领资料**

– 精灵图（Spriting）和资源内联（inlining）减少了请求数但增加了每次请求的报文大小，但不利于缓存（精灵图中某个图标改动就要重新请求整个精灵图，资源内联同理）

> 资源内联：内嵌css、js等资源



- 对于HTTP/1来说，因为它存在`队头阻塞`，所以将多个小的`资源合并`可以有效的缓解这种情况的出现
- 但对于HTTP/2来说，它已经引入了流的概念实现了基于单个TCP连接的多路复用，也就是说解决了`HTTP的队头阻塞`，它不需要通过资源合并来缓解
- 域名分片是指利用多个域名和同一个IP地址建立TCP连接，巧妙地避开了浏览器对并发连接数的限制

- 对于HTTP/1来说，因为它没有`多路复用`，所以这样能很好的缓解因为`丢包重发`而导致的`队头阻塞`
- 但对于HTTP/2来说，多建立的TCP连接完全是浪费资源（两端的静态表和动态表，TCP连接的成本等）

作者回复: great。



钱

2020-04-04

迁移到HTTP/2感觉成本小收益高，好像没有什么理由不迁移一下，除了弱网环境下性能差一些，是否还有什么不利影响呢？或者不想改变也是一种原因，谁知道新东西是否有什么坑呢？

作者回复: 协议本身很好，但相关的运维、测试、兼容成本必须要考虑。



keep it simple

2020-02-28

看上去迁移到HTTP/2完全不需要改应用代码呀，只需要修改nginx配置就行

作者回复: 是的，http/2对上层应用完全兼容，没有迁移成本，可以立即获得性能提升。

当然了，你还是要做一些测试，业务逻辑之外可能还有很多事情要做。



领资料



红军

2019-08-12

如果浏览器比较老，只支持http1，访问http2服务器可以工作吗

作者回复: 当然可以，服务器会根据alpn告诉客户端使用http/1。

