.

Measurement-based Analysis of TCP/IP Processing Requirements

Srihari Makineni Ravi Iyer
Communications Technology Lab
Intel Corporation
{srihari.makineni, ravishankar.iyer}@intel.com

Abstract – *With the advent of 10Gbps Ethernet technology, industry is taking a hard look at TCP/IP stack implementations and trying to figure out how to scale TCP/IP processing to higher speeds with least amount of processing power. In this paper, we focus on the following: 1) measuring and analyzing the current performance and architectural requirements of TCP/IP packet processing and 2) understanding the processing requirements for TCP/IP at 10Gbps and identifying the challenges that need to be addressed. To accomplish this we have measured the performance of Microsoft Windows* 2003 Server O/S TCP/IP stack running on the state-of-the art low power Intel® Pentium® M processor [5] in a server configuration. Based on these measurements and our analysis, we show that the cache misses and associated memory stalls will be the gating factors to achieve higher processing speeds. We show the performance improvements achievable via reduced memory copies and/or latency hiding techniques like multithreading.*

I. INTRODUCTION

TCP/IP over Ethernet is the most dominant packet processing protocol in datacenters and on the Internet. It is shown in the paper [13] that the networking requirements of commercial server workloads represented by the popular benchmarks like TPC-C, TPCW and SpecWeb99 are significant and so is the TCP/IP processing overhead on these workloads. Hence it is important to understand the TCP/IP processing characteristics and the requirements of this processing as it scales to 10Gbps speeds with the least possible amount of processing overhead. Analysis done on TCP/IP in the past [2,4] has shown that only a small fraction of the computing cycles are required by the actual TCP/IP protocol processing and that the majority of cycles are spent in dealing with the Operating System, managing the buffers and and passing the data back and forth between the stack and the end-user applications. Many improvements and optimizations have been done to speed up the TCP/IP packet processing over the years. CPU-intensive functions such as checksum calculation and segmentation of large chunks of data into right-sized packets have been offloaded to the Network Interface Card (NIC). Interrupt coalescing by the NIC devices to minimize the number of interrupts sent to the CPU is also reducing the burden on processors. In addition to these NIC features, some OS advancements such as asynchronous I/O and pre-posted buffers are also speeding up TCP/IP packet processing. While these optimizations combined with the increases in CPU processing power are sufficient for the current 100 Mbps and 1Gbps Ethernet bandwidth levels, these may not be sufficient to meet a sudden jump in available bandwidth by a factor of 10 with the arrival of 10Gbps Ethernet technology.

In this paper, we focus on transmit and receive side processing components of TCP/IP which are some times also referred to as data or common path processing. We have performed detailed measurements on Intel® Pentium® M microprocessor to understand execution time characteristics of TCP/IP processing. Based on our measurements, we analyze the architectural requirements of TCP/IP processing in future datacenter servers (expected to require around 10Gbps soon). We do not cover connection management aspects of TCP/IP processing here but intend to study these later using traffic generators like GEIST [8]. For a detailed description of the TCP/IP protocols, please refer to RFC [9].

II. CURRENT TCP/IP PERFORMANCE

In this section, we first provide an overview of the TCP/IP performance in terms of the throughput and CPU utilization. We then characterize TCP/IP processing by analyzing information gathered from the processor's performance monitoring events. To accomplish this, we have used NTttcp, a tool based on a TCP/IP benchmark program called ttcp [12].

.

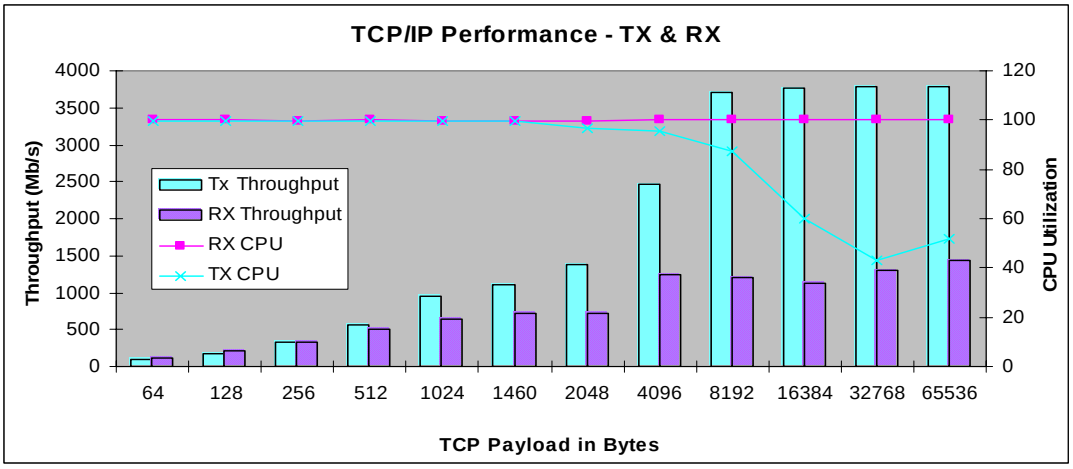


Figure 1. Today's TCP/IP Processing Performance -- Throughput and CPU Utilization

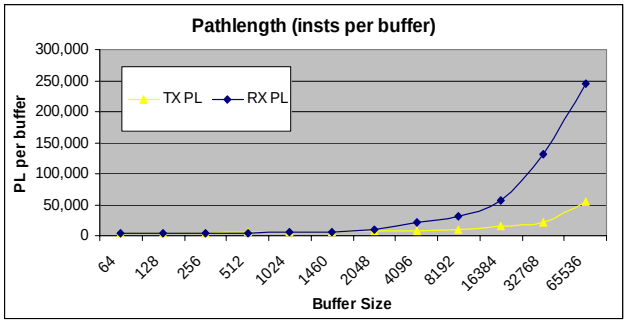
This is used to test the performance of the Microsoft Windows* TCP/IP stack. We have used another tool called EMON for gathering information on processor behavior when running NTttcp. EMON collects information like number of instructions retired, cache misses, TLB misses and etc. from the processor. For information on system configuration used in the tests, various modes of TCP/IP operation and parameters we have used for, please refer to [3].

A. Overview of TCP/IP Performance

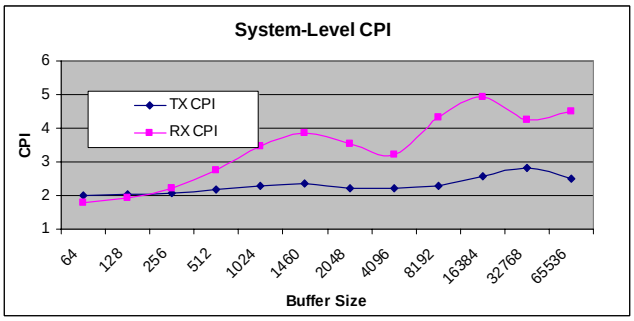
Figure 1 shows the observed transmit and receive-side throughput and CPU utilization over a range of TCP/IP payload (application data) sizes. The reason for lower receive side performance over the transmit side for payload sizes larger than 512 bytes is due to memory copy that happens when copying the data from NIC buffer to application provided buffer. Also, the TCP/IP stack sometimes ends up copying the incoming data into the socket buffer if the application is not posting the buffers fast enough. To eliminate the possibility of this intermediate copy we ran NTttcp again with six outstanding buffers ("-a 6") and set the socket buffer to zero and observed that the throughput for a 32KB payload went up from 1300MB to 1400MB. For payload sizes less than 512 bytes the receive side fares slightly better than the transmit side because, for the smaller payload sizes the TCP/IP stack on the transmit side coalesces these payloads into a larger payload and hence results in longer code path. Higher transmit side throughput and lower CPU utilization for payload sizes larger than 1460 bytes are due to the fact that we are using the Large Segment Offload (LSO) [13] option and the socket buffer is set to zero thus disabling any buffering in the TCP/IP stack. Also, when transmitting these larger payloads, the client machines have started becoming the bottleneck as one of the processors on these multi processor machines is processing all the NIC interrupts and hence is at 100% utilization while the others are under utilized. This bottleneck can be eliminated by employing more clients.

B. Architectural Characteristics

Figure 2 shows the number of instructions retired by the processor for each payload size (termed as path length or PL) and the CPU cycles required per instruction (CPI) respectively. It is apparent that as the payload size increases, the PL increases significantly. A comparison of transmit and receive side PLs reveal that the receive code path executes significantly more instructions (almost 5x) at large buffer sizes. This explains the larger throughput numbers for transmit over receive. Higher PL when receiving larger buffer sizes (> 1460 bytes) is due to the fact that this large application buffer is segmented into smaller packets (a 64 Kbytes of application buffer generates ~44.8 TCP/IP packets on LAN) before transmitting and hence the receive side has to process several individual packets per an application buffer. At small buffer sizes, however,



(a) Path Length



(b) CPI Characteristics

Figure 2. Execution Characteristics of TCP/IP Processing

TCP Payload in bytes	#of Instructions per payload		# of branches per payload		Memory accesses per payload		TLB Misses Per payload		Instruction Cache Misses per payload		Data Cache misses per payload	
	TX	RX	TX	RX	TX	RX	TX	RX	TX	RX	TX	RX
64	4,536	3,759	843.6	768.1	1.0	3.1	9.0	6.2	5.9	9.1	39.2	17.9
128	4,305	3,892	863.0	788.3	1.6	6.5	7.2	6.3	7.2	9.6	40.9	28.2
256	4,600	4,134	918.9	844.2	3.5	13.1	10.4	6.8	9.4	11.2	48.0	46.0
512	5,088	4,607	1,026.4	947.6	4.5	25.9	8.3	7.7	15.3	14.2	62.1	82.3
1024	5,597	5,608	1,128.9	1,142.5	5.7	50.9	9.7	10.1	21.6	21.5	80.6	153.1
1460	6,713	6,473	1,359.6	1,329.6	7.7	73.0	11.7	11.0	34.2	25.8	102.1	202.3
2048	7,978	9,779	1,590.1	2,023.3	7.7	101.5	24.4	13.5	54.8	39.7	158.3	304.6
4096	8,540	21,547	1,765.7	4,515.3	9.2	219.1	25.7	43.9	56.1	109.1	179.0	791.5
8192	10,675	31,955	2,167.8	6,436.0	12.6	474.6	31.5	56.6	83.9	135.2	217.2	1,600.7
16384	14,766	56,225	3,028.8	11,754.8	20.9	1,079.5	45.5	97.0	140.7	227.7	299.7	3,330.2
32768	22,320	130,440	4,604.7	22,680.7	34.0	2,286.7	64.6	176.1	196.0	409.4	426.9	7,196.8
65536	55,597	245,148	11,172.9	51,144.1	114.0	4,758.8	96.0	331.5	296.7	751.2	958.4	14,415.3

Table 1. Transmit & Receive-Side Processing Characteristics

the PL for transmit and receive is quite similar as well as much lower.

Looking at the CPI numbers, the observation is that the transmit CPI is relatively flat as compared to the increase in payload size. The increase in receive CPI with the buffer size is due to the fact that receive side processing involves at least one memory copy of the received data which causes several cache misses. The observed dip in the receive CPI for 2KB, 4KB and 32KB buffer sizes is due to a reduction in the number of L1 and L2 misses for these buffer sizes. We are further investigating the cause for lower cache misses for these three buffer sizes.

Table 1 summarizes some additional architectural characteristics of the TCP/IP packet processing. From the data, we note that the number of memory accesses per payload are significantly higher for receive-side processing. This leads to the higher receive-side CPI shown in Figure 2(b) and the lower performance of the receive operation as compared to transmit in Figure 1.

III. Analyzing Challenges in TCP/IP Processing

In this section we analyze the requirements for TCP/IP processing in a future datacenter. Our previous analysis of server networking characteristics [13] has shown that the datacenter server applications require TCP/IP processing close to 10Gbps in the near future. To achieve this processing rate efficiently, we study the effect of platform evolution (processor frequency improvements and memory latencies improvements) on transmit and receive performance scaling. We then show that some specific functions of this processing definitely need to be sped up in order to make the TCP/IP processing at 10Gbps a reality on the general purpose processors. Our analysis can also shed light on the features required for packet processing

engines that are being investigated by various research projects [11].

A. *Effect of Frequency Improvements*

In order to understand the effect of frequency improvements, we measured the performance of receive and transmit-side processing on the same platform employing the same processor at two different frequencies (1600MHz and 600MHz). Table 4 shows the frequency scaling of TCP/IP packet processing. We define frequency scaling as the ratio of the increase in performance with respect to the increase in CPU frequency.

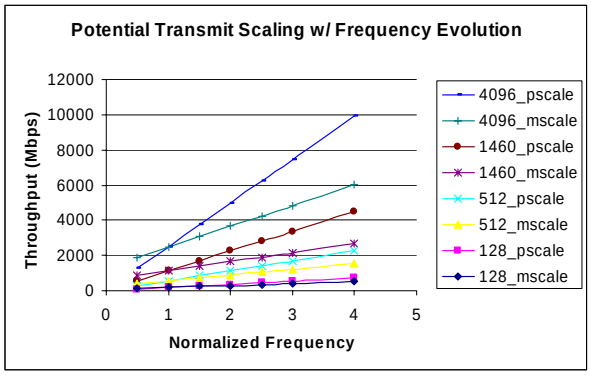
TCP Payload in Bytes	Transmit	Receive
64	63%	68%
128	64%	64%
256	64%	56%
512	56%	46%
1024	52%	37%
1460	46%	33%
2048	56%	41%
4096	47%	55%

Table 2. Frequency Scaling Efficiency

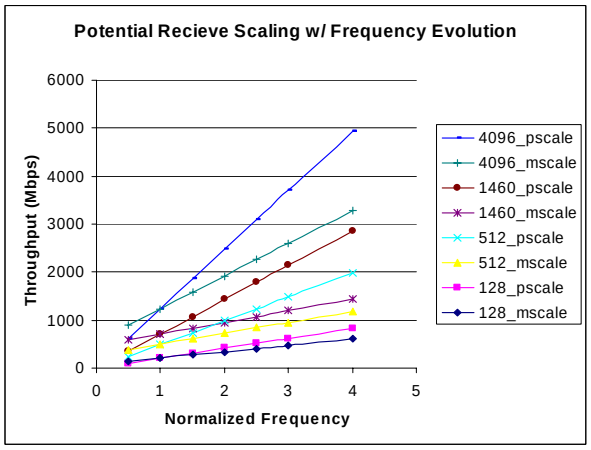
It should be noted that frequency scaling experiments illustrate the limitations posed by a constant memory latency on performance since we used the same platform configuration. Based on this frequency scaling, we show the expected increase in performance as a function of frequency in Figure 3. The curves denoted by the suffix ("_mscale") denote the performance as the frequency is increased in the X-axis. In contrast, the curves denoted by the suffix ("_pscale") show the scaling achievable if the memory access speed also scaled at the same rate as the CPU frequency. From Figure 3(a), we find that the transmit performance can support 10Gbps only at 4096 payload size and, furthermore, only if we achieved a constant CPI over future platform generations. For the receive-side, we are hardly able to achieve 5Gbps for the same case. With sub-optimal memory latency improvements, the (realistic) performance scenario seems far from the desired processing rate.

B. *Memory Latency Improvements*

To address the issue of memory latency for receive-side processing, we next study the performance improvement achievable assuming significant memory latency improvements in the long term. Some of the technologies that support such a potential improvement include integration of memory controllers on to the processor die, the use of faster processor-memory



(a) Transmit side scaling



(b) Receive side scaling

Figure 3. Performance Scaling with CPU Frequency Evolution

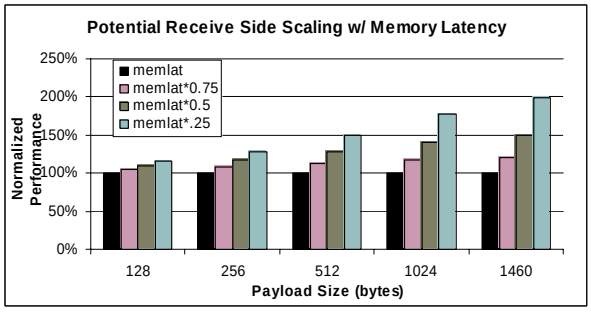


Figure 4. Performance Scaling with Memory Access Time

.

interconnects (faster parallel busses and /or faster serial link technologies). Figure 4 shows the performance benefits achievable with reduction in memory access latency. The TCP/IP processing performance is shown for five different payload sizes (128 bytes to 1460 bytes). For each payload size, the performance is shown normalized to the base memory latency (in today's platform). As the memory latency reduces from 75% to 50% and then to 25% of what it is now (in clocks), improvements from 5% to 100% are achievable. The performance benefits are larger for higher payload sizes since the time spent accessing memory is higher in these cases (as payload size increases, so does the amount of memory copy).

C. Potential Challenges / Solutions

As shown in the previous subsection, memory stall time can be huge problem for receive-side processing. Table 3 starts to analyze the problem of memory copies by studying the percentage of memory misses spent in copying data from one payload size to another. As shown in the Table 3, we find that 43% to 65% of cache misses are due to copies. Surprisingly, we also find that the performance of cache misses reduces as the payload size is increased.

	Measured Misses per Buffer		Copy-Related Misses (assuming src/dest not in cache)		% of Cache Misses related to Copies	
	TX	RX	TX	RX	TX	RX
64	1.0	3.1	-	2	negl	65%
128	1.6	6.5	-	4	negl	61%
256	3.5	13.1	-	8	negl	61%
512	4.5	25.9	-	16	negl	62%
1024	5.7	50.9	-	32	negl	63%
1460	7.7	73.0	-	46	negl	63%
2048	7.7	101.5	-	64	negl	63%
4096	9.2	219.1	-	128	negl	58%
8192	12.6	474.6	-	256	negl	54%
16384	20.9	1079.5	-	512	negl	47%
32768	34.0	2286.7	-	1024	negl	45%
65536	114.0	4758.8	-	2048	negl	43%

Table 3. Analyzing Memory Copies

In order to address the memory stall time, there are two types of approaches that need to be investigated -- (1) reducing the number of memory accesses through better cache management schemes and prefetching schemes and (2) hiding the memory latency by overlapping it with other useful work i.e. multithreading for example. To study the performance potential of multithreading on TCP/IP processing performance, we measured the performance of TCP/IP paths on Intel's Pentium® 4 processor with Hyper-Threading (HT) enabled and disabled. The performance benefits of HT are illustrated in Figure 5. As shown, we find that HT helps significantly in overlapping execution and hiding memory latency, resulting in a performance benefit from ~20% to over 50%.

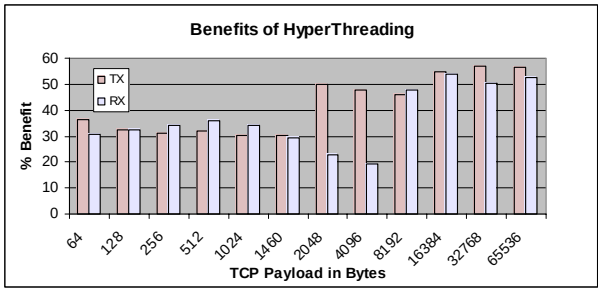


Figure 5. Benefits of HT Technology

IV. CONCLUSIONS AND FUTURE WORK

In this paper, we presented a detailed analysis of TCP/IP transmit and receive side performance in terms of the achieved throughput, CPU utilization and the architectural characteristics like Cycles Per Instruction (CPI), number of instructions executed for a given TCP payload size (also known as Path Length or PL) and etc. We then computed the architectural requirements of TCP/IP processing at 10Gbps and compared these requirements with the natural evolution of processor and memory technologies and identified some key issues that need to be addressed to effectively scale the TCP/IP stacks to 10Gbps speeds.

Future work in this area is multi-fold. We are looking into several threading technologies like SMP, CMP and etc. as a way to hide the memory latencies. We are also looking into various hardware assists that help accelerate the memory copies, and mechanisms to bring data into cache shortly before it is needed.

ACKNOWLEDGEMENTS

We would like to express our thanks to Michael Espig for providing us the necessary system infrastructure to be able to performance these measurements. We would also like to Dave Minturn and Ramesh Illikkal for his insight into the TCP/IP protocol stacks and other members of our team for their helpful input on this study.

.

NOTICES

® is a trademark or registered trademark of Intel Corporation or its subsidiaries in the United States and other countries.
** Other names and brands may be claimed as the property of others.*

REFERENCES

[1] J. Chase et. al., "End System Optimizations for High-Speed TCP", IEEE Communications, Special Issue on High-Speed TCP, June 2000.

[2] D. Clark et. al., "An analysis of TCP Processing overhead", IEEE Communications, June 1989.

[3] A. Earls, "TCP Offload Engines Finally Arrive", Storage Magazine, March 2002.

[4] A. Foong et al., "TCP Performance Analysis Re-visited," IEEE International Symposium on Performance Analysis of Software and Systems, March 2003.

[5] S. Gochman, et al., "The Intel® Pentium® M Processor: Microarchitecture and Performance." Intel Technology Journal. http://developer.intel.com/technology/itj/, May 2003.

[6] R. Huggahalli and R. Iyer, "Direct Cache Access for Coherent Network I/O", submitted to an international conference, 2003.

[7] K. Kant, "TCP offload performance for front-end servers," to appear in Globecom, San Francisco, 2003.

[8] K. Kant, V. Tewari and R. Iyer, "GEIST – A Generator for E-commerce and Internet Server Traffic," 2001 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS), Oct 2001

[9] J. B. Postel, "Transmission Control Protocol", RFC 793, Information Sciences Institute, Sept. 1981.

[10] M. Rangarajan et al., "TCP Servers: Offloading TCP/IP Processing in Internet Servers. Design, Implementation, and Performance," Rutgers University, Dept of Computer Science Technical Report, DCS-TR-481, March 2002.

[11] G. Regnier et al., "ETA: Experience with an Intel Xeon Processor as a Packet Processing Engine," A Symposium on High Performance Interconnects (HOT Interconnects), Stanford, 2003.

[12] "The TTTCP Benchmark", http://ftp.arl.mil/~mike/ttcp.html

[13] S. Makineni and R. Iyer, "Performance Characterization of TCP/IP Processing in Commercial Server Workloads, to appear in the 6th IEEE Workshop on Workload Characterization (WWC-6), Austin, Oct 2003.