# **/proc/sys/net目录**

　　所有的TCP/IP参数都位于/proc/sys/net目录下（请注意，对/proc/sys/net目录下内容的修改都是临时的，任何修改在系统重启后都会丢失），例如下面这些重要的参数：

|  |  |  |  |
| --- | --- | --- | --- |
| **参数（路径+文件）** | **描述** | **默认值** | **优化值** |
| /proc/sys/net/core/rmem\_default | 默认的TCP数据接收窗口大小（字节）。 | 229376 | 256960 |
| /proc/sys/net/core/rmem\_max | 最大的TCP数据接收窗口（字节）。 | 131071 | 513920 |
| /proc/sys/net/core/wmem\_default | 默认的TCP数据发送窗口大小（字节）。 | 229376 | 256960 |
| /proc/sys/net/core/wmem\_max | 最大的TCP数据发送窗口（字节）。 | 131071 | 513920 |
| /proc/sys/net/core/netdev\_max\_backlog | 在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。 | 1000 | 2000 |
| /proc/sys/net/core/somaxconn | 定义了系统中每一个端口最大的监听队列的长度，这是个全局的参数。 | 128 | 2048 |
| /proc/sys/net/core/optmem\_max | 表示每个套接字所允许的最大缓冲区的大小。 | 20480 | 81920 |
| /proc/sys/net/ipv4/tcp\_mem | 确定TCP栈应该如何反映内存使用，每个值的单位都是内存页（通常是4KB）。第一个值是内存使用的下限；第二个值是内存压力模式开始对缓冲区使用应用压力的上限；第三个值是内存使用的上限。在这个层次上可以将报文丢弃，从而减少对内存的使用。对于较大的BDP可以增大这些值（注意，其单位是内存页而不是字节）。 | 94011  125351  188022 | 131072  262144  524288 |
| /proc/sys/net/ipv4/tcp\_rmem | 为自动调优定义socket使用的内存。第一个值是为socket接收缓冲区分配的最少字节数；第二个值是默认值（该值会被rmem\_default覆盖），缓冲区在系统负载不重的情况下可以增长到这个值；第三个值是接收缓冲区空间的最大字节数（该值会被rmem\_max覆盖）。 | 4096  87380  4011232 | 8760  256960  4088000 |
| /proc/sys/net/ipv4/tcp\_wmem | 为自动调优定义socket使用的内存。第一个值是为socket发送缓冲区分配的最少字节数；第二个值是默认值（该值会被wmem\_default覆盖），缓冲区在系统负载不重的情况下可以增长到这个值；第三个值是发送缓冲区空间的最大字节数（该值会被wmem\_max覆盖）。 | 4096  16384  4011232 | 8760  256960  4088000 |
| /proc/sys/net/ipv4/tcp\_keepalive\_time | TCP发送keepalive探测消息的间隔时间（秒），用于确认TCP连接是否有效。 | 7200 | 1800 |
| /proc/sys/net/ipv4/tcp\_keepalive\_intvl | 探测消息未获得响应时，重发该消息的间隔时间（秒）。 | 75 | 30 |
| /proc/sys/net/ipv4/tcp\_keepalive\_probes | 在认定TCP连接失效之前，最多发送多少个keepalive探测消息。 | 9 | 3 |
| /proc/sys/net/ipv4/tcp\_sack | 启用有选择的应答（1表示启用），通过有选择地应答乱序接收到的报文来提高性能，让发送者只发送丢失的报文段，（对于广域网通信来说）这个选项应该启用，但是会增加对CPU的占用。 | 1 | 1 |
| /proc/sys/net/ipv4/tcp\_fack | 启用转发应答，可以进行有选择应答（SACK）从而减少拥塞情况的发生，这个选项也应该启用。 | 1 | 1 |
| /proc/sys/net/ipv4/tcp\_timestamps | TCP时间戳（会在TCP包头增加12个字节），以一种比重发超时更精确的方法（参考RFC 1323）来启用对RTT 的计算，为实现更好的性能应该启用这个选项。 | 1 | 1 |
| /proc/sys/net/ipv4/tcp\_window\_scaling | 启用RFC 1323定义的window scaling，要支持超过64KB的TCP窗口，必须启用该值（1表示启用），TCP窗口最大至1GB，TCP连接双方都启用时才生效。 | 1 | 1 |
| /proc/sys/net/ipv4/tcp\_syncookies | 表示是否打开TCP同步标签（syncookie），内核必须打开了CONFIG\_SYN\_COOKIES项进行编译，同步标签可以防止一个套接字在有过多试图连接到达时引起过载。 | 1 | 1 |
| /proc/sys/net/ipv4/tcp\_tw\_reuse | 表示是否允许将处于TIME-WAIT状态的socket（TIME-WAIT的端口）用于新的TCP连接 。 | 0 | 1 |
| /proc/sys/net/ipv4/tcp\_tw\_recycle | 能够更快地回收TIME-WAIT套接字。 | 0 | 1 |
| /proc/sys/net/ipv4/tcp\_fin\_timeout | 对于本端断开的socket连接，TCP保持在FIN-WAIT-2状态的时间（秒）。对方可能会断开连接或一直不结束连接或不可预料的进程死亡。 | 60 | 30 |
| /proc/sys/net/ipv4/ip\_local\_port\_range | 表示TCP/UDP协议允许使用的本地端口号 | 32768  61000 | 1024  65000 |
| /proc/sys/net/ipv4/tcp\_max\_syn\_backlog | 对于还未获得对方确认的连接请求，可保存在队列中的最大数目。如果服务器经常出现过载，可以尝试增加这个数字。 | 2048 | 2048 |
| /proc/sys/net/ipv4/tcp\_low\_latency | 允许TCP/IP栈适应在高吞吐量情况下低延时的情况，这个选项应该禁用。 | 0 |  |
| /proc/sys/net/ipv4/tcp\_westwood | 启用发送者端的拥塞控制算法，它可以维护对吞吐量的评估，并试图对带宽的整体利用情况进行优化，对于WAN 通信来说应该启用这个选项。 | 0 |  |
| /proc/sys/net/ipv4/tcp\_bic | 为快速长距离网络启用Binary Increase Congestion，这样可以更好地利用以GB速度进行操作的链接，对于WAN通信应该启用这个选项。 | 1 |  |

**/etc/sysctl.conf文件**

　　/etc/sysctl.conf是一个允许你改变正在运行中的Linux系统的接口。它包含一些TCP/IP堆栈和虚拟内存系统的高级选项，可用来控制Linux网络配置，由于/proc/sys/net目录内容的临时性，建议把TCPIP参数的修改添加到/etc/sysctl.conf文件, 然后保存文件，使用命令“/sbin/sysctl –p”使之立即生效。具体修改方案参照上文：

net.core.rmem\_default = 256960

net.core.rmem\_max = 513920

net.core.wmem\_default = 256960

net.core.wmem\_max = 513920

net.core.netdev\_max\_backlog = 2000

net.core.somaxconn = 2048

net.core.optmem\_max = 81920

net.ipv4.tcp\_mem = 131072  262144  524288

net.ipv4.tcp\_rmem = 8760  256960  4088000

net.ipv4.tcp\_wmem = 8760  256960  4088000

net.ipv4.tcp\_keepalive\_time = 1800

net.ipv4.tcp\_keepalive\_intvl = 30

net.ipv4.tcp\_keepalive\_probes = 3

net.ipv4.tcp\_sack = 1

net.ipv4.tcp\_fack = 1

net.ipv4.tcp\_timestamps = 1

net.ipv4.tcp\_window\_scaling = 1

net.ipv4.tcp\_syncookies = 1

net.ipv4.tcp\_tw\_reuse = 1

net.ipv4.tcp\_tw\_recycle = 1

net.ipv4.tcp\_fin\_timeout = 30

net.ipv4.ip\_local\_port\_range = 1024  65000

net.ipv4.tcp\_max\_syn\_backlog = 2048

转至：<http://www.cnblogs.com/fczjuever/archive/2013/04/17/3026694.html>

[LINUX 下tcp 和 udp 套接字收发缓冲区的大小决定规则](http://blog.csdn.net/qiaoliang328/article/details/7580555)

1. tcp 收发缓冲区默认值

[root@qljt core]# cat /proc/sys/net/ipv4/tcp\_rmem    
4096    87380   4161536

87380  ：tcp接收缓冲区的默认值

[root@qljt core]# cat /proc/sys/net/ipv4/tcp\_wmem   
4096    16384   4161536

16384  ： tcp 发送缓冲区的默认值

2. tcp 或udp收发缓冲区最大值

[root@qljt core]# cat /proc/sys/net/core/rmem\_max   
131071

131071：tcp 或 udp 接收缓冲区最大可设置值的一半。

也就是说调用 setsockopt(s, SOL\_SOCKET, SO\_RCVBUF, &rcv\_size, &optlen);  时rcv\_size 如果超过 131071，那么

getsockopt(s, SOL\_SOCKET, SO\_RCVBUF, &rcv\_size, &optlen); 去到的值就等于 131071 \* 2 = 262142

[root@qljt core]# cat /proc/sys/net/core/wmem\_max    
131071

131071：tcp 或 udp 发送缓冲区最大可设置值得一半。

跟上面同一个道理

3. udp收发缓冲区默认值

[root@qljt core]# cat /proc/sys/net/core/rmem\_default     
111616：udp接收缓冲区的默认值

[root@qljt core]# cat /proc/sys/net/core/wmem\_default   
111616

111616：udp发送缓冲区的默认值

4. tcp 或udp收发缓冲区最小值

tcp 或udp接收缓冲区的最小值为 256 bytes，由内核的宏决定；

tcp 或udp发送缓冲区的最小值为 2048 bytes，由内核的宏决定

# ****/proc/sys/net目录****

　　所有的TCP/IP参数都位于/proc/sys/net目录下（请注意，对/proc/sys/net目录下内容的修改都是临时的，任何修改在系统重启后都会丢失），例如下面这些重要的参数：

|  |  |  |  |
| --- | --- | --- | --- |
| **参数（路径+文件）** | **描述** | **默认值** | **优化值** |
| /proc/sys/net/core/rmem\_default | 默认的TCP数据接收窗口大小（字节）。 | 229376 | 256960 |
| /proc/sys/net/core/rmem\_max | 最大的TCP数据接收窗口（字节）。 | 131071 | 513920 |
| /proc/sys/net/core/wmem\_default | 默认的TCP数据发送窗口大小（字节）。 | 229376 | 256960 |
| /proc/sys/net/core/wmem\_max | 最大的TCP数据发送窗口（字节）。 | 131071 | 513920 |
| /proc/sys/net/core/netdev\_max\_backlog | 在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。 | 1000 | 2000 |
| /proc/sys/net/core/somaxconn | 定义了系统中每一个端口最大的监听队列的长度，这是个全局的参数。 | 128 | 2048 |
| /proc/sys/net/core/optmem\_max | 表示每个套接字所允许的最大缓冲区的大小。 | 20480 | 81920 |
| /proc/sys/net/ipv4/tcp\_mem | 确定TCP栈应该如何反映内存使用，每个值的单位都是内存页（通常是4KB）。第一个值是内存使用的下限；第二个值是内存压力模式开始对缓冲区使用应用压力的上限；第三个值是内存使用的上限。在这个层次上可以将报文丢弃，从而减少对内存的使用。对于较大的BDP可以增大这些值（注意，其单位是内存页而不是字节）。 | 94011  125351  188022 | 131072  262144  524288 |
| /proc/sys/net/ipv4/tcp\_rmem | 为自动调优定义socket使用的内存。第一个值是为socket接收缓冲区分配的最少字节数；第二个值是默认值（该值会被rmem\_default覆盖），缓冲区在系统负载不重的情况下可以增长到这个值；第三个值是接收缓冲区空间的最大字节数（该值会被rmem\_max覆盖）。 | 4096  87380  4011232 | 8760  256960  4088000 |
| /proc/sys/net/ipv4/tcp\_wmem | 为自动调优定义socket使用的内存。第一个值是为socket发送缓冲区分配的最少字节数；第二个值是默认值（该值会被wmem\_default覆盖），缓冲区在系统负载不重的情况下可以增长到这个值；第三个值是发送缓冲区空间的最大字节数（该值会被wmem\_max覆盖）。 | 4096  16384  4011232 | 8760  256960  4088000 |
| /proc/sys/net/ipv4/tcp\_keepalive\_time | TCP发送keepalive探测消息的间隔时间（秒），用于确认TCP连接是否有效。 | 7200 | 1800 |
| /proc/sys/net/ipv4/tcp\_keepalive\_intvl | 探测消息未获得响应时，重发该消息的间隔时间（秒）。 | 75 | 30 |
| /proc/sys/net/ipv4/tcp\_keepalive\_probes | 在认定TCP连接失效之前，最多发送多少个keepalive探测消息。 | 9 | 3 |
| /proc/sys/net/ipv4/tcp\_sack | 启用有选择的应答（1表示启用），通过有选择地应答乱序接收到的报文来提高性能，让发送者只发送丢失的报文段，（对于广域网通信来说）这个选项应该启用，但是会增加对CPU的占用。 | 1 | 1 |
| /proc/sys/net/ipv4/tcp\_fack | 启用转发应答，可以进行有选择应答（SACK）从而减少拥塞情况的发生，这个选项也应该启用。 | 1 | 1 |
| /proc/sys/net/ipv4/tcp\_timestamps | TCP时间戳（会在TCP包头增加12个字节），以一种比重发超时更精确的方法（参考RFC 1323）来启用对RTT 的计算，为实现更好的性能应该启用这个选项。 | 1 | 1 |
| /proc/sys/net/ipv4/tcp\_window\_scaling | 启用RFC 1323定义的window scaling，要支持超过64KB的TCP窗口，必须启用该值（1表示启用），TCP窗口最大至1GB，TCP连接双方都启用时才生效。 | 1 | 1 |
| /proc/sys/net/ipv4/tcp\_syncookies | 表示是否打开TCP同步标签（syncookie），内核必须打开了CONFIG\_SYN\_COOKIES项进行编译，同步标签可以防止一个套接字在有过多试图连接到达时引起过载。 | 1 | 1 |
| /proc/sys/net/ipv4/tcp\_tw\_reuse | 表示是否允许将处于TIME-WAIT状态的socket（TIME-WAIT的端口）用于新的TCP连接 。 | 0 | 1 |
| /proc/sys/net/ipv4/tcp\_tw\_recycle | 能够更快地回收TIME-WAIT套接字。 | 0 | 1 |
| /proc/sys/net/ipv4/tcp\_fin\_timeout | 对于本端断开的socket连接，TCP保持在FIN-WAIT-2状态的时间（秒）。对方可能会断开连接或一直不结束连接或不可预料的进程死亡。 | 60 | 30 |
| /proc/sys/net/ipv4/ip\_local\_port\_range | 表示TCP/UDP协议允许使用的本地端口号 | 32768  61000 | 1024  65000 |
| /proc/sys/net/ipv4/tcp\_max\_syn\_backlog | 对于还未获得对方确认的连接请求，可保存在队列中的最大数目。如果服务器经常出现过载，可以尝试增加这个数字。 | 2048 | 2048 |
| /proc/sys/net/ipv4/tcp\_low\_latency | 允许TCP/IP栈适应在高吞吐量情况下低延时的情况，这个选项应该禁用。 | 0 |  |
| /proc/sys/net/ipv4/tcp\_westwood | 启用发送者端的拥塞控制算法，它可以维护对吞吐量的评估，并试图对带宽的整体利用情况进行优化，对于WAN 通信来说应该启用这个选项。 | 0 |  |
| /proc/sys/net/ipv4/tcp\_bic | 为快速长距离网络启用Binary Increase Congestion，这样可以更好地利用以GB速度进行操作的链接，对于WAN通信应该启用这个选项。 | 1 |  |

**/etc/sysctl.conf文件**

　　/etc/sysctl.conf是一个允许你改变正在运行中的Linux系统的接口。它包含一些TCP/IP堆栈和虚拟内存系统的高级选项，可用来控制Linux网络配置，由于/proc/sys/net目录内容的临时性，建议把TCPIP参数的修改添加到/etc/sysctl.conf文件, 然后保存文件，使用命令“/sbin/sysctl –p”使之立即生效。具体修改方案参照上文：

net.core.rmem\_default = 256960

net.core.rmem\_max = 513920

net.core.wmem\_default = 256960

net.core.wmem\_max = 513920

net.core.netdev\_max\_backlog = 2000

net.core.somaxconn = 2048

net.core.optmem\_max = 81920

net.ipv4.tcp\_mem = 131072  262144  524288

net.ipv4.tcp\_rmem = 8760  256960  4088000

net.ipv4.tcp\_wmem = 8760  256960  4088000

net.ipv4.tcp\_keepalive\_time = 1800

net.ipv4.tcp\_keepalive\_intvl = 30

net.ipv4.tcp\_keepalive\_probes = 3

net.ipv4.tcp\_sack = 1

net.ipv4.tcp\_fack = 1

net.ipv4.tcp\_timestamps = 1

net.ipv4.tcp\_window\_scaling = 1

net.ipv4.tcp\_syncookies = 1

net.ipv4.tcp\_tw\_reuse = 1

net.ipv4.tcp\_tw\_recycle = 1

net.ipv4.tcp\_fin\_timeout = 30

net.ipv4.ip\_local\_port\_range = 1024  65000

net.ipv4.tcp\_max\_syn\_backlog = 2048

Doc2：

   可调优的内核变量存在两种主要接口：sysctl命令和/proc文件系统，proc中与进程无关的所有信息都被移植到sysfs中。IPV4协议栈的sysctl参数主要是sysctl.net.core、sysctl.net.ipv4，对应的/proc文件系统是/proc/sys/net/ipv4和/proc/sys/net/core。只有内核在编译时包含了特定的属性，该参数才会出现在内核中。

    对于内核参数应该谨慎调节，这些参数通常会影响到系统的整体性能。内核在启动时会根据系统的资源情况来初始化特定的变量，这种初始化的调节一般会满足通常的性能需求。

    应用程序通过socket系统调用和远程主机进行通讯，每一个socket都有一个读写缓冲区。读缓冲区保存了远程主机发送过来的数据，如果缓冲区已满，则数据会被丢弃，写缓冲期保存了要发送到远程主机的数据，如果写缓冲区已慢，则系统的应用程序在写入数据时会阻塞。可知，缓冲区是有大小的。

**socket缓冲区默认大小**：  
/proc/sys/net/core/rmem\_default     对应net.core.rmem\_default  
/proc/sys/net/core/wmem\_default     对应net.core.wmem\_default  
    上面是各种类型socket的默认读写缓冲区大小，然而对于特定类型的socket则可以设置独立的值覆盖默认值大小。例如tcp类型的socket就可以用/proc/sys/net/ipv4/tcp\_rmem和tcp\_wmem来覆盖。

**socket缓冲区最大值：**  
/proc/sys/net/core/rmem\_max        对应net.core.rmem\_max  
/proc/sys/net/core/wmem\_max        对应net.core.wmem\_max

/proc/sys/net/core/netdev\_max\_backlog    对应 net.core.netdev\_max\_backlog  
    该参数定义了当接口收到包的速率大于内核处理包的速率时，设备的输入队列中的最大报文数。

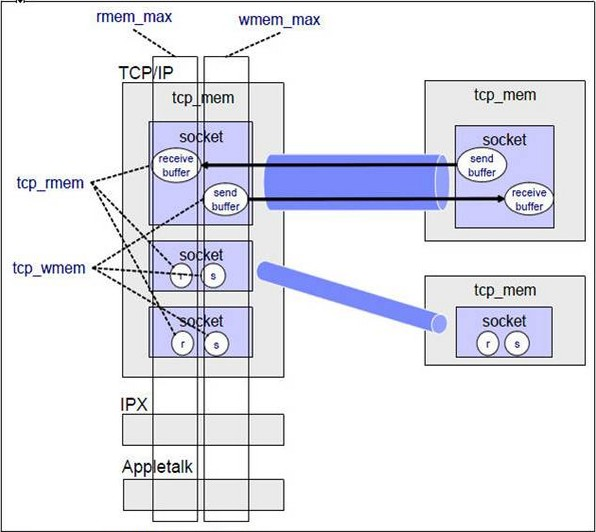
/proc/sys/net/core/somaxconn        对应 net.core.somaxconn  
    通过listen系统调用可以指定的最大accept队列backlog，当排队的请求连接大于该值时，后续进来的请求连接会被丢弃。

/proc/sys/net/core/optmem\_max          对应 net.core.optmem\_max  
    每个socket的副缓冲区大小。

**TCP/IPV4内核参数：**  
    在创建socket的时候会指定socke协议和地址类型。TCP socket缓冲区大小是他自己控制而不是由core内核缓冲区控制。  
/proc/sys/net/ipv4/tcp\_rmem     对应net.ipv4.tcp\_rmem  
/proc/sys/net/ipv4/tcp\_wmem     对应net.ipv4.tcp\_wmem  
    以上是TCP socket的读写缓冲区的设置，每一项里面都有三个值，第一个值是缓冲区最小值，中间值是缓冲区的默认值，最后一个是缓冲区的最大值，虽然缓冲区的值不受core缓冲区的值的限制，但是缓冲区的最大值仍旧受限于core的最大值。

/proc/sys/net/ipv4/tcp\_mem    
    该内核参数也是包括三个值，用来定义内存管理的范围，第一个值的意思是当page数低于该值时，TCP并不认为他为内存压力，第二个值是进入内存的压力区域时所达到的页数，第三个值是所有TCP sockets所允许使用的最大page数，超过该值后，会丢弃后续报文。page是以页面为单位的，为系统中socket全局分配的内存容量。

socket的结构如下图：



/proc/sys/net/ipv4/tcp\_window\_scaling      对应net.ipv4.tcp\_window\_scaling  
    管理TCP的窗口缩放特性，因为在tcp头部中声明接收缓冲区的长度为26位，因此窗口不能大于64K，如果大于64K，就要打开窗口缩放。

/proc/sys/net/ipv4/tcp\_sack    对应net.ipv4.tcp\_sack  
    管理TCP的选择性应答，允许接收端向发送端传递关于字节流中丢失的序列号，减少了段丢失时需要重传的段数目，当段丢失频繁时，sack是很有益的。

/proc/sys/net/ipv4/tcp\_dsack   对应net.ipv4.tcp\_dsack  
    是对sack的改进，能够检测不必要的重传。

/proc/sys/net/ipv4/tcp\_fack    对应net.ipv4.tcp\_fack  
    对sack协议加以完善，改进tcp的拥塞控制机制。

**TCP的连接管理：**  
/proc/sys/net/ipv4/tcp\_max\_syn\_backlog    对应net.ipv4.tcp\_max\_syn\_backlog  
    每一个连接请求(SYN报文)都需要排队，直至本地服务器接收，该变量就是控制每个端口的 TCP SYN队列长度的。如果连接请求多余该值，则请求会被丢弃。

/proc/sys/net/ipv4/tcp\_syn\_retries    对应net.ipv4.tcp\_syn\_retries  
    控制内核向某个输入的SYN/ACK段重新发送相应的次数，低值可以更好的检测到远程主机的连接失败。可以修改为3

/proc/sys/net/ipv4/tcp\_retries1    对应net.ipv4.tcp\_retries1  
    该变量设置放弃回应一个tcp连接请求前，需要进行多少次重试。

/proc/sys/net/ipv4/tcp\_retries2    对应net.ipv4.tcp\_retries2  
    控制内核向已经建立连接的远程主机重新发送数据的次数，低值可以更早的检测到与远程主机失效的连接，因此服务器可以更快的释放该连接，可以修改为5

**TCP连接的保持：**  
/proc/sys/net/ipv4/tcp\_keepalive\_time        对应net.ipv4.tcp\_keepalive\_time  
    如果在该参数指定的秒数内连接始终处于空闲状态，则内核向客户端发起对该主机的探测

/proc/sys/net/ipv4/tcp\_keepalive\_intvl    对应net.ipv4.tcp\_keepalive\_intvl  
    该参数以秒为单位，规定内核向远程主机发送探测指针的时间间隔

/proc/sys/net/ipv4/tcp\_keepalive\_probes   对应net.ipv4.tcp\_keepalive\_probes  
    该参数规定内核为了检测远程主机的存活而发送的探测指针的数量，如果探测指针的数量已经使用完毕仍旧没有得到客户端的响应，即断定客户端不可达，关闭与该客户端的连接，释放相关资源。

/proc/sys/net/ipv4/ip\_local\_port\_range   对应net.ipv4.ip\_local\_port\_range  
    规定了tcp/udp可用的本地端口的范围。

**TCP连接的回收：**  
/proc/sys/net/ipv4/tcp\_max\_tw\_buckets     对应net.ipv4.tcp\_max\_tw\_buckets  
   该参数设置系统的TIME\_WAIT的数量，如果超过默认值则会被立即清除。

/proc/sys/net/ipv4/tcp\_tw\_reuse           对应net.ipv4.tcp\_tw\_reuse  
   该参数设置TIME\_WAIT重用，可以让处于TIME\_WAIT的连接用于新的tcp连接

/proc/sys/net/ipv4/tcp\_tw\_recycle         对应net.ipv4.tcp\_tw\_recycle  
   该参数设置tcp连接中TIME\_WAIT的快速回收。

/proc/sys/net/ipv4/tcp\_fin\_timeout       对应net.ipv4.tcp\_fin\_timeout  
   设置TIME\_WAIT2进入CLOSED的等待时间。

/proc/sys/net/ipv4/route/max\_size  
   内核所允许的最大路由数目。

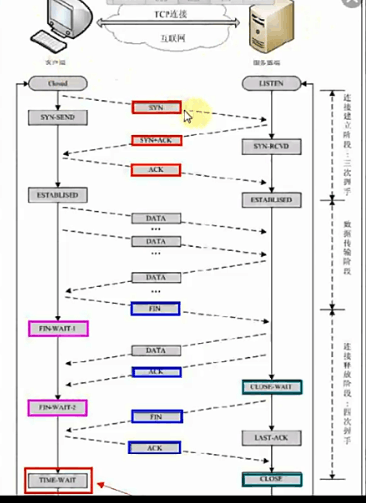
/proc/sys/net/ipv4/ip\_forward  
   接口间转发报文

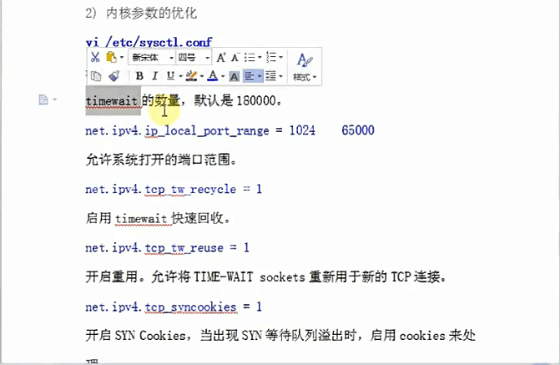
/proc/sys/net/ipv4/ip\_default\_ttl  
   报文可以经过的最大跳数

**虚拟内存参数：**  
/proc/sys/vm/  
  
  
   在linux kernel 2.6.25之前通过ulimit -n(setrlimit(RLIMIT\_NOFILE))设置每个进程的最大打开文件句柄数不能超过NR\_OPEN(1024\*1024),也就是100多w(除非重新编译内核)，而在25之后，内核导出了一个sys接口可以修改这个最大值/proc/sys/fs/nr\_open。shell里不能直接更改，是因为登录的时候pam已经从limits.conf中设置了上限，ulimit命令只能在低于上限的范围内发挥了。

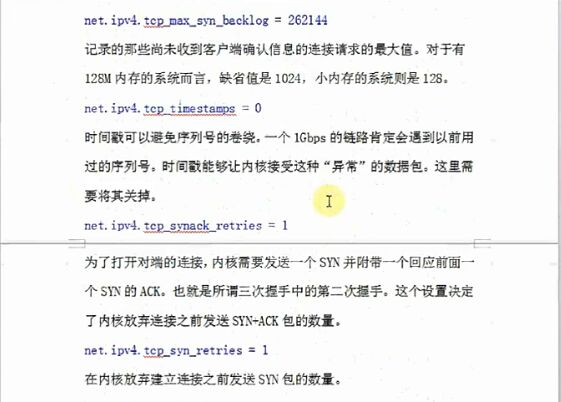
**Linux中查看socket状态：**  
cat /proc/net/sockstat #（这个是ipv4的）  
  
sockets: used 137  
TCP: inuse 49 orphan 0 tw 3272 alloc 52 mem 46  
UDP: inuse 1 mem 0  
RAW: inuse 0  
FRAG: inuse 0 memory 0  
说明：  
sockets: used：已使用的所有协议套接字总量  
TCP: inuse：正在使用（正在侦听）的TCP套接字数量。其值≤ netstat –lnt | grep ^tcp | wc –l  
TCP: orphan：无主（不属于任何进程）的TCP连接数（无用、待销毁的TCP socket数）  
TCP: tw：等待关闭的TCP连接数。其值等于netstat –ant | grep TIME\_WAIT | wc –l  
TCP：alloc(allocated)：已分配（已建立、已申请到sk\_buff）的TCP套接字数量。其值等于netstat –ant | grep ^tcp | wc –l  
TCP：mem：套接字缓冲区使用量（单位不详。用scp实测，速度在4803.9kB/s时：其值=11，netstat –ant 中相应的22端口的Recv-Q＝0，Send-Q≈400）  
UDP：inuse：正在使用的UDP套接字数量  
RAW：  
FRAG：使用的IP段数量

参考：http://www.mjmwired.net/kernel/Documentation/sysctl/









TCP/IP的设计者如此设计，主要原因有两个：

　　防止上一次连接中的包迷路后重新出现，影响新的连接（经过2MSL时间后，上一次连接中所有重复的包都会消失）。

　　为了可靠地关闭TCP连接。主动关闭方发送的最后一个ACK（FIN）有可能会丢失，如果丢失，被动方会重新发FIN，这时如果主动方处于CLOSED状态，就会响应RST而不是ACK。所以主动方要处于TIME\_WAIT状态，而不能是CLOSED状态。另外，TIME\_WAIT并不会占用很大的资源，除非受到攻击。

　　在Squid服务器中可输入查看当前连接统计数的命令，如下所示：

|  |
| --- |
| * netstat -n| awk '/^tcp/ {++S［$NF］} END {for(a in S) print a, S［a］}' * LAST\_ACK 14 * SYN\_RECV 348 * ESTABLISHED 70 * FIN\_WAIT1 229 * FIN\_WAIT2 30 * CLOSING 33 * TIME\_WAIT 18122 |

　　CLOSED：无连接是活动的或正在进行中的。

　　LISTEN：服务器在等待进入呼叫。

　　SYN\_RECV：一个连接请求已经到达，等待确认。

　　SYN\_SENT：应用已经开始，打开一个连接。

　　ESTABLISHED：正常数据传输状态。

　　FIN\_WAIT1：应用说它已经完成。

　　FIN\_WAIT2：另一边已同意释放。

　　CLOSING：两边同时尝试关闭。

　　TIME\_WAIT：另一边已初始化一个释放。

　　LAST\_ACK：等待所有分组死掉。

　　也就是说，这条命令可以把当前系统的网络连接状态分类汇总。

　　在**Linux**下高并发的Squid服务器中，TCP TIME\_WAIT套接字数量经常可达两三万，服务器很容易就会被拖死。不过，我们可以通过修改Linux内核参数来减少Squid服务器的TIME\_WAIT套接字数量，命令如下所示：

|  |
| --- |
| vim /etc/sysctl.conf |

　　然后，增加以下参数：

|  |
| --- |
| * net.ipv4.tcp\_fin\_timeout = 30 * net.ipv4.tcp\_keepalive\_time = 1200 * net.ipv4.tcp\_syncookies = 1 * net.ipv4.tcp\_tw\_reuse = 1 * net.ipv4.tcp\_tw\_recycle = 1 * net.ipv4.ip\_local\_port\_range = 1024 65000 * net.ipv4.tcp\_max\_syn\_backlog = 8192 * net.ipv4.tcp\_max\_tw\_buckets = 5000 |

分类: [TCP/IP](http://www.cnblogs.com/zengkefu/category/724337.html" \t "http://www.cnblogs.com/zengkefu/p/_blank)

本文内容包括****Linux****下对于 ****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。

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)。

清单 3. 调优：在 TCP/IP 栈中启用 IP 转发

1. [root@camus]# cat /proc/sys/net/ipv4/ip\_forward
3. 0
5. [root@camus]# echo "1" **>** /poc/sys/net/ipv4/ip\_forward
7. [root@camus]# cat /proc/sys/net/ipv4/ip\_forward
9. 1
11. [root@camus]#

表 1 给出了几个可调节的参数，它们可以帮助您提高 Linux TCP/IP 栈的性能。

|  |  |  |
| --- | --- | --- |
| 可调节的参数 | 默认值 | 选项说明 |
| |  |  |  | | --- | --- | --- | |  |  |  | | /proc/sys/net/core/rmem\_default |  |  | | "110592" | 定义默认的接收窗口大小；对于更大的BDP 来说，这个大小也应该更大。 |
| /proc/sys/net/core/rmem\_max | "110592" | 定义接收窗口的最大大小；对于更大的 BDP 来说，这个大小也应该更大。 |
| /proc/sys/net/core/wmem\_default | "110592" | 定义默认的发送窗口大小；对于更大的 BDP 来说，这个大小也应该更大。 |
| /proc/sys/net/core/wmem\_max | "110592" | 定义发送窗口的最大大小；对于更大的 BDP 来说，这个大小也应该更大。 |
| |  |  | | --- | --- | | /proc/sys/net/ipv4/tcp\_window\_scaling |  | | |  |  | | --- | --- | |  | "1" | | 启用 RFC 1323定义的 window scaling；要支持超过 64KB 的窗口，必须启用该值。 |
| /proc/sys/net/ipv4/tcp\_sack | "1" | 启用有选择的应答（Selective Acknowledgment），这可以通过有选择地应答乱序接收到的报文来提高性能（这样可以让发送者只发送丢失的报文段）；（对于广域网通信来说）这个选项应该启用，但是这会增加对CPU 的占用。 |
| /proc/sys/net/ipv4/tcp\_fack | "1" | 启用转发应答（Forward Acknowledgment），这可以进行有选择应答（SACK）从而减少拥塞情况的发生；这个选项也应该启用。 |
| /proc/sys/net/ipv4/tcp\_timestamps | "1" | 以一种比重发超时更精确的方法（请参阅 RFC 1323）来启用对 RTT 的计算；为了实现更好的性能应该启用这个选项。 |
| /proc/sys/net/ipv4/tcp\_mem | "24576 32768 49152" | 确定 TCP 栈应该如何反映内存使用；每个值的单位都是内存页（通常是4KB）。第一个值是内存使用的下限。第二个值是内存压力模式开始对缓冲区使用应用压力的上限。第三个值是内存上限。在这个层次上可以将报文丢弃，从而减少对内存的使用。对于较大的 BDP 可以增大这些值（但是要记住，其单位是内存页，而不是字节）。 |
| /proc/sys/net/ipv4/tcp\_wmem | "4096 16384 131072" | 为自动调优定义每个 socket 使用的内存。第一个值是为socket 的发送缓冲区分配的最少字节数。第二个值是默认值（该值会被wmem\_default 覆盖），缓冲区在系统负载不重的情况下可以增长到这个值。第三个值是发送缓冲区空间的最大字节数（该值会被wmem\_max 覆盖）。 |
| /proc/sys/net/ipv4/tcp\_rmem | "4096 87380 174760" | 与 tcp\_wmem类似，不过它表示的是为自动调优所使用的接收缓冲区的值。 |
| /proc/sys/net/ipv4/tcp\_low\_latency | "0" | 允许 TCP/IP 栈适应在高吞吐量情况下低延时的情况；这个选项应该禁用。 |

 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。