提高Linux系统性能加速网络应用程序 PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/145/2021_2022__E6_8F_90_E9_AB_98Linu_c103_145320.htm 在开发 socket 应用程序时， 首要任务通常是确保可靠性并满足一些特定的需求。利用本文中给出的 4 个提示，您就可以从头开始为实现最佳性能来设计并开发 socket 程序。本文内容包括对于 Sockets API 的使用、两个可以提高性能的 socket 选项以及 GNU/Linux 优化。 为了能够开发性能卓越的应用程序，请遵循以下技巧： 最小化报文传输的延时。 最小化系统调用的负载。 为 Bandwidth Delay Product 调节 TCP 窗口。 动态优化 GNU/Linux TCP/IP 栈。 技巧 1. 最小化报文传输的延时 在通过 TCP socket 进行通信时，数据都拆分成了数据块，这样它们就可以封装到给定连接的 TCP payload（指 TCP 数据包中的有效负荷）中了。TCP payload 的大小取决于几个因素（例如最大报文长度和路径），但是这些因素在连接发起时都是已知的。为了达到最好的性能，我们的目标是使用尽可能多的可用数据来填充每个报文。当没有足够的数据来填充 payload 时（也称为最大报文段长度（maximum segment size）或 MSS），TCP 就会采用 Nagle 算法自动将一些小的缓冲区连接到一个报文段中。这样可以通过最小化所发送的报文的数量来提高应用程序的效率，并减轻整体的网络拥塞问题。 尽管 John Nagle 的算法可以通过将这些数据连接成更大的报文来最小化所发送的报文的数量，但是有时您可能希望只发送一些较小的报文。一个简单的例子是 telnet 程序，它让用户可以与远程系统进行交互，这通常都是通过一个 shell 来进行的。如果用户被要求用

发送报文之前输入的字符来填充某个报文段，那么这种方法就绝对不能满足我们的需要。 另外一个例子是 HTTP 协议。通常，客户机浏览器会产生一个小请求（一条 HTTP 请求消息），然后 Web 服务器就会返回一个更大的响应（Web 页面）。 解决方案 您应该考虑的第一件事情是 Nagle 算法满足一种需求。由于这种算法对数据进行合并，试图构成一个完整的 TCP 报文段，因此它会引入一些延时。但是这种算法可以最小化在线路上发送的报文的数量，因此可以最小化网络拥塞的问题。 但是在需要最小化传输延时的情况中，Sockets API 可以提供一种解决方案。要禁用 Nagle 算法，您可以设置 TCP_NODELAY socket 选项，如清单 1 所示。 int sock, flag, ret. /* Create new stream socket */ sock = socket(AF_INET, SOCK_STREAM, 0). /* Disable the Nagle (TCP No Delay) algorithm */ flag = 1. ret = setsockopt(sock, IPPROTO_TCP, TCP_NODELAY, (char *)reg. Windowsamp.sock_buf_size, sizeof(sock_buf_size)). ret = setsockopt(sock, SOL_SOCKET, SO_RCVBUF, (char *)&amp.sock_buf_size, sizeof(sock_buf_size)). 清单 2. 手动设置发送和接收 socket 缓冲区大小 在 Linux 2.6 内核中，发送缓冲区的大小是由调用用户来定义的，但是接收缓冲区会自动加倍。您可以进行 getsockopt 调用来验证每个缓冲区的大小。 就 window scaling 来说，TCP 最初可以支持最大为 64KB 的窗口（使用 16 位的值来定义窗口的大小）。采用 window scaling（RFC 1323）扩展之后，您就可以使用 32 位的值来表示窗口的大小了。GNU/Linux 中提供的 TCP/IP 栈可以支持这个选项（以及其他一些选项）。 提示：Linux 内核还包括了自动对这些 socket 缓冲区进行优化的能力（请参

阅下面 表 1 中的 tcp_rmem 和 tcp_wmem），不过这些选项会对整个栈造成影响。如果您只需要为一个连接或一类连接调节窗口的大小，那么这种机制也许不能满足您的需要了。 技巧 4. 动态优化 GNU/Linux TCP/IP 栈 标准的 GNU/Linux 发行版试图对各种部署情况都进行优化。这意味着标准的发行版可能并没有对您的环境进行特殊的优化。 解决方案 GNU/Linux 提供了很多可调节的内核参数，您可以使用这些参数为您自己的用途对操作系统进行动态配置。下面我们来了解一下影响 socket 性能的一些更重要的选项。 在 /proc 虚拟文件系统中存在一些可调节的内核参数。这个文件系统中的每个文件都表示一个或多个参数，它们可以通过 cat 工具进行读取，或使用 echo 命令进行修改。清单 3 展示了如何查询或启用一个可调节的参数（在这种情况中，可以在 TCP/IP 栈中启用 IP 转发）。 [root@camus]# cat /proc/sys/net/ipv4/ip_forward 0 [root@camus]# echo "1" > /poc/sys/net/ipv4/ip_forward [root@camus]# cat /proc/sys/net/ipv4/ip_forward 1 [root@camus]# 清单 3. 调优：在 TCP/IP 栈中启用 IP 转发 与任何调优努力一样，最好的方法实际上就是不断进行实验。您的应用程序的行为、处理器的速度以及可用内存的多少都会影响到这些参数影响性能的方式。在某些情况中，您认为有益的操作可能恰恰是有害的（反之亦然）。因此，我们需要逐一试验各个选项，然后检查每个选项的结果。换而言之，我们需要相信自 己的经验，但是对每次修改都要进行验证。 提示：下面介绍一个有关永久性配置的问题。注意，如果您重新启动了 GNU/Linux 系统，那么您所需要的任何可调节的内核参数都会恢复成默认值。

为了将您所设置的值作为这些参数的默认值，可以使用 /etc/sysctl.conf 在系统启动时将这些参数配置成您所设置的值。 GNU/Linux 工具 GNU/Linux 对我非常有吸引力，这是因为其中有很多工具可以使用。尽管其中大部分都是命令行工具，但是它们都非常有用，而且非常直观。GNU/Linux 提供了几个工具 有些是 GNU/Linux 自己提供的，有些是开放源码软件 用于调试网络应用程序，测量带宽/吞吐量，以及检查链接的使用情况。 ping 这是用于检查主机的可用性的最常用的工具，但是也可以用于识别带宽延时产品计算的 RTT。 traceroute 打印某个连接到网络主机所经过的包括一系列路由器和网关的路径（路由），从而确定每个 hop 之间的延时。 netstat 确定有关网络子系统、协议和连接的各种统计信息。 tcpdump 显示一个或多个连接的协议级的报文跟踪信息；其中还包括时间信息，您可以使用这些信息来研究不同协议服务的报文时间。 netlog 为应用程序提供一些有关网络性能方面的信息。 nettimer 为瓶颈链接带宽生成一个度量标准；可以用于协议的自动优化。 Ethereal 以一个易于使用的图形化界面提供了 tcpump（报文跟踪）的特性。 iperf 测量 TCP 和 UDP 的网络性能；测量最大带宽，并汇报延时和数据报的丢失情况。 结束语 尝试使用本文中介绍的技巧和技术来提高 socket 应用程序的性能，包括通过禁用 Nagle 算法来减少传输延时，通过设置缓冲区的大小来提高 socket 带宽的利用，通过最小化系统调用的个数来降低系统调用的负载，以及使用可调节的内核参数来优化 Linux 的 TCP/IP 栈。 在进行优化时还需要考虑应用程序的特性。例如，您的应用程序是基于 LAN 的还是会通过 Internet 进行通信？如果您的应用程序仅

仅会在 LAN 内部进行操作，那么增大 socket 缓冲区的大小可能不会带来太大的改进，不过启用巨帧却一定会极大地改进性能！ 最后，还要使用 tcpdump 或 Ethereal 来检查优化之后的结果。在报文级看到的变化可以帮助展示使用这些技术进行优化之后所取得的成功效果。