Tag Archives: tcp_max_tw_buckets

DEC
29
2014

Linux Tuning For SIP Routers – Part 4 (Networking)

.

Introduction

This is Part 4 of “Linux Tuning For SIP Routers” topic. In Part 1 i have talked about *Interrupts and IRQ Tuning*. In Part 2 i ha talked about *File System Tuning*, *Journaling File System, and Swappiness* . In Part 3 i have talked about *OOM killer*, *Priva Momory*, *and Shared Memory Tuning*. In this part i will talk about *Network tuning*.

Network Adapter Settings

To check the setting of your network card, type: “ethtool X” where X is the interface name. The tool “ethtool” is used to read a write the settings of the network card. To update the settings:

- Find the name of the network interface: # ifconfig
- Get the current settings: # ethtool X
- Do your change using ethtool (See the man page: # man ethtool). For example to change the speed to 1000, and t mode to full duplex: # ethtool -s p4p2 speed 1000 duplex full autoneg off (where p4p2 is my interface name).

Note: Changes must be supported by the network adapter, otherwise it will give an error message “Cannot set new setting Invalid argument”

To make the changes permanent for p4p2, set the environment variable ETHTOOL_OPTS:

- If you are using bash : # vi /etc/sysconfig/network-scripts/ifcfg-p4p1 (note “ifcfg-p4p1” will be different)
- Add/update this line: ETHTOOL_OPTS=speed 1000 duplex full
- Restart the network service: # systemctl restart network.service

“txqueuelen” Parameter

The **kernel** parameter “txqueuelen” is the size of the transmission queue related to the interface. The default value is 10 **frames**. The kernel stores the departing frames in this queue (the frames are not loaded onto the NIC’s buffer yet). Tuning th value is important to avoid loosing frames due to the lack of space in the transmission queue. Use high value (long queue) f high speed interfaces (≈> 10 Gb).

To get the current value of txqueuelen: # ifconfig p4p2 where p4p2 is the interface name. The output will look like:

> p4p2: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500 ether 0c:54:a5:08:45:a7 **txqueuelen 1000** (Ethernet) F packets 0 bytes 0 (0.0 B) RX errors 0 dropped 0 overruns 0 frame 0 TX packets 0 bytes 0 (0.0 B) TX errors 0 droppec overruns 0 carrier 0 collisions 0 device interrupt 19

You can also execute “# ip link” to get the txqueuelen value (qlen 1000).

To temporarily change the value to 3000 (for ≈10 Gb card): # ifconfig p4p2 txqueuelen 3000

To permanently change it, add/change the command “/sbin/ifconfig p4p2 txqueuelen 3000” in the file “/etc/rc.d/rc.local”. If this f is not exist, do the following if you have Fedora Red Hat:

- # vi /etc/rc.d/rc.local
- # chmod a+x /etc/rc.d/rc.local
- At the top of the file, add the bash interpreter: #!/bin/sh
- At the end of the file add exit 0
- Between the previous two lines add the command: /sbin/ifconfig p4p2 txqueuelen 3000
- Save the file and reboot (The command will be executed at system startup).

“netdev_max_backlog” Parameter

The **kernel** parameter “netdev_max_backlog” is the maximum size of the receive queue. The received frames will be stored this queue after taking them from the ring buffer on the NIC. Use high value for high speed cards to prevent loosing packets. real time application like SIP router, long queue must be assigned with high speed CPU otherwise the data in the queue will out of date (old).

To get the current value of netdev_max_backlog: # /proc/sys/net/core/netdev_max_backlog. The default value is 1000 frames.

To temporarily change the size to 300000 frame: # echo 300000 > /proc/sys/net/core/netdev_max_backlog

To permanently change the value:

- Edit the file “/etc/sysctl.conf“
- Add the line: net.core.netdev_max_backlog=300000
- Load the change which you did: # sysctl -p

Note

- Do a test before going to the deployment.
- To change the ring buffer size on the NIC: # ethtool -g p4p2 where p4p2 is the interface name. This operation must supported by the NIC.

Kernel Parameters For the Socket Buffer

Usually SIP uses TCP or UDP to carry the SIP signaling messages over the internet (<=> TCP/UDP sockets). The receive buf (socket receive buffer) holds the received data until it is read by the application. The send buffer (socket transmit buffer) holds t data until it is read by the underling protocol in the network stack.

The **default (initial)** and maximum size (In bytes) of the receive socket buffer:

```
# cat /proc/sys/net/core/rmem_default

# cat /proc/sys/net/core/rmem_max
```

You can manipulate this value in the application by calling the function “setsockopt” and specify the option name “SO_RCVBU with the socket level set to SOL_SOCKET. When you specify a value for this option, the kernel will doubles this value. So t doubled value should be less than the maximum (rmem_max).

The **default (initial)** and maximum size (In bytes) of the send socket buffer:

```
# cat /proc/sys/net/core/wmem_default # cat /proc/sys/net/core/wmem_max
```

You can manipulate this value in the application by calling the function “setsockopt” with the option name “SO_SNDBUF”. Choo the value which is suitable to he speed of the network card. For example, to set the maximum to 10 MB (10 * 1024 * 10 Bytes)=10,485,760 for the receive buffer and send buffer:

Temporarily:

```
# echo 10485760 > /proc/sys/net/core/rmem_max

# echo 10485760 > /proc/sys/net/core/wmem_max
```

Permanently:

- Edit the file “/etc/sysctl.conf“

- Add the lines:

    net.core.rmem_max=10485760

    net.core.wmem_max=10485760

- Save the file and reload the changes: # sysctl -p

To change the defaults:

Temporarily:

    # echo 10485760 > /proc/sys/net/core/rmem_default

    # echo 10485760 > /proc/sys/net/core/wmem_default

Permanently:

- Edit the file "/etc/sysctl.conf"
- Add the lines:

    net.core.rmem_default=10485760

    net.core.wmem_default=10485760

- Save the file and reload the changes: # sysctl -p

Also You need to set the minimum size, initial size, and maximum size for the protocols. To read the current values (Assumi IPv4 and TCP):

    # cat /proc/sys/net/ipv4/tcp_rmem

The output look like this: 4096 87380 6291456 (min, initial, max). To read write buffer settings: # c /proc/sys/net/ipv4/tcp_wmem

You can do the same for UDP (udp_rmem and udp_wmem) **To change the values:** Temporarily:

    # echo 10240 87380 10485760 > /proc/sys/net/ipv4/tcp_rmem

    # echo 10240 87380 10485760 > /proc/sys/net/ipv4/tcp_rmem

Permanently:

- Edit the file "/etc/sysctl.conf"
- Add the lines:

    net.ipv4.tcp_rmem= 10240 87380 10485760

    net.ipv4.tcp_wmem= 10240 87380 10485760

- Save the file and reload the changes: # sysctl -p

TCP Kernel Parameters

- Disable TCP timestamp (RFC 1321): TCP timestamp feature allows round trip time measurement (<=> Adding 8 bytes TCP header). To avoid this overhead we disable this feature: net.ipv4.tcp_timestamps = 0

- Enable window scaling: net.ipv4.tcp_window_scaling = 1
- Disable select acknowledgements (SACK): net.ipv4.tcp_sack = 1
- Disable cache metrics so the initial conditions of the closed connections will not be saved to be used in near futu connections:

  net.ipv4.tcp_no_metrics_save = 1

  '1' means disable caching.

- Tune the value of "backlog" (maximum queue length of pending connections "Waiting Acknowledgment"):

  tcp_max_syn_backlog= 300000

- Set the value of *somaxconn*. This is the Max value of the backlog. The default value is 128. If the *backlog* is greater than somaxconn, it will truncated to it.

  Temporarily: # echo 300000 > /proc/sys/net/core/somaxconn

  Permanently: Add the line: net.core.somaxconn= 300000 in the file /etc/sysctl.conf. Reload the change (# sysctl -p).

- " TIME_WAIT" TCP socket state is the state where the socket is closed but waiting to handle the packets which are still the network. The parameter tcp_max_tw_buckets is the maximum number of sockets in "TIME_WAIT" state. After reachi this number the system will start destroying the socket in this state. To get the default value:

  # cat /proc/sys/net/ipv4/tcp_max_tw_buckets.

Increase the value to 2000000 (2 million). Increasing this value leads to requirement of more memory.

Temporarily:

echo 2000000 > /proc/sys/net/ipv4/tcp_max_tw_buckets

Permanently: Add the line: net.ipv4.tcp_max_tw_buckets= 2000000 in the file /etc/sysctl.conf. Reload the change (# sysctl -p).

- Other TCP parameters that you can change as above:

  net.ipv4.tcp_keepalive_time (how often the keepalive packets will be sent to keep the connection alive).

  net.ipv4.tcp_keepalive_intvl (time to wait for a reply on each keepalive probe).

  net.ipv4.tcp_retries2 (how many times to retry before killing an alive TCP connection).

  net.ipv4.tcp_syn_retries (how many times to retransmit the initial SYN packet).

Number of Open Files

When the SIP router is high loaded, this means it has a lot of open sockets (Each socket is a file). To increase the number sockets we increase the number of open files (<=> number of file handles). To know the maximum number of file handles on t entire system:

cat /proc/sys/fs/file-max. The output will be a number: 391884

To get the current usage of file handles: # cat /proc/sys/fs/file-nr. The output will look like: 10112 0 391884 First numb (10112) is the total number of allocated file handles. Second number (0) is the currently used file handles (2.4 kkernel) currently unused file handles (2.6 kernel). Third number (391884) is the maximum number of file handles.

To change the maximum number of open files temporarily: # echo 65536 > /proc/sys/fs/file-max

To change the maximum number of open files permanently: Add the line “fs.file-max=65536” in the file “/etc/sysctl.conf”.

SHELL Limits

The maximum number of open files “/proc/sys/fs/file-max” is for the whole system. This number is for all users so if you want allow specific user (lets say opensips) to open X files where X < file-max, you can use “ulimit” to do that:

- As **root**, edit the file: # vi /etc/security/limits.conf
- Add/Change the following lines:

opensips	**soft**	nofile	**4096**
opensips	**hard**	nofile	**40000**

At the start time, the opensips can open till 4096 files (soft limit). If opensips gets an error message about running out of f handles, then opensips user can execute “ulimit -n New_Number” to increase the number of file handles to New_Number. Tl New_Number must be less than the hard limit (the second line) and the hard limit must be less than the file-max.

To **avoid** getting the error message (“out of file handles”), let opensips in its start scripts execute “ulimit -n New_Numbe Choosing the New_Number depends on your test to know how much your opensips needs. You can as root set permanently “s limit=hard limit = New_Number” in the file ” # vi /etc/security/limits.conf”.

Do not set the hard limit to a value equal to /proc/sys/fs/file-max, then the entire system might run out of file handles. Rememb file-max is for the whole system.

Note

- Don’t use the numbers in this article. Do your tests on your system.
- Use network performance testing tools like netperf, pktgen, and mpstat.
- You always have to check how to optimize your hardware:
  - If the Network Interface Card support Flow Control (e.g e1000 network interface card).
  - Some NIC drivers support interrupt coalescing (multiple interrupts can be coalesced into one kernel interrupts). If your NIC support this, you can activate it as following: # ethtool -C p4p2 where p4p2 is the interface.

More Information

- setsockopt && getsockopt Man Page

By Binan AL Halabi • Posted in Linux, Testing & Performance • Tagged /proc/sys/net/core/, autoneg, duplex, ethtool, ETHTOOL_OPTS, file handles, file handles usage, file-max, file-nr, hard limit, interrupt coalescing, net.core., net.ipv4.tcp_sack, net.ipv4.tcp_timestamps, netdev_max_backlog, network adapter, network stack, network-scripts, open files, rmem_default, rmem_max, SEMMNI, signaling, sip, soft limit, SOL_SOCKET, somaxconn, SO_RCVBUF, SO_SNDBUF, speed, sysctl, TCP, TCP timestamp, tcp_max_syn_backlog, tcp_max_tw_buckets, tcp_no_metrics_save, tcp_rmem, tcp_window_scaling, tcp_wmem, test, TIME_WAIT, txqueuelen, UDP, ulimit, wmem_default, wmem_max