优化 Linux 的内核参数来提高服务器并发处理能力

PS：在服务器硬件资源额定有限的情况下，最大的压榨服务器的性能，提高服务器的并发处理能力，是很多运维技术人员思考的问题。要提高 *Linux* 系统下的负载能力，可以使用 *nginx* 等原生并发处理能力就很强的 *web* 服务器，如果使用 *Apache* 的可以启用其 *Worker* 模式，来提高其并发处理能力。除此之外，在考虑节省成本的情况下，可以修改 *Linux* 的内核相关 *TCP* 参数，来最大的提高服务器性能。当然，最基础的提高负载问题，还是升级服务器硬件了，这是最根本的。

Linux 系统下，*TCP* 连接断开后，会以 *TIME_WAIT* 状态保留一定的时间，然后才会释放端口。当并发请求过多的时候，就会产生大量的 *TIME_WAIT* 状态的连接，无法及时断开的话，会占用大量的端口资源和服务器资源。这个时候我们可以优化 *TCP* 的内核参数，来及时将 *TIME_WAIT* 状态的端口清理掉。

本文介绍的方法只对拥有大量 *TIME_WAIT* 状态的连接导致系统资源消耗有效，如果不是这种情况下，效果可能不明显。可以使用 *netstat* 命令去查 *TIME_WAIT* 状态的连接状态，输入下面的组合命令，查看当前 *TCP* 连接的状态和对应的连接数量：

```
netstat -n | awk '/^tcp/ {++S[$NF]} END {for(a in S) print a, S[a]}'
```

这个命令会输出类似下面的结果：

```
LAST_ACK 16
SYN_RECV 348
ESTABLISHED 70
FIN_WAIT1 229
FIN_WAIT2 30
CLOSING 33
TIME_WAIT 18098
```

我们只用关心 *TIME_WAIT* 的个数，在这里可以看到，有*18000*多个 *TIME_WAIT*，这样就占用了*18000*多个端口。要知道端口的数量只有*65535*个，占用一个少一个，会严重的影响到后继的新连接。这种情况下，我们就有必要调整下 *Linux* 的 *TCP* 内核参数，让系统更快的释放 *TIME_WAIT* 连接。

用 *vim* 打开配置文件：*vim /etc/sysctl.conf*

在这个文件中，加入下面的几行内容：

```
net.ipv4.tcp_syncookies = 1
```

```
net.ipv4.tcp_tw_reuse = 1
net.ipv4.tcp_tw_recycle = 1
net.ipv4.tcp_fin_timeout = 30
```

输入下面的命令，让内核参数生效：

```
sysctl -p
```

简单的说明上面的参数的含义：

net.ipv4.tcp_syncookies = 1

#表示开启 *SYN Cookies*。当出现 *SYN* 等待队列溢出时，启用 *cookies* 来处理，可防范少量 *SYN* 攻击，默认为*0*，表示关闭；

net.ipv4.tcp_tw_reuse = 1

#表示开启重用。允许将 *TIME-WAIT sockets* 重新用于新的 *TCP* 连接，默认为*0*，表示关闭；

net.ipv4.tcp_tw_recycle = 1

#表示开启 *TCP* 连接中 *TIME-WAIT sockets* 的快速回收，默认为*0*，表示关闭；

net.ipv4.tcp_fin_timeout

#修改系統默认的 *TIMEOUT* 时间。

在经过这样的调整之后，除了会进一步提升服务器的负载能力之外，还能够防御小流量程度的 *DoS*、*CC* 和 *SYN* 攻击。

此外，如果你的连接数本身就很多，我们可以再优化一下 *TCP* 的可使用端口范围，进一步提升服务器的并发能力。依然是往上面的参数文件中，加入下面这些配置：

```
net.ipv4.tcp_keepalive_time = 1200
net.ipv4.ip_local_port_range = 10000 65000
net.ipv4.tcp_max_syn_backlog = 8192
net.ipv4.tcp_max_tw_buckets = 5000
```

#这几个参数，建议只在流量非常大的服务器上开启，会有显著的效果。一般的流量小的服务器上，没有必要去设置这几个参数。

net.ipv4.tcp_keepalive_time = 1200

#表示当 *keepalive* 起用的时候，*TCP* 发送 *keepalive* 消息的频度。缺省是*2*小时，改为*20*分钟。

net.ipv4.ip_local_port_range = 10000 65000

#表示用于向外连接的端口范围。缺省情况下很小：*32768*到*61000*，改为*10000*到*65000*。（注意：这里不要将最低值设的太低，否则可能会占用掉正常的端口！）

net.ipv4.tcp_max_syn_backlog = 8192

#表示 *SYN* 队列的长度，默认为*1024*，加大队列长度为*8192*，可以容纳更多等待连接的网络连接数。

net.ipv4.tcp_max_tw_buckets = 6000

#表示系统同时保持 *TIME_WAIT* 的最大数量，如果超过这个数字，*TIME_WAIT* 将立刻被清除并打印警告信息。默 认为*180000*，改为*6000*。对于 *Apache*、*Nginx* 等服务器，上几行的参数可以很好地减少 *TIME_WAIT* 套接字数量，但是对于 *Squid*，效果却不大。此项参数可以控制 *TIME_WAIT* 的最大数量，避免 *Squid* 服务器被大量的 *TIME_WAIT* 拖死。

内核其他 *TCP* 参数说明：

net.ipv4.tcp_max_syn_backlog = 65536

#记录的那些尚未收到客户端确认信息的连接请求的最大值。对于有*128M* 内存的系统而言，缺省值是*1024*，小内存的系统则是*128*。

net.core.netdev_max_backlog = 32768

#每个网络接口接收数据包的速率比内核处理这些包的速率快时，允许送到队列的数据包的最大数目。

net.core.somaxconn = 32768

#web 应用中 *listen* 函数的 *backlog* 默认会给我们内核参数的 *net.core.somaxconn* 限制到*128*，而 *nginx* 定义的 *NGX_LISTEN_BACKLOG* 默认为*511*，所以有必要调整这个值。

```
net.core.wmem_default = 8388608

net.core.rmem_default = 8388608

net.core.rmem_max = 16777216                #最大 socket 读 buffer,可参考的优化值:873200

net.core.wmem_max = 16777216                #最大 socket 写 buffer,可参考的优化值:873200

net.ipv4.tcp_timestsmps = 0
```

#时间戳可以避免序列号的卷绕。一个*1Gbps* 的链路肯定会遇到以前用过的序列号。时间戳能够让内核接受这种“异常”的数据包。这里需要将其关掉。

net.ipv4.tcp_synack_retries = 2

#为了打开对端的连接，内核需要发送一个 *SYN* 并附带一个回应前面一个 *SYN* 的 *ACK*。也就是所谓三次握手中的第二次握手。这个设置决定了内核放弃连接之前发送 *SYN+ACK* 包的数量。

net.ipv4.tcp_syn_retries = 2

#在内核放弃建立连接之前发送 *SYN* 包的数量。

#net.ipv4.tcp_tw_len = 1

net.ipv4.tcp_tw_reuse = 1

开启重用。允许将 *TIME-WAIT sockets* 重新用于新的 *TCP* 连接。

net.ipv4.tcp_wmem = 8192 436600 873200

 # TCP 写 *buffer,*可参考的优化值*: 8192 436600 873200*

net.ipv4.tcp_rmem = 32768 436600 873200

 # TCP 读 *buffer,*可参考的优化值*: 32768 436600 873200*

net.ipv4.tcp_mem = 94500000 91500000 92700000

同样有*3*个值*,*意思是*:*

*net.ipv4.tcp_mem[0]:*低于此值，*TCP* 没有内存压力。

*net.ipv4.tcp_mem[1]:*在此值下，进入内存压力阶段。

*net.ipv4.tcp_mem[2]:*高于此值，*TCP* 拒绝分配 *socket*。

上述内存单位是页，而不是字节。可参考的优化值是*:786432 1048576 1572864*

net.ipv4.tcp_max_orphans = 3276800

#系统中最多有多少个 *TCP* 套接字不被关联到任何一个用户文件句柄上。

如果超过这个数字，连接将即刻被复位并打印出警告信息。

这个限制仅仅是为了防止简单的 *DoS* 攻击，不能过分依靠它或者人为地减小这个值，

更应该增加这个值*(*如果增加了内存之后*)*。

net.ipv4.tcp_fin_timeout = 30

#如果套接字由本端要求关闭，这个参数决定了它保持在 *FIN-WAIT-2*状态的时间。对端可以出错并永远不关闭连接，甚至意外当机。缺省值是*60*秒。*2.2* 内核的通常值是*180*秒，你可以按这个设置，但要记住的是，即使你的机器是一个轻载的 *WEB* 服务器，也有因为大量的死套接字而内存溢出的风险，*FIN- WAIT-2*的危险性比 *FIN-WAIT-1*要小，因为它最多只能吃掉*1.5K* 内存，但是它们的生存期长些。

经过这样的优化配置之后，你的服务器的 *TCP* 并发处理能力会显著提高。以上配置仅供参考，用于生产环境请根据自己的实际情况。