

# 17 | 排队也要讲效率：HTTP的连接管理

2019-07-05 Chrono

《透视HTTP协议》

课程介绍 >



讲述：Chrono

时长 10:23 大小 14.27M



在🔗第 14 讲里，我曾经提到过 HTTP 的性能问题，用了六个字来概括：“不算差，不够好”。同时，我也谈到了“队头阻塞”，但由于时间的限制没有展开来细讲，这次就来好好地看看 HTTP 在连接这方面的表现。

HTTP 的连接管理也算得上是个“老生常谈”的话题了，你一定曾经听说过“短连接”“长连接”之类的名词，今天让我们一起来把它们弄清楚。

## 短连接

HTTP 协议最初 (0.9/1.0) 是个非常简单的协议，通信过程也采用了简单的“请求 - 应答”方式。

它底层的数据传输基于 TCP/IP，每次发送请求前需要先与服务器建立连接，收到响应报文后会立即关闭连接。

领资料



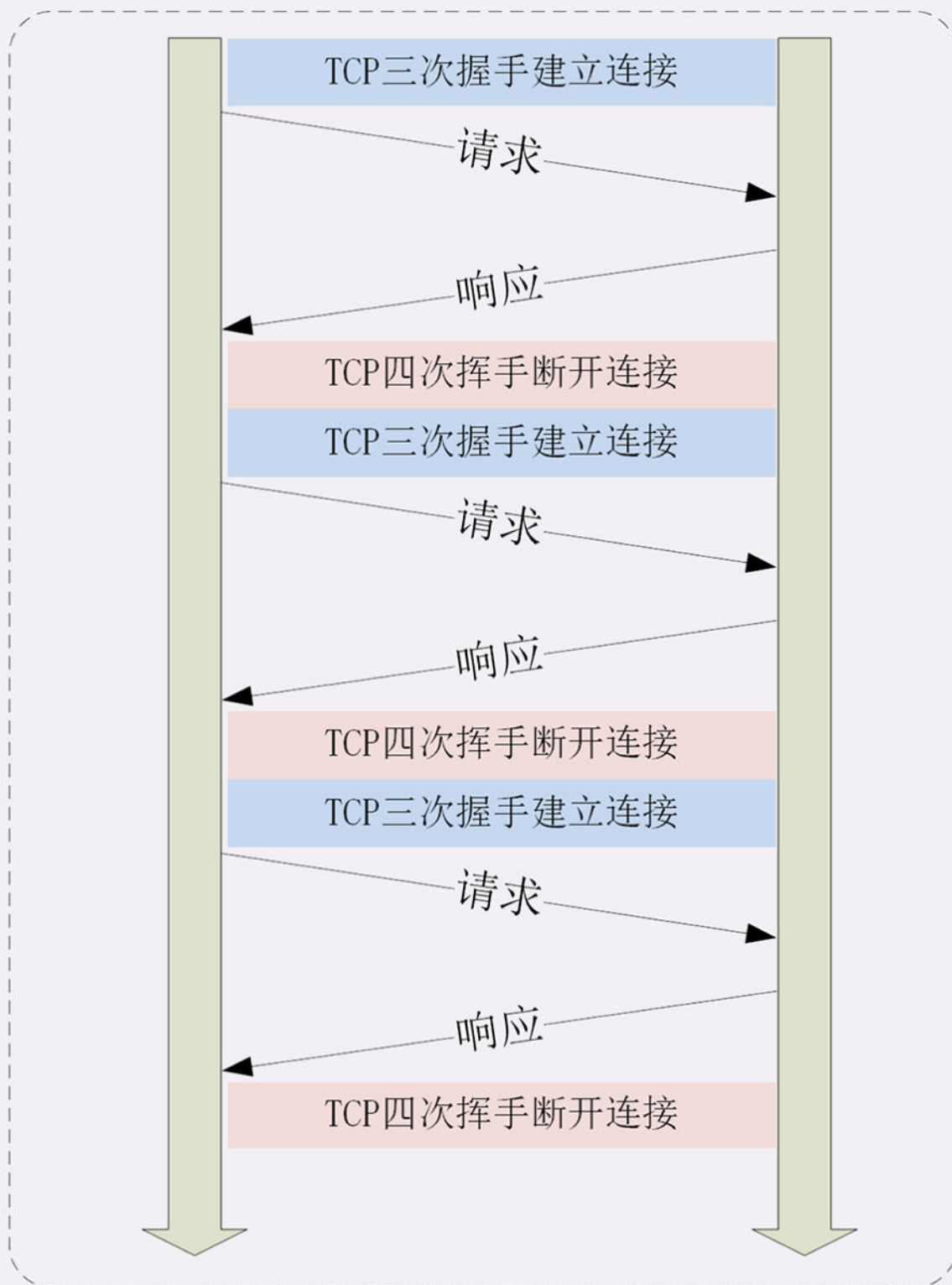
因为客户端与服务器的整个连接过程很短暂，不会与服务器保持长时间的连接状态，所以就被称为“**短连接**”（short-lived connections）。早期的 HTTP 协议也被称为是“**无连接**”的协议。

短连接的缺点相当严重，因为在 TCP 协议里，建立连接和关闭连接都是非常“昂贵”的操作。TCP 建立连接要有“三次握手”，发送 3 个数据包，需要 1 个 RTT；关闭连接是“四次挥手”，4 个数据包需要 2 个 RTT。

而 HTTP 的一次简单“请求 – 响应”通常只需要 4 个包，如果不算服务器内部的处理时间，最多是 2 个 RTT。这么算下来，浪费的时间就是“ $3 \div 5 = 60\%$ ”，有三分之二的时间被浪费掉了，传输效率低得惊人。

领资料





单纯地从理论上讲，TCP 协议你可能还不太好理解，我就拿打卡考勤机来做个形象的比喻吧。

假设你的公司买了一台打卡机，放在前台，因为这台机器比较贵，所以专门做了一个保护罩盖着它，公司要求每次上下班打卡时都要先打开盖子，打卡后再盖上盖子。

可是偏偏这个盖子非常牢固，打开关闭要费很大力气，打卡可能只要 1 秒钟，而开关盖子却需要四五秒钟，大部分时间都浪费在了毫无意义的开关盖子操作上了。

可想而知，平常还好说，一到上下班的点在打卡机前就会排起长队，每个人都要重复“开盖 – 打卡 – 关盖”的三个步骤，你说着急不着急。

在这个比喻里，打卡机就相当于服务器，盖子的开关就是 TCP 的连接与关闭，而每个打卡的人就是 HTTP 请求，很显然，短连接的缺点严重制约了服务器的服务能力，导致它无法处理更多的请求。

## 长连接

针对短连接暴露出的缺点，HTTP 协议就提出了“**长连接**”的通信方式，也叫“持久连接”（persistent connections）、“连接保活”（keep alive）、“连接复用”（connection reuse）。

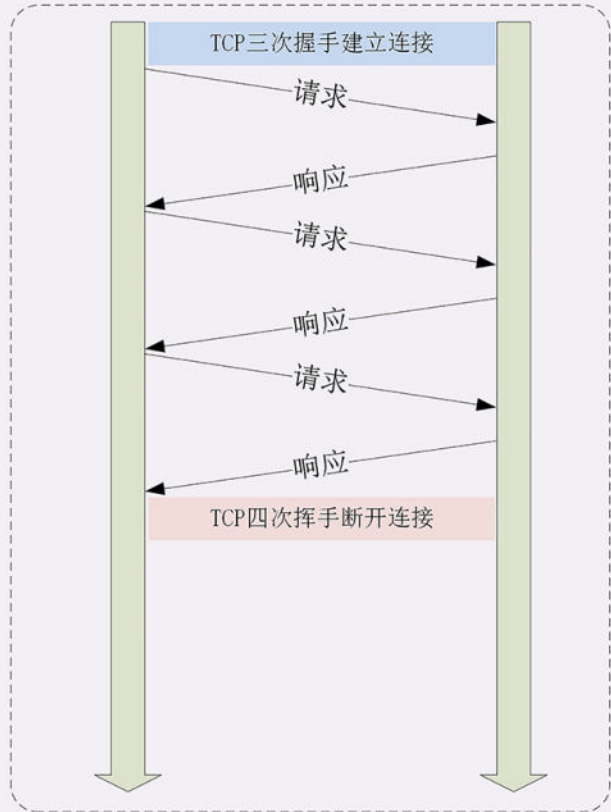
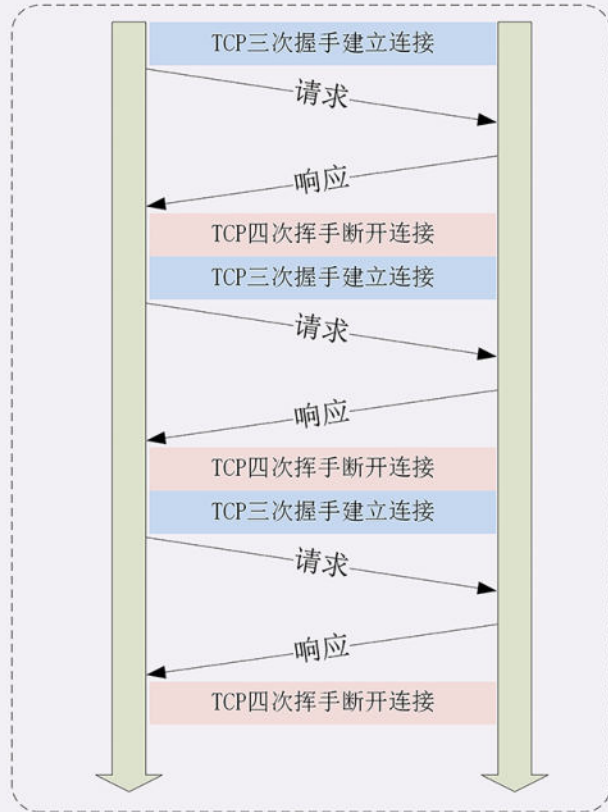
其实解决办法也很简单，用的就是“**成本均摊**”的思路，既然 TCP 的连接和关闭非常耗时间，那么就把这个时间成本由原来的一个“请求 – 应答”均摊到多个“请求 – 应答”上。

这样虽然不能改善 TCP 的连接效率，但基于“**分母效应**”，每个“请求 – 应答”的无效时间就会降低不少，整体传输效率也就提高了。

这里我画了一个短连接与长连接的对比示意图。

领资料





在短连接里发送了三次 HTTP“请求 - 应答”，每次都会浪费 60% 的 RTT 时间。而在长连接的情况下，同样发送三次请求，因为只在第一次时建立连接，在最后一次时关闭连接，所以浪费率就是“ $3 \div 9 \approx 33\%$ ”，降低了差不多一半的时间损耗。显然，如果在这个长连接上发送的请求越多，分母就越大，利用率也就越高。

继续用刚才的打卡机的比喻，公司也觉得这种反复“开盖 - 打卡 - 关盖”的操作太“反人类”了，于是颁布了新规定，早上打开盖子后就不用关上了，可以自由打卡，到下班后再关上盖子。

这样打卡的效率（即服务能力）就大幅度提升了，原来一次打卡需要五六秒钟，现在只要一秒就可以了，上下班时排长队的景象一去不返，大家都开心。

## 连接相关的头字段



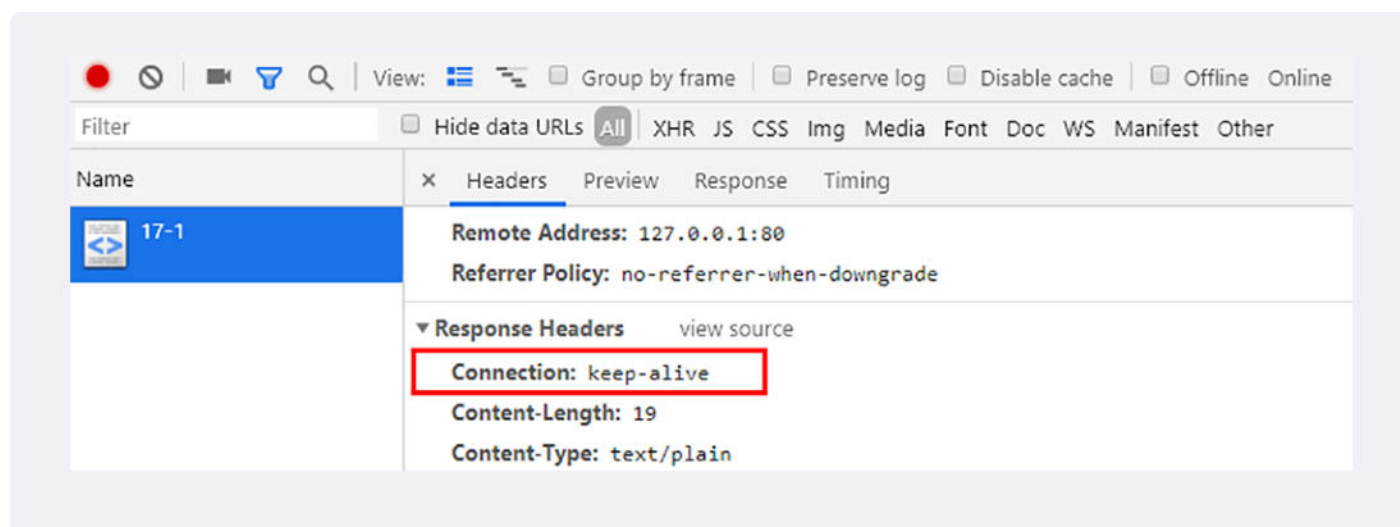


由于长连接对性能的改善效果非常显著，所以在 HTTP/1.1 中的连接都会默认启用长连接。不需要用什么特殊的头字段指定，只要向服务器发送了第一次请求，后续的请求都会重复利用第一次打开的 TCP 连接，也就是长连接，在这个连接上收发数据。

当然，我们也可以在请求头里明确地要求使用长连接机制，使用的字段是 **Connection**，值是“**keep-alive**”。

不过不管客户端是否显式要求长连接，如果服务器支持长连接，它总会在响应报文里放一个“**Connection: keep-alive**”字段，告诉客户端：“我是支持长连接的，接下来就用这个 TCP 一直收发数据吧”。

你可以在实验环境里访问 URI“/17-1”，用 Chrome 看一下服务器返回的响应头：



不过长连接也有一些小缺点，问题就出在它的“长”字上。

因为 TCP 连接长时间不关闭，服务器必须在内存里保存它的状态，这就占用了服务器的资源。如果有大量的空闲长连接只连不发，就会很快耗尽服务器的资源，导致服务器无法为真正有需要的用户提供服务。

所以，长连接也需要在恰当的时间关闭，不能永远保持与服务器的连接，这在客户端或者服务器都可以做到。

在客户端，可以在请求头里加上“**Connection: close**”字段，告诉服务器：“这次通信后就关闭连接”。服务器看到这个字段，就知道客户端要主动关闭连接，于是在响应报文里也加上这个字段，发送之后就调用 Socket API 关闭 TCP 连接。



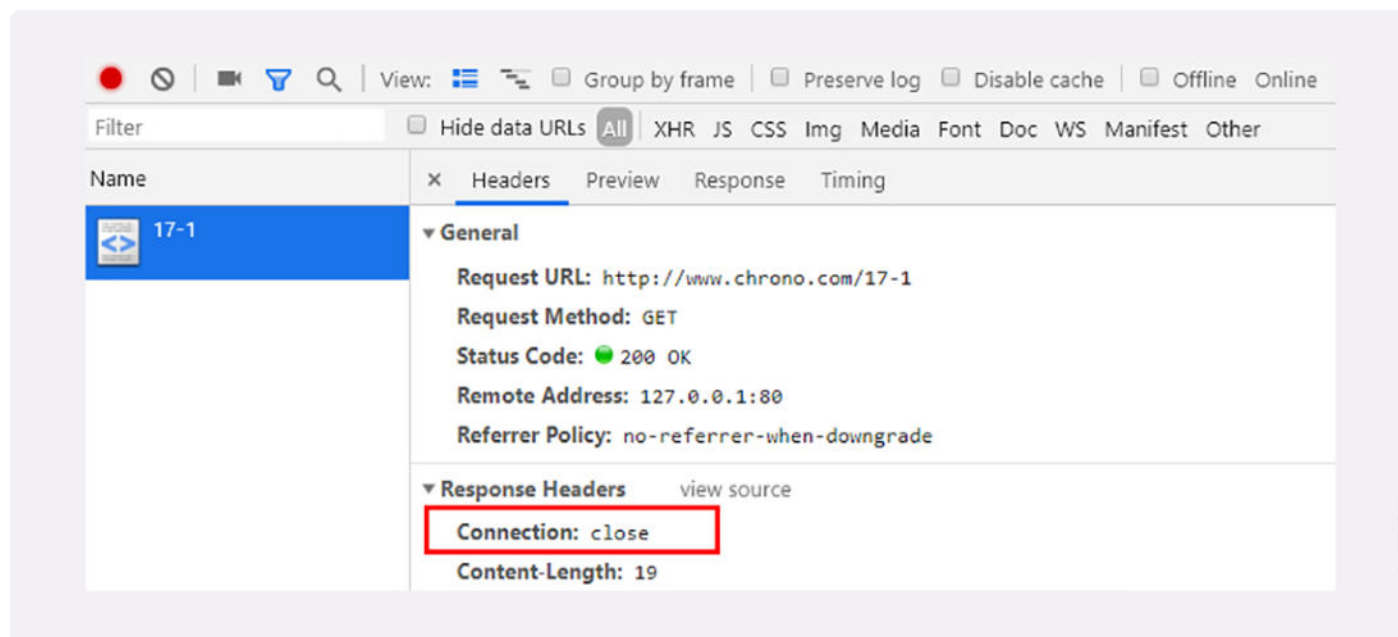
服务器端通常不会主动关闭连接，但也可以使用一些策略。拿 Nginx 来举例，它有两种方式：

1. 使用“keepalive\_timeout”指令，设置长连接的超时时间，如果在一段时间内连接上没有任何数据收发就主动断开连接，避免空闲连接占用系统资源。
2. 使用“keepalive\_requests”指令，设置长连接上可发送的最大请求次数。比如设置成 1000，那么当 Nginx 在这个连接上处理了 1000 个请求后，也会主动断开连接。

另外，客户端和服务端都可以在报文里附加通用头字段“Keep-Alive: timeout=value”，限定长连接的超时时间。但这个字段的约束力并不强，通信的双方可能并不会遵守，所以不太常见。

我们的实验环境配置了“keepalive\_timeout 60”和“keepalive\_requests 5”，意思是空闲连接最多 60 秒，最多发送 5 个请求。所以，如果连续刷新五次页面，就能看到响应头里的“Connection: close”了。

把这个过程用 Wireshark 抓一下包，就能够更清晰地看到整个长连接中的握手、收发数据与挥手过程，在课后你可以再实际操作看看。



## 队头阻塞

看完了短连接和长连接，接下来就要说到著名的“队头阻塞”（Head-of-line blocking，也叫“队首阻塞”）了。

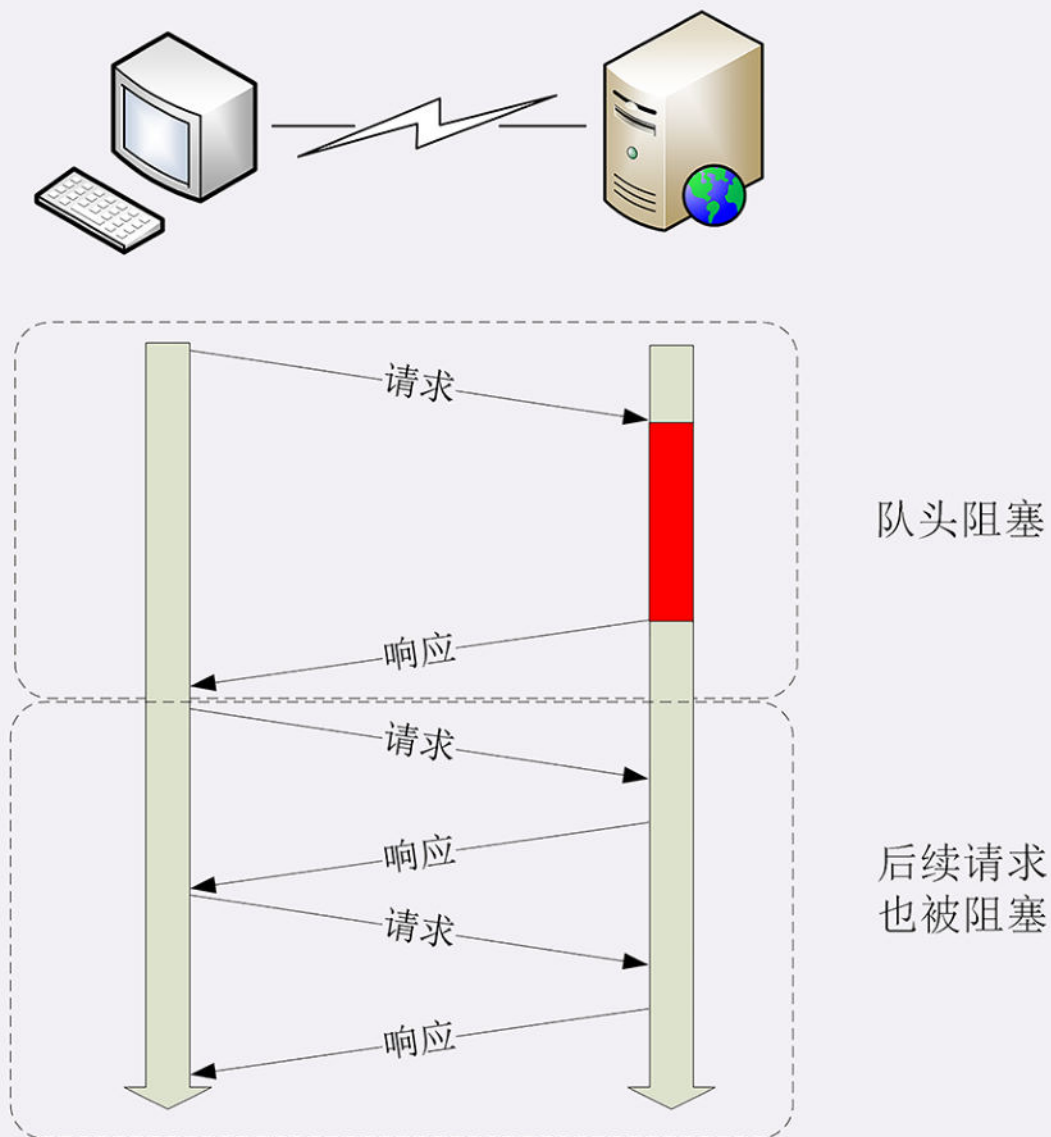
领资料



“队头阻塞”与短连接和长连接无关，而是由 HTTP 基本的“请求 - 应答”模型所导致的。

因为 HTTP 规定报文必须是“一发一收”，这就形成了一个先进先出的“串行”队列。队列里的请求没有轻重缓急的优先级，只有入队的先后顺序，排在最前面的请求被最优先处理。

如果队首的请求因为处理的太慢耽误了时间，那么队列里后面的所有请求也不得不跟着一起等待，结果就是其他的请求承担了不应有的时间成本。



领资料

还是用打卡机做个比喻。

上班的时间点上，大家都在排队打卡，可这个时候偏偏最前面的那个人遇到了打卡机故障，怎么也不能打卡成功，急得满头大汗。等找人把打卡机修好，后面排队的所有人全迟到了。



## 性能优化

因为“请求 – 应答”模型不能变，所以“队头阻塞”问题在 HTTP/1.1 里无法解决，只能缓解，有什么办法呢？

公司里可以再多买几台打卡机放在前台，这样大家可以不用挤在一个队伍里，分散打卡，一个队伍偶尔阻塞也不要紧，可以改换到其他不阻塞的队伍。

这在 HTTP 里就是“**并发连接**”（concurrent connections），也就是同时对一个域名发起多个长连接，用数量来解决质量的问题。

但这种方式也存在缺陷。如果每个客户端都想自己快，建立很多个连接，用户数×并发数就会是个天文数字。服务器的资源根本就扛不住，或者被服务器认为是恶意攻击，反而会造成“拒绝服务”。

所以，HTTP 协议建议客户端使用并发，但不能“滥用”并发。RFC2616 里明确限制每个客户端最多并发 2 个连接。不过实践证明这个数字实在是太小了，众多浏览器都“无视”标准，把这个上限提高到了 6~8。后来修订的 RFC7230 也就“顺水推舟”，取消了这个“2”的限制。

但“并发连接”所压榨出的性能也跟不上高速发展的互联网无止境的需求，还有什么别的办法吗？

公司发展的太快了，员工越来越多，上下班打卡成了迫在眉睫的大问题。前台空间有限，放不下更多的打卡机了，怎么办？那就多开几个打卡的地方，每个楼层、办公区的入口也放上三四台打卡机，把人进一步分流，不要都往前台挤。

这个就是“**域名分片**”（domain sharding）技术，还是用数量来解决质量的思路。

HTTP 协议和浏览器不是限制并发连接数量吗？好，那我就多开几个域名，比如 shard1.chrono.com、shard2.chrono.com，而这些域名都指向同一台服务器 www.chrono.com，这样实际长连接的数量就又上去了，真是“美滋滋”。不过实在是有点“上有政策，下有对策”的味道。

## 小结

这一讲中我们学习了 HTTP 协议里的短连接和长连接，简单小结一下今天的内容：

领资料



1. 早期的 HTTP 协议使用短连接，收到响应后就立即关闭连接，效率很低；
2. HTTP/1.1 默认启用长连接，在一个连接上收发多个请求响应，提高了传输效率；
3. 服务器会发送“Connection: keep-alive”字段表示启用了长连接；
4. 报文头里如果有“Connection: close”就意味着长连接即将关闭；
5. 过多的长连接会占用服务器资源，所以服务器会用一些策略有选择地关闭长连接；
6. “队头阻塞”问题会导致性能下降，可以用“并发连接”和“域名分片”技术缓解。

## 课下作业

1. 在开发基于 HTTP 协议的客户端时应该如何选择使用的连接模式呢？短连接还是长连接？
2. 应当如何降低长连接对服务器的负面影响呢？

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。



## 课外小贴士

01 因为 TCP 协议还有“慢启动”“拥塞窗口”等特性，通常新建立的“冷连接”会比打开了一段时间的“热连接”要慢一些，所以长连接比短连接还多了这一层的优势。

02 在长连接中的一个重要问题是如何正确地区分多个报文的开始和结束，所以最好总使用“Content-Length”头明确响应实体的长度

领资料



Content-Length 来明确响应主体的长度，正确标记报文结束。如果是流式传输，body 长度不能立即确定，就必须用分块传输编码。


03 利用 HTTP 的长连接特性对服务器发起大量请求，导致服务器最终耗尽资源“拒绝服务”，这就是常说的 DDoS。

04 HTTP 的连接管理还有第三种方式 pipeline（管道，或者叫流水线），它在长连接的基础上又进了一步，可以批量发送请求批量接收响应，但因为存在一些问题，Chrome、Firefox 等浏览器都没有实现它，已经被事实上“废弃”了。

05 Connection 字段还有一个取值：“Connection: Upgrade”，配合状态码 101 表示协议升级，例如从 HTTP 切换到 WebSocket。

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

领资料



 赞 27

 提建议

上一篇 16 | 把大象装进冰箱：HTTP传输大文件的方法

下一篇 18 | 四通八达：HTTP的重定向和跳转

## 学习推荐

# JVM + NIO + Spring

## 各大厂面试题及知识点详解

限时免费



## 精选留言 (82)

写留言



TerryGoForIt

2019-07-05

老师您好，我想对于 队首阻塞 的问题，应该从 TCP 层面去解释会比较好一点吧。

> 以下引用自《Web 性能权威指南》

每个 TCP 分组都会带着一个唯一的序列号被发出，而所有分组必须按顺序传送到接收端。如果中途有一个分组没能到达接收端，那么后续分组必须保存到接收端的 TCP 缓冲区，等待丢失的分组重发并到达接收端。这一切都发生在 TCP 层，应用程序对 TCP 重发和缓冲区中排队分组一无所知，必须等待分组全部到达才能访问数据。在此之前，应用程序只能在通过套接字读数据时感觉到延迟交互。这种效应称为 TCP 的队首阻塞。

领资料

作者回复: 队头阻塞在http和tcp层次都有，原因不同。

你说的是tcp队头阻塞，而http的队头阻塞是因为它的请求-应答模式，当然它运行在tcp上，就有两种队头阻塞。



**信信**

2019-07-06

老师能解释下，为什么tcp握手1个rtt，挥手2个rtt吗？

作者回复：一个来回就是1rtt，三次回收准确来说是1.5个rtt，四次挥手是两个来回，所以是2rtt。



44

**qzmone**

2019-08-05

"多开几个域名，比如 shard1.chrono.com、shard2.chrono.com，而这些域名都指向同一台服务器 www.chrono.com " 老师，多开几个域名，最终都是指向一个服务器，那跟都直接连一个服务器的效果一样吧，我感觉对服务器的性能要求一样呀，没有减少后端的压力

作者回复：域名分片解决的是客户端并发的问题，可以创建更多的连接。

比如浏览器限制一个域名最多6个连接，域名分3片，那么浏览器就可以同时建立18个连接，显然就能够并发更多请求，获取数据也就快了。



39

**锦**

2019-07-05

一般使用长连接，除非明确知道只会发送一个请求，比如游戏内连接兑换码服务进行礼包兑换。

- 1，服务器端设置keepalive\_timeout表示多长时间没有数据则关闭连接。
- 2，服务器端设置keepalive\_requests，表示该连接上处理多少个请求后关闭连接。
- 3，服务器端设置最大连接数，当连接达到上限之后拒绝连接，也可以采用限流措施等。
- 4，客户端设置keepalive\_requests，表示该连接上发送多少个连接后关闭连接。
- 5，客户端设置keepalive\_timeout，表示多长时间没有数据发送则关闭连接。
- 6，客户端设置响应超时后重试次数，当次数达到上限后关闭连接。

作者回复：总结的不错。



25

**领资料****披荆斩棘KK**

2019-07-05

老师，请问高并发请求和并发连接有什么关系吗？



负载均衡解决高并发问题是并发连接吗？

作者回复：“高并发请求”是服务器端的概念，意思是同时有多个客户端连接服务器。

课程里的“并发连接”是客户端的概念，意思是一个浏览器并发多个连接，访问服务器。

负载均衡是服务器端的概念，就是把大量的客户端连接均匀分散到集群里的多台服务器。

共 2 条评论 >

👍 23



恒

2019-10-13

老师看下我总结的对不对，谢谢！

短连接：每次“请求-响应”都先建立tcp连接，完后关闭连接。这样三次握手1.5个rtt，“请求-响应”2个rtt(里面有两个ack)，四次挥手2个rtt，效率极低。适用于少次请求，例如客户端只会对某个服务器发送几次请求后就不再发送。

长连接：建立tcp连接后不立即关闭，后续http请求复用这个tcp连接。http/1.1默认开启。如果有大量的空闲长连接只连不发占用资源，可能导致耗尽资源“拒绝服务”即DDoS。因此服务器常会设置超时时间或最大请求数。

这里的“连接”其实是对某个域名的，而不是某个ip或主机。而浏览器对单个域名的并发连接数量有限制，一般为6~8个，所以为了进一步提高连接数就有了“域名分片”技术，即将原来一个域名分成多个域名，但最后指向的服务器还是原来那一台。

例如把www.chrono.com分成shard1.chrono.com, 和 shard2.chrono.com, 但还是指向原来那台服务器。这虽然提高了客户端的并发数，但反而增加了服务器端的压力。

连接相关头字段

Connection: keep-alive

在请求头里表明要求使用长连接，在响应头里表明支持使用长连接。

Connection: close

在请求头里表明告诉服务器这次通信后关闭长连接，在响应头里表明服务器将关闭长连接。

[Connection: Upgrade，配合101状态码表示协议升级，例如从http切换到WebSocket]

作者回复：非常完整详细，good。

共 2 条评论 >

👍 16



鸟人

2019-07-29

拒绝服务应该是dos ddos是分布式dos

领资料



作者回复: 感谢指正。



12



小炭

2019-11-15

浪费的时间就是“ $3 \div 5 = 60\%$ ”这个算法不是很理解，分子/分母是怎么来的

作者回复: tcp握手是1个rtt，挥手是2个rtt，这就是3个没有数据传输的时间，实际的数据传输是2个rtt。

一次短连接总共5个rtt，所以浪费的就是tcp握手挥手的3个，除以总共的5个。

共 4 条评论 >

10



独钓寒江雪

2020-02-01

其实感觉本节用地铁站做例子更好：

地铁站相当于服务器；

地铁站有多个出入口，对应域名分片；

每个口有多条道，每条道对应一个连接；

每条道有开和关，对应握手和挥手；

人刷卡通过就传递了数据信息。

作者回复: 也挺好，多用不同的比喻可以加强理解。



9



-W.LI-

2019-07-05

谢谢老师!讲的很形象，有个问题，长链接。课后小贴士写了区分请求和应答。一个长链接，同一时间只能发送一个请求是吗?等到收到服务器响应以后才能被别的请求复用?假如有一个视频的请求一直占着。在分段传输的间隙能发送别的请求么?

“域名分片”(domain sharding)技术，具体怎么实现啊后面仔细会讲么?那个域名的比喻没看懂。是一个浏览器持有同一个服务的，多个负载的链接的意思么(一台服务器8个最多)服务器有集群浏览器创建了 $8 \times n$ 个链接。每台负载最多只让一个浏览器连了8个。

域名解析不是DNS服务器做的么，不是直接解析成IP的么还能域名指向域名么?完全不懂

领资料



作者回复: 1.是的, http是“半双工”, 只能一来一回收发数据, 这就是队头阻塞的根源。

2.域名分片其实很简单, 就是申请多个域名, 但这些域名最后都映射到同一个网站服务器, 这样就可以突破浏览器的限制, 让连接数是8\*域名的个数。

3.最后我说的不太确切, 应该是shard1、shard2的域名都指向www.xxx.com域名对应的ip地址。

4.域名也可以解析成别名 (cname), 用于cdn负载均衡, 后面会讲到。



8



### 衬衫的价格是19美元

2019-07-05

服务器或者客户端是怎么判断一个连接的呢? 是不是有一个id来对应一个连接? 一个连接具体是什么东西呢? 是双方在内存中开辟的空间吗?

作者回复: http里的连接通常就是tcp连接, 也就是调用socket api打开的一个套接字, 可以理解成一个流式文件的句柄, 可读可写, 但数据都是在网络上。

想要理解清楚应该去看一下tcp/ip相关的资料。

共 2 条评论 >

7



### MClink

2020-04-11

老师, 我们都知道tcp要三次握手来保证连接成功, 但是老是有人问为什么是三次, 不是四次, 不是五次, 如果面试官这么问的话, 我们应该怎么回答才能是较为准确呢? 我也只能说两次无法确认, 三次足够确认, 四次就多余了, 每三次就是一个有效的循环

作者回复: 因为tcp要一来一回才能确认消息能够正确收发, a端一个来回, 然后b端一个来回, 最少要三次通信, 所以这是最经济的做法。

共 2 条评论 >

6

领资料



### 小M

2021-03-07

个人理解, 不能把http的队首阻塞归结于请求-响应机制。

tcp是全双工的, 完全可以在一个tcp连接上双端同时进行收发。http只是在tcp连接上流动的一堆数据而已, 应用层的数据不存在队首阻塞。问题在于http协议自身: 它无法明确的标识出某个rsp是哪个req的。如果服务端不等待上一个req-rsp结束就发出另一个rsp, 那么客户端无



法区分收到的数据。

单连接上的多路复用也是基于请求-响应机制的，虽然一个连接上同时流动着多个req-rsp的数据，但是应用层协议有序号可以区分出rsp是哪个req的。

作者回复: 说的没错，req-rsp这个就是请求响应机制，要求必须一来一回，也就造成了消息必须排队处理。

到了http/2，一个流上只跑一个请求响应，而多个流并行，这个队头阻塞的问题也就迎刃而解了。



👍 5



**Hale**

2020-04-22

队头阻塞 是http协议层实现的，还是tcp 中listen(blocklog) 实现的？二者有关系吗？

作者回复:

队头阻塞有两个层面，一个是HTTP的长连接排队请求处理，另一个是TCP的丢包重传机制。

可以再看后面的HTTP/2和HTTP/3。



👍 4



**Happy-Coming**

2019-12-21

打卡机器比喻非常棒

作者回复: 其实只要是现实中的排队，就免不了队头阻塞。



👍 4



**answer宫**

2019-11-21

老师 ,请问为什么此请求是两个RTT啊,我理解一个就够了啊,一次请求,一次响应,一来一回就ok了吧,没有想通为什么是2个RTT

作者回复: 在tcp层次，发出一个包会返回一个确认收到的ack包，这就是一个rtt。

如果在http层次，那么看不到ack，所以一个请求一个响应是一个rtt。

层次不同，对rtt的理解就不一样，这里说的是tcp层次。

领资料



**李扬翼**

2019-09-30

老师，http2在应用层解决了队头阻塞，那TCP传输层的队头阻塞如何解决呢？

作者回复: 这就要看QUIC和HTTP/3了，请参考飞翔篇。



3

**徐海浪**

2019-07-07

1. 在开发基于 HTTP 协议的客户端时应该如何选择使用的连接模式呢？短连接还是长连接？根据请求的频繁程度来选择连接模式。一次性的请求用短连接，频繁与服务端交互的用长连接。

2. 应当如何降低长连接对服务器的负面影响呢？

长连接会长期占用服务器资源，根据服务器性能设置连接数和长连接超时时间，保证服务器TCP资源使用处于正常范围。

作者回复: √



3

**钱**

2020-03-29

1: 在开发基于 HTTP 协议的客户端时应该如何选择使用的连接模式呢？短连接还是长连接？这个视业务场景而定，只要一次交互就行了，也就是只有一次来回就OK，那就短连接，否则就长连接。

罗老师，请教一下短连接和长连接，表面上的长短是指连接存活的长短，不过如果使用短连接的请求和响应时间比较长时，短连接的存活时间也可能比长连接长的吧？连接的长短核心在于关闭连接的机制而非实际存活长短是吧？当然，一般而言长连接的存活时间比短连接长的。另外，长连接和回话的存活时间是两个不同的概念吧？他们有什么联系嘛？

2: 应当如何降低长连接对服务器的负面影响呢？

使用一定的保护措施，比如：文中讲的按超时时间或请求次数来关闭连接，也可以搞一个连接池来防护服务器。

老师的例子很棒，生动形象跃然于脑，另外，浏览器建立多个连接看评论服务器端口是一个浏览器是随机的，换言之多个连接在浏览器侧是多个端口在服务器侧是一个端口，这是为什么？感觉有些疑惑，另外请教一下一个端口能建立多少个连接？长连接的超时时间一般设置为多少，怎么考虑的？

[领资料](#)



作者回复:

1.是的，短连接指的是一次请求响应后就关闭连接，而不是指时间长短。

2.长连接是在一个连接里发多次请求，而会话是连接之上的概念，可能跨越多个连接，是业务层面的。

3.这方面已经有很多方法了，你说的对。

4.这个是tcp协议的原因，客户端用任意端口，服务器端用固定端口。并发的连接数受服务器的能力、资源限制，如果是Nginx就可以开几万、几十万个并发连接。

5.看具体情况，如果用户频繁交互就长一些，如果服务器能力差就短一些，Nginx默认是60秒。



WL

2019-07-05

请问一下老师域名分片技术是不是让一个浏览器跟不同的域名都建立长连接, 而这些域名都指向同一个服务器集群?

作者回复: 前半句对，后半句不一定，也可能是镜像集群。



领资料

