如何优化TCP：三个角度：三次握手、四次挥手、数据传输

一、TCP三次握手的性能提升

三次握手建立连接的首要目的是同步序列号

1. 客户端优化

SYN\_SENT状态的优化

根据网络的稳定性和目标服务器的繁忙程度修改SYN的重传次数（tcp\_syn\_retries），调整客户端的三次握手时间上限

2.服务端优化

服务端SYN半连接队列溢出后会导致后续连接被丢弃，可以通告netstat -s观察半连接队列溢出的情况，如果SYN半连接队列溢出情况比较严重可以通过tcp\_max\_syn\_backlog、somaxconn、backlog参数来调整半连接队列的大小，还可以开启syncookies在不使用SYN半连接队列的情况下成功建立连接

通过tcp\_synack\_retries控制服务端回复SYN+ACK的重传次数

通过ss -lnt查看服务端进程的accept队列长度，如果accept队列溢出系统默认丢弃ACK，把tcp\_abort\_on\_overflow设置为1表示用RST通知客户端链接额建立失败

如果accept队列溢出严重可以通过listen函数的backlog参数和somaxconn系统参数提高队列大小

3.绕过三次握手

TCP Fast Open功能可以绕过三次握手，通过cookie减少握手带来的1个RTT的时间消耗

二、四次挥手的性能提升

1.主动方的优化

FIN\_WAIT1:主动方发起FIN报文没有收到ACK会重传，通过tcp\_orphan\_retries参数控制重传次数

FIN\_WAIT2：主动方收到ACK后进入FIN\_WAIT2状态，根据关闭方式不同优化方式也不同：

close函数关闭的连接是孤儿连接（不能收发）如果tcp\_fin\_timeout秒内没有收到对方的FIN报文连接就直接关闭。同时为了应对孤儿连接占用太多的资源，tcp\_max\_orphans定义了最大孤儿连接的数量，超过时连接就直接释放

如果是shutdown关闭的连接不受此参数限制

TIME\_WAIT：主动方接收FIN报文后回复ACK进入TIME\_WAIT状态，这一状态会持续1分钟，为了防止其占用太多资源，tcp\_max\_tw\_buckets定义了最大数量，超过时连接也会直接释放。

还可以通过设置tcp\_tw\_reuse和tcp\_timestamps在客户端将TIME\_WAIT状态的端口复用于新连接

2.被动方优化

CLOSE\_WAIT：出现大量CLOSE\_WAIT状态的连接应从程序中寻找问题

LAST\_ACK：通过tcp\_orphana\_retries控制重发FIN报文的次数

三、TCP传输数据的性能提升

TCP头部的窗口大小为16位，最多能表达65535字节也就是64KB大小，在当今的高速网络下不够用，TCP选项字段中定义了窗口扩大因子，可以将窗口最多扩大到1GB，Linux通过tcp\_window\_scaling控制（默认开启）

网络的传输能力有限，发送超过网络处理能力的报文会被路由器直接丢弃，因此缓冲区的内存并不是越大越好。

发送缓冲区的大小决定了发送窗口的上限，发送窗口决定了已发送未确认的飞行报文的上限，发送缓冲区不能超过带宽时延积（RTT\*带宽）

内核缓冲区决定了滑动窗口的上限，缓冲区可以分为发送缓冲区tcp\_wmem和接收缓冲区tcp\_rmem，Linux会对缓冲区动态调节，我们应把缓冲区的上限设置为带宽时延积，发送缓冲区的调整功能是自动的，接收缓冲区通过把tcp\_moderate\_rcvbuf设置为1来开启