

TCP 连接传输速率限制因素的测量和诊断

吕国晗, 严程, 李星

(清华大学电子工程系 NGN 实验室, 北京 100084)

摘要: TCP 连接的传输速率是很重要的一个网络性能指标. 由于很多应用的性能都取决于它, 我们往往需要找出 TCP 连接传输速率的瓶颈并进而提高. 然而, TCP 连接传输速率受到多种因素如丢包、瓶颈带宽以及发送接收端缓存的限制, 分析起来比较复杂. 本文描述了我们设计的一个工具, 它通过分析 TCP 数据 trace 文件来确定连接发送窗口的受限因素并测量连接的 RTT, 使我们很容易确定连接传输速率的受限因素. 实际网络测量表明该工具能够分辨多种传输速率限制因素.

关键词: TCP; 速率瓶颈; 故障诊断

中图分类号: TP 393.04

文献标识码: A

文章编号: 0438-0479(2007)S2-0188-04

TCP 连接的传输速率是最重要的网络性能指标之一. 网页浏览、文件下载共享以及在线视频播放等很多应用层程序的性能都取决于 TCP 连接的传输速率. 在实际网络故障诊断和性能优化中, 确定限制 TCP 速率的因素, 才能够找到提高连接速率的办法.

与 UDP 协议不同, TCP 使用一套较为复杂的拥塞流量控制机制来进行数据传输, 因此影响一条 TCP 传输速率的因素也是多方面的, 比如网络丢包、路径延迟、发送端接收端数据缓存大小等都有可能影响 TCP 连接的速率. 这使得仅仅测量连接的丢包率和 RTT 还不能很准确的确定限制 TCP 连接速率的原因.

发送窗口和数据包 RTT 是决定一条 TCP 连接传输速率的两个基本参数, 而发送窗口又受发送缓存、拥塞窗口以及接收端通告窗口的共同限制. 我们实现了一个工具, 它分析一条 TCP 连接数据发送端的 trace, 输出发送窗口的受限因素和连接的平均 RTT, 使得我们能够很方便地确定该连接传输速率的受限因素.

1 相关工作

tcpeval^[1]是一个用于分析 TCP 连接中各种延迟对连接总延迟影响的工具. 它首先确定连接中的关键路径, 其次对关键路径上的各种不同类型延迟进行定量分析, 找出影响连接总延迟的主要因素. 但该方法没有分析发送窗口变化的原因, 因此无法全面分析 TCP

速率的限制因素. tcp trace^[2]是另一个常用的 TCP 连接分析工具, 它可以给出 TCP 连接的一些基本统计参数, 比如重传包个数, RTT, 发送窗口等, 以及这些参数随时间变化的曲线. 虽然这些统计数据能够帮助我们分析影响连接速率的因素, 但工具没有对这些数据进行进一步分析.

2 速率限制因素的分析方法

TCP 连接的平均速率 $T = \overline{wnd} / \overline{RTT}$, 其中 \overline{wnd} 连接的发送窗口 (wnd) 的时间平均, \overline{RTT} 数据包的平均 RTT. 可以看出连接的速率和 \overline{wnd} 成正比和 \overline{RTT} 成反比. 根据 \overline{wnd} 和 \overline{RTT} 之间的关系, TCP 速率限制因素可以分为两类:

(1) \overline{wnd} 和 \overline{RTT} 之间独立, \overline{RTT} 不随 \overline{wnd} 的变化而变化. 此时速率取决于 \overline{wnd} . \overline{wnd} 是指连接某一时刻在网络中的数据包数量 (即那些已经发送但尚未收到确认的数据包个数). 它取值于拥塞窗口 (cwnd)、发送端缓存 (sbuf) 和接收端窗口 (rwnd) 中的最小值. 一旦知道每一时刻的 \overline{wnd} 和其它 3 个参数, 我们就能够判断此时 \overline{wnd} 具体是受哪个参数的限制.

(2) \overline{wnd} 和 \overline{RTT} 正相关. 此时增大 \overline{wnd} 无助于提高连接速率, 因为此时瓶颈链路塞满, 多增加的数据包进入路由器缓存, 导致数据包排队延迟的增加.

我们的分析方法需要确定该连接中每一时刻的 \overline{wnd} , cwnd, sbuf, rwnd 以及数据包的 RTT, 并通过将 \overline{wnd} 和 cwnd, sbuf, rwnd 比较来确定 \overline{wnd} 的限制因素. 事实上, 这些参数都包含在发送端的 TCP 协议栈中. 虽然个别操作系统如 FreeBSD 的 TCPDEBUG 能够输出此类状态变量, 但大多数操作系统没有提供此类

收稿日期: 2007-08-20

基金项目: 863 项目 (2006AA01Z201130) 资助

作者简介: 吕国晗, 男, 博士.

Email: lguohan@gmail.com

接口让上层应用直接获取传输层某些状态变量,即使提供也没有一个统一的标准^[3]。为了能够使工具适用任何操作系统,我们采用在发送端采集并分析数据包的办法来估计以上参数。在下面章节中,我们将分别讨论这些量的测量和估计方法。

2.1 确定发送窗口

数据发送端在收到一个确认包后即更新发送窗口的上下界,在发送窗口允许的范围内发送新数据包。因为使用的数据包 trace是在发送端采集的,且发送端从收到确认包到发送新数据包的延迟非常短,因此可以将紧接在一个确认包之后的数据包看作是由该确认包引发的数据包。发送窗口可以用等式 $wnd = snd_max \div snd_una$ 来计算,其中 snd_max 是该确认包引发数据包的最大序列号, snd_una 是该确认包的确认序列号。

2.2 确定发送窗口的限制因素

接收窗口 ($rwnd$) 包含在确认包的 TCP 包头中,可以直接通过分析包头获得。 $cwnd$ 和 $sbuf$ 需要根据分析发送窗口的变化来估计。当没有丢包时,发送端每收到一个确认包都会增加 $cwnd$,具体增加的值取决于发送端处于慢启动或者线性增加状态。无论是哪种状态,当发送窗口受限于 $cwnd$ 时,我们可以观察到每收到一定数量的确认包发送窗口就会增加一个数据包。而当发送窗口受限于 $sbuf$ 时,网络中数据包的总字节数已经等于发送缓存,此时无论收到多少个确认包发送窗口都不会再增加。因此,我们只需要参照 TCP 协议每收到一个确认包后就增加 $cwnd$,当 $cwnd$ 明显大于 wnd 时可以确定此时发送窗口不再受 $cwnd$ 而是 $sbuf$ 的限制。

图 1 给出了确定发送窗口限制因素的伪代码。其中 mss 是连接单个数据包的最大长度。我们使用 TCP 线性增长算法来增大 $cwnd$,这使得当 TCP 处于慢启动时,实际连接的 $cwnd$ 增长会比我们的算法快,此时算法只需要重置 $cwnd$ 就可以重新对真实 $cwnd$ 进行同步。这种重置方法使得我们的算法不需要和实际 TCP 协议栈使用的 $cwnd$ 增长算法一模一样,因此可以适用于不同 TCP 实现。

2.3 测量 RTT

接收端收到数据包后会发送确认包,该确认包的确认序列号等于收到数据包的最大序列号 + 1。根据这个准则,我们可以将确认包与引发它的数据包对应起来,并使用下面等式来计算一个数据包的 RTT:

$$RTT = T_D \div T_A,$$

其中 T_D 是数据包的发送时间, T_A 是发送端收到确认

```

if (wnd > cwnd) cwnd = wnd;
else cwnd += mss*mss/cwnd
if (wnd > rwnd - 2*mss)
wlimit = RCV_LIMIT;
else if (wnd < cwnd - 2*mss)
wlimit = SND_LIMIT;
else
wlimit = CWND_LIMIT;

```

图 1 发送窗口限制因素分析的伪代码

包的时间。需要指出的是,只有当接收端接收到的数据包序列号是连续时,发送的确认序号才等于该数据包的最大序列号 + 1。当有数据包丢包或者乱序时,接收端发送 DupACK 确认包,其确认序列号是当前收到连续数据包的最大序列号 + 1。在发送端仅仅根据确认序号无法将这个确认包和引发它的数据包对应起来。此时,我们可以利用 TCP SACK[4] 选项来匹配。限于篇幅限制,具体的算法我们不展开讨论。由于一般情况下丢包是少数,对于大多数的确认包,我们都可以不使用 SACK 选项进行成功的匹配,而忽略掉 DupACK 确认包对估计连接平均的 RTT 影响很小。

3 仿真实验

如图 2 所示,实验床由 3 台主机和 1 台 Cisco 7300 路由器构成。主机 A 和 C 使用 Linux 2.6 内核。我们通过配置路由器 R 来模拟不同的网络带宽,通过配置 FreeBSD 主机 B 的 dummynet 模块产生网络延迟。在每次实验中,我们在 A -> C 进行 10 s 的 TCP 文件传输,然后使用工具确定连接的速率限制因素并得到 Slimit, Rlimit, Climit 3 个参数,分别表示发送窗口受发送缓存、接收窗口和拥塞窗口限制的时长占整个连接时长的百分比。我们进行了 3 种典型网络环境下的测试,其中 3.1 和 3.2 节是 wnd 和 RTT 独立的情况,3.3 节是 wnd 和 RTT 正相关的情况。

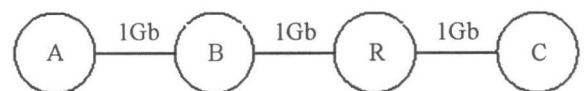


图 2 实验床

3.1 网络丢包

我们配置主机 B 在 A -> C 之间产生 0.1, 0.01, 0.001 的随机丢包,并通过改变接收端窗口来观察接收端窗口对速率的影响。从表 1 可以看出,当接收窗

口为 200 kB 时,连接主要受拥塞窗口的限制,其主要原因是网络丢包.对于接收窗口为 20 kB 的情况,当丢包率为 0.1 时,连接仍然主要受拥塞窗口的限制;当丢包率为 0.01 时,连接主要受拥塞窗口和接收窗口的共同限制;当丢包率为 0.001 时,连接 90% 的时间都受接收窗口的限制.在后两种丢包率下,20 kB 拥塞窗口限制了连接速率,当丢包率为 0.001 时,连接速率甚至不到此前的 50%.

表 1 网络丢包对传输速率的影响

	Rcwnd = 200kB			Rcwnd = 20kB		
	0.1	0.01	0.001	0.1	0.01	0.001
Loss rate	0.1	0.01	0.001	0.1	0.01	0.001
Rate (Mb)	2	73	190	2	53	98
wnd (kB)	2	13	38	2	10	18
R limit (%)	0	0	0	0	28	90
C limit (%)	27	90	99	23	64	10

3.2 大带宽延迟积链路

我们配置主机 B 使其在 C - > A 之间产生 100 ms 延迟,连接的 RTT 在 100 ms 以上.根据公式 $T = \text{wnd} / \text{RTT}$,此时需要很大的窗口才能达到高速.通过修改 A 和 C 主机 Linux 操作系统发送接收缓存的系统设置,我们控制接收窗口在 100 ~ 500 kB 之间取值.如表 2 所示,当窗口小于等于 300 kB 时,连接没有丢包,连接 90% 以上的时间数据发送都受接收端窗口的限制.由于 RTT 基本不变,连接速率随接收端窗口的增加而线性增长.在过程中,我们没有发现发送窗口受发送端缓存限制的现象,这是因为 Linux 2.6 内核当发送窗口接近缓存容量的时候能够动态增大发送缓存^[5].当窗口大于等于 400 kB 时,连接有大约 0.2% 的数据丢包.文献^[6]指出大窗口 TCP 即使经历极少量的丢包,都会使其长时间处在窗口线性增长的阶段,导致发送窗口大小不能及时增加,传输速率远小于链路的瓶颈带宽.对于这两种情况,工具显示有超过 95% 的时间数据发送受到拥塞窗口的限制,而它们的速率甚至小于 300 kB 时的传输速率.

3.3 链路瓶颈带宽

我们配置路由器在 A - > C 之间设置瓶颈带宽为 100 Mbit/s 从表 3 可以看出,工具显示有超过 95% 的时间数据发送分别受接收端窗口或者发送端缓存的限制,但只有当窗口为 10 kB 时,连接速率才远小于瓶颈带宽.当窗口为 50 kB 和 100 kB 的情况,连接速率基本等于瓶颈带宽.此时增大发送端缓存或接收端窗口

的同时数据包的 RTT 也在等比增加,因此传输速率没有提高.所以,虽然发送缓存或接收窗口限制了发送窗口,但此时的发送窗口已经足够塞满链路,真正限制连接速率的是瓶颈带宽.

表 2 不同接收端窗口下的连接速率

Rcwnd/ kB	Rate/ Mb	wnd/ kB	RTT/ ms	R limit/ %	C limit/ %
100	6.6	84.2	102.1	92	6
200	13.0	166	102.3	91	8
300	19.2	247	102.7	90	9
400	14.8	190	105.0	1	96
500	17.8	230	103.7	0	97

表 3 不同发送缓存和接收窗口下的速率等连接参数

	Sndbuf (kB)			Rcwnd (kB)		
	10	50	100	10	50	100
Rate (Mb)	53	91.5	90	43	95	95
wnd (kB)	13	65	125	8.7	49	100
RTT (ms)	1.7	5.6	10.9	1.6	4.1	8.3
R limit (%)	99.8	97.7	91.3	100	100	99.9
S limit (%)	0	0	0	0	0	0

4 结论和今后的工作

本文提出了一种通过分析 TCP 连接数据包 trace 文件来确定限制其传输速率因素的方法,该方法通过测量发送窗口、接收窗口,估计发送端拥塞窗口来确定发送窗口的受限因素,并结合数据包 RTT 来确定传输速率的受限因素.在几种有代表性网络环境下的测试表明该方法能够分辨不同的速率受限因素.今后,我们将在实际网络环境中进行测量,并将此类分析扩展到对接收端数据包 trace 文件的支持.

参考文献:

[1] Barford Paul, Crovella Mark. Critical path analysis of TCP transactions[J], Transactions on Networking, 2001, 9(3): 238 - 248.

[2] <http://jark.cs.ohio.edu/software/tcptrace/>.

[3] Jeffrey Mogul, Brakmo L, Lowell D E, et al. Unveiling the Transport[C]. Hotnet II, 2003.

[4] Mathis M, Mahdavi J, Floyd S, et al. RFC2018 TCP selective acknowledgement options[S]. 1996.

[5] <http://www-didc.lbl.gov/TCP-tuning/linux.html>

[6] Floyd S. RFC 3649 High-speed TCP for large congestion windows[S]. 2003.

The Measurement and Diagnosis of TCP Throughput Limiting Factors

LÜ Guo-han, YAN Cheng, LI Xing

(Lab of NGN, Department of EE, Tsinghua University, Beijing 100084, China)

Abstract: TCP throughput is one of important network performance metrics. As the performance of many applications depends on it, we usually need to identify its limiting factors in order to increase it. However, TCP throughput is affected by many factors such as loss, bottleneck bandwidth, and sender/receiver buffer. In this paper, we describe our tool which identifies the limiting factor of sender window size, and measures the average RTT of the connection based on packet traces dumped at the TCP sender side. The output of the tool enable us to determine the limiting factors of TCP throughput. Testbed measurements validated our method.

Key words: TCP; rate limit; troubleshooting