Impact of the End-System and Affinities on the Throughput of High-Speed Flows

Nathan Hanford, Vishal Ahuja,
Matthew K Farrens, and Dipak Ghosal
Department of Computer Science
University of California
Davis, CA
{nhanford, vahuja, mkfarrens,
dghosal}@ucdavis.edu

Mehmet Balman,
Eric Pouyoul, and Brian Tierney
ESnet
Lawrence Berkeley Labs
Berkeley, CA
mbalman@lbl.gov, lomax@es.net,
bltierney@es.net

ABSTRACT

Network throughput is scaling-up to higher data rates while processors are scaling-out to multiple cores. In order to optimize high speed data transfer into multicore end-systems, network adapter offloads and performance tuning have received a great deal of attention. However, much of this attention is focused on how to set the tuning parameters and which offloads to select for higher performance and not why they do (or do not) work. In this study we have attempted to address two issues that impact the data transfer performance. First is the impact of the processor core affinity (or core binding) which determines the choice of which processor core or cores handle certain tasks in a network- or I/O-heavy application running on a multicore end-system. Second issue is the impact of Ethernet pause frames which provides a link layer flow control in addition to the end-to-end flow control provided by TCP. The goal of our research is to delve deeper into why these tuning suggestions and this offload exist, and how they affect the end-to-end performance and efficiency of a single, large TCP flow.

Categories and Subject Descriptors

C.2.2 [**Computer-Communication Networks**]: Network Protocols; C.2.4 [**Distributed Systems**]: Client/server; C.2.5 [**Local and Wide-Area Networks**]: Internet (e.g., TCP/IP); C.2.m [**Miscellaneous**]: Network performance analysis

Keywords

40 Gbps Network, ESnet, Multi-core Affinitization, End-system Performance, high-speed network, Flow Affinity, Application Affinity, RPS, RFS

1. INTRODUCTION

Several physical constraints have contributed to a processing core to hit a clock speed "wall". On the other hand, the data rates in optical fiber networks have continued to increase, with the physical realities of scattering, absorption and dispersion being ameliorated by better optics and precision equipment. A similar situation holds for short-range copper networks with better quality conductors and better shielding. TCP is a reliable, connection-oriented protocol which guarantees in-order delivery of data from a sender to a receiver. There is a certain amount of sophistication required to implement the functionalities of the TCP protocol which are all instrumented in the end-system since it is an end-to-end protocol. As a result, most of the efficiencies that improve upon current TCP implementations fall into two categories. First, there are offloads which attempt to push TCP functions at (or along with) the lower layers of the protocol stack (usually hardware, firmware, or drivers) in order to achieve greater efficiency at the transport layer. Second, there are tuning parameters, which place more sophistication at the upper layers (software, systems, and systems management). Within the category of offloads, this work focuses on pause frames interesting. Pause frames in Ethernet support link layer flow control by allowing a receiver to inform the upstream node or router to pause sending data as it requires more time to process data that it has already received.

Within the category of tuning parameters, this work focuses on affinity. Affinity (or core binding) is essentially the decision regarding which resources to use on which processor in a networked multiprocessor system. Message Passing in the Linux network receive process in modern systems principally allows for two possibilities: First, there is interrupt processing (usually with coalescing). In this process, the NIC interrupts the procesor once it has received a certain number of packets. Then, the NIC transmits the packets to the processor through DMA and the NIC driver and the OS kernel continue the protocol processing until the data is ready for the application [2]. Second, there is NIC polling (known in Linux as the New API for networks (NAPI)), where the Kernel polls the NIC to see if there is any network data to receive, and if there is, processes the data up the network stack as mentioned above. In either case, there are two types of affinity: 1) Flow affinity which determines the core that is interrupted to process the network flow and 2) Application affinity which determines the core that executes the application process that receives the network data. Flow affinity is set by modifying the hexadecimal core descriptor in /proc/sys/<irq#>/smp_affinity. Application affinity can be set through taskset or similar tools.

2. EXPERIMENTAL SETUP

There are many valid arguments in favor of various NIC offloads. Furthermore, NIC manufacturers typically offer many tuning suggestions to get the most out of the high-performance hardware. However, light is rarely shed on the empirical rationale for these tuning suggestions and offloads. A valuable resource for tuning parameters obtained from careful experimentation on

ANCS'14, October 20–21, 2014, Los Angeles, CA, USA.
ACM 978-1-4503-2839-5/14/10.
http://dx.doi.org/10.1145/2658260.2661772.

ESnet's 100 Gbps testbed is available at http://fasterdata.es.net. ESnet has published a number of papers detailing the experiments that have led to their tuning suggestions. For our experiments, we considered data transfer between two identical 12-core Sandy Bridge host with 40 Gbps NICs connected over the 100 Gbps ESnet testbed. We used iperf3 in zero-copy mode to exhaustively test all 144 combinations of Flow and Application affinity with pause frames on and then off. Finally, we used Oprofile on the receiver to monitor a variety of hardware counters in order to identify the cause of the data transfer performance.

3. SUMMARY OF RESULTS

With pause frames on, we observed packet losses at the upstream router. These losses resulted in TCP congestion control to be invoked which contributed to large throughput variance in the experimental runs. It should be noted that this is a somewhat unusual scenario because of the "quiet" nature of the testbed. Specifically, there was no other traffic through the router and as a result it could dedicate a large buffer to the single flow. However, it still serves as a reminder to avoid setting pause frames on in pause frame enabled routers as they could lead to buffer bloat and potential losses.

Both our current and previous work [3] concluded that there exists three different performance categories corresponding to the following affinitization: 1) Same Socket Same Core (i.e., both Flow and Application affinitized to the same core) reaches a throughput of around 20 Gbps; 2) Different Sockets (thus Different Cores) reaches a throughput of around 28 Gbps; and 3) Same Socket Different Cores reaches a throughput of around 39 Gbps. While changing the OS (from CentOS running a 2.6 kernel to Fedora running a 3.13 kernel) and updating the NIC driver improved the performance, the relative performance for the three affinitization settings remained the same.

Oprofile hardware counter results showed that the main resource consumed was the CPU. This was reflected both in terms of the instructions retired and unhalted clock cycles. When these counters were divided by the amount of data transferred, the efficiency of different affinitizations could be compared. This is shown in Figure 1 for the case with pause frames on. The case where flow and application were affinitized to the same core had by far the worst efficiency measured in terms of CPU-utilization per Gigabyte transferred. Due to the correlation between cache and memory transactions and CPU utilization, it appears that the NIC driver could be spinning while waiting for memory. A possible explanation is that using two different cores on the same socket doubles the amount of L1 cache available to the NIC driver and the application in comparison to the case when they are both on affinitized to the same core.

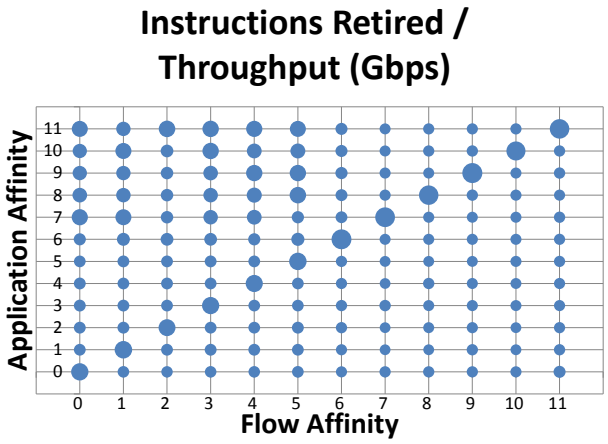

Figure 1: The width of the bubbles represents the amount of instructions retired divided by the throughput in Gbps for each of the 144 tests.

4. CONCLUSION AND FUTURE WORK

One of the most important results of the clock speed wall, (or the "hiatus" of Moore's law) is that the line between intra-system and inter-system communication is rapidly blurring. For one processor core to communicate with another, data must traverse an intra-system (on-chip) network. For large-scale data replication and coherency, data must traverse a WAN. How are these networks meaningfully different? WAN data continues to become less of a limiting factor, and routers and networks are becoming more reliable and more easily reconfigurable. At the same time, intra-system networks are becoming more complex (due to scale-out systems and virtualization), and perhaps less reliable (as energy conservation occasionally demands that parts of a chip could be slowed down, or turned off altogether). When discussing affinitization, it becomes obvious that despite these changes, distance and locality still matter, whether the network is "large" or "small". Therefore, in the future, the most efficient solution may be not only to integrate a NIC onto the processor die [4], but perhaps even integrate the functionality with existing I/O structures, such as the north bridge. However, the feasibility of doing so may be years away.

In the meantime, other NICs and especially, other NIC drivers are being tested in similar ways to see if results are similar, and if generalizations can be made. The relatively recent advancement in NIC drivers that automatically switch between interrupt coalescing and NAPI is also being tested. Finally, results for practical, multi-stream TCP, and UDT GridFTP transfers are being tested along these lines. One future goal could be to implement a lightweight middleware tool that could optimize affinitization on a larger scale, extending the work that has been carried out on Cache Aware Affinitization Daemon [1].

5. ACKNOWLEDGMENTS

This research used resources of the ESnet Testbed, which is supported by the Office of Science of the U.S. Department of Energy under contract DE-AC02-05CH11231. This research was also supported by NSF grant CNS-0917315.

6. REFERENCES

[1] Ahuja, V., Farrens, M., and Ghosal, D. Cache-aware affinitization on commodity multicores for high-speed network flows. In *Proceedings of the eighth ACM/IEEE symposium on Architectures for networking and communications systems* (2012), ACM, pp. 39–48.

[2] Benvenuti, C. *Understanding Linux Network Internals*. O'Reilly Media, 2005.

[3] Hanford, N., Ahuja, V., Balman, M., Farrens, M. K., Ghosal, D., Pouyoul, E., and Tierney, B. Characterizing the impact of end-system affinities on the end-to-end performance of high-speed flows. In *Proceedings of the Third International Workshop on Network-Aware Data Management* (New York, NY, USA, 2013), NDM '13, ACM, pp. 1:1–1:10.

[4] Liao, G., Zhu, X., and Bhuyan, L. A new server i/o architecture for high speed networks. In *High Performance Computer Architecture (HPCA), 2011 IEEE 17th International Symposium on* (2011), IEEE, pp. 255–265.