Performance Measurement of an Integrated NIC Architecture with 10GbE

Guangdeng Liao
Department of Computer Science and Engineering
University of California, Riverside
Riverside, California, USA
gliao@cs.ucr.edu

Laxmi Bhuyan
Department of Computer Science and Engineering
University of California, Riverside
Riverside, California, USA
bhuyan@cs.ucr.edu

Abstract— **The deployment of 10 Gigabit Ethernet (10GbE) connections to servers has been hampered by the "fast-network-slow-host" phenomenon. Recently, the integration of network interfaces (INICs) is proposed to tackle the performance mismatch. While significant advantages over PCI-based discrete NICs (DNICs) were shown in prior work using simulation methodologies, it is still unclear how INICs perform on real machines with 10GbE.**

This paper is the first to study the impact of INICs by extensive evaluations through micro-benchmarks on a highly threaded Sun Niagara 2 processor. The processor is the industry's first "system on a chip," integrating two 10GbE NICs. We observe that the INIC only shows its advantage over the DNIC with large I/O sizes. It improves 7.5% network bandwidth while saving 20% relative CPU utilization. We characterize the system behaviors to fully understand the performance benefits with respect to different number of connections, OS overhead, instruction counts, and cache misses etc. All of our studies reveal that there is a benefit of integrating NICs onto CPUs, but the gain is somewhat marginal. More aggressive integrated NIC designs should be adopted for higher speed networks like the upcoming 40GbE and 100GbE.

Keywords: 10GbE, Integrated NIC, Discrete NIC, Performace Evaluation, Characterization, Sun Niagara 2.

I. INTRODUCTION

Ethernet continues to be the most widely used network architecture today for its low component cost and backward compatibility with the existing Ethernet infrastructure. As of 2006, Gigabit Ethernet-based clusters make up 176 (or 35.2%) of the top-500 supercomputers [17]. Unfortunately, even as nearly all server platforms completed the transition to Gigabit Ethernet, the adoption of 10 Gigabit Ethernet (10GbE) has been limited to a few niche applications [18]. The use of 10GbE has been constrained by the processing capability of general purpose platforms [7, 11].

Prior work [1, 2, 3, 5, 14] for improving the processing capability broadly falls into two categories: 1) dedicating an embedded CPU to a NIC and 2) integrating a NIC onto a CPU. TCP Offload Engine (TOE), a popular approach for high speed networks, belongs to the first category. It offloads the whole network stack running on operating systems (OS) into NICs in the form of firmware. However, TOE lacks flexibility and the ability to take advantage of technology-driven performance improvements as easily as host CPUs [5]. Recently, an alternative approach to integrating NICs onto CPUs has been shown to be more promising and is gaining more and more popularity in both academia and industry [3, 14, 15]. Compared to TOE, the integration of NICs does not require modifying the legacy network stack, and provides high flexibility and good compatibility with the OS.

Existing work on the integration of NICs (INICs) was evaluated by simulation [1, 2, 3]. Although simulation is flexible, it is hard to fully simulate the bandwidth and latency of memory and system bus protocols in real machines. It is also difficult for simulators to capture the whole OS behaviors. Hence, evaluations on real machines become critically important and are complementary to simulators. Papers [1, 2] claimed that INICs can significantly improve network processing efficiency in comparison with discrete NICs (DNICs) connected via a PCI-E bus, due to the smaller latency of accessing I/O registers. However, how integrated NICs perform on real machines remains unclear. A detailed performance evaluation and characterization are required to answer it.

Sun released the UltraSPARC T2 processor (a.k.a Niagara 2) [15] by envisioning the benefits of the integration of NICs. The processor is a highly threaded processor consisting of small cores and the industry's first "system on a chip" integrating two 10GbE NICs. The integration can reduce the latency of accessing I/O registers, though the overhead of accessing network packets is not eliminated because they are still sourced and destined to memory rather than caches [15]. In this paper, we present extensive evaluations with 10GbE, and compare the INIC with the DNIC in detail. To make a fair comparison, the INIC has the same design as the DNIC except for its proximity to CPUs in our experiments. Our experiments reveal that the INIC improves network efficiency only with large I/O sizes, with 7.5% higher bandwidth and 20% less CPU utilization. Our CPU breakdown confirms the previous observation from the simulator that the driver overhead is largely reduced (50% reduction in our experiments, and up to 80% reduction in [2]). The reduction is contributed to the smaller latency of accessing I/O registers.

1550-4794/09 $26.00 © 2009 IEEE
DOI 10.1109/HOTI.2009.16

Besides confirming the simulator-based finding above, through a detailed performance characterization, we also unexpectedly observe that the INIC significantly affects the behaviors of the OS scheduler and CPU caches. We notice that the INIC reduces context switches by 40% in comparison with the DNIC. Our in-depth analysis shows that more cross-calls (or inter-processor interrupts) are incurred by the OS scheduler, and correspondingly result in more frequent context switches in the DNIC. Additionally, the longer packet processing latency in the DNIC directly translates to longer residential life cycles of packets in caches. It could result in cache pollution and thus incur higher cache miss rates. With the combination of influence of more context switches on caches, our evaluation shows that the INIC has 25% lower L1 data cache and 7.6% lower L2 cache miss rates.

In our experiments, we observe that the smaller latency of accessing I/O registers itself does not help processing by a large extent. The different behaviors of the OS scheduler and CPU caches incurred by the smaller latency mainly contribute to the performance gain. It is in contrary to the previous observation that the reduced driver overhead can lead to the performance improvement up to 58% [2]. To satisfy the processing requirement introduced by higher network traffic rates, more aggressive INIC designs like the new CPU/NIC interaction mechanism should be considered.

The remainder of this paper is organized as follows. Section 2 describes the background knowledge about the integration of NICs. Section 3 presents the experimental methodology. Section 4 and 5 show our performance evaluation and detailed characterization. Finally, we conclude our paper in section 6.

II. BACKGROUND

A. Integration of NICs

It is well known that TCP/IP over Ethernet is a dominant overhead for commercial web and data servers [19, 20]. Researchers gradually realized that a comprehensive solution across hardware platforms and software stacks is necessary to eliminate the overhead [1, 2]. Existing work for improving TCP/IP performance falls into two categories: TOE [5] and the integration of NICs [1, 2]. Although TOE reduces the communication overhead between processors and NICs, it lacks scalability due to the limited processing and memory capacity. It also requires extensive modification of OS and development of firmware in NICs. Recently, a counter-TOE approach is to integrate NICs onto CPUs. It is envisioned as the next generational network infrastructure. It not only reduces the latency of accessing I/O registers, but also leverages extensive resources in multi-core CPUs.

Binkert [1, 2] made a first attempt to couple a simple NIC with a CPU for high bandwidth networks. They claimed that the device driver is one of the dominant overheads for processing high speed networks and an integrated NIC can eliminate the overhead. Additionally, they also go further to redesign the integrated NIC to eliminate the overheads of DMA descriptor management and data copy. Evaluation on their full system simulator M5 [2, 3] showed the driver overhead is reduced up to 80% even without any redesign, thus improving performance up to 58%.

The Joint Network Interface Controller (JNIC) [14], a collaborative research project between HP and Intel, also attempted to explore high performance in-data-center communications over Ethernet by integrating a NIC. They built a system prototype by attaching a 1GbE NIC on the front side bus to mimic the integration. Apparently, the integration is drawing more and more attention to eliminating the disparity between host computation capacity and high speed networks.

B. Sun Niagara 2

The Niagara 2 processor is the industry's first "system on a chip," packing the most small underpowered cores and threads, and integrating all the key functions of a server on a single chip: computing, networking, security and I/O [15].

As shown in Figure 1, it has two 10 GbE NICs (NIU in the figure) with a few features. All the data is sourced from and destined to memory, DMA in the parlance. This means a core sets up the transfer and gets out of the way. The path to memory goes from the Ethernet unit (NIU), to the system interface unit (SIU), directly into the L2 or the crossbar. The CPU sets up DMA for packet transfers from the NIC to memory.

Niagara 2, known for its massive amount of parallelism, contains eight small SPARC physical processor cores and each core has full hardware support for eight hardware threads. There are total 64 hardware threads or CPUs from the OS perspective. Additionally, each core has a 64-entry fully associative ITLB, a 128-entry fully associative DTLB, a 16K L1 I (instruction) cache and an 8K L1 D (data) cache with associativity of the Icache upped to eight. The Dcache has four-way associativity and is write-through, and all of the cores share a 4MB L2 cache. This is divided into 8 banks and each bank is 16-way associative.

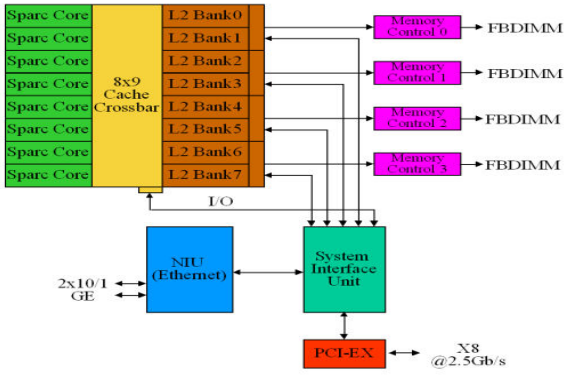

Figure 1. Niagara 2 Architecture

III. EXPERIMENT METHODOLOGY

A. Testbed Setup

Our experimental testbed consists of a Sun T5120 server connected to an Intel® Quad Core DP Xeon® server, which

functions as a System Under Test (SUT) and a stressor respectively. The Sun server has a Niagara 2 processor, which has 64 hardware threads and each hardware thread is operating at 1.2GHz. The Intel server is a two-processor platform based on the quad-core Intel® Xeon® processor 5300 series with 8 MB of L2 cache per processor [8]. Both of the machines are equipped with 16GB DRAM.

TABLE I. INIC VS DNIC

Features	NIU(INIC)	Neptune (DNIC)
Transmit DMA Channels	8	12
Receive DMA Channels	8	8
Bus interface	No	8 lane PCI Express
Bus bandwidth limit	No	16 Gbits/s each direction
Transmit Packet Classification	Software	Software
Receive Packet Classification	Hardware	Hardware

In order to compare the integrated NIC with the discrete NIC, we used two 10GbE network adapters in the SUN server: a discrete Sun 10GbE PCI-E NIC (a.k.a Neptune) [16] and an on-chip 10GbE Network Interface Unit (a.k.a NIU) [15]. The on-chip NIU has the same physical design as Neptune except it has half less DMA transmit channels. More information is shown in Table 1. They use the same device driver, and trigger an interrupt after the number of received packets reaches 32 or 8 NIC hardware clocks have elapsed since the last packet was received. We also installed two Intel 10GbE Server Adapters (a.k.a Oplin) [9] in the stressor system to connect two network adapters in the Sun server. All of discrete NICs connect to hosts through PCI-E x8, a 16+16 Gigabit/s full-duplex I/O fabric that is fast enough to keep up with the 10+10 Gigabit/s full-duplex network port.

B. *Server Software*

The SUT runs the Solaris 10 Operating System while the stressor runs Vanilla Linux kernel 2.6.22. In Solaris 10, a STREAMS-based network stack is replaced by a new architecture named FireEngine [6] which provided better connection affinity to CPUs, greatly reducing the connection setup cost and the cost of per-packet processing. It merges all protocol layers into one STREAMS module that is fully multithreaded.

In order to optimize network processing with the 10GbE network, we use 16 soft rings per 10GbE NIC by setting the parameter *ip_soft_rings_cnt* for the driver. Soft rings are kernel threads that offload processing of received packets from the interrupt CPU, thus preventing the interrupt CPU from becoming the bottleneck. We also set *ddi_msix_alloc_limit* to 8 so that received interrupts can target 8 different CPUs. Besides, we retain the default settings in the device driver without specific performance tuning on interrupt coalescing, write combining etc. All protocol and system relevant settings are at default.

Micro-benchmarks were used in our experiments to easily identify the performance benefits and avoid system noises from commercial applications [7, 11], We selected Iperf [10] and NetPIPE [13] as micro-benchmarks for measuring bandwidth and ping-pong latency respectively. Because peak bandwidth can be achieved by more than 16 connections, Iperf is run with 32 parallel connections on 64 CPUs for 60 seconds in all our experiments, unless otherwise stated.

In our experiments, the utility *vmstat* is used for capturing the corresponding CPU utilization. We ran profiling tools *er_kernel* and *er_print* to collect and analyze the system functions overhead. Meanwhile, tools *busstat* and *cpustat* were chosen to obtain memory traffic and hardware performance counter statistical information while running the benchmark.

IV. PERFORMANCE EVALUATION

In Figure 2, we show how the INIC and the DNIC perform with various I/O sizes while receiving packets. The bar in the figure represents achievable network bandwidth, and the line stands for the corresponding CPU utilization. It can be observed that the INIC can achieve 8.97 Gbps bandwidth while consuming 27% CPU utilization with large I/O sizes. Correspondingly, 8.31 Gbps bandwidth is obtained by the DNIC with 35% CPU utilization. The INIC obtains 7.5% higher bandwidth and saves 20% relative CPU utilization on average for large I/O sizes (>1KB). The efficiency of the INIC is close to the DNIC with small packets. All of the results reveal that the integration improves network efficiency in the receive side only with large I/O sizes.

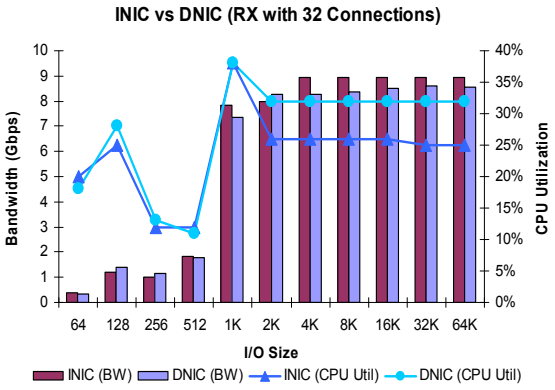

Figure 2. Bandwidth & CPU Utilization (RX)

We studied the performance comparison of the DNIC and the INIC while transmitting packets in Figure 3. Because less time is required in the driver for the INIC to transmit packets, it is expected that the higher transmitting bandwidth could be obtained by the INIC than the DNIC. However, the INIC does not show noticeable benefits to the application in terms of network efficiency. It is possibly because: first, the number of transmit DMA channels in NIU is half less than that in the Neptune 10GbE card (8 TX DMA channels in the INIC and 12 TX DMA channels in the DINC). Fewer channels could reduce the capacity of transmitting packets. Second, the

transmit side is much less latency-sensitive than the receive side [12, 19, 20].

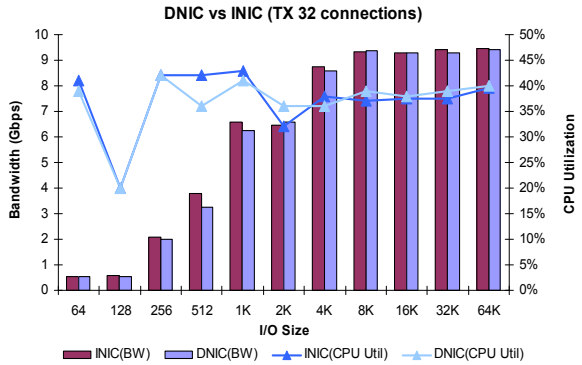

Figure 3. Bandwidth & CPU Utilization (TX)

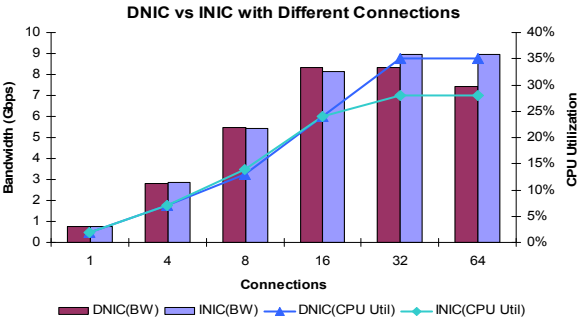

Figure 4. Performance with Various Connections

To ease and expedite our analysis of the above observation in receive side, we conducted experiments for comparing the INIC with the DNIC by running Iperf with varying number of connections rather than 32 connections. Figure 4 illustrates the comparison from one single connection to 64 connections with 64KB messages. The following observations can be made from the figure: 1) greater than 16 connections are required for both the INIC and the DNIC to achieve peak bandwidth. It is due to low performance of a single hardware thread in Niagara 2; 2) differing from the INIC, the DNIC with 64 connections downgrades 10% bandwidth compared to 32 connections; 3) the INIC improves network efficiency only with greater than or equal to 32 connections.

Similarly, we also studied the performance comparison by running 32 connections with varying number of CPUs or hardware threads in Figure 5. We observe from the figure that the benefits only come when more than 16 CPUs are used in our experiments. With the combination of Figure 4, we can draw two conclusions: 1) the integration could affect the system behaviors with a large number of connections, and different system behaviors mainly cause the performance difference, and 2) the benefits can only be achieved with large number of CPUs, and thus are tied to the highly threaded Sun system.

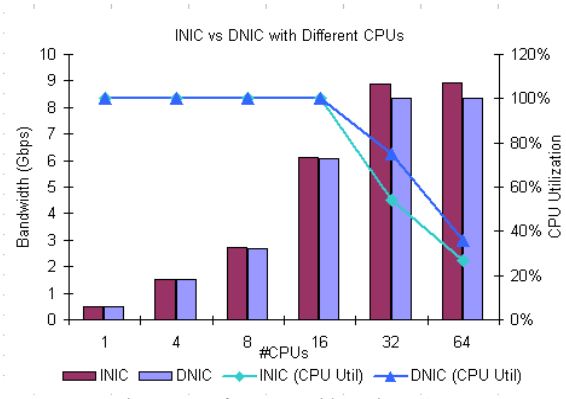

Figure 5. Performance with Various CPUs

High bandwidth and low latency are two main metrics in modern networking servers. We also conducted experiments to compare ping-pong latency by configuring the SUT with the INIC or the DNIC while retaining the same configuration in the stressor. The micro-benchmark NetPIPE was used to measure the latency. Since large I/Os are segmented into small packets less than MTU (Maximum Transfer Unit, 1.5KB by default), we focus on packets less than MTU for the ping-pong latency test. Our results in Figure 6 show that the INIC can achieve a lower latency by saving 6 µs. It is due to the smaller latency of accessing I/O registers and eliminating PCI-E bus latency.

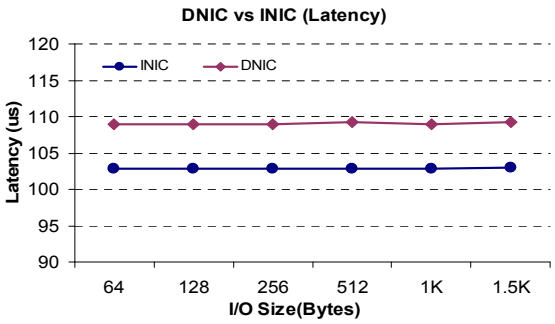

Figure 6. Ping-Pong Latency

V. Detailed Performance Characterization

To further understand the benefits of the INIC, we profiled the system for both the kernel and application function calls as well as the assembly code. We used the test case with a 64KB I/O size and 32 concurrent connections in Figure 2. The data gathered was grouped into the following components to determine their impacts on performance: device driver, socket, buffer management, network stack, kernel, data copy and user application Iperf.

CPU overhead breakdown per packet is calculated and presented in Figure 7. We observe that 28 µs and 20 µs are required for processing one received packet in the DNIC and the INIC respectively.

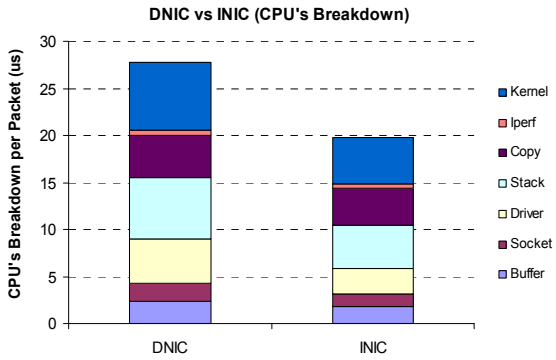

Figure 7. CPU Overhead Breakdown

The comparison in the figure reveals that the CPU overhead on the driver is reduced from 4.7 µs to 2.6 µs by the integration. Our profiled result shows that the overhead on the interrupt handler *nxge_rx_intr,* which frequently operates on NIC registers, is reduced by 10X. The copy component remains the same when we switch between the DNIC to the INIC configuration. It is because all packets in the INIC are sourced and destined to memory rather than caches. The data copy from kernel to user buffers in both configurations incurs compulsory cache misses to fetch payloads from memory into caches. The overhead on the copy component is eliminated only if packets are delivered to caches. Our findings so far confirm the observations in prior work [1, 2] even though they differ in absolute benefits.

We also observe that the INIC also reduces the overheads on network stack, buffer management, socket and kernel. These unexpected improvements comprise up to 75% of the total overhead reduction and thus mainly contribute to the performance benefits. We found that the different behaviors of the OS scheduler and CPU caches lead to these benefits.

A. Impacts on the OS Scheduler

Since the benefits of the INIC over the DNIC changes as the number of connections increases, we carefully characterize the system behaviors with varying number of connections to understand the benefits by the INIC.

1) Instruction Breakdown

First, we did an architectural characterization by instruction for packet processing along various connections. In the DNIC, instructions are broken down into 5 types of instructions: load, store, atomics, software count instructions and all other instructions as shown in Figure 8 (Note that the received packet is less than 1.5KB because large messages in the sender side are segmented into packets smaller than MTU). As shown in Figure 8, about 3500 instructions are required to process a packet with less than 32 connections, but increase to 4500 instructions for 32 and 64 connections. The instruction breakdown shows that the instruction types of load, store and other instructions, increase proportionally. Figure 9 shows the similar behavior for the INIC, but contrary to the DNIC, increased connections do not significantly increase instructions per packet. The higher instructions per packet directly translate to the higher CPU utilization of the DNIC with a large number of connections.

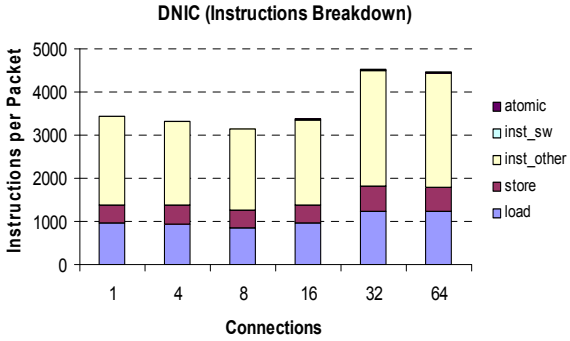

Figure 8. Instruction Breakdown (DNIC)

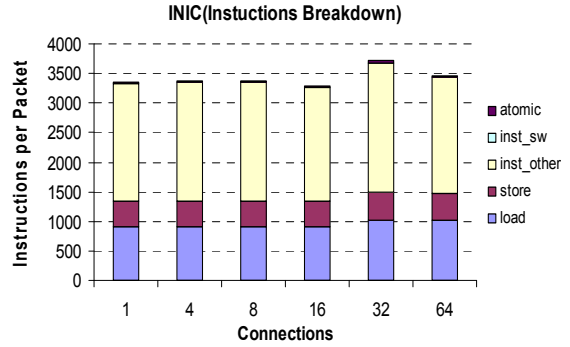

Figure 9. Instruction Breakdown (INIC)

2) Context Switch Rate

Because the same device driver and network stack are used, the INIC and the DNIC have the same code path while processing packets. The increased instructions are incurred by other components in the OS. The increased load and store operations reveal that more context switches could be required by the DNIC. Hence, we studied the OS scheduler's behaviors while processing packets along various connections. Average context switches per second are presented in Figure 10. The figure confirms our deduction that more context switches are incurred by the DNIC with more than 16 connections.

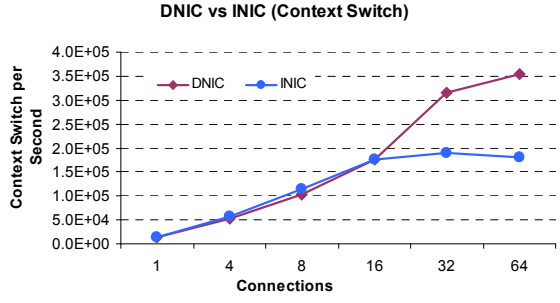

Figure 10. Context Switches with Various Connections

3) Interrupt Rates

Since the micro-benchmark was used in our experiment, the lightweight execution in applications does not incur system noise and yields few context switches. Context switches are mainly caused by system interrupts. Hence, we studied system interrupts per second along various connections in Figure 11. The result lines up with the observation in Figure 10. Both the INIC and the DNIC have comparable interrupt rates with less than 32 connections. When we come to the scenario beyond 16 connections, the DNIC largely increases the interrupt rate but the INIC keeps the same interrupt rate. The higher interrupt rate results in more context switches. To study the increased interrupts, we breakdown system interrupts with 32 connections into interrupts from the NIC, cross-calls, and all other system interrupts in Figure 12.

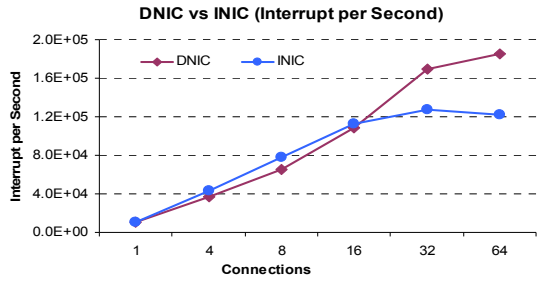

Figure 11. Interrupts per Second

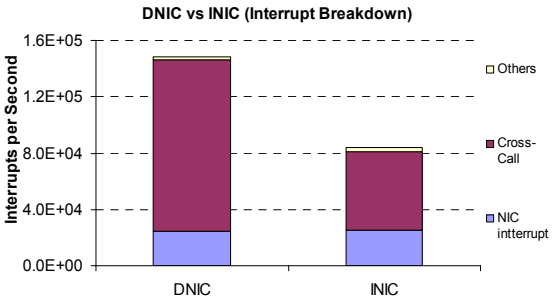

Figure 12. System Interrupts Breakdown

We notice that the INIC sent slightly more interrupts than the DNIC because of the higher bandwidth. However, the system with the DNIC is interrupted much more frequently than with the INIC by cross-calls. We used the Dtrace utility [4] to count the number of cross-calls incurred by various system components. It shows that more than the 96% cross-calls are from the OS scheduler. The scheduler uses cross-calls to notify other CPUs of running tasks or threads immediately.

We also profiled the usage for all 64 CPUs from the OS perspective and found that more CPUs were used by the system with the DNIC. Specifically, only 18 CPUs were free with the DNIC, while 31 CPUs are available with the INIC. (Note that one connection might require more than one CPU for processing in the Sun system [6, 15]). The result reveals that the OS scheduler with the DNIC uses the cross-calls to distribute threads to more CPUs as compared to the INIC. It is because the lower processing latency with the integration makes running cores more efficient and lowers the likelihood that packets are dispatched to other idling cores.

B. *Impacts on the CPU Caches*

Since lower processing latency intuitively embeds shorter residential life cycles of network data in caches, the integration could also bring impacts on CPU caches. We studied cache behaviors in the system with the INIC and with the DNIC respectively.

Starting from the instruction cache, we show the instruction misses per packet in Figure 13. More context switches incur higher miss rates beyond 16 connections. We studied the instruction misses in L2 cache in Figure 14 to investigate the impacts of those misses on the unified L2 cache. Their performance is similar but misses happen very rarely in larger L2 cache.

We also show data behaviors in both L2 and L1 data caches. We captured data misses per packet in L2 cache for both the DNIC and the INIC in Figure 15. It shows they have comparable miss rates with less than 32 connections. When it comes to beyond 16 connections, the INIC has 7.6% reduction of misses. The misses in the data cache behave similarly as shown in Figure 16, but we see a much larger gap between the DNIC and the INIC. The INIC has 180 fewer misses or 42% reduction of misses at most.

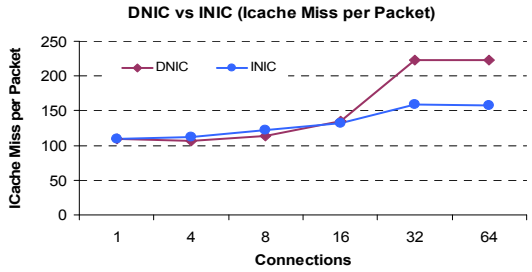

Figure 13. Icache Misses per Packet

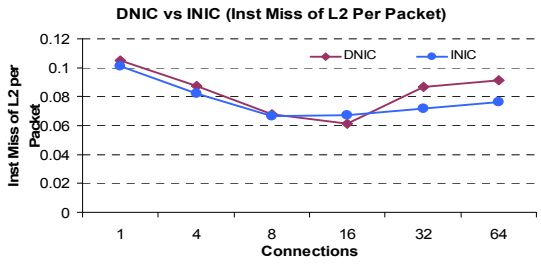

Figure 14. Instruction Misses per Packet in L2

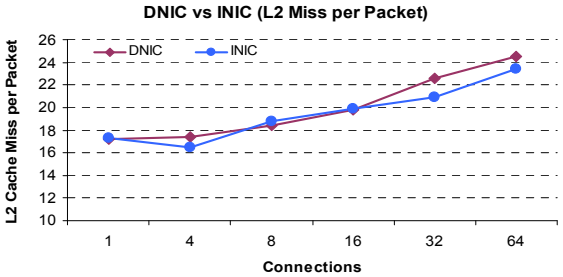

Figure 15. Data Misses per Packet in L2

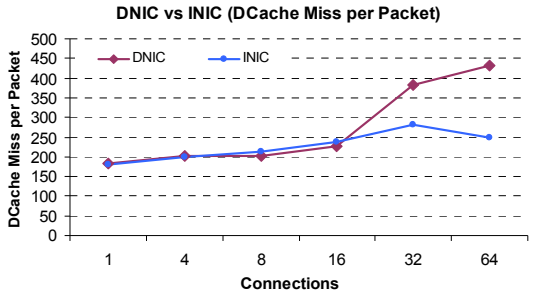

Figure 16. Data Cache Misses per Packet

In our system, the L2 cache is a 4MB cache and the total data cache size of eight cores is 64KB. They can accommodate up to 64 and 1 64KB I/O sizes respectively. We need control plane data structures such as TCP Control Block (TCB) and headers, descriptors etc during packet processing. With the increased connections, we actually need more cache size for simultaneous control plane processing. For example, different connections need to lookup different entries in the TCB. Hence, the smaller access latency to I/O registers in the INIC is beneficial. The smaller latency means that packets can be provided for upper level processing faster than the DNIC, correspondingly resulting in smaller processing latency. Hence, in the same time interval, less packet footprints are left in caches with the INIC and more cache spaces can be used for other data. The above behavior could incur the lower miss rate with the INIC. Two conclusions can be drawn from our analysis: 1) the smaller latency could explain the difference between cache misses, and 2) the difference caused by the smaller latency is sensitive to the cache size. It explains why the difference on data cache is much larger than that on L2 cache.

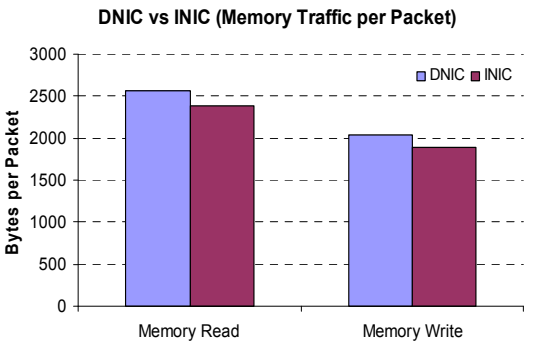

Figure 17. Memory Traffic per Packet

Last but not least, we also captured traffic on the memory bus. More cache misses would lead to more memory accesses and thus increase memory read traffic. We gathered the memory traffic for both read and write operations with the INIC and the DNIC while running Iperf for 60 seconds. The memory traffic, normalized to per packet in Figure 17, shows that the DNIC incurs more memory read and write accesses.

Although both the behaviors of the OS scheduler and CPU caches are influenced by the integration, we believe that there is some correlation between them. Besides the impact of different processing latency on CPU caches, more context switches also change the working data set in caches and thus incur some cache misses. Unfortunately, the paper now is unable to quantify their impacts on CPU caches.

VI. Conclusion and Future Work

The integration of NICs has been proposed for improving the processing capacity of general purpose platforms with high speed networks. Their performance benefits over discrete NICs are usually verified by simulators.

In this paper, we conducted extensive evaluations on a real machine with integrated NICs to study their performance benefits over discrete NICs. A detailed performance characterization is provided to understand the benefits. We show that the smaller latency of accessing I/O registers with the integration positively affects the behavior of the OS scheduler and CPU caches through fewer context switches and cache misses. Through our thorough studies, we make the following observations: 1) the driver overhead is largely reduced due to the smaller latency of accessing I/O registers, which confirms the observation from prior simulation-based work, 2) the INIC affects the behaviors of the OS scheduler and CPU caches, mainly resulting in the performance benefits, which is unexpected, 3) the performance benefits of the INIC are tied to the highly threaded Sun system. SUN Niagara 2 uses a simple Integrated NIC architecture. We believe that the performance can be further improved by redesigning the interaction mechanism between CPUs and NICs in the next generation I/O infrastructure. In future, we plan to study the performance benefits of the integrated NIC with real workloads like NFS and web servers.

Acknowledgment

The research was supported by NSF grants CCF-0811834 and NEDG-0832108, and a grant from Intel. We would like to thank Ram Huggahalli, Xia Zhu, Amit Kumar and Steve King for providing us the Sun Niagara 2 equipment and invaluable guidance while we were conducting experiments. We would also like to thank my labmates Danhua Guo, Chen Tian and Dennis Jeffery for their useful comments on this paper.

References

[1] Nathan Binkert, Ali G. Saidi, Steven K. Reinhardt, Integrated network interfaces for high-bandwidth TCP/IP. Proceedings of the 2006 ASPLOS Conference. December 2006.

[2] Nathan L. Binkert, Lisa R. Hsu, Ali G. Saidi et al., Performance Analysis of System Overheads in TCP/IP Workloads. Proc. 14th Int'l Conf. on Parallel Architectures and Compilation Techniques (PACT), Sept. 2005.

[3] Nathan. L. Binkert, Ronaldo G. Dreslinski, Lisa R. Hsu et al., The M5 simulator: Modeling networked systems. IEEE Micro, Jul/Aug 2006.

[4] Dtrace, http://en.wikipedia.org/wiki/DTrace.

[5] Doug Freimuth, Elbert Hu, Jason LaVoie et al., Server network scalability and TCP offload, Proceedings of the USENIX Annual Technical Conference, Anaheim, CA. 2005.

[6] Fireengine, http://www.sun.com/bigadmin/content/networkperf/FireEngine_WP.pdf.

[7] Ram Huggahalli, Ravi Iyer, Scott Tetrick, Direct Cache Access for High Bandwidth Network I/O, 32nd Annual International Symposium on Computer Architecture (ISCA'05), 2005.

[8] Intel Core 2 Extreme quad-core processor, http://www.intel.com/products/processor/core2XE/.

[9] Intel 10 Gigabit Ethernet Controllers, http://download.intel.com/design/network/prodbrf/317796.pdf

[10] Iperf, http://dast.nlanr.net/Projects/Iperf

[11] Amit Kumar, Ram Huggahalli, Impact of Cache Coherence Protocols on the Processing of Network Traffic. 40th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO-40), 2007.

[12] Srihari Makineni, Ravi Iyer, Architectural characterization of TCP/IP packet processing on the Pentium M microprocessor. 10th HPCA, 2004.

[13] Netpipe. http://www.scl.ameslab.gov/netpipe/

[14] Michael Schlansker, Nagabhushan Chitlur, Erwin Oertli et al., High-Performance Ethernet-Based Communications for Future Multi-Core Processors. Proceedings of the 2007 SuperComputing Conference. November 2007.

[15] Sun Niagara 2, http://www.sun.com/processors/niagara/index.jsp.

[16] Sun 10GbE multithread Networking Cards, http://www.sun.com/products/networking/ethernet/10gigethernet/

[17] Top500 Supercomputer List, http://www.top500.org.

[18] Worldwide Ethernet Semiconductor 2006–2011 Forecast, Research Report # IDC204254, November 2006.

[19] Haiyong Xie, Li Zhao, Laxmi Bhuyan, Architectural Analysis and Instruction Set Optimization for Network Protocol Processors, Proc. IEEE ISSS+CODES, October 2003.

[20] Li Zhao, Ramesh Illikkal, Srihari Makineni, Laxmi Bhuyan, TCP/IP Cache Characterization in Commercial Server Worloads, CAECW-7.