使用 http/2
提升性能的7个建议

前言

> 原文出处： https://www.nginx.com/blog/7-tips-for-faster-http2-performance/

历史悠久的超文本传输协议，即HTTP标准，最近版本升级了。HTTP/2在2015年5月被批准，目前已经在很多Web浏览器和服务器中得到实现（包括NGINX Plus和开源NGINX）。大约有三分之二的浏览器已经支持HTTP/2，而且这个比例每月都在增加。

HTTP/2构建在Google SPDY协议基础之上，Chrome将在2016年年初停止对后者的支持。NGINX是最早支持SPDY的，如今同样率先支持了HTTP/2。为此，我们还发布了详尽的白皮书（PDF），介绍了HTTP/2以及它如何基于SPDY构建，并展示了如何实现这个新协议。

HTTP/2的重要特性完全源自SPDY。

- HTTP/2是二进制（而文本）协议，因此更简洁高效；
- 它针对每个域只使用一个多路复用的连接，而不是每个文件一个连接；
- 首部使用特制的HPACK协议（而非SPDY中使用的gzip）压缩；
- HTTP/2设计了复杂的优先级排定规则，帮助浏览器首先请求最急需的文件，而NGINX已经支持（SPDY的方案要简单一些）。

现在，你需要决定是否迁移到HTTP/2，而其中关键是知道如何最大限度地利用它。这篇文章会带你了解从性能角度考虑为什么要做这个决定，以及如何实现。接下来我们要逐一讨论关于HTTP/2性能的7个小建议。

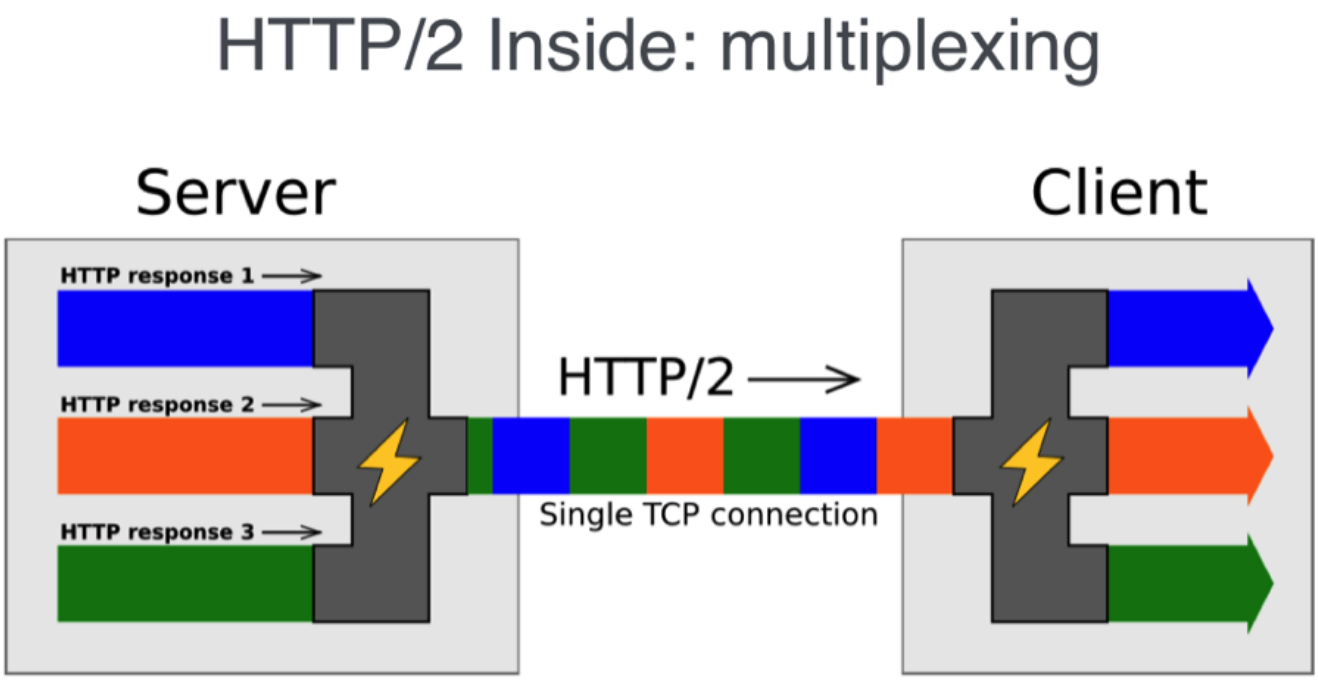
建议一：现在是否需要迁移到HTTP/2

实现HTTP/2很简单，看看我们的白皮书就明白了（PDF）。不过，HTTP/2并不是万能的银弹，它只对某些Web应用有用，对另外一些则没那么有用。

如果你使用SSL/TLS（以后简称TLS），那么HTTP/2可以提升网站性能。如果你没有，那在使用HTTP/2之前要先支持TLS。这时候，使用TLS的性能损耗大致可以被使用HTTP/2的性能提升抵销。不过还是建议你在实际应用之前先测试一下。

HTTP/2有五大优势。

1. 每个服务器只用一个连接。HTTP/2对每个服务器只使用一个连接，而不是每个文件一个连接。这样，就省掉了多次建立连接的时间，这个时间对TLS尤其明显，因为TLS连接费时间。
2. 加速TLS交付。HTTP/2只需一次耗时的TLS握手，并且通过一个连接上的多路利用实现最佳性能。HTTP/2还会压缩首部数据，省掉HTTP/1.x时代所需的一些优化工作，比如拼接文件，从而提高缓存利用率。
3. 简化Web应用。使用HTTP/2可以让Web开发者省很多事，因为不用再做那些针对HTTP/1.x的优化工作了。
4. 适合内容混杂的页面。HTTP/2特别适合混合了HTML、CSS、JavaScript、图片和有限多媒体的传统页面。浏览器可以优先安排那些重要的文件请求，让页面的关键部分先出现，快出现。
5. 更安全。通过减少TLS的性能损失，可以让更多应用使用TLS，从而让用户信息更安全。



HTTP/2的多路复用示意图

相应地，HTTP/2也有五个不足之处。

1. 单连接开销比较大。HPACK数据压缩算法会更新两端的查找表。这样可以让连接有状态，而破坏状态

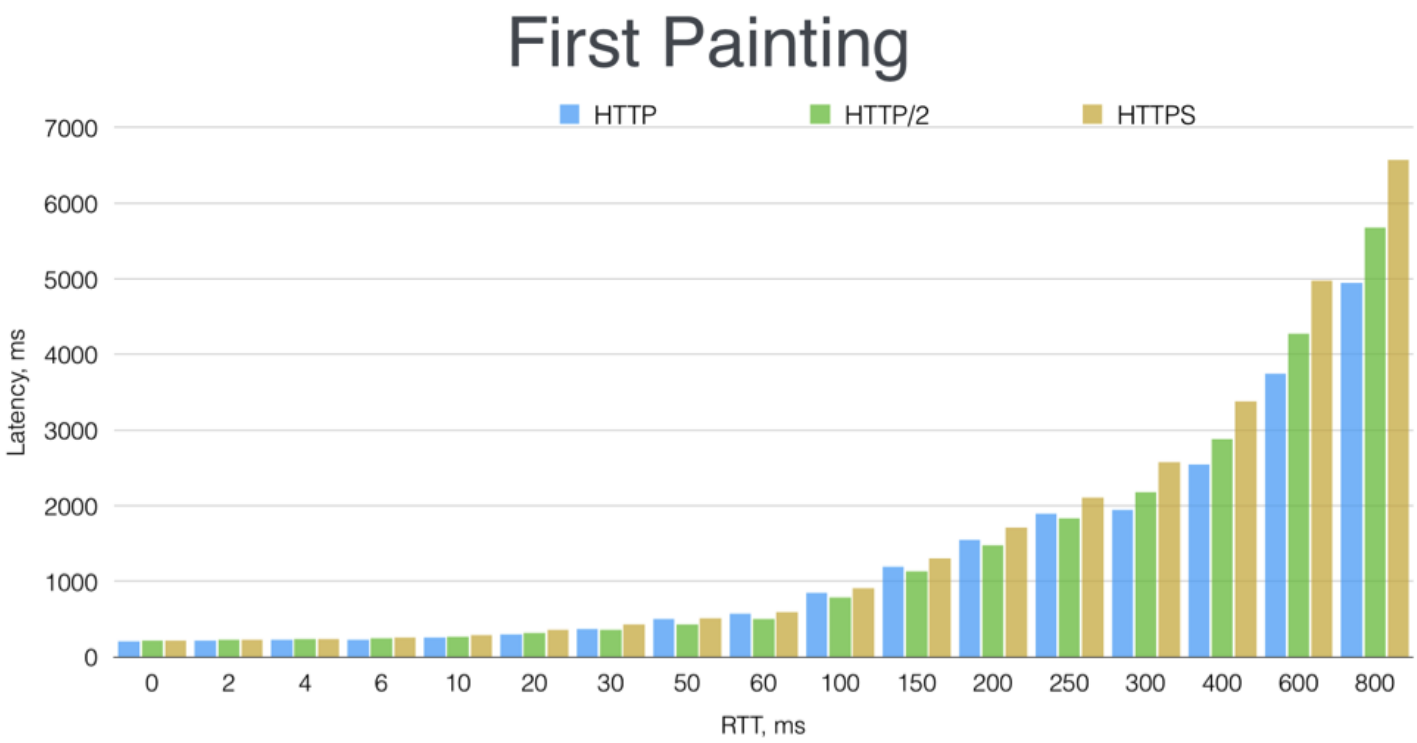
就意味着要重建查找表，另外单连接占用内存较多。

2. 你可能不需要SSL。如果你的数据不需要保护，或者已经使用DRM或其他编码进行保护了，那么TLS的安全性对你可能无所谓。
3. 需要抛弃针对HTTP/1.x的优化。HTTP/1.x优化在支持HTTP/2的浏览器中会影响性能，因此可能需要花时间把它们推倒重来。
4. 对下载大文件不利。如果你的应用主要提供大文件下载或者流媒体播放，那可能不想用TLS，而且在只有一个流的情况下，多路复用也体现不出什么优势。
5. 你的客户也许不在乎。你的客户很可能不在乎他分享的自家猫咪的视频是否受到TLS和HTTP/2的保护。

总之，一切要看性能。这方面，有好消息也有坏消息。

好消息是我们在内部对NGINX做过测试，结果从理论上能够得到印证：对于要通过典型网络延迟请求的混合内容网页，HTTP/2的性能好于HTTP/1.x和HTTPS。基于连接的RTT，结果可以分三种情况。

- 很低的RTT（0-20ms）：HTTP/1.x、HTTP/2和HTTPS基本无差别。
- 典型网络RTT（30-250ms）：HTTP/2比HTTP/1.x快，而且它们都比HTTPS快。美国两个相邻城市间的RTT约为30 ms，而东西海岸间（约3000英里）则约为70 ms。东京到伦敦间最短路径的RTT大约240 ms。
- 高RTT（300ms及以上）：HTTP/1.x比HTTP/2快，后者又比HTTPS快。



这张图显示了首次渲染的时间，也就是用户第一次在自己屏幕上看到网页内容的时间。这个时间一般认为关系到用户对网站响应速度的感知。

要想了解我们测试的更多内容，请看这个HTTP/2的介绍视频，来源是nginx.conf 2015。

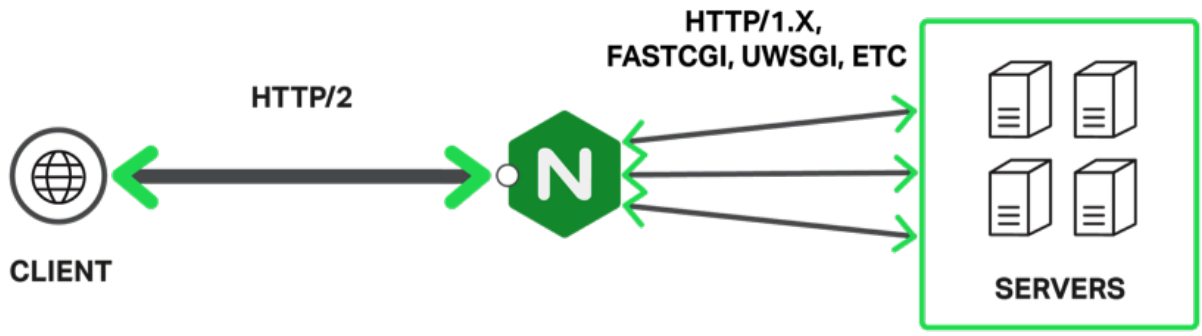
然而，每个网页都不相同，实际上每个用户的会话也不一样。如果你托管流媒体或提供大文件下载，那你

的决定可能不一样，甚至相反。

你最终可能发现投入产出比并不明显。如果是这样，那你得多学习一下，针对自己的内容多做一些测试，然后咱们可以聊一聊。（想找点资料？可以看看NGINX网络研讨：What’ s New in HTTP/2? ）。

建议二：终止HTTP/2和TLS

终止协议意味着客户端使用期望的协议连接代理服务器，比如TLS或HTTP/2，然后代理服务器再去连接应用服务器、数据库服务器等，但不需要使用相同的协议，如下图所示。



使用独立的服务器终止协议意味着使用多服务器架构。多服务器可能是多个物理服务器、多个虚拟服务器，或者AWS这样的云环境中的多个虚拟服务器实例。多服务器就比单服务器复杂，或者比应用服务器/数据库服务器的组合复杂。不过，多服务器架构有很多好处，而且很多流量大的网站也必须用这种架构。

配置了服务器或者虚拟服务器之后，很多事情都成为可能。新服务器可以分担其他服务器的负载，可用于负载平衡、静态文件缓存和其他用途。另外，也可以让添加和替换应用服务器或其他服务器更容易。

NGINX和NGINX Plus经常被用来终止TLS和HTTP/2协议、负载平衡。已有环境不必改动，除非要把NGINX服务器挪到前端。

建议三：考虑从SPDY开始

SPDY是HTTP/2的上一代，总体性能相同。因为它已经出现好几年了，所以有很多浏览器支持SPDY却不支持HTTP/2。不过，在本文写作时，这个支持上的差距正在缩小。具体来说，有三分之二的浏览器支持HTTP/2，而有五分之四的浏览器支持SPDY。

如果你着急采用新的Web传输协议，又想尽可能覆盖更多用户，可以先从SPDY开始。然后到2016年初，即谷歌不再支持SPDY的时候，再切换到HTTP/2，很简单，至少在NGINX中如此。那时候，更多用户会拥有支持HTTP/2的浏览器，而你已经为其中大部分用户提供了很好的性能。

建议四：找出为HTTP/1.x优化的代码

在决定采用HTTP/2之前，首先得知道你的代码有哪些是针对HTTP/1.x优化过的。大概有四方面的优化。

1. 分域存储。为了实现并行请求文件，你可能把文件分散到了不同的域里，CDN会自动这么做。但分域存储会影响HTTP/2的性能，建议使用HTTP/2友好的分域存储（建议七），只针对HTTP/1.x用户分域。
2. 雪碧图。雪碧图把很多图片拼成一个文件，然后通过代码按需取得每个图片。雪碧图在HTTP/2的环境下没太大用处，但还是有点用的。
3. 拼接的代码文件。与使用雪碧图的原因类似，很多独立的文件也会被弄成一个，然后浏览器再从其中找到并运行需要的文件。
4. 插入行内的文件。CSS代码、JavaScript代码，甚至图片等被直接插到HTML文件中的内容。这样可以减少文件传输，代价是初始HTML文件较大。

后面三种优化都涉及把小文件塞进一个较大的文件里，目的是减少新建连接的初始化和握手，这些操作对TLS而言非常费时间。

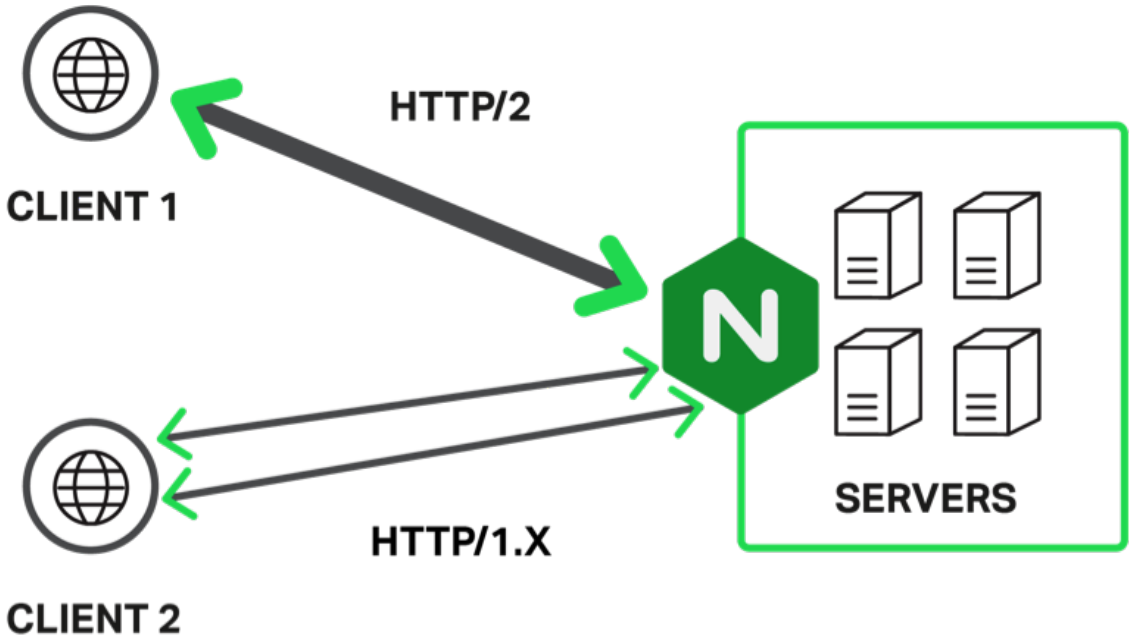
第一种优化即分域存储恰恰相反，强制打开多个连接，目的是并行地从不同的域获取文件。这两种看似矛盾的技术对于HTTP/1.x下的站点却十分有效。然而，要用好这两种技术，必须投入大量时间、精力和资源，用于实现、管理和运维。

在采用HTTP/2之前，需要找出应用了这些优化的代码，分析一下它们会不会影响你的应用设计和工作流程。这样在迁移到HTTP/2之后，就可以着手改造它们，甚至撤销某些优化。

建议五：部署HTTP/2或SPDY

事实上，部署HTTP/2或SPDY并不难。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，参见这里了解如何启用HTTP/2（PDF）。浏览器和服务器会协商采用什么协议，如果浏览器支持HTTP/2（而且也在使用TLS），就会使用HTTP/2。

配置完服务器后，使用支持HTTP/2浏览器的用户就会基于HTTP/2运行你的应用，而使用旧版本浏览器的用户则会继续使用HTTP/1.x运行你的应用，如下图所示。如果你的网站流量非常大，那么应该监测改变前后的性能，对于性能降低的情况，可能就得撤销更改。



注意：使用HTTP/2及其单连接之后，NGINX某些配置的重要性会很明显，特别要注意的是 output_buffers 、 proxy_buffers 和 ssl_buffer_size 等指令，多测试一下。参见general configuration notes，特定的SSL建议（在这里 and here），以及NGINX关于SSL性能的白皮书（PDF）。

注意：使用HTTP/2传输密文要格外注意。HTTP/2的RFC中有一个长长的列表，列出了要避免的加密套件。建议你自己也搞一个表格，启用 ssl_buffer_size ，然后在所有常用的浏览器版本下测试你想用的加密套件。

建议六：再谈HTTP/1.x优化

你说奇怪不，撤销和修改针对HTTP/1.x优化的代码居然是实现HTTP/2最有创意的部分。这里面有几个问题要注意，因为很多事怎么做都是可以的。

在开始运作之前，必须考虑旧版本浏览器用户是否好过。之后，可以采取三个策略撤销和修改HTTP/1.x的优化。

- 什么也不用做。假如你并没有针对HTTP/1.x做过优化，或者只做过少量优化，那么你几乎什么也不用做，就可以直接迁移到HTTP/2。
- 有选择地去做。第二种情况是减少合并某些文件，而不是完全不合并。比如，牵扯到很多场景的雪碧图就不用动，而被塞得满满的HTML可能就要分离出来一些。
- 完全撤销HTTP/1.x优化（不过请先参考建议七中关于分域存储的建议）。可以不再做以前做过的任何优化。

缓存还是普适的。理论上，缓存操作非常适合小文件特别多的情况。但是，小文件多也意味着文件I/O多。因此一些相近文件的合并还是必要的，一方面要考虑工作流程，另一方面要考虑应用性能。建议多关注一下其他人在过渡到HTTP/2过程中的一些经验。

建议七：实现智能分域

分域存储可能是最极端但也最成功的HTTP/1.x优化策略。它能够提升HTTP/1.x下的应用性能，但在HTTP/2之下，其性能提升可以忽略不讲（因为只有一个连接。）

对HTTP/2友好的分域，要保证以下两点。

- 让多个域名解析到同一个IP。
- 确保证书包含通配符，以便所有分域名都可以使用，适当的多域证书当然也可以。

具体细节，请参考这里。

有了这些保障，分域还会继续对HTTP/1.x有效，即域名仍然可以触发浏览器创建更多连接，但对HTTP/2则无效，因为这些域名会被看成同一个域，一个连接就可以访问所有域名了。

小结

HTTP/2和TLS组合可以提升你的站点性能，并且让用户觉得你的网站很安全。无论你是率先在自己的应用里实现HTTP/2，还是要赶超竞争对手，都可以又快又好地实现对HTTP/2的支持。

希望这篇文章能让你以最少的努力获得最大的HTTP/2性能收益，而且从此你可以把注意力集中到编写更快、更有效、更安全的应用上，让自己的应用更容易维护和运维。

参考资源

- 要全面了解HTTP/2，可以看看NGINX的白皮书（PDF）。
- 在*Can I use*网站中可以查到浏览器对各种前端技术的支持情况，包括SPDY和HTTP/2。
- 要了解我们测试的细节，参考这里HTTP/2 presentation。
- NGINX有一个Web研讨班：What’ s New in HTTP/2?，讨论了核心特性并给出了实现建议。
- 要了解NGINX关于性能的建议，请参考我们的博客：10 Tips for 10x Application Performance。

扫码关注w3ctech微信公众号