Speeding up TCP/IP: Faster Processors are not Enough

Evangelos P. Markatos *
Institute of Computer Science (ICS)
Foundation for Research & Technology – Hellas (FORTH)
P.O.Box 1385, Heraklio, Crete, GR-711-10 GREECE
http://archvlsi.ics.forth.gr markatos@csi.forth.gr
Appears in the IEEE Int. Perf. Comp. and Comm. Conf. (IPCCC 2002)

Abstract

Over the last decade we have been witnessing a significant increase in the capabilities of our computing and communication systems. On the one hand, processor speeds have been increasing exponentially, doubling every 18 months or so, while network bandwidth, has followed a similar (if not higher) rate of improvement, doubling every 9-12 months, or so. Unfortunately, applications that communicate frequently using standard protocols like TCP/IP do not seem to improve at similar rates.

In our attempt to understand the magnitude and reasons for this gap between processor speed and interprocess communication performance, we study the execution of TCP/IP on several processors and operating systems that span a time interval of more than eight years. Our main conclusion is that TCP/IP performance does not scale comparably to processor speed, and this poor scalability is magnified and propagated to higher-level protocols like HTTP.

1 Introduction

Over the last decade we have been witnessing a tremendous increase in the capabilities of our computing and communication systems. Processor speeds have been increasing exponentially, doubling every 18 months or so [5]. This rate of increase, which is also known as Moore's Law, has been sustained since the seventies and is expected to continue to hold (at least) for the near future. Similarly, network bandwidth, has followed a similar (if not higher) rate of improvement, doubling every 9-12 months, or so, as indicated by Gilder's Law [3].

Based on Moore's Law and Gilder's Law, one would conclude that the performance of distributed/networked applications would improve at similar rates, i.e. doubling every 9-18 months. Unfortunately, this is not the case, partly because, interprocess communication does not improve at the rates predicted by Moore's and Gilder's laws. In this paper we study the performance improvements in TCP/IP-based interprocess communication over the last decade, aiming to answer the following questions:

- *Will future networked applications be able to capitalize on Moore's and Gilder's laws?* i.e. double their performance every 9-18 months?
- *Does interprocess communication performance improve at rates comparable to those suggested by Moore's and Gilder's laws?*
- *What are the most significant bottlenecks in the performance of TCP/IP-based interprocess communication?*

The rest of the paper is organized as follows: Section 2 places our work in the context of previous work and recent technology trends. Section 3 presents our experimental measurements, section 4 discusses the major factors that limit TCP/IP performance, and section 5 concludes the paper.

2 Related Work

TCP/IP performance had received significant interest in the past [1, 11, 12]. Most of this prior work has focused on studying and improving TCP/IP throughput and latency over limited bandwidth, possibly wireless, and usually congested networks. However, technology trends suggest that, network bandwidth is neither the most limited, nor the most precious commodity in a distributed system. Recent results suggest that raw network bandwidth has been improving much faster than processor speeds [3]. This trend between processor speeds and network bandwidth is expected to continue in the near future, and therefore the gap between TCP/IP performance and processor speeds will continue to widen. In fact, some researchers, have

*The author is also with the University of Crete.

started working on *network processors*, a new breed of special-purpose processors that are specifically tuned for executing network-related tasks [4]. Our hope is that by studying TCP/IP performance on traditional general-purpose processors we will be able to identify and improve on the main factors that contribute to its limited scalability.

3 Experiments

3.1 Experimental Environment

To quantify the performance improvement of interprocess communication over the last several years, we have used a variety of computers based on SPARC, Alpha, and Pentium Processors. The oldest and slowest of the computers used (our base case) is a 40 MHz SPARCstation 10 rated at 0.96 SPEC Int95. The most recent and fastest computer used is a 1200 MHz Pentium rated at 51 SPEC Int95. These computers represent a time spectrum more than eight years apart. [1]

To accurately characterize the interprocess communication performance of the studied computing systems, we use the popular benchmarks `lmbench` [8] and `ttcp` [13]. The traditional performance metrics that have been used in the literature to characterize the performance of communicating systems, are *bandwidth*, *latency*, and, in some cases, *wall clock time*. However, in this paper we are not only interested in the actual performance of the studied communicating systems, but also in the improvement of this performance over the past years. Therefore, instead of reporting the actual bandwidth and/or latency measured, we report the *improvement* of the bandwidth (or latency) compared to the bandwidth (or latency) of our base case.

3.2 Kernel Entry-Exit Overhead

Most interprocess communication operations usually require the assistance of the operating system kernel. [2] For example, all the operations that send and receive data, that manipulate sockets, and in general that communicate with other processes require operating system calls. Therefore, it is important to understand how the performance of system calls that are in the critical path of interprocess communication has improved over the last decade. Thus, in our first benchmark, we use `lmbench` to measure the cost of an empty operating system call: that is, the latency to enter and exit the operating system kernel. Figure 1 plots the cost (latency) of an empty operating system call (normalized to the cost of an empty operating system call of our baseline system). We plot the kernel entry-exit latency as a function of the processor speed (measured in SPECint95 SPECmarks). We immediately see that the performance of the empty operating system call does not scale well with processor speed. For example, we see that although PENTIUM processors have gotten more than 50 times faster (compared to our baseline), the cost of entering the operating system kernel has improved at best only by a factor of 15. Similarly, although SPARC processors have gotten faster by more than a factor of 25, the overhead of entering the kernel has improved only by a factor of less than 3.

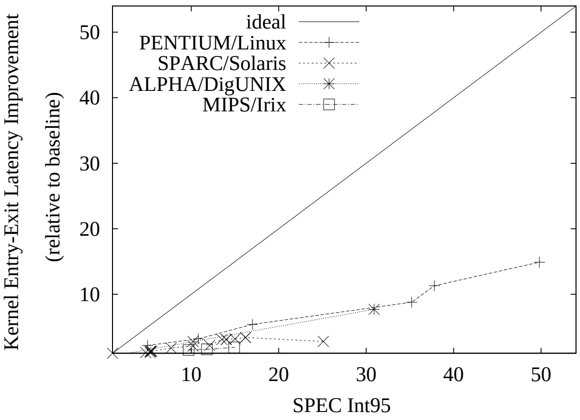

Figure 1: **Kernel Entry-Exit latency.**

Although this disparity between processor speed and operating system performance has been reported before [10], our results indicate that the disparity is getting worse. For example, Ousterhout, in his paper in the early 90's [10], reported that kernel entry/exit performance had relatively improved only 50%-80% compared to the processor's speed, while our results (in the late 90's and early 2000's) suggest that kernel entry/exit performance has relatively improved only 12%-28% compared to the processor's speed. Therefore, operating system kernel performance not only continues to get relatively worse compared to processor speed, but it does so at a higher rate.

3.3 TCP/IP in Localhost

Although kernel entry-exit performance lies in the critical path of interprocess communication and may be significant for short data transfers, long data transfers are dominated by the overhead of communication protocol execution. In our next experiment we use `lmbench` to study the performance of TCP/IP between communicating processes that run on the same computer. To reduce any initialization overheads, data are

[1] More details about the computers used can be found in [7].

[2] For some noticeable exceptions in the area of high-speed communication for workstation clusters see [9] and [6].

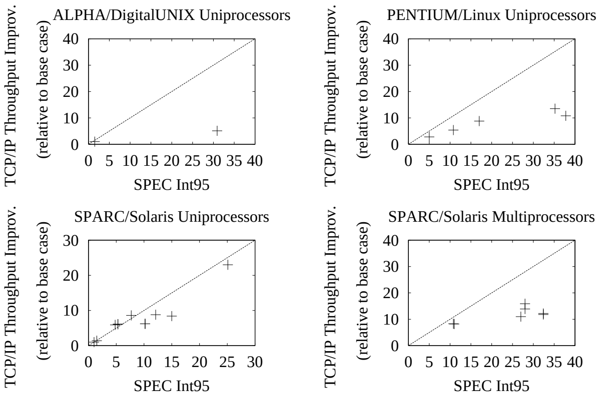

Figure 2: **TCP/IP bandwidth between processors located in the same host as a function of processor speed.** Message size = 64 Kbytes.

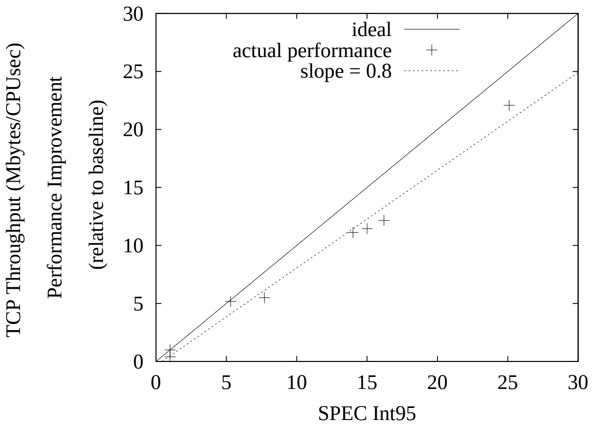

Figure 3: **TCP/IP Bandwidth: 100 Mbps LAN.**

given to the socket layer in chunks 64 Kbytes. Figure 2 plots the TCP/IP throughput achieved in each experiment (expressed as improvement relative to the TCP/IP throughput of the base case). We see that in Alpha-based and in Pentium-based computers the TCP/IP throughput scales very poorly. For example, although processor speed has improved by more than a factor of 35 in both cases, TCP/IP throughput has improved only by a factor of 5 for Alphas, and by a factor of 12 for Pentiums. Contrary to these trends, SPARC-based uniprocessors have somewhat better performance. In most of them (esp. the slow ones), TCP/IP throughput scales linearly with processor speed, a trend, however, that seems to decline in computers rated faster than 10 SPECmarks.

3.4 TCP/IP in a Local Area Network

Although TCP/IP performance between communicating processes located on the same computer revealed significant insight about the scalability of TCP/IP execution, it is important to understand, what is the scalability of TCP/IP execution between communicating processes that reside in different computers, connected through a network. Therefore, in our next experiment we investigate the performance scaling of TCP/IP-based data transfers between processes communicating over a 100 Mbps and a 10 Mbps Local Area Network, using the `ttcp` benchmark. We use various computers as sources of traffic (only one computer transmits at-a-time). The traffic destination for the 10 Mbps network is an ULTRASPARC-1 clocked at 167 MHz connected to a 10 Mbps Ethernet adapter, and the traffic destination for the 100 Mbps network is an ULTRA-4 clocked at 400 MHz.

The performance metric we use in this experiment is not the number of Mbytes transfered per second, but the number of Mbytes transfered per sender's CPU-second, (expressed in Mbytes per CPUsec), that is, the ratio of the amount of data transfered over the (sender's) CPU seconds (including both kernel and user time) that were required for the transfer. Thus, we essentially measure how much data were transfered for each second of CPU time invested. For example, if we transfer 100 Mbytes of data, over a period of 10 seconds, during which the sender has invested 2 seconds of computing power, then the performance we report is 100/2=50 Mbytes/CPUsec. [3]

Figure 3 shows the achieved throughput (as an improvement over the achieved throughput of our base case) for the 100 Mbps LAN. We see that TCP/IP performance generally scales well with processor speed. Although it does not scale exactly the same as processor speed, the line that fits the data has a slope of 0.8, which implies that TCP/IP execution follows processor performance within 80%.

Figure 4 shows the achieved throughput (in Mbytes per CPUsec) over the achieved throughput of our base case for the 10 Mbps LAN. We can see easily that

[3] We did not use the traditional definition of throughput, that is the "Mbytes transfered over the wall clock time elapsed", because, the computers we use are rather fast and can easily saturate the 10/100 Mbps Ethernet network that was connecting them. Therefore, the interconnection network, by being a bottleneck, it would not let us explore how TCP/IP execution scales on the different processors studied. In order to understand the scaling of TCP/IP execution we needed to shift the bottleneck from the Ethernet network to the processor executing the TCP/IP. Thus, measuring "Mbytes per CPUsec", instead of "Mbytes per sec", puts the processor, instead of the network, in the spotlight.

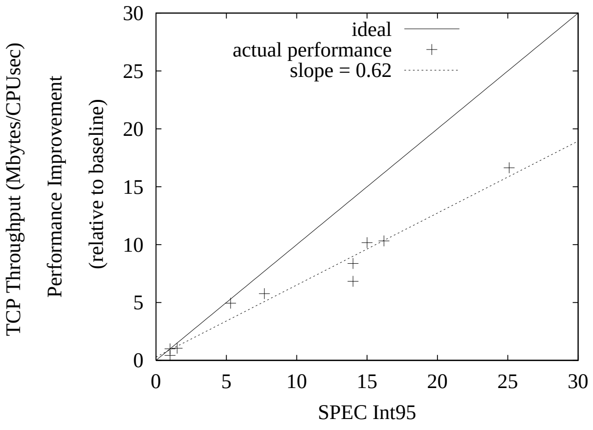

Figure 4: **TCP/IP Bandwidth: 10 Mbps LAN.**

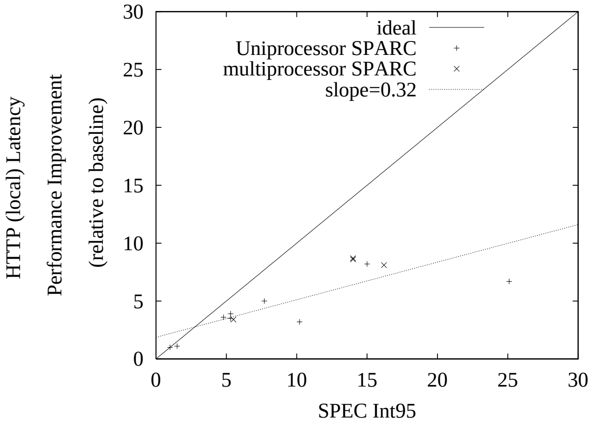

Figure 5: **HTTP latency in localhost as a function of processor speed.**

TCP/IP performance, in this case scales rather poorly with processor speed. Actually, although processor speeds have improved by a factor of 25, the achieved TCP/IP bandwidth has improved only by a factor of 17. As can be seen in figure 4 we fitted the measured data with a straight line, whose slope turned out to be 0.62. This implies that TCP/IP execution scaled (about) 60% as fast as processor speeds, a trend which holds also for communication over Metropolitan Area Networks and Wide Area Networks [7].

3.5 HTTP Performance

Our experiments so far have demonstrated that TCP/IP execution does not scale as well as processors do. It is interesting to know, however, whether this poor scalability propagates to upper-level protocols as well, or it is a minor detail confined within the TCP/IP software. To understand the effect of this poor scalability to upper-level protocols, we use `lat_http`, one of the programs of `lmbench` to measure the performance of the `http` protocol. In our setting, `lat_http` initiates a very simple web server and a client on the same computer. The client requests an html page (about 10 Kbytes large), and its 12 embedded images that are about 40 Kbytes in total. The benchmarks measures the latency to receive all these files. Figure 5 plots the performance measured by `lat_http` (normalized to the base case). We see that HTTP performance scales about 30% as fast as processor speeds - much worse than TCP/IP performance. The reason behind this obviously bad performance is that `lat_http` transfers small objects: HTML pages and images that are no more than a few Kbytes large, while our previous TCP/IP benchmarks transfered chunks of data 64 Kbytes large. Recent studies report that the average file size on the web is 8 Kbytes, while the median file is even smaller: only 3 Kbytes large [2]. When transferring such small files, operating systems are not able to optimize the transfers and necessarily suffer large initialization overheads.

4 Why isn't TCP/IP getting faster as fast as hardware?

All our experiments so far indicate that TCP/IP processing does not get faster as fast as hardware does. There are several reasons that contribute to this performance disparity.

Architectural innovations do not necessarily apply to protocol processing. Recent processors incorporate several architectural innovations, including large caches, out-of-order execution, deep pipelines, and superscalar execution, all of which can not necessarily be exploited by networking code. For example, although large caches improve the performance of programs that repeatedly access large amounts of data, TCP/IP code typically does not exhibit large amount of temporal locality. That is, TCP/IP, and similar communication protocols, do not repeatedly access their data several times, and therefore large caches may not improve their performance significantly. To make matters worse, recent highly optimized protocols (i.e. zero-copy protocols) reduce the number of accesses they make to their data, and therefore, they take even less advantage of the large caches that may exist. Therefore, processors do get faster, but not for protocol-processing type of applications.

Operating system performance lags behind processor speed. This is partly because, recent processors include lots of registers, deep pipelines, and in general a large amount of state, all of which needs to be saved and restored during context switches. Therefore, operating system activities on recent processors, get relatively slower compared to similar activities on older processors.

Memory bandwidth can be a limiting factor. Although protocol processing for small messages is dominated by operating-system related overheads, large message transfers can be limited by memory bandwidth. It is possible that significant improvements in processor speed do not translate in improvements in interprocess communication performance if not accompanied by similar improvements in memory systems. For example, in Fig. 2 we see that there are several computers that are rated between 7 and 15 SPECmarks, that all achieve the same TCP/IP throughput of 8-9 (times better than the base case). Among these computers there is one a SPARC Ultra 5-10 at 440 MHz, and one SPARC Ultra 4 clocked at 83 MHz with a 2-way interleaved memory. The SPARC Ultra 5-10 at 440 MHz, although it is significantly faster than the SPARC Ultra 4, it has about half the memory bandwidth, which turns out to be the limited factor in TCP/IP performance.

5 Conclusions - Future Work

In this paper we experimentally studied the performance cost of TCP/IP in several different processors, ranging from an old 40 MHz SPARC, to a recent 1200 MHZ Pentium. Our results suggest that the disparity between processor speed and TCP/IP performance is large and will probably continue to widen. This calls for new performance metrics which characterize not only the computing capacity, but also the communication capabilities of modern processors [7]. Overall, it becomes increasingly important to understand and optimize the execution of TCP/IP in particular, and protocol software in general, on recent (and future) processors.

Acknowledgments

This project was supported in part by IST project SCAMPI (IST-2001-32404) funded by the European Union. We would like to thank Dionisions Pnevmatikatos and Catherine Chronaki for their constructive comments in earlier versions of this paper. Katerina Gialama, Antonis Danalis, Eva Kalyviannaki, and Xeni Asimakopoulou provided the lmbench results for the LINUX operating system.

References

[1] Hari Balakrishnan, Venkata N. Padmanabhan, and Randy H. Katz. The Effects of Asymmetry on TCP Performance. In *Mobile Computing and Networking*, pages 77–89, 1997.

[2] Paul Barford, Azer Bestavros, Adam Bradley, and Mark Crovella. Web Client Access Patterns: Characteristics and Caching Implications. *World Wide Web Journal*, 2:15–28, 1999.

[3] Jay S. Bayne. Unleashing the POWER of Networks. http://www.johnsoncontrols.com/ Metasys/articles/article7.htm.

[4] Intel Corporation. Intel IXP1200 Network Processor (white paper), 2000. http://developer.intel.com/design/ network/products/npfamily/ixp1200.html.

[5] J. L. Hennessy and D. A. Patterson. *Computer Architecture: A Quantitative Approach (second edition)*. Morgan Kaufmann Publishers, 1996.

[6] M. G.H. Katevenis, E. P. Markatos, G. Kalokairinos, and A. Dollas. Telegraphos: A Substrate for High Performance Computing on Workstation Clusters. *Journal of Parallel and Distributed Computing*, 43(2):94–108, June 1997.

[7] E.P. Markatos. Speeding up TCP/IP: Faster Processors are not Enough. Technical Report TR297, Institute of Computer Science, FORTH, December 2001.

[8] L. McVoy and C. Staelin. lmbench: Portable Tools for Performance Analysis. In *Proc. of the 1996 Usenix Technical Conference*, pages 279–294, January 1996.

[9] S. Mukherjee and Mark D. Hill. A Survey of User-Level of Network Interfaces for System Area Networks. Technical report, Computer Science Department - University of Wisconsin-Madison, 1997.

[10] J.K. Ousterhout. Why aren't Operating Systems Getting Faster As Fast As Hardware? In *Proceedings of the Summer 1990 Usenix Technical Conference*, pages 247–256, June 1990.

[11] C. Partridge and T. Shepard. TCP Performance over Satellite Links. *IEEE Network*, 11(5):44–99, 1997.

[12] Michael Perloff and Kurt Reiss. Improvements to TCP Performance in High-Speed ATM Networks. *Communications of the ACM*, 38(2):90–100, 1995.

[13] USNA. TTCP: a Test of TCP and UDP Performance, 1984.