AIX

TCP/IP Performance Tuning for the RISC System/6000

By Rudy Chukran

This article shows AIX administrators how to improve the performance of AIX systems that have significant TCP/IP network workloads.

Users often complain to system administrators that their systems are too slow. This common complaint forces administrators to become proficient in fixing performance problems. Although many administrators think nothing of tackling overloaded systems by applying load balancing techniques and balancing disk data, even some of the best system administrators shudder when someone complains about the sluggishness of the network. The system administrators who are most skilled in the art of network administration are often viewed with awe.

Network administration, particularly network performance tuning, consists of a system administrator applying a logical method of fault isolation, and knowing exactly what to alter. This article discusses how buffer space can affect TCP/IP performance, provides instructions on how to assess the constraints on buffer space, and suggests solutions to the problems posed by constrained space. Finally, this article discusses some topics that can help system administrators when tuning other TCP/IP networks. It describes both specific and general insights into performance tuning the RISC System/6000.

Rudy Chukran

TCP/IP Concepts

Two major protocols in the TCP/IP suite are Transmission Control Protocol (TCP) and User Datagram Protocol (UDP).

TCP is a reliable delivery protocol that implements data transmission pacing, positive acknowledgement, timeout, and retransmission. Data pacing prevents the receiver from becoming flooded with data, resulting in data loss. The receiver determines how rapidly to send data by telling the sender how much receive buffer space is available. Shortage of receive buffer space can degrade TCP throughput by reducing the window size to less than optimum. At the same time, the sender attempts to maintain an optimum amount of data sent, pending acknowledgement; that is, the sender sends multiple data items before receiving the acknowledgment. This technique, called *sliding window acknowledgement*, allows the sender to maximize throughput by keeping the network constantly busy rather than waiting for acknowledgments. The effective throughput of transmitting messages can approach 100% of the network bandwidth. If the receiver does not have enough buffer space, this shortage is communicated to the sender, which can slow down data transmission.

The sender's job is to detect whether the receiver acknowledges receipt or whether data must be resent. Retransmission severely degrades the effective throughput of TCP transmissions. If a successful transmission requires two tries, then the effective throughput is half that of a normal transmission. If the user considers the time it takes to determine whether the transmission has been received, then the effective throughput is reduced still further.

In contrast, UDP is an unreliable delivery protocol since it has no automatic, built-in acknowledgement and retransmission protocol. For reliable transmission, the application program must detect whether the receiver has received a transmission, and resend the information if it has not been

```
TOKEN STATISTICS (tr0) :

Hardware Address: 10:00:5a:a8:83:6c
Transmit Byte Count: 81279.0    Receive Byte Count: 1288514.0
Transmit Frame Count: 724.0     Receive Frame Count: 16602.0
Transmit Error Count: 0         Receive Error Count: 0
Max NetIDs in use: 1            Max Transmits queued: 0
Max Receives queued: 0          Max Stat Blks queued: 0
Interrupts lost: 0              WDT Interrupts lost: 0
Timeout Ints lost: 0            Status lost: 0
Receive Packets Lost: 0         No Mbuf Errors: 0
No Mbuf Extension Errors: 0     Receive Int Count: 19748
Transmit Int Count: 724         Packets Rejected No NetID: 3148
Packets Accepted Valid NetID: 16602     Overflow Packets Received: 0
Packets Transmitted and Adapter Errors Detected: 0
```

Figure 1. Results of a netstat -v command

received. Even in this case, retransmission should be avoided for the same reasons as discussed for TCP. (For a more complete description of TCP/IP concepts, see the following: Comer, Douglas E. *Internetworking with TCP/IP*, Volume I, 2nd ed. Prentice-Hall, 1991.)

AIX Network Bottlenecks

Several different points in the network subsystem can limit network performance. This section examines how data buffering affects throughput at each of these points, from the network device and to the application.

Network Device Hardware Errors

The device driver controlling the network adapter device has both an input and an output queue for data frames. The device driver also keeps track of errors.

The `v` option of the `netstat` command shows network device statistics. Figure 1 shows the results of a `netstat -v` command for a Token-Ring device.

Any indication of errors that may have occurred is important. `Mbuf` errors indicate `mbuf` buffer shortages. The lost items also indicate some type of problem, generally with the network adapter itself. For hardware problems, the user should investigate whether any errors occurred earlier. The `errpt` command can display a one-line summary of each error; for a detailed report, use the `-a` option of `errpt`.

An excerpt of the `errpt -a` command output in Figure 2 shows a potential problem that should be investigated by a hardware maintenance specialist. The specialist should also investigate possible cabling errors that could cause unreliable transmission.

```
ERROR LABEL:    TOK_ESERR
ERROR ID:       FCA960CE

Date/Time:       Sat Nov 27 14:59:47
Sequence Number: 7615
Machine ID:      000071873000
Node ID:         chukran
Class:           S
Type:            TEMP
Resource Name:   tok0

Error Description
EXCESSIVE TOKEN-RING ERRORS
Probable Causes
TOKEN-RING FAULT DOMAIN

Failure Causes
TOKEN-RING FAULT DOMAIN
COMMUNICATIONS PROGRAM
```

Figure 2. Results of an errpt -a command

Network Device Queues

To determine if the network device input and output queues are performing efficiently, look at the next higher layer in the chain—the network interface. `Netstat` with the `-I` option provides statistics for the network interface.

The `Oerrs` column is nonzero. It shows that the network interface code encountered errors sending out data packets to the network device. These errors indicate packet loss because the input queue was too small. Insufficient depth of

```
netstat -I tr0
Name  Mtu   Network     Address         Ipkts    Ierrs
Opkts    Oerrs
tr0   1492  <Link>                      25666365     0
```

Figure 3. Netstat gives statistics for the network interface

```
              Change / Show a Token-Ring Network Interface

Type or select values in entry fields.
Press Enter AFTER making all desired changes.
                                                    [Entry
Fields]
Network Interface Name                               tr0
INTERNET ADDRESS (dotted decimal)
[129.35.17.105]
Network MASK (hexadecimal or dotted decimal)
[255.255.240.0]
Current STATE                                        detach
+
Use Address Resolution Protocol (ARP)?               yes
+
Enable Hardware LOOPBACK Mode?                       no
+
BROADCAST ADDRESS (dotted decimal)                  []
Confine BROADCAST to LOCAL Token-Ring?               no
```

Figure 4. Changing a Token-Ring network interface

```
            Change/Show Characteristics of a Token-Ring
Adapter

Type or select values in entry fields.
Press Enter AFTER making all desired changes.
                                                      [Entry
Fields]
 Token-Ring adapter                                  tok0
 Description                                         Token-Ring
High-Perfor>
 Status                                              Available
 Location                                            00-07
 RECEIVE DATA TRANSFER OFFSET                       [92]
+#
 TRANSMIT queue size                                [30]
+#
 RECEIVE queue size                                 [30]
+#
 STATUS BLOCK queue size                            [10]
```

Figure 5. Changing a Token-Ring adapter

the network device input queue is the most common TCP/IP performance problem that can be solved by increasing queue depth. Since it is not possible to identify the number of packets that were lost at any instant, the simplest approach is to specify the maximum size for both.

First, using the `smit chinet` command, disable the network interface before proceeding. The menu shown in Figure 4 then appears. Specify `detach` as the Current State.

Next, change the appropriate network adapter; for example, changing a Token-Ring adapter is shown in Figure 5.

The default queue size is 30. To find the maximum, move the cursor to the appropriate item and press F4 to identify the values that are allowed. After changing the queue lengths, re-enable the network interface by changing the interface state to "up" using the screen called "Change/Show a Token-Ring Network Interface." If the device is not Token Ring, a different `smit` fastpath must be specified.

IP Input Queue

The next buffer limit to investigate is the IP input queue limit, which can be queried with the `no` command:

```
no -o ipqmaxlen
      ipqmaxlen = 50
```

There is no easy way to query the limit if this queue has been overrun, which results in packet loss. The easiest way to handle a potential IP queue overrun is to make it larger, such as doubling the default size and observing if this has a positive effect on throughput as shown:

```
no -o ipqmaxlen=100
```

TCP Send and Receive Space

Receiving buffers (called TCP send and receive spaces in AIX/6000™) limits the optimum TCP window size. Since insufficient size of TCP receive space limits the window size, it also limits TCP throughput. Fortunately, it is easy to calculate an upper limit on the size of the TCP receive space. First, use the `ping` command to determine the round-trip time of a packet to a typical host. Next, determine the theoretical maximum bandwidth of the transmission medium being used. Then perform this calculation:

maximum window = round trip delay * maximum network bandwidth

Test this equation by substituting typical numbers, such as a round trip delay of 23 ms and an Ethernet bandwidth of 10 Mbits/second = 1.25 MB/second:

maximum window = 23 ms * 1.25 MB/second = 28750 bytes

The window cannot become larger than this, so TCP receive space must be 28750.

A shortage of TCP send space also slows down transmission by causing the sending process to become blocked while waiting for the queued data to be sent. A larger TCP send space enables the socket system call code to copy larger chunks of data from user buffers into kernel buffers. If the TCP send space size matches the receive space size, performance will be optimized.

```
no -o tcp_sendspace = 28750
no -o tcp_recvspace = 28750
```

UDP send and receive spaces serve the same purpose as their TCP counterparts. Since there is no sliding window feature in UDP, shortage of buffer space will have a detrimental effect: loss of data. If the application retransmits this data, the result is a decrease in effective throughput. Change the UDP spaces using the `no` command according to the example provided for TCP spaces.

```
no -o udp_sendspace = 64000
no -o udp_recvspace = 64000
```

Socket Buffer Limits

Increasing either TCP or UDP spaces requires an increase in `sb_max`, a limit on the total buffer space that can be occupied by data for a socket. The `sb_max` value must be greater than either send or receive space; it should be twice the `tcp_sendspace` size for 4 KB socket data transfers and eight times for 512 byte transfers. Determining the amount of data the program is trying to transfer will be discussed later.

Mbuf Network Buffers

Various buffer limits restrict data flow in parts of the network subsystem. The buffer allocation subsystem performs duties for the entire TCP/IP subsystem. The network message buffers are called `mbufs`. The `mbuf` pool has allocation limits that serve as a guide to efficient memory allocation. This section discusses how to determine if these limits should be changed and how to manipulate them.

First, `mbufs` can exist in two states: occupied with pending data or free of data. The sum of these `mbufs` comprises the allocated `mbufs` from application buffers that are unique because they are pinned into RAM. That is, RAM is dedicated to the pinned `mbufs`, and they cannot be paged out to make room for other virtual memory. Allocating more `mbufs` than are truly needed can take RAM from other parts of the system. Over-allocation of `mbufs` makes the total RAM appear smaller.

`Mbufs` can have two sizes. Small `mbufs` (256 bytes) are used for small messages. Larger size `mbuf` clusters (4096 bytes) are used for larger messages. A small message can be placed into a maximum of two small `mbufs`. If the message becomes too large to fit into two `mbufs`, then an `mbuf` cluster will be used.

The limits for `mbufs` can be shown using the `no` command. An edited example of the `no` command output follows:

```
no -a
lowclust = 21
lowmbuf = 188
thewall = 2048
mb_cl_hiwat = 42
```

The total allocated number of `mbufs` cannot exceed the value of `thewall`, which is measured in kilobytes. If the number of free `mbufs` drops below `lowmbuf`, then a special kernel process is dispatched to asynchronously allocate more `mbufs`. If the number of free clusters drops below `lowclust`, the same kernel process tries to allocate more clusters. As occupied clusters become free, the number of free bytes may exceed `mb_cl_hiwat`. When this occurs, the kernel process deallocates clusters to allow the RAM space to be unpinned and subsequently used by other programs. That is, the kernel process will attempt to maintain the free cluster list to range between a high-water and low-water mark. Small `mbufs` are never deallocated because there is no high-water mark defined for them.

Figure 6 shows the different types of allocated `mbufs` and how the high-water and low-water marks are related to one another.

A natural question arises regarding the value of `thewall`. If the value is sufficient, should the kernel process manage the `mbuf` pool automatically without problems? The answer is a qualified "yes" if `mbuf` allocation could happen instantaneously. But in reality, pinning memory can cause page faults that take many milliseconds. During

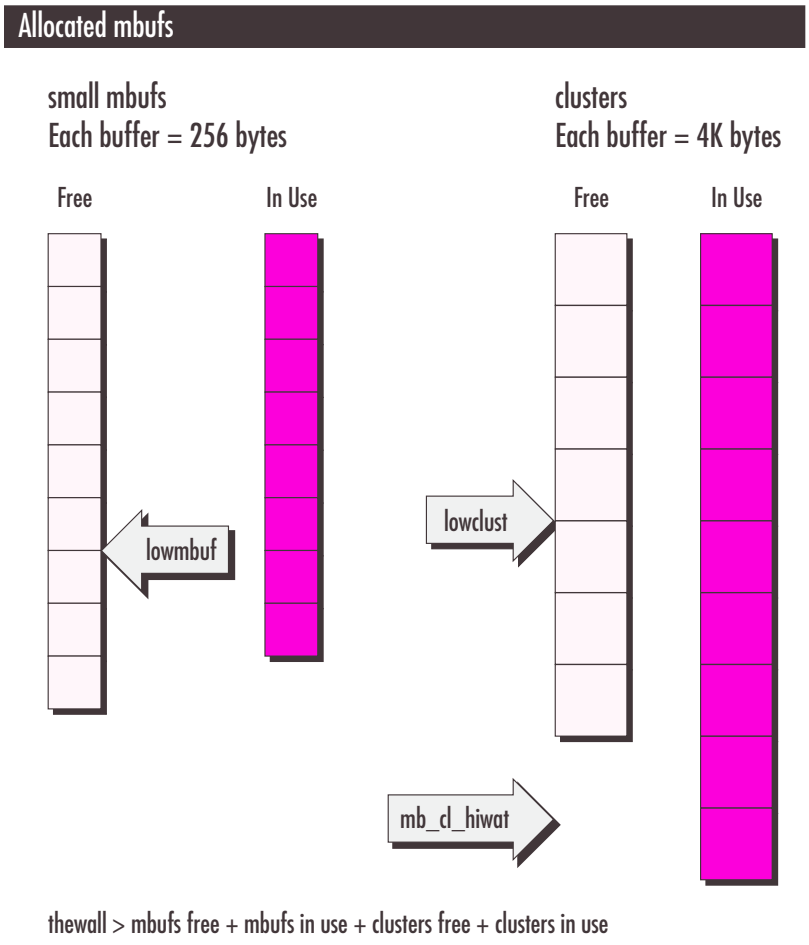

Figure 6. Types of allocated mbufs

```
netstat -m
467 mbufs in use:
       17 mbufs allocated to data
       58 mbufs allocated to packet headers
       139 mbufs allocated to socket structures
       227 mbufs allocated to protocol control blocks
       14 mbufs allocated to routing table entries
       10 mbufs allocated to socket names and addresses
       2 mbufs allocated to interface addresses
16/115 mapped pages in use
576 KB allocated to network (31% in use)
0 requests for memory denied
0 requests for memory delayed
0 calls to protocol drain routines
```

Figure 7. Netstat command using the m option

this time, a large burst of data packets could arrive, deplete the remaining free clusters, and then be lost before the kernel process could allocate enough memory.

The solution is to determine if `mbufs` are being depleted. The `netstat` command is employed using the `m` option, shown in Figure 7.

When requests for memory are denied, a network interrupt routine drops packets because of a depleted `mbuf` or cluster-free list. When requests for memory are delayed, an application blocks on a send awaiting free `mbufs` or clusters. The solution is to increase `thewall`, `lowclust`, and `mb_cl_hiwat`. For the first attempt, try doubling these limits; then determine if this increase eliminates `mbuf` depletion. Several increases may be required to effectively eliminate `mbuf` depletion.

Changing network limits with the `no` command is not permanent; the change is not preserved when the system is rebooted. For the change to be permanent, it is necessary to put any `no` commands that change defaults into a shell script that is invoked every time the system is booted. Place the shell script invocation in the `/etc/rc.net` file.

Netpmon Network Statistics Report

A system statistics tool included in AIX/6000 (`netpmon`) can provide a list of hosts to which the system has connected; information about the traffic flow and data transmission times; and statistics on TCP, UDP, and NFS® connections. Since the report produces pages of data, users can run the report using the following command:

```
netpmon -o netpmon.out
sleep 300
trcstop
```

`Netpmon` can show network traffic statistics sorted by host. It can also detect traffic to other networks. Hosts on the same network communicate directly with each other; hosts on different networks must be routed through a router, which can introduce delays that affect network throughput. Operations to some hosts may seem to perform better than others as a result of comparing same-network operations to operations across the network. `Netpmon` will show host traffic at both the device layers and at the TCP and UDP layers. If hosts appearing in the device section of the `netpmon` report do not appear in the TCP section, the host is an intermediary. It is a good idea to investigate performance of the intermediary host.

MTU Fragmentation

Another factor to consider when dealing with multiple networks is the concept of packet fragmentation caused by different MTU sizes across different networks. A packet traveling through a gateway from one network to a network with a smaller MTU size will be divided into more than one fragment. These fragments will travel through any succeeding networks and be reassembled at the final destination host. This packet fragmentation can put an additional CPU utilization burden on gateways and add to congestion and network delay. To resolve this problem, try configuring routes between hosts on different networks that follow networks using the same MTU sizes. See *Internetworking with TCP/IP* for more details about packet fragmentation.

When MTUs are on the same network, the general rule is that larger is better. Since there is significant CPU utilization overhead to move packets, larger MTU sizes generally produce better effective throughput, but only up to a point. Experimentation is the best way to determine optimal size. *AIX Version 3.2 for RISC System/6000 Performance Monitoring and Tuning Guide* (SC23-2365) discusses which MTU sizes are optimal for which media. Remember that increasing the MTU size will require an increase in send and receive spaces and in `sb_max` to accommodate the larger MTU.

***Rudy Chukran**, IBM Corporation, RISC System/6000 Division, 11400 Burnet Road, Austin, TX 78758. Internet: chukran@austin.ibm.com. Mr. Chukran joined IBM in 1974 and has been working with AIX since 1983. He advises software vendors about topics related to performance and porting. He has a BS in Electrical Engineering from Carnegie-Mellon® University.*

How to Download *AIXpert* Electronic Information

EVIS

IBM's Electronic Vendor Information System (EVIS) is available toll-free at (800) 453-9030. The system supports modem speeds from 300 to 19,200 bps. Communications settings should be N-8-1. To access the limited public information available on EVIS, use the following account:

ID: `vendor`
Password: `14theking`

For articles from *AIXpert*, look in the `aix/aixpert` directory.

Internet in Europe

Use the following procedure to obtain *AIXpert* supplemental information from a European ftp Server:

1. `ftp asterix.fi.upm.es (138.100.8.6)`
2. Login with

   User Name: `anonymous`
   Password: `username@subdomain.domain` (your email address)
3. Change directories to `pub/docs/ibm/aixpert`.
4. All files are in ASCII format unless they have a `.Z` or `.tar` suffix (larger files may be compressed).

Internet in the U.S.

Use the following procedure to obtain *AIXpert* supplement information on Internet in the U.S.:

1. `ftp evis.svo.com (192.195.29.12)`
2. Login with

   User Name: `anonymous`
   Password: `username@subdomain.domain` (your email address)
3. Change directories to `evis/aixpert`.
4. All files are in ASCII format.