

Increasing Performance in Commodity IP

Patricia Gilfeather, Principal Investigator
Scalable Systems Lab
Department of Computer Science
University of New Mexico
pfeather@cs.unm.edu

Arthur B. Maccabe
Scalable Systems Lab
Department of Computer Science
University of New Mexico
maccabe@cs.unm.edu

Todd Underwood
Oso Grande Technologies
todd@nm.net

Abstract

Clusters built from commodity hardware and software have several advantages over more traditional supercomputers. Commodity clusters are cheap and ubiquitous, and so they are easier to design, program and maintain. However, even as high-speed networks reach 10Gb/s speeds and modern computer architectures reach past 1Ghz, new commodity clusters are not able to harness this power.

One of the bottlenecks to high-performance computing using commodity hardware and protocols is interrupt pressure. We examine several methods used to relieve interrupt pressure in practice and propose a new solution, offloading IP fragmentation and reassembly onto a network interface card (NIC). Our proposed solution provides the performance gains of lower communication overhead and high bandwidth without sacrificing the advantages of using commodity hardware and protocols.

1 Introduction

On commodity hardware, 100 Mb/s Ethernet adapters are capable of near line-rate TCP/IP performance (90%) with low CPU utilization (15-20%). Projecting this performance out to Gigabit speeds, we can expect to see bandwidth between about 450 Mb/s and 600 Mb/s at near 100% CPU utilization. Bottlenecks to high speed performance no longer reside with the network infrastructure. With modern commodity hardware supporting memory-to-IO bandwidths of 480MB/s (3840 Mb/s), even memory performance is not the limiting factor. In fact, interrupt handling and the resulting CPU utilization is the major performance issue in efficient high-speed networks.

There have been several moderately successful attempts

at addressing the new performance bottlenecks in high-performance local area network computing. The first, most-obvious choice, is to use protocols more suited to high-speed networking, and there are several well-established protocols in the research community. However, since IP remains the most common protocol within LAN applications and certainly in the WAN world, IP implementations are widespread and therefore inherently appealing. Changing protocols means maintaining protocol stacks separately for the WAN (which remains TCP/IP) and the LAN and also requires a great deal of re-engineering. We sought to implement a scalable and efficient way of addressing the major bottleneck in high-speed networks, while still maintaining all of the advantages of a commodity protocol.

1.1 Interrupt Pressure

Interrupt pressure is the most significant concern for high-speed Ethernet networks, yet it is largely overlooked as a performance bottleneck. Table 1 presents the minimum inter-arrival times for 1500 byte packets at various network speeds. The minimum inter-arrival time is calculated by assuming a constant stream of 1500 byte packets. For example, the inter-arrival time for a 10Mb/s network is calculated as follows:

$$1500\text{bytes} \times 8\text{bits/byte} \times 1\mu\text{sec}/10\text{bit} = 1200\mu\text{sec}$$

If the network interface generates an interrupt for each 1500-byte packet, a processor with a 1Gb/s network interface should be capable of handling an interrupt every 12 μsec . Although estimates of interrupt handling latency vary considerably and should be measured on each system, the PCI 2.1 specification estimates typical 33MHz latencies of 10 μsec to 20 μsec . Expected latencies on 66MHz buses

Network Speed	Inter-arrival time
10 Mb/s	1200 μ sec
100 Mb/s	120 μ sec
1 Gb/s	12 μ sec
10 Gb/s	1.2 μ sec

Table 1. Minimum Inter-arrival Times for 1500 byte Packets

should be lower. Even an extremely efficient operating system cannot hope to manage interrupts arriving at 12 μ sec, and the situation becomes utterly hopeless for 10 Gb/s networks.

1.2 Performance Metrics

The primary performance characteristic we must address is lowering communication overhead on the host processor as all high-performance computing is linearly sensitive to increases in CPU overhead.[5] Traditionally, however, bandwidth has been the measure of success in networking and we must certainly maintain competitive bandwidth since decreasing communication overhead by simply decreasing the amount of communication does not reflect a systemic improvement. A true measure of this performance is *effective utilization* which is calculated by dividing bandwidth by CPU utilization. We hope to achieve a high effective utilization which means that we are gaining the most bandwidth from the least amount of CPU utilization. As we lower communication overhead and maintain high bandwidth, our effective utilization will increase.

Any method for relieving interrupt pressure must also allow for *application-level flexibility of protocols, messaging styles or performance needs*. Some applications need consistent latency (low jitter) for synchronization. On the other hand, some applications will be less sensitive to latencies than to bulk data flow. Also, different high-performance applications may need different high-level protocols. MPI is typically implemented on TCP whereas many multi-media applications use UDP. Applications and application programmers must have the flexibility to emphasize their different needs.

Additionally, we must be aware of the performance advantages associated with commodity protocols like IP. We expect that by using commodity protocols we will *increase our interoperability*. Interoperability is not a binary measurement. The high-performance computing community may choose a protocol that is fully compliant with respect to IP and can take advantage of IP routing, but which is only partially interoperable with respect to TCP. For example, such a protocol might take advantage of TCP's reliability

without implementing the congestion control necessary for protocol completeness. Researchers may choose a trade-off with respect to compliance, completeness or global interoperability.

Practically, the most important performance metric in industry is resource usage. Here we want to decrease the amount of time needed to perform large calculations, but we also want to *minimize cost*. Commodity protocols decrease resource consumption in terms of power, space, hardware, software, training, programming and maintenance. As we make tradeoffs for better performance in other areas, we may lose performance with respect to cost.

In the next section we use these metrics to compare three approaches to reducing interrupt pressure: interrupt coalescing, jumbo frames, and end-point fragmentation. In the final section we discuss other approaches to offloading additional communication processing onto the NIC.

2 Reducing Interrupt Pressures

2.1 Interrupt Coalescing

We have several options for reducing interrupt pressures due to communication. Most conservatively, we can coalesce interrupts. The use of algorithms to reduce (or coalesce) the number of interrupts has been widespread. In this approach, the receiving NIC only interrupts the host after a specified amount of time or number of arriving packets. As we see in Figure 1, while effective utilization increases slightly with interrupt coalescing, the amount is not significant. In fact, improper adjustment of the interrupt coalescing algorithm can have detrimental effects. Another significant disadvantage is that small packets of a latency-bound application are not allowed to move directly through the NIC to the IP stack on the host. Instead, they are subject to being coalesced and latency-bound applications can see their performance suffer so the application-level flexibility is severely compromised.

Interrupt coalescing is scalable and interoperable. It solves the problem of interrupt pressure cleanly and elegantly. However, reducing the number of interrupts by holding messages for a short time does not significantly increase the effective utilization of the host processor. While this technique does maintain bandwidth and interoperability and is inexpensive, its greatest disadvantage is the high variance in latency which leads to a significant loss in flexibility of the application. These results are summarized in Table 2.

2.2 Jumbo Frames

Standard Ethernet Frames are 1518 bytes, including the MAC header and the CRC trailer. Jumbo frames, supported by many Gigabit NIC and switch manufacturers, increase

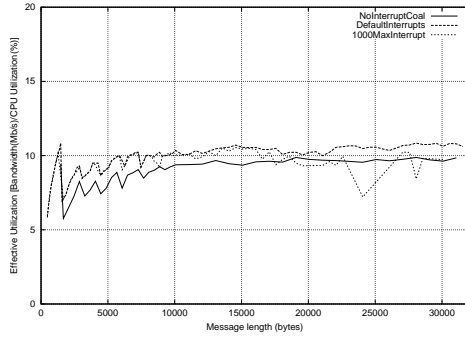


Figure 1. Effective utilization using no interrupt coalescing, default interrupt coalescing and extreme interrupt coalescing

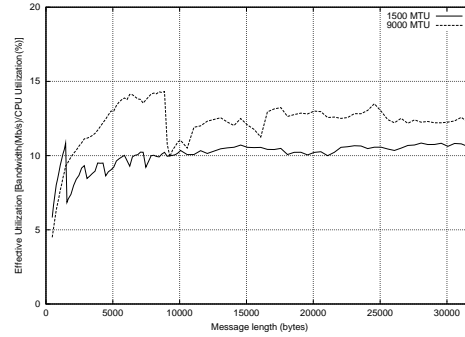


Figure 2. Effective utilization for 1500 byte Ethernet frames and 9000 byte Ethernet frames

Performance Metrics	Interrupt Coalescing
increased effective utilization	fair
application-level flexibility	<i>poor</i>
interoperability	good
cost	good

Table 2. Performance parameters for Interrupt Coalescing

Performance Metrics	Jumbo Frames
increased effective utilization	fair
application-level flexibility	good
interoperability	<i>poor</i>
cost	<i>poor</i>

Table 3. Performance parameters for Jumbo Frames

the size of an Ethernet frame to 9018 bytes. This reduces the number of packets necessary to transport a large message and thereby reduces the number of interrupts at the receiving and transporting hosts. Figure 2 shows that jumbo frames do relieve both interrupt inter-arrival and stack processing pressures and thereby increase the effective utilization of the host processor.

Using jumbo frames increases effective utilization in high-performance computing nodes. Unfortunately, this method requires special routers and special TCP/IP stacks. This need decreases interoperability and severely increases costs. Finally, frames of only 9000 bytes will not contribute significantly to solving comparable problems created by a 10 Gb/s network, where inter-arrival times will only increase from $1.2\mu s$ to $7.2\mu s$ and the bottleneck will return. Jumbo frames are inadequate primarily because they are not fully interoperable (and therefore costly) and they are not scalable. These results are summarized in Table 3.

2.3 Offload Communication Work onto the Network Interface Card

More and more work is being offloaded onto network interface cards [NICs], even in the commodity world of TCP/IP over Ethernet. One of the most successful exam-

ples of offloading communication processing is crypto hardware accelerators which have made virtual private networks (VPN) feasible. While a great deal of the work is offloaded directly into hardware (e.g. FPGA's), cards like the Myrinet NICs provide extensive computing resources and a rich development environment. Offloading work onto the NIC is a powerful method to decrease CPU utilization on compute nodes, but there is always the pressures of available resources on the NIC versus price per NIC.

Certainly, MAC assist, cyclic redundancy checks, and checksumming are the most common parts of the the TCP/IP protocol stack to be moved to the NIC, but many other elements of TCP/IP are being migrated off the host processor. For example, interrupt coalescing has been successfully deployed in hardware and recently researchers and companies have been offloading the entire TCP/IP protocol stack onto hardware. We see the trade-offs from offloading some of the IP stack onto the NIC versus offloading all of it onto the NIC.

2.3.1 Offloading IP Fragmentation and Reassembly

Fragmentation has been virtually nonexistent since Kent and Mogul [3] identified serious problems with intermediate fragmentation (where routers fragment packets too large

for the outgoing MTU of the next link) and proposed path MTU discovery in order to avoid any fragmentation by intermediate routers between two endpoints. Endpoint fragmentation is the only kind of fragmentation allowed in IP version 6.

Endpoint fragmentation does not share most of the disadvantages of intermediate fragmentation. Endpoint fragmentation is already commonly used by applications such as NFS that benefit from large datagrams being sent and received at higher levels of the protocol stack (in the case of NFS because disk blocks are the smallest unit of currency for disks). Moreover, fragmentation and reassembly are tasks that are extremely well-suited for implementation on a reasonably powerful NIC such as the Acenic. They are easy to separate from the rest of the IP stack. This is accomplished by transparently accepting packets larger than the real MTU of the link and fragmenting on the card. Almost no modification of the driver is necessary.

Figure 3 shows that offloading fragmentation and reassembly onto the NIC significantly increases the effective utilization of the host. Effective utilization increased because this method maintained bandwidths within about 10% of bandwidths using a 1500 byte MTU but reduced the amount of time the host spent processing communication by about 50%. In fact, offloading fragmentation and reassembly was more successful in relieving the interrupt pressure bottleneck than the use of interrupt coalescing or jumbo frames.

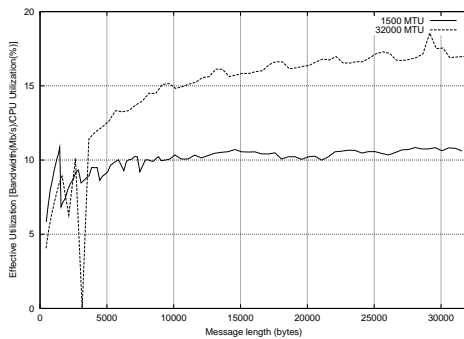


Figure 3. Effective utilization for unmodified firmware at 1500 byte MTU and offloaded fragmentation with driver MTU of 32000 bytes

We were able to implement fragmentation and reassembly on the NIC to demonstrate that it is possible to drastically impact the performance of a commodity protocol without significantly compromising its advantages[2]. Offloading a small part of the protocol led to a successful relieving of interrupt pressure while maintaining bandwidth, flexibility, interoperability and cost. Table 4 summarizes

these results.

Performance Metrics	Offloaded Fragmentation
increased effective utilization	good
application-level flexibility	good
interoperability	good
cost	good

Table 4. Performance parameters for Offloaded Fragmentation and Reassembly

3 Discussion

3.1 Can We Do Better?

Since offloading a small part of the TCP/IP protocol onto a NIC is so successful, we naturally wonder whether offloading *more* of the TCP/IP stack would prove even more beneficial. In fact, the trend has been toward offloading all of the TCP/IP stack onto the NIC. However, there are serious limitations in offloading the entire stack. Applications can no longer choose which protocol to use (TCP or UDP) as only TCP gives any performance advantages. Also, applications cannot affect policy decisions at the TCP level. These decisions range from congestion control strategies to the use of fast retransmits or selective acknowledgments. Additionally, TCP is currently very dynamic and there is no way to update the TCP stack to account for changes to the protocol which will drastically affect interoperability. Finally, power consumption has become a very real part of the cost of a system and the hardware and power necessary for such sophisticated NICs can become prohibitive.

Table 5 shows that while we definitely get a performance advantage[1], we lose the advantages of commodity protocols.

Performance Metrics	Offloaded TCP/IP
increased effective utilization	good
application-level flexibility	poor
interoperability	fair
cost	poor

Table 5. Performance parameters for Offloaded TCP/IP

3.2 Summary

Using the interrupt coalescing method for relieving interrupt pressure required that small messages wait to be pro-

cessed. Because offloaded fragmentation and reassembly assumes an MTU of up to 32000 bytes, small messages are unaffected by the method. Offloading leads to a larger performance gain than interrupt coalescing and also maintains application-level flexibility by not interfering with small messages.

The jumbo frames method for relieving interrupt pressure required special IP routers that allowed 9000 byte frames. Because offloaded fragmentation and reassembly assumes a wire MTU of 1500 bytes, no special routers are needed. This not only makes this method more interoperable, but it also makes the infrastructure required for this method much cheaper.

Finally, while offloading the TCP/IP stack provides the same performance advantages of offloading fragmentation and reassembly, the cost in loss of flexibility and resource usage is high. As TCP and IP continue to evolve, NICs with current implementations of TCP and IP "hard-coded" in silicon may quickly become obsolete.

Offloading IP fragmentation and reassembly increases the effective utilization of a host processor by reducing interrupts while maintaining high bandwidth. Furthermore, this method allows for application flexibility by not imposing latency restriction and remains fully interoperable with low resource usage. As we see in Table 6, offloading a portion of the IP protocol proves to be the best current solution.

Performance Metrics	Interrupt Coal	Jumbo Frames	Offload Frag	Offload TCP
utilization	fair	fair	good	good
flexibility	poor	good	good	poor
interoperability	good	poor	good	poor
low cost	good	poor	good	poor

Table 6. Performance Metrics for All Solutions

3.3 What's Next?

We have found that offloading some implementation aspects of the TCP/IP stack will lower communication overhead and thereby increase performance without sacrificing the advantages that commodity protocols offer. The next question is whether we have chosen the most effective part of the stack to offload. There are several other choices that do not affect policy choices and might give increased performance. If there is a way to cleanly split the TCP and IP stacks, can we get better performance by offloading all of the IP stack and still not lose flexibility? There is no clean modularity in TCP/IP stacks today so there is no obvious way to split the IP and TCP stacks. Also, there is no method

for maintaining the zero-copy efficiency within the stacks if we split them in this manner.

We can also chose to continue the trend toward offloading all of the TCP/IP stacks and perhaps move back into the realm of communication co-processors. We can potentially gain the increased bandwidth and lower communication overhead on the host processor that we see with offloaded TCP/IP but commodity processors are cheaper and more flexible than intelligent NICs. In order for this to be a feasible solution, we need more lightweight operating systems and more flexible schedulers and memory systems.

Finally, the most promising aspect of the TCP/IP stack to offload may be the congestion control mechanism in TCP. This allows for all data flow management to be offloaded so the host can concentrate on the data. This also could lead to a separation of headers and data at the NIC which would allow for operation-bypass mechanisms to be implemented[4]. Here we are reaching toward a trade-off of interoperability for the chance to get around the fundamental limitation of commodity protocols. TCP and IP assume memory management. We have removed the interrupt pressure bottleneck, but the memory-copy bottleneck still remains. The protocol itself assumes that we must copy once from system space into application space and this memory copy is our final bottleneck and in order to get around it we must sacrifice interoperability.

References

- [1] A. Gallatin. Duke trapeze/myrinet drivers. Web: <http://www.cs.duke.edu/ari/trapeze/ip/>, Oct. 2000.
- [2] P. Gilfeather and T. Underwood. Fragmentation and high performance ip. In *Proc. of the 15th International Parallel and Distributed Processing Symposium*, April 2001.
- [3] C. A. Kent and J. C. Mogul. Fragmentation considered harmful. *WRL Technical Report 87/3*, Dec. 1987.
- [4] C. Kurmann, M. Muller, F. Rauch, and T. Stricker. Speculative defragmentation — a technique to improve the communication software efficiency for gigabit ethernet. In *Proc. 9th IEEE Symp. High Performance Distr. Comp.*, pages 131–138, August 2000.
- [5] R. P. Martin, A. M. Vahdat, D. E. Culler, and T. E. Anderson. Effects of Communication Latency, Overhead, and Bandwidth in a Cluster Architecture. In *Proceedings of the 24th International Symposium on Computer Architecture*, pages 85–97, Denver, Colorado, June 1997.