VM/ESA TCP/IP Performance Improvements

Last Updated May 17, 1998

Bill Bitner
IBM Endicott
1701 North St.
Endicott, NY 13760
607-752-6022
bitner@vnet.ibm.com

Romney White
IBM Endicott
1701 North St.
Endicott, NY 13760
607-755-8276
romney@vnet.ibm.com

Disclaimer

The information contained in this document has not been submitted to any formal IBM test and is distributed on an "as is" basis without any warranty either express or implied. The use of this information or the implementation of any of these techniques is a customer responsibility and depends on the customer's ability to evaluate and integrate them into the operational environment. While each item may have been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will be obtained elsewhere. Customers attempting to adapt these techniques to their own environment do so at their own risk.

In this document, any references made to an IBM licensed program are not intended to state or imply that only IBM's licensed program may be used; any functionally equivalent program may be used instead.

Any performance data contained in this document was determined in a controlled environment and, therefore, the results which may be obtained in other operating environments may vary significantly.

Users of this document should verify the applicable data for their specific environments.

It is possible that this material may contain references to, or information about, IBM products (machines and programs), programming, or services that are not announced in your country or not yet announced by IBM. Such references or information should not be construed to mean that IBM intends to announce such IBM products, programming, or services.

Should the speaker start getting too silly, IBM will deny any knowledge of his association with the corporation.

Trademarks

The following are trademarks of the IBM Corporation:

- IBM
- OfficeVision
- VM/ESA
- ACF/VTAM

Introduction

- Advances in the VM/ESA TCP/IP Performance World
  - Experience
  - Improvement
  - Documentation
- Will focus on TCP/IP function level 310
  - But also include some 2.4.0 information and prototype work
- Comparison to SNA

This presentation will discuss some of the advances in performance for TCP/IP, in terms of both throughput, measurement, and understanding. The majority of this will involve the TCP/IP function level 310 changes, but there will be some discussion of measurements on TCP/IP 2.4.0. In addition, measurements comparing TCP/IP to SNA for terminal support will be presented.

Stack Processor Usage Savings

- Processor usage reduced by about 2%
  - Compiled with "no-check"
  - avoid processing for client notification when the client does not want to be notified
- Additional savings on processors with the checksum (CKSM) instruction
  - MVS folks claim 1.5 instructions per checksum word

TCP/IP 310 saw an improvement in performance in terms of processor usage by the stack (TCPIP virtual machine). This was measured to be about 2%. Several factors contributed to this and are listed in the foil. Several of these involved avoiding unnecessary checking and processing.
In addition, the exploitation of the checksum (CKSM) instruction was added for processors that support CKSM.
For Checksum, the old way involved:
AL total,word
BC nooverflow
ALR total,one
New way eliminates the BC and the ALR (which is executed about half the time) and thus the 1.5 instruction saving per word.

Stack Real Storage Savings

- In prior releases, overestimating the buffer pool sizes resulted in increase storage requirements
- In 310 excess buffers have no appreciable effect on TCPIP storage
  - only looks at used connections where possible
  - hash table allocation redesigned to avoid large system effect
  - Heavy part of consistency checking is now optional
  - other improvements

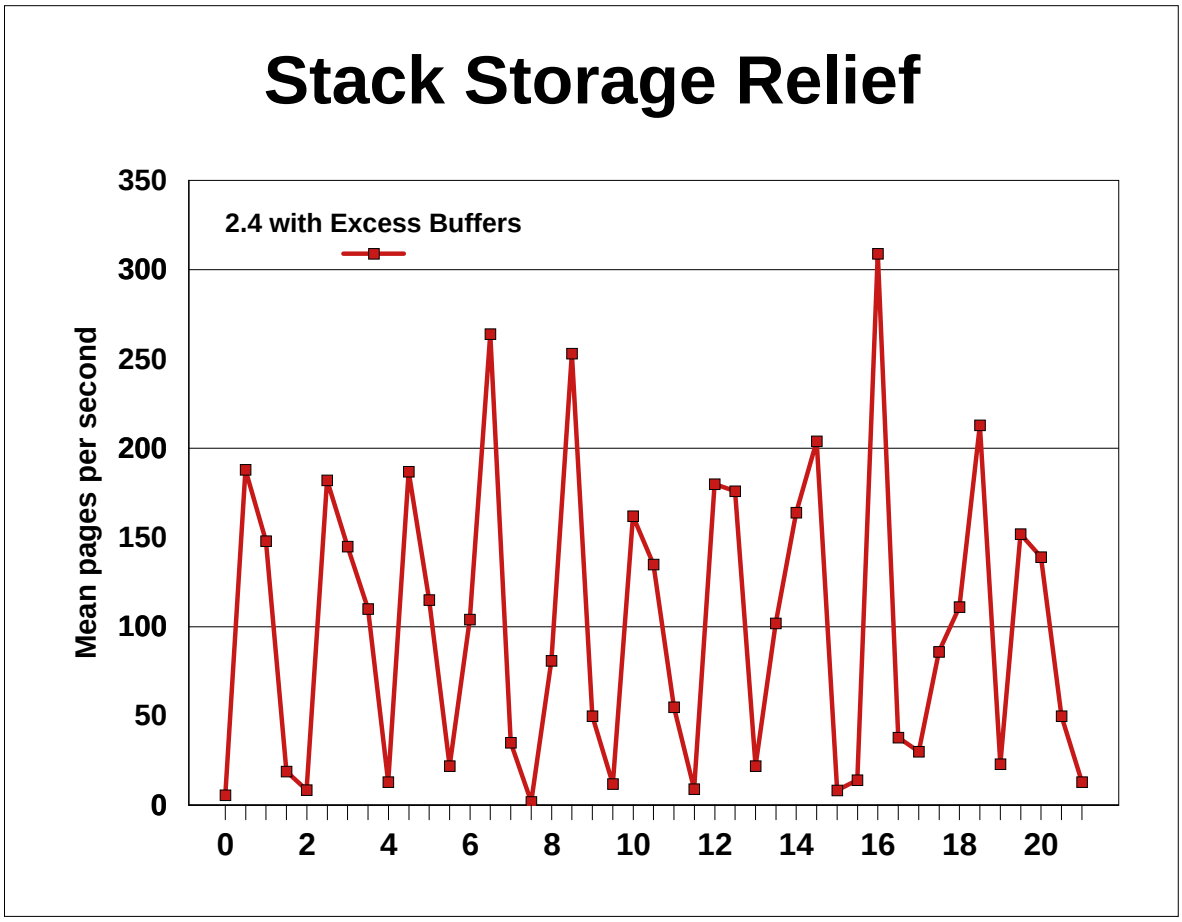
While doing our first measurements with Telnet, we found that overestimating the buffer pool sizes resulted in increased virtual storage requirements. This led the TCPIP stack machine to page unless large numbers of pages were reserved for it.
TCP/IP 310 removes the adverse effect with improvements in storage and connection management. In addition, the largest parts of consistency checking is now optional (with default OFF). We found that this checking seldom added to the RAS characteristics.

Storage Relief Measurements

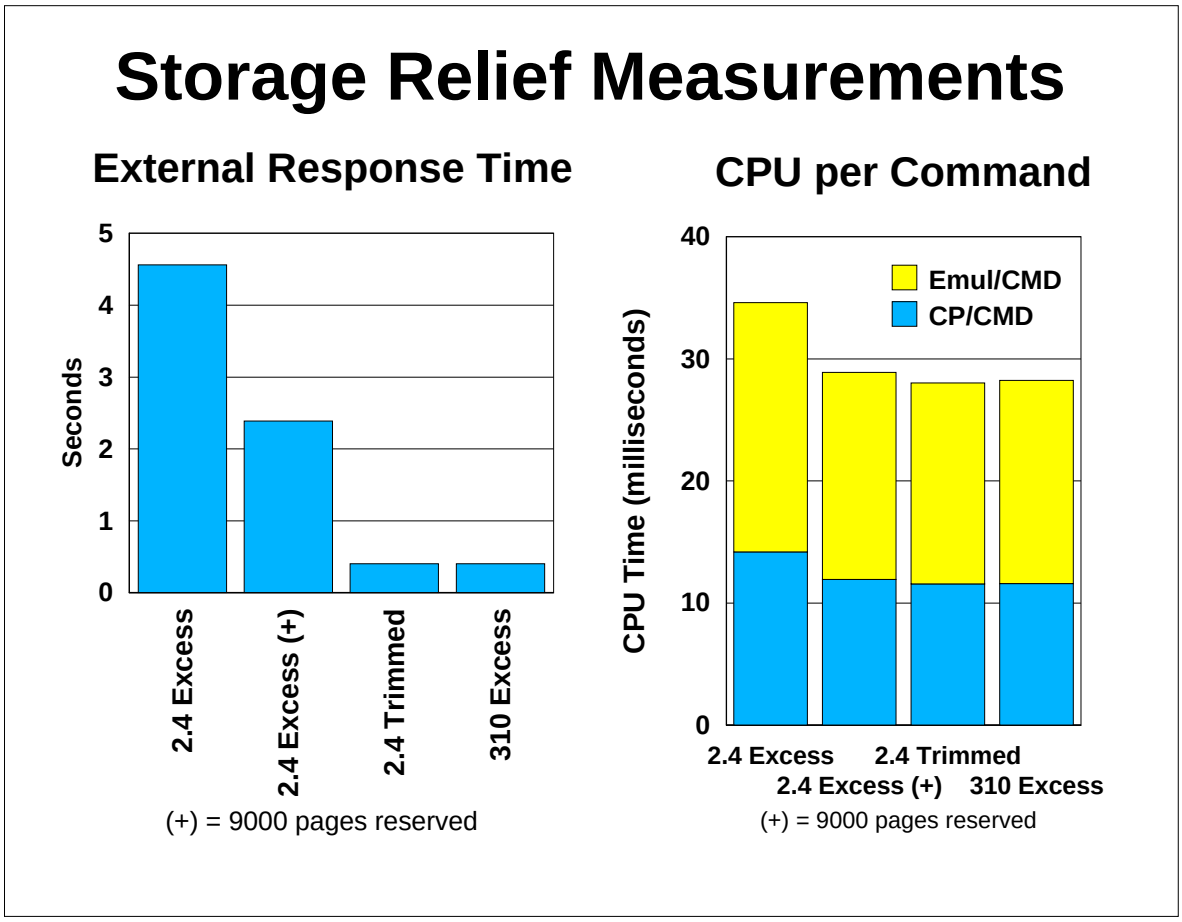
Buffer Pool Size	Excess	Trimmed
ACB	10000	1000
AddressTranslation	1500	150
CCB	500	20
DataBuffer	15000 8192	150 8192
Envelope	5000	300
IPRoute	300	100
LargeEnvelope	500	10
RCB	10	10
SCB	5000	20
SKCB	5000	10
SmallDataBuffer	15000 2048	2500 2048
TCB	10000	2000
TinyDataBuffer	5000	20
UCB	10	10

Two configurations to show impact of storage relief.

The table above shows two configurations that were used in testing. The first is where the number of buffers were set in excess. This was the first time we had done TCP/IP measurements such as these and we thought bigger would be better. As we would see, and you will in foils that follow, this was a mistake. The second configuration shown is more reasonable.



This graph shows what we first saw when measuring a Telnet environment on TCP/IP 2.4 with excess buffers. Periodically, there would be huge spikes in the paging activity due to consistency checking and other data structure management. After the enhancements in function level 310, the graph is basically a flat line near 0.

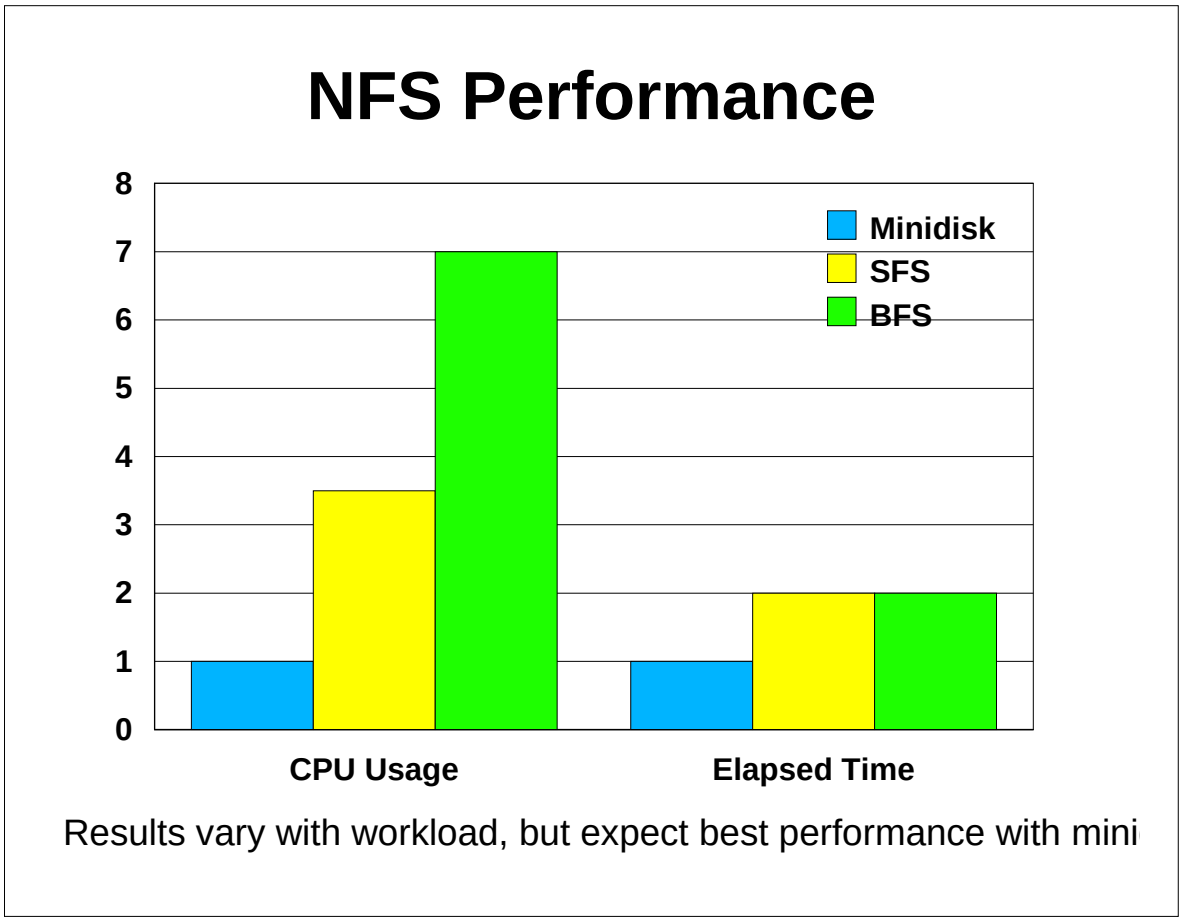


The two charts here show the impact of the storage enhancements. On the left, you see the external response time for a CMS interactive workload. The first bar shows awful performance with the excess buffers driving up paging. The second bar shows the same TCP/IP 2.4 system but this time with pages reserved (over 35 meg). The third bar is where we started gettng smart and trimmed the buffers down to a reasonable level. And finally you see the equivalent response time on 310, but this is with excess buffers! The chart on the right shows the CPU per command for corresponding runs. Notice the extra overhead is in both CP and Emul, so it is not just the effects of paging by CP.

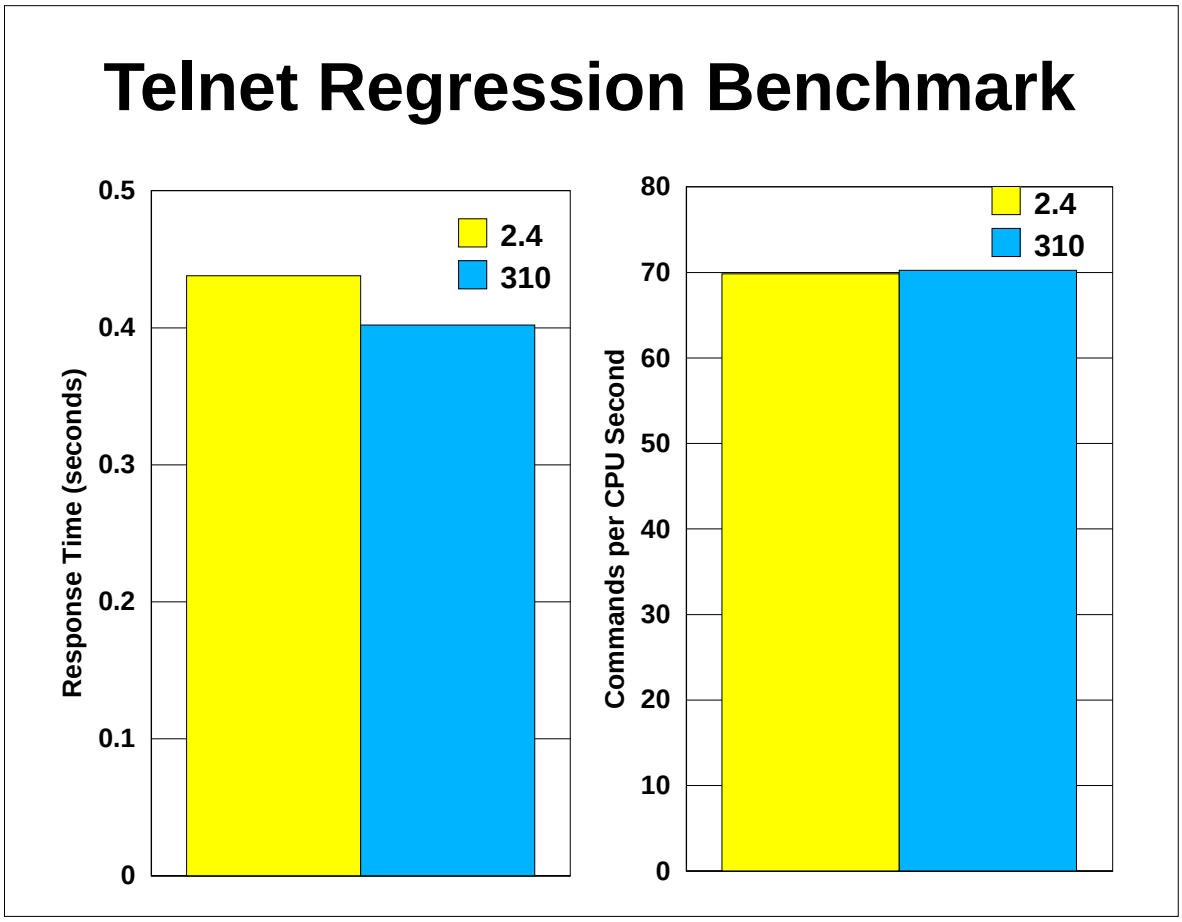
RFC 1323 - Long Fat Network

- 310 Stack implements RFC 1323
- Protocol extension that allows for window sizes exceeding 64KB
- Benefit dependent on
  - high bandwidth and high latency
  - both ends having RFC 1323 implemented (and enabled)
- Benefit seen as higher maximum throughput by increasing the amount of data that can be in the pipe without an acknowledgment

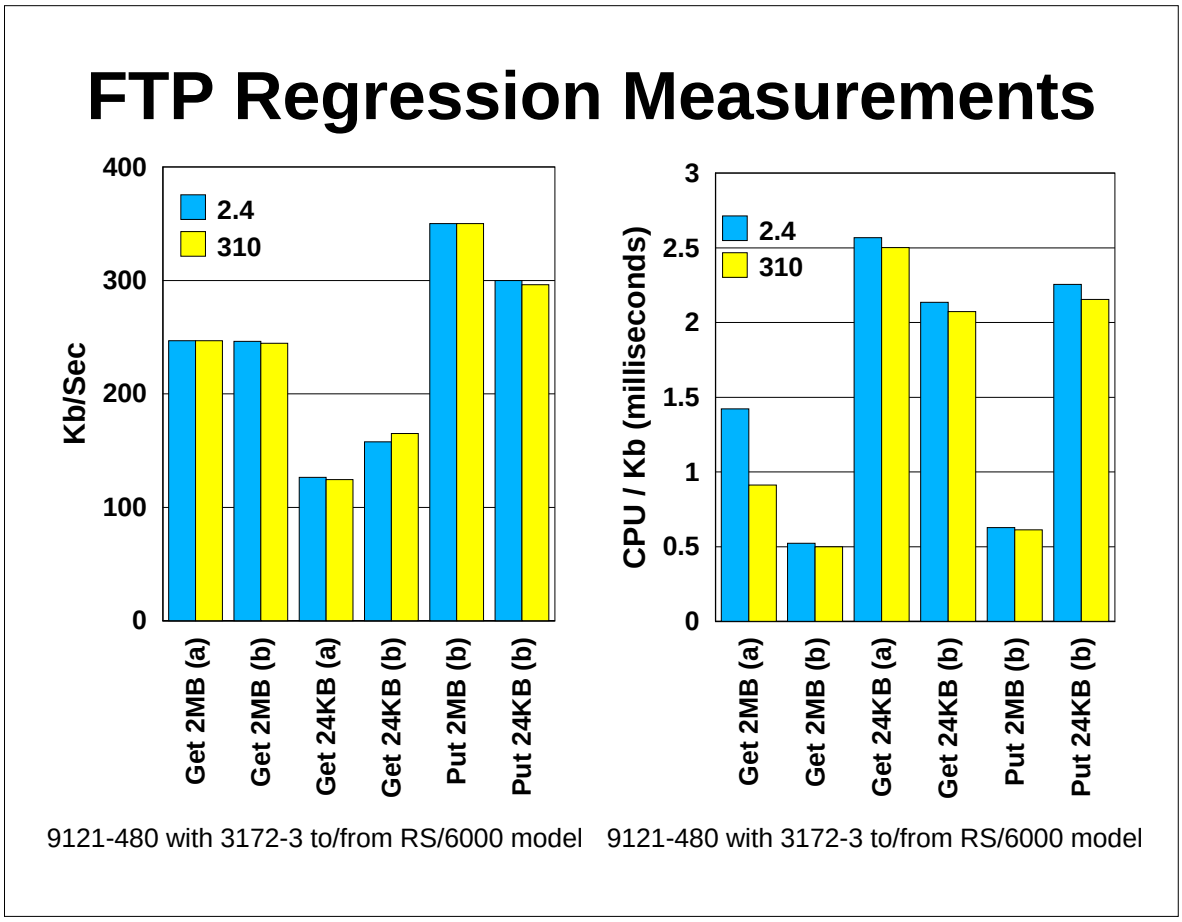
RFC 1323 or Long Fat Network was implemented in the 310 Stack. This protocol extension allows for larger window sizes to be used, which allows for more data to be in the pipe (network) before the stack needs to wait for an acknowledgment. Since window size is dependent on both sides of the connection, the other end must also have RFC 1323 implemented. It also needs to be enabled. To see the most benefit with this enhancement, you need a network that has a high bandwidth and also a long round trip time.



Support in NFS for TCPIP 310 included that of the Byte File System (BFS). As seen in this chart, performance with the BFS is more expensive in terms of processor resources than using NFS with SFS and minidisk, but still provides respectable elapsed times. Of course, the exact results are dependent on the workload.



Measurements were made to compare TCP/IP 2.4 to 310 for a Telnet environment. As we see here, the response time improved somewhat in 310 while CPU per command stayed about the same. The improvement in response time goes back to the storage and processor improvements.

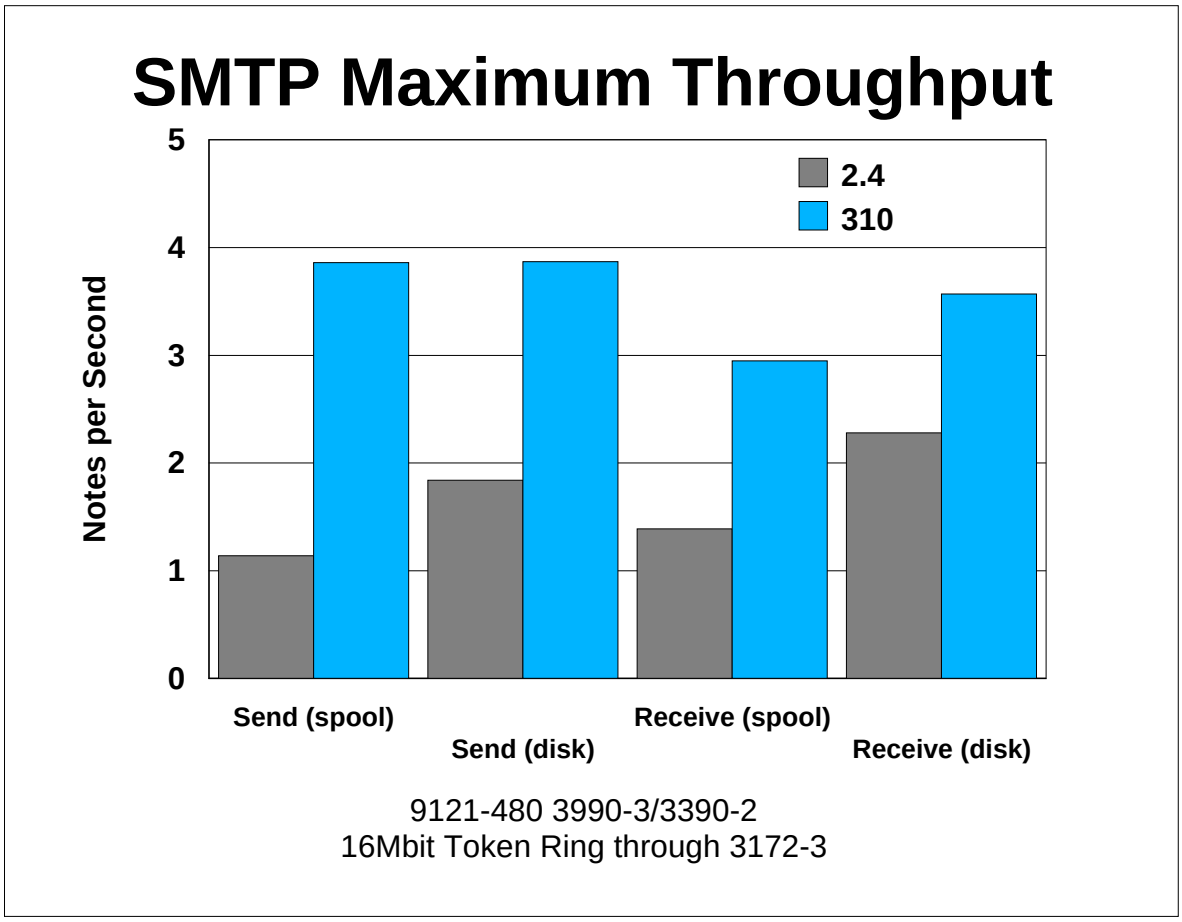


Measurements were also made to check the performance of TCP/IP in an FTP environment. Here you see several measurements made with 2 file sizes (2MB and 24KB) in both ascii and binary transfer mode. This was measured with both Put and Get FTP functions. The chart on the left shows kilobits per second (throughput) while the chart on the left shows CPU / kilobit (overhead). The throughput was the same or slightly better. Note that RFC 1323 was not applicable in these runs. Overhead was similar. The 2MB Get in ascii did improve more than expected. Our investigation has not shown why this would happen. It remains an anomaly. Your mileage may vary.

Increase SMTP Capacity

- SMTP server redesigned
  - Fewer minidisk I/O
    - stat files managed differently
  - Read spool files asynchronously with *SPL
- Performance Benefit dependent on
  - log file written to spool or disk (disk still better performer)
  - access time for SMTP A-disk
  - Spool system performance
  - Processor availability

SMTP was known as a poor performer. Several changes resulted in up to a 3.4 fold improvement based on our measurements. It particular, a majority of the minidisk I/O was removed by managing the Stat files differently. In addition, SMTP now reads spool files asynchronously with *SPL. Configuring your log file to be written to disk is still the better configuration choice for performance. As we minimize the constraint on minidisk I/O performance is now also dependent on processor and spool resources.



The graph here shows the results of performance measurements for the new SMTP. Separate measurements were made for receiving mail and sending mail. Both these scenarios were tested with the logging to spool and to minidisk. The biggest improvement was seen in sending data with logging to spool.

Management Improvements

- Stack and TFTPD contribute to CP monitor data via APPLDATA domain
  - Stack exploits extended CP diagnose x'DC' interface to contribute event and configuration data
- Were not able to get SMTP monitoring in this release
- Improvements in NETSTAT

Both the Stack and TFTPD virtual machines will contribute data to the CP monitor via the APPLDATA domain. The Stack actually uses new functions in diagnose x'DC' added in VM/ESA 2.3.0 to contribute event and config type records in addition to the existing sample records.
The NETSTAT command had several enhancements to it, including increased selectivity and scrolling in fullscreen displays.
We had hoped to add SMTP monitoring in this release, but that was not possible.

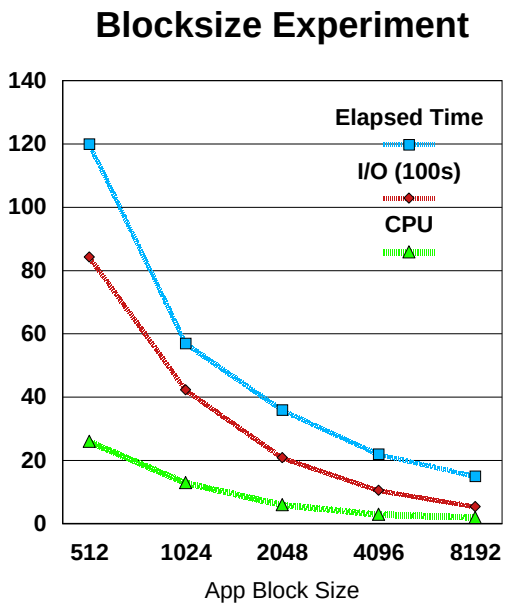
NETSTAT INTERVAL Example

```
  04/14/98            VM TCP/IP Real Time Network Monitor              08:10:50
             Bytes     Bytes Local                                      Idle
User Id        Out       In  Port   Foreign Socket        State         Time
--------   -------   ------- ------ ------------------    -----------   -------
INTCLIEN    141217      3920 TELNET 9.130.25.79..2862     Established   0:00:00
INTCLIEN     66679       835 TELNET 9.130.57.70..2856     Established   0:00:01
INTCLIEN      5235       232 TELNET 9.130.64.59..1032     Established   0:00:01
INTCLIEN     68898     82321 TELNET 9.130.25.2..1436      Established   0:00:02
ROUTED          32   7390376 520    *..*                  UDP           0:00:02
INTCLIEN    241710    316357 TELNET 9.130.57.61..1031     Established   0:00:03
INTCLIEN     39738     38326 TELNET 9.130.57.18..1032     Established   0:00:03
INTCLIEN     26691       276 TELNET 9.130.58.45..1047     Established   0:00:03
INTCLIEN    271330      5127 TELNET 9.130.79.123..1145    Established   0:00:04
INTCLIEN     71917      7081 TELNET 9.130.61.21..1684     Established   0:00:04
INTCLIEN    175681      3584 TELNET 9.130.58.55..1026     Established   0:00:05
INTCLIEN      6333       435 TELNET 9.130.25.35..2712     Established   0:00:05
BEACHMA         16      3229 50490  9.130.57.43..5998     Established   0:00:05
INTCLIEN   1659991     24932 TELNET 9.130.61.21..1027     Established   0:00:08
FTPSERVE         1         1 FTP-D  9.20.26.74..1883      Established   0:00:09
INTCLIEN         0         0 TELNET *..*                  Listen        0:00:09
INTCLIEN    238512    212388 TELNET 9.130.57.49..1029     Established   0:00:09
INTCLIEN    351783      3658 TELNET 9.130.58.11..1729     Established   0:00:11
 Refresh interval: 20 seconds.                            TCBs in Use:216
 1=Usr 2=Sock 3=Quit 4=BOut 5=BIn 6=St 7=Up 8=Dwn 9=Save 10=T/B 11=Ip@ 12=Rfsh
```

This is an example of the improved NETSTAT INTERVAL command. This would appear as a fullscreen display that is scrollable via PF7 and PF8. Note that you can sort based on various fields that map to PF keys. Or you can save the data via PF9.

TFTP with Various Blocksizes

- Early measurements for the Network Station
- Download 2M kernel from 9121-320 with 3172 to 16 Mbit TR
- MTU size = 2000
- Blocksize important even when greater than MTU size



Use this chart not for details on the network station, but for what is illustrated with it. In this experiment, there was a net station attached by a 16Mbit IBM token ring to a 9121-320 via a 3172 to a 16 Mbit IBM Token Ring. The TFTP server on VM was used to download a 2 meg kernel file. The blocksize used for the transfer was varied over time. The number of requests made to move the 2 meg file is directly proportional to the blocksize, and this leads to the number of roundtrips required for handling the download. The MTU size in this case was 2000 bytes, but even when the blocksize exceeded the MTU size, the steady improvement continued.

Migration from VTAM to Telnet

- Tough to be apples to apples and real world
- Took a two step approach
  - Measured VTAM and Telnet on 9121-480 with a CTCA connection
  - Measured Telnet on 9121-480 with 3172-3 Interconnect Controller and 16Mbit IBM Token Ring.
- Configured with target of 90% processor utilization (2000 users for VTAM vs. 1800 for Telnet)
- Measured with TCP/IP 2.4.0

We wanted to show a comparison between supporting terminals via VTAM/VSCS and via Telnet. We started by making a measurement with Telnet where a CTCA was used since this is the configuration we use for our VTAM environments. However, since few customers run TCP/IP with CTCAs, we then did a measurement with a 3172-3 connected to a 16Mbit IBM Token Ring. Our measurements are configured to reach a target of 90% processor utilization. Due to slightly higher costs in TCP/IP, this resulted in slightly fewer users for the Telnet scenario. All these measurements were made with TCP/IP 2.4.

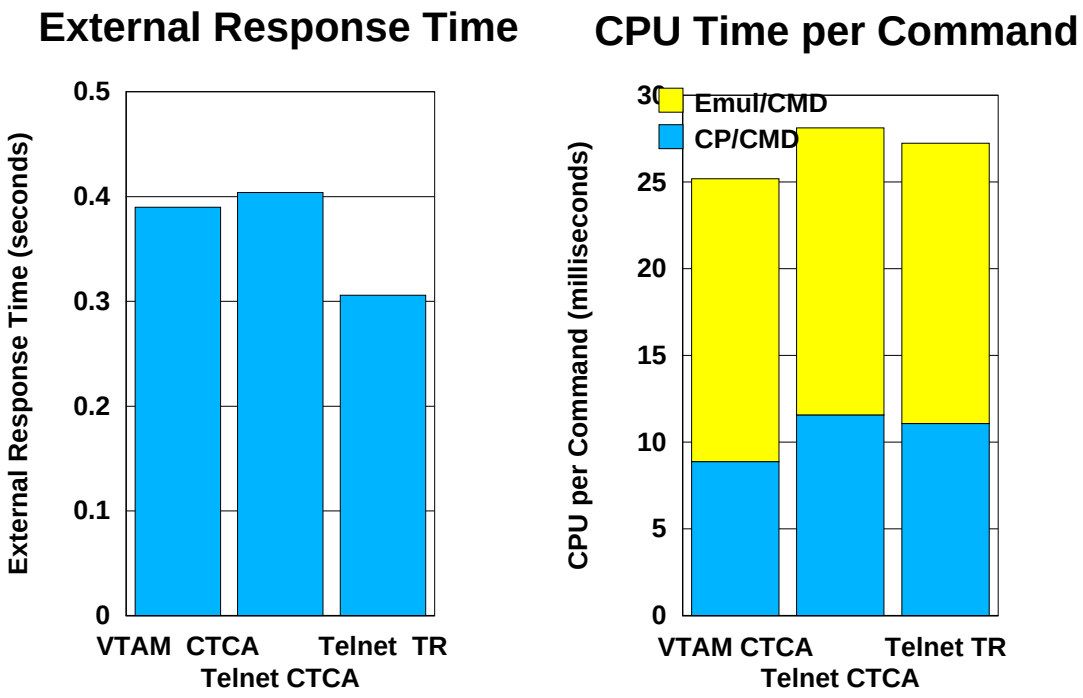
Migration from VTAM to Telnet

- Subsecond response time in all cases
- CPU requirements slightly higher for Telnet
- Storage requirements were higher for Telnet
  - Reserved more pages for TCPIP than for VTAM

Scenario	Server Machine Working Set Size
VTAM CTCA	547
TCP/IP CTCA	2700
TCP/IP TR	2700

One thing to note in this type of migration is that TCP/IP requires more virtual storage than a comparable VTAM network. This is seen in the table above. Note however, that some improvement in this area was made in TCP/IP function level 310.

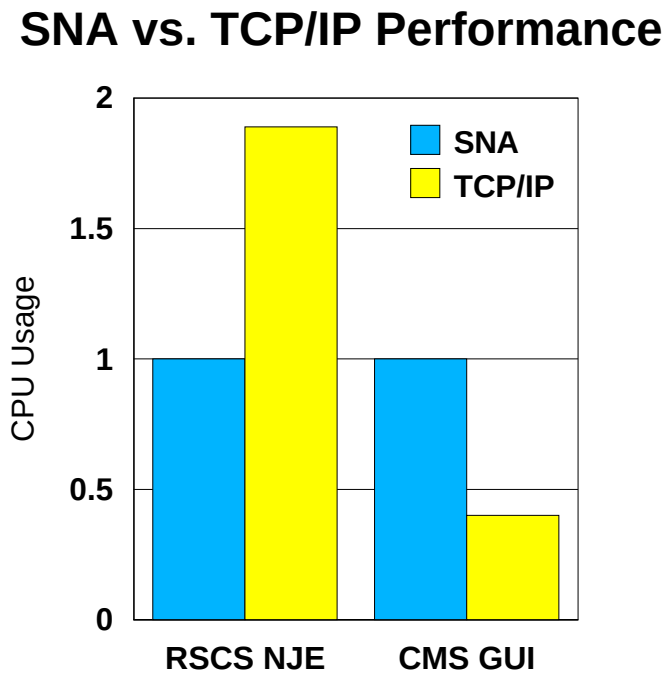
VTAM vs. Telnet Measurements



In the charts shown here we see that TCP/IP does well compared to VTAM in terms of response time, particularly for the Token Ring environment. However, there are increases in processor time per command for the TCP/IP environment in both CP and emulation time.

More Comparisons with SNA

- SNA wins in RSCS line driver scenario
- TCP/IP wins in CMS GUI environment



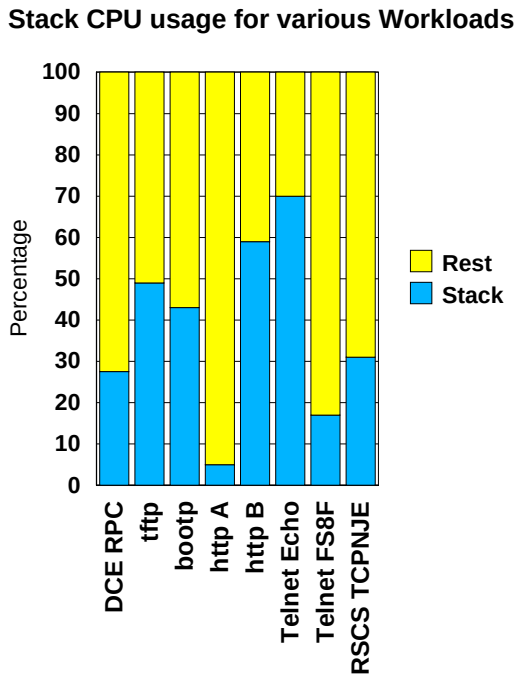
In addition to the terminal support comparison between SNA and TCP/IP, there is some older data worth mentioning in terms of comparisons.

The first was when we did some RSCS testing a while ago where we measured over SNANJE and TCPNJE lines for a RSCS workload of ours. In this case, SNA was the better performer. In the testing of the CMSDESK VM GUI application, we found that better performance was achieved when configured with TCP/IP than in an SNA APPC configuration.

Stack Comments

- Limited by single processor speeds
- Two flavors:
  - UDP
  - TCP

Note: Telnet case includes Telnet running in the TCPIP machine with the stack.



The TCPIP virtual machine is not able to run as a virtual MP, so it is limited by the capacity of a single processor. Note that Telnet is different in that the Telnet code also runs in the TCPIP server machine along with the stack.

The chart shows how big a part the stack plays in various workloads. It shows the percentage of host resources used by the stack. Remember the single processor limitation. Based on that, this DCE RPC workload would be hard pressed to run on anything above a 4-way. Note, that this is worst case, basically a null RPC.

Other Investigation

- The following is being investigated, but no commitments at this time.
- Limited virtual multiprocessor support
  - Allow some CP functions of the Stack to run on an alternate virtual processor.

Knowing that the stack being limited to a single virtual processor could be a bottleneck. Investigation of alternatives showed that a limited form of virtual multiprocessor support was possible by using an alternate virtual processor to handle certain CP functions. The initial work is interesting, and will continue to be investigated as resources permit.

Summary

- Better Performance
  - Stack improvements
  - RFC 1323 Long Fat Networks
  - SMTP improvements
- Better flexibility
  - Stack storage relief
- Improved monitoring ability
- Increased our knowledge base

I feel more comfortable and confident talking about TCP/IP performance today than I did a year ago. We have seen performance enhancements put into the product both in the Stack and SMTP. This enhancements have not just been in terms of throughput, but also in terms of flexibility, management, and monitoring. We also know a lot more than we did a year ago. Thanks for your support.