**LINUX服务器TCP连接数调优**

假设一台Linux服务器有上百万的请求数，服务器的连接数高有设置好的话，很容易出现服务器负载不过来的问题，本文我们来看看如何优化Linux的连接数来应付大数据请求。

默认的Linux服务器文件描述符等打开最大是1024，用 ulimit -a 查看：

[viewuser@~]$ ulimit -a  
core file size          (blocks, -c) 0   #coredump 文件大小  
data seg size           (kbytes, -d) unlimited  
scheduling priority             (-e) 0  
file size               (blocks, -f) unlimited  
pending signals                 (-i) 255622  
max locked memory       (kbytes, -l) 64  
max memory size         (kbytes, -m) unlimited  
open files                      (-n) 1024   #打开文件数量，root账户无限制  
pipe size            (512 bytes, -p) 8  
POSIX message queues     (bytes, -q) 819200  
real-time priority              (-r) 0  
stack size              (kbytes, -s) 8192  
cpu time               (seconds, -t) unlimited  
max user processes              (-u) 4096   #root用户本项是无限  
virtual memory          (kbytes, -v) unlimited  
file locks                      (-x) unlimited

如果超过了连接数量，可以在 /var/log/message 里面看到类似：

May 14 16:13:52 hostname kernel: nf\_conntrack: table full, dropping packet

的信息，基本可以判定是fd不够用引起的。（服务器受到攻击也会有这个信息）

设置要求：假设我们要设置为200W最大打开文件描述符

**1、修改 nr\_open 限制** （用途：能够配置nofile最大数）

cat /proc/sys/fs/nr\_open

Linux 内核 2.6.25 以前，在内核里面宏定义是1024\*1024，最大只能是100w（1048576），所以不要设置更大的值，如果Linux内核大于 2.6.25 则可以设置更大值。

设置方法：

sudo bash -c ‘echo 2000000 > /proc/sys/fs/nr\_open’

注意：只有修改了 nr\_open 限制，才能修改下面的限制。（如果 nr\_open 的默认现有值如果高于我们的200w，那么可以不用修改）

**2、打开文件描述符限制：**修改 limits.conf 的nofile软硬打开文件限制（用途：tcp连接数）

(1) 临时生效

如果想针对当前登陆session临时生效文件描述符修改，可以直接使用 ulimit 命令：

ulimit -SHn 2000000

再执行相应的程序就能够正常使用不会超过限制，但是重启服务器会失效。  
如果想一直生效，可以把这个内容保存到启动里面，同步到 ： /etc/rc.local 文件

sudo echo “ulimit -SHn 2000000” >> /etc/rc.local

注意：如果需要让 /etc/rc.local 下次启动生效，务必记得有该文件必须有执行权限：sudo chmod +x /etc/rc.local

下次启动会自动执行这句，也是可以正常使用的。

(2) 永久生效

文件位置：/etc/security/limits.conf

查找 nofile ，如果没有，则在自己最后加上：

2.6.25 及以前内核设置为100W：

\*        soft     nofile  1000000  
\*        hard    nofile  1000000

2.6.25 以后版本内核可以设置为200W：

\*        soft     nofile  2000000  
\*        hard    nofile  2000000

设置后保存本文件。（本操作必须重启才生效，如果无法重启，会无法生效，不确定是否使用 /sbin/sysctl -p 是否可以直接生效）

说 明：如果需要 limits.conf生效，有部分需要加载/lib/security/pam\_limits.so才能生效（默认情况一般不关心），如果需要关注，则 需要在 /etc/pam.d/login 在末尾追加 session required /lib/security/pam\_limits.so ，但是目前新版内核应该都没问题问题，可以忽略。

**3、打开进程限制：**修改 limits.conf 中的nproc限制 （用途：进程数）

说明：如果你对进程总数量没有特殊要求，可以不修改本选项，如果你是一个高性能多进程的server，需要很多进程来处理，那么可以修改本选项。  
ulimit -a 里可以看到 max user processes 如果值是比较大的，可以不用设置 nproc 项。  
配置文件：/etc/security/limits.d/20-nproc.conf  （RHEL 7/CentOS 7，如果是 RHEL6.x/CentOS6.x 文件在 /etc/security/limits.d/90-nproc.conf）

\*          soft    nproc  4096  
root     soft    nproc  unlimited

就是root无限（实际root用户限制是：255622），其他非root用户是4096个进程。

说明：

硬限制表明soft限制中所能设定的最大值。 soft限制指的是当前系统生效的设置值。 hard限制值可以被普通用户降低。但是不能增加。 soft限制不能设置的比hard限制更高。 只有root用户才能够增加hard限制值。  
当增加文件限制描述，可以简单的把当前值双倍。 例子如下， 如果你要提高默认值1024， 最好提高到2048， 如果还要继续增加， 就需要设置成4096。

**4、修改 file-max 选项**（用途：可分配文件句柄数目）

file-max 价值：指定了可以分配的文件句柄的最大数目（可以使用 /proc/sys/fs/file-nr 文件查看到当前已经使用的文件句柄和总句柄数。）

(1) 临时生效：

文件路径：/proc/sys/fs/file-max

cat /proc/sys/fs/file-max

3252210

如果要修改，直接覆盖文件：（比如改成200w）

sudo echo 2000000 > /proc/sys/fs/file-max

注意：如果你想每次启动都自动执行上面的命令，可以在系统启动配置文件/etc/rc.local里面添加一句命令：（跟永久生效差不多）

echo 2000000 > /proc/sys/fs/file-max

或者直接Shell全搞定：

echo “echo 2000000 > /proc/sys/fs/file-max” >> /etc/rc.local

注意：如果需要让 /etc/rc.local 下次启动生效，务必记得有该文件必须有执行权限：sudo chmod +x /etc/rc.local

(2) 永久生效：

修改配置文件，文件位置：/etc/sysctl.conf

打开配置文件到最末尾，如果配置文件里没有则可以直接添加：

sudo echo “fs.file-max = 2000000” >>/etc/sysctl.conf

配置文件生效：sudo /sbin/sysctl -p

**5、修改TCP等相关选项**

配置文件：/etc/sysctl.conf

修改选项：

net.core.somaxconn = 2048  
net.core.rmem\_default = 262144  
net.core.wmem\_default = 262144  
net.core.rmem\_max = 16777216  
net.core.wmem\_max = 16777216  
net.core.netdev\_max\_backlog = 20000  
net.ipv4.tcp\_rmem = 4096 4096 16777216  
net.ipv4.tcp\_wmem = 4096 4096 16777216  
net.ipv4.tcp\_mem = 786432 2097152 3145728  
net.ipv4.tcp\_max\_syn\_backlog = 16384  
net.ipv4.tcp\_fin\_timeout = 30  
net.ipv4.tcp\_keepalive\_time = 300  
net.ipv4.tcp\_max\_tw\_buckets = 5000  
net.ipv4.tcp\_tw\_reuse = 1  
net.ipv4.tcp\_tw\_recycle = 1  
net.ipv4.tcp\_syncookies = 1  
net.ipv4.tcp\_max\_orphans = 131072  
fs.file-max = 2000000  
fs.inotify.max\_user\_watches = 16384  
net.netfilter.nf\_conntrack\_max = 6553500   #本选项在一些版本下无效，可以删除  
net.netfilter.nf\_conntrack\_tcp\_timeout\_established = 1200     #本选项在一些版本下无效，可以删除

配置文件生效：sudo /sbin/sysctl -p

以上选项也可以直接给 /proc/sys/net/ 目录下面按照各个选项可以直接使用类似于 echo VALUE > /proc/sys/net/core/wmem\_max 来直接修改内存临时值生效。

主要看这几项：

net.ipv4.tcp\_rmem 用来配置读缓冲的大小，三个值，第一个是这个读缓冲的最小值，第三个是最大值，中间的是默认值。我们可以在程序中修改读缓冲的大小，但是不能超过最小与最 大。为了使每个socket所使用的内存数最小，我这里设置默认值为4096；  
net.ipv4.tcp\_wmem 用来配置写缓冲的大小。读缓冲与写缓冲在大小，直接影响到socket在内核中内存的占用；  
net.ipv4.tcp\_mem 则是配置tcp的内存大小，其单位是页，而不是字节。当超过第二个值时，TCP进入 pressure模式，此时TCP尝试稳定其内存的使用，当小于第一个值时，就退出pressure模式。当内存占用超过第三个值时，TCP就拒绝分配 socket了，查看dmesg，会打出很多的日志“TCP: too many of orphaned sockets”；  
net.ipv4.tcp\_max\_orphans 这个值也要设置一下，这个值表示系统所能处理不属于任何进程的 socket数量，当我们需要快速建立大量连接时，就需要关注下这个值了。当不属于任何进程的socket的数量大于这个值时，dmesg就会看 到”too many of orphaned sockets”；  
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?默认的TIMEOUT时间；  
net.ipv4.tcp\_max\_syn\_backlog 进入SYN包的最大请求队列.默认1024.对重负载服务器,增加该值显然有好处.可调整到16384；  
net.ipv4.tcp\_keepalive\_time = 300 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300秒；  
net.ipv4.tcp\_max\_tw\_buckets = 5000 表示系统同时保持TIME\_WAIT套接字的最大数量，如果超过这个数字，TIME\_WAIT套接字将立刻被清除并打印警告信息。默认为180000，改为5000；  
fs.file-max = 2000000 是指能够打开的文件描述符的最大数量，如果系统报错：too many file opened，就需要修改本值（本值必须跟 /etc/security/limits.conf 一块修改才生效）；  
fs.inotify.max\_user\_watches = 16384 设置文件系统变化监听上线。如果在没满各种正常的情况下，还出现tail -f这种watch事件报错No space left on device就是这个值不够了；

注意：如果是客户端程序，为了更好的访问server程序不是卡在端口分配上，建议把客户端的端口（port\_range）范围开大一些：

修改文件：/etc/sysctl.conf

net.ipv4.ip\_local\_port\_range = 1024 65535

配置生效：sudo /sbin/sysctl -p

如果是客户端，其他文件打开限制等可以参考上面的来设置。

**6、其他一些配置**

(1) 打开core文件

如果为了观察程序是否正常，出现问题后生成相应映像文件，可以开启coredump相关的操作，可以打开：（非必须，如果线上环境，担心影响稳定性，可以考虑不开启）

配置文件：/etc/security/limits.conf

修改配置文件：

增加：

\*        soft     core   102400  
\*        hard    core   2048003

建议设置为无限大小：

\*        soft     core   unlimited  
\*        hard    core   unlimited

然后重启机器生效（不确定是否可以使用 /sbin/sysctl -p 生效），使用： ulimit -a 或 ulimit -c 查看结果，后续如果程序出现栈溢出等都会生成coredump文件，方便用gdb等追查问题原因。

(2) 修改其他 limits.conf 配置

如 果想临时当前会话里让 /etc/security/limits.conf 生效，可以直接使用 ulimit 命令进行修改，在当前session就直接生效（退出登陆或者重启失效，为了永久生效，必须直接修改 /etc/security/limits.conf 文件）

ulimit -SHc unlimited   #修改coredump文件大小，修改完当前session就生效了，启动的程序都会直接可用这个新配置值  
ulimit -SHn 10000000  #修改打开文件数量限制为100W，修改完当前session就生效  
ulimit -SHu 4096  #修改当前用户打开进程数量限制为4096个，修改完后当前session直接生效

#修改完成后使用 ulimit -a 可以查看修改的效果，需要用就生效修改 /etc/security/limits.conf 文件，然后重启服务器生效 #

**优化Linux的内核参数来提高服务器并发处理能力**

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?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\_timestsm[ps](http://www.111cn.net/fw/photo.html) = 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。