Optimizing Linux Cluster Performance by Exploring the Correlation between Application Characteristics and Gigabit Ethernet Device Parameters

Onur Celebioglu, Tau Leng, Victor Mashayekhi

{onur_celebioglu, tau_leng, victor_mashayekhi}@dell.com
March 2004

Cluster interconnect performance is typically characterized by latency and throughput. However, not only latency and throughput but also the CPU utilization of an interconnect are important attributes that affect overall system performance. In our studies, we have run cluster benchmarks with two device drivers with different throughput and latency characteristics. We have observed that point-to-point performance tests such as throughput and latency cannot be translated directly into application performance. We also tried to further tune the performance of the system by changing the interrupt coalescing parameters one of the drivers. Finally we used this data to understand the correlation between an application's characteristics and interconnect performance attributes.

1. Introduction

One of the key components that make up High Performance Computing Cluster (HPCC) is the cluster interconnect. Cluster interconnect is the medium where Inter-Process Communication (IPC) messages or data is exchanged within the cluster. There are several important factors in choosing the interconnect, including latency, performance, price per port and communication characteristics of the application.

The performance of a cluster interconnect is typically defined by its latency and bandwidth. The host CPU utilization, although tougher to measure, is also a quite important performance attribute that affects the overall system performance. The impact of each of these parameters, i.e., latency, bandwidth, CPU utilization, on the overall system performance depends on the applications' computation/communication mix.

2. Motivation

There are many cluster interconnect technologies, such as Myrinet, Quadrics, SCI and InfiniBand, that provide low latency and high bandwidth communications. Most of these technologies use techniques such as protocol offloading and Remote Direct Memory Access (RDMA) transfers to minimize latency and host CPU utilization in message passing. Although most Gigabit Ethernet adapters do not posses these capabilities today, Gigabit Ethernet is still one of the most widely used interconnects in

high performance cluster computing. Some of the factors that make Ethernet an attractive technology for cluster computing are its price, ease of use and the fact that it is a standards-based, commodity component.

The OS TCP/IP protocol processing latency, host side hardware latency and switching latency are the main factors that constitute the overall latency of an Ethernet-based cluster interconnect. The interconnect bandwidth is typically determined by the sustainable bandwidth through the PCI bus, over-subscription ratio at the switch and number of frames that can be handled by the switch.

CPU utilization of an interconnect is also a very important attribute that impacts overall system performance besides latency and throughput. Typically, TCP/IP is the protocol used over Ethernet networks for Message Passing Interface (MPI) communications. In fact, there is a high CPU overhead associated with processing the TCP/IP protocol on the host CPU. Furthermore, interrupts must be generated to move data in and out of network adapter due to lack of RDMA capabilities in today's Ethernet adapters. Technologies such as zero-copy send offload [7] and interrupt coalescing [1] help alleviate some of this problem. In order to be used, these technologies must be supported by the hardware and the OS. The device driver is the enabling factor that exploits these technologies and contributes to efficient passing of messages over Ethernet networks. The efficiency of the device driver itself may be an important factor in the overall system performance of a HPCC for a specific combination of hardware, middleware and application.

Gigabit Ethernet without interrupt coalescing may produce a significant number of interrupts. Interrupt coalescing is a mechanism used to reduce the CPU overhead by grouping multiple packets in a single interrupt. Similarly, at the other end, the CPU gets notified about the departure of a group of packets through a single interrupt. This mechanism was already being used in networks before Gigabit Ethernet [5].

In this study, we focused on the Ethernet device driver and did a performance evaluation of different device drivers. We used Dell PowerEdge™ 1750 servers in our studies which have embedded dual Gigabit Ethernet adapters based on the Broadcom 5704 chip [6]. We used the tg3 driver present in the RedHat Linux 9 and the bcm5700 drivers from Broadcom to do our experiments. Our initial findings suggested that different device drivers may exhibit vastly varying performance characteristics in terms of latency, throughput and CPU utilization. Then we tried to further tune the performance of the system by changing the interrupt coalescing parameters of the bcm5700 driver. Finally, we focused on communication & computation mix of the benchmarks in order to understand the correlation between application characteristics and performance metrics of the device driver.

3. Experimental Environment

We conducted our experiments on a cluster of Dell™ PowerEdge™ 1750 servers. The cluster's configuration consisted of 33 identically configured servers, one used as a master node and 32 used as compute nodes. Each server was equipped with dual 3.06 GHz Intel® Xeon™ processors and 4GB of RAM. We used the embedded dual NICs on the PowerEdge 1750 for our testing. Each server was running Red Hat®

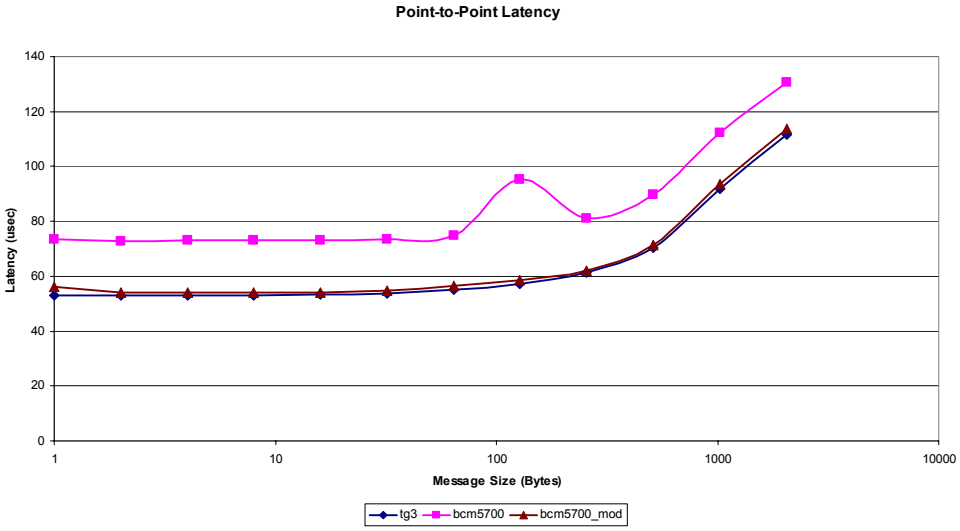

Figure 1. Point-to-point half round-trip time with varying message size

Linux 9 with 2.4.20-20.9smp kernel. We used an Extreme BlackDiamond 6808 as the Gigabit Ethernet switch. The benchmark programs were compiled with Intel C or FORTRAN compilers.

First, we used the Pallas MPI benchmarks' ping-pong test [4] to assess the point-to-point performance characteristics using bcm5700 v6.2.11 and tg3 v1.5 drivers. This benchmark sends multiple messages back and forth between two end-nodes. It then reports the overall throughput and half-round trip time for varying message sizes. Then we used NAS Parallel Benchmarks (NPB) and High Performance Linpack (HPL) to understand the impact of using these device drivers on the overall system performance. Finally, we examined the correlation between application's characteristics and interconnect's performance attributes.

4. Results and Analysis

4.1. Point-to-Point Performance Metrics

Figure 1 and Figure 2 show the point-to-point link performance characteristics. In both charts, x-axis denotes the message size in Bytes. In Figure 1, y-axis is the latency in microseconds and in Figure 2, y-axis is the throughput in Mbytes/sec. The tests were performed using tg3 driver, bcm5700 driver and bcm5700 driver with modified interrupt coalescing parameters.

As can be seen from Figure 1, there was ~20 usec latency difference between bcm5700 and tg3 driver. The bcm5700 drivers also did not provide as much bandwidth as tg3 drivers in the PMB PingPong tests as can be seen in Figure 2. In the light

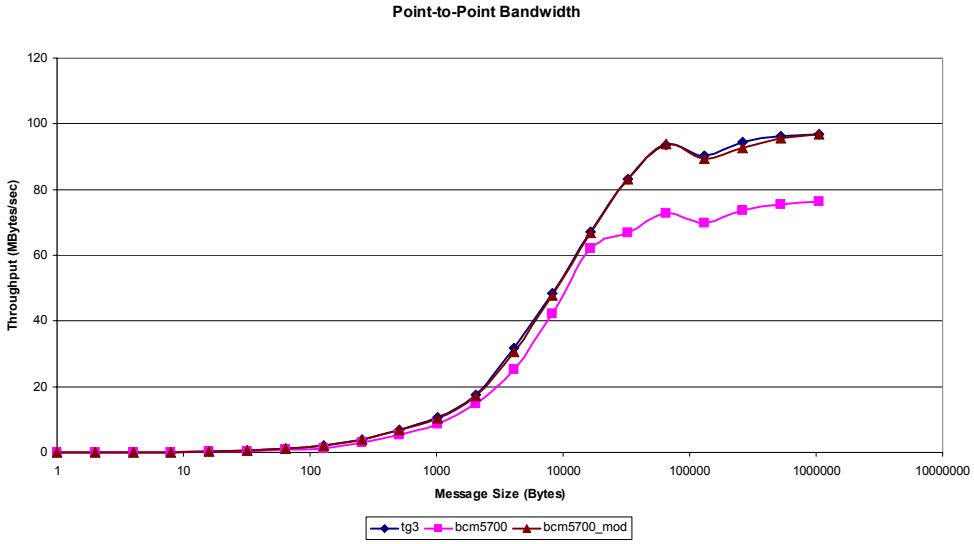

Figure 2. Point-to-point bandwidth with varying message size.

of these results one might expect an HPCC application to perform better with the tg3 driver; however, as demonstrated in the following sections we discovered this not to be true. The reason is that not only latency and throughput but also the CPU utilization of an interconnect are important attributes that affect the overall system performance. We investigated the CPU utilization for communication using the netperf program and observed that using bcm5700 driver results in significantly lower CPU utilization compared to using the tg3 driver as can be seen in Table 1.

Modifying the bcm5700 driver interrupt coalescing parameters brought the bcm5700 latency and throughput to a close level with the tg3 driver. The performance results with this modification are plotted in Figures 1 and 2 marked as bcm5700_mod. This modification resulted in performance improvement with some applications as described in the following section.

Although point-to-point performance tests give good indications of interconnect performance characteristics, they cannot be translated directly into application performance. The reason is that each application uses a different communication pattern and has varying computational characteristics. Depending on metrics such as message size, message frequency, traffic pattern and computation intensity, different applications are affected differently by the basic performance characteristics of the interconnect. In order to understand the overall system impact of the Ethernet device driver, we used common HPC benchmarks, which are mainly derived from real-life HPC applications.

Table 1. CPU utilization with different drivers using netperf

TG3

Socket Size bytes	Message Size bytes	Send Throughput Mbits/s	Recv local CPU %T	Send remote CPU %T
262142	4096	940.79	**61.98**	**83.67**
262142	8192	940.84	**55.69**	**87.05**
262142	32768	940.9	**49.38**	**87.47**

BCM5700

Socket Size bytes	Message Size bytes	Send Throughput Mbits/s	Recv local CPU %T	Send remote CPU %T
262142	4096	940.64	**51.8**	**19.05**
262142	8192	940.94	**44.52**	**19.53**
262142	32768	940.72	**41.75**	**19.42**

3.2. NAS Parallel Benchmarks (NPB)

The first MPI program we used for the system level experiments was the NAS Parallel Benchmark (NPB) suite [3]. NPB is a commonly used benchmark suite in the High Performance Computing arena, developed by NASA Advanced Supercomputing Division. These benchmarks are derived from computational fluid dynamics (CFD) applications. Since each program represents a specific part of CFD applications, from the benchmark results the performance characteristics of the system from various aspects can be realized. For our study, we used the results of four of the eight programs, the CG, FT, LU and IS, to facilitate our explanations.

In our experiments, we ran the NPB on the test cluster over Ethernet by using tg3, bcm5700 and the bcm5700 with modified parameters. For all the runs, we used Mpich-1.2.5. We ran our tests on a cluster of 32 servers as described in section 3.

Integer Sort (IS) performs an integer sorting operation that is important in *particle method* codes. This benchmark tests both integer computation speed and communication performance. This problem is unique in that floating-point arithmetic is not involved. Significant data communication, however, is required.

Figure 3 displays the relative performance reported by each benchmark normalized with respect to the performance obtained with that benchmark using tg3 driver. NPB reports the performance in millions of operations per second. For each benchmark in Figure 3, the result with tg3 driver is set as base and relative performance using other drivers is plotted. As can be seen, the best performance with IS was obtained with the tg3 driver. The bcm5700 drivers yielded 66% of the performance obtained using tg3. Although bcm5700 driver had lower CPU utilization, the fact that IS is more communication intensive rather than computation intensive caused the tg3 drivers to give better results. With the modified interrupt coalescing parameters, the performance of bcm5700 drivers in the IS benchmark improved dramatically, by nearly 30%.

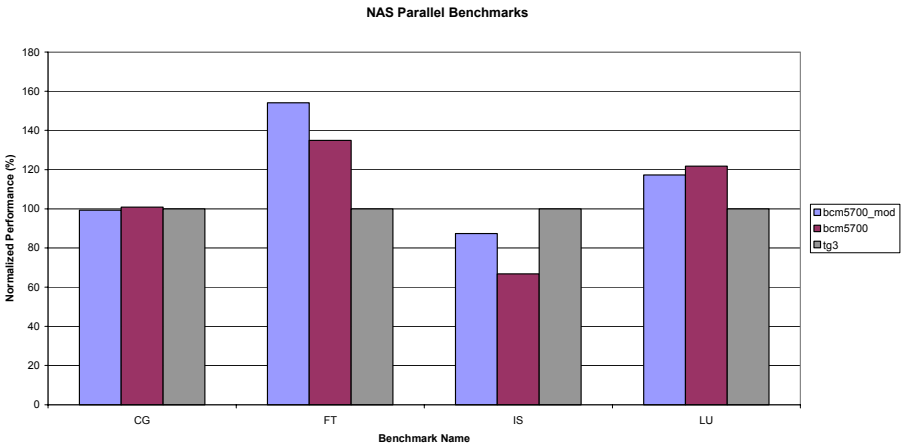

Figure 3. NAS Parallel Benchmark results for CG, FT, IS and LU tests

The behavior observed for FT was quite the opposite of IS. In FT, the ***3-D FFT PD benchmark***, a 3-D partial differential equation is solved using FFTs, the Fast Fourrier Transform. FT requires intensive float-point operations. It also is a good test of long-distance communication performance. With FT, using the bcm5700 driver resulted in 34% better performance than using the tg3 driver. This result may seem surprising at first glance if we only look at the point-to-point latency and throughput tests with these drivers. However, it is quite expected given the fact that FT is a highly computation intensive benchmark. Using the bcm5700 for communication frees up CPU resources that can be utilized to achieve better performance. In our tests, the modification of interrupt coalescing parameters for bcm5700 resulted in better performance for the FT benchmark.

The Lower-Upper (LU) diagonal benchmark employs a symmetric successive over-relaxation (SSOR) numerical scheme to solve a regular sparse, block 5x5 lower and upper triangular matrix system. Moreover, the communication is fine-grained; therefore, LU gives a good indication of the cluster interconnect latency. In this case however, for similar reasons to FT, bcm5700 provided better results although tg3 driver exhibited lower latency. Changing the interrupt coalescing parameters for bcm5700 did not have a significant effect on the performance.

The Conjugate Gradient (CG) benchmark uses a conjugate gradient method to compute an approximation to the smallest eigenvalue of a large, sparse symmetric positive definite matrix. CG is one of the most communication intensive benchmarks of NPB together with IS. Different than IS, CG also involves some floating-point computation. The fact that CG has both intense communications as well as heavy computation caused all test cases to give similar performance results.

A previous study by Venkataramiah, et.al [2] analyzed the NPB suite and measured the percentage time spent by CPU doing communications and computation. In their tests, IS had the highest communication ratio where 38% of CPU time was spent doing communications. [2] CG was the second most communication intensive with 23.1% and LU was far behind with only 4% CPU time in communications. [2] (They

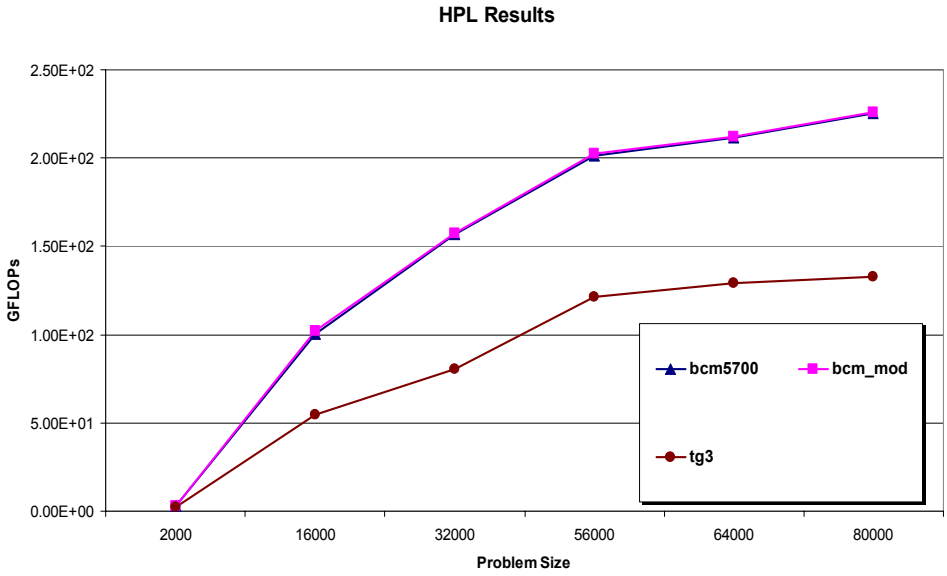

Figure 4. High Performance Linpack (HPL) performance

elected not to include FT in their testing). If we look at these results and focus on LU, CG and IS in Figure 3, we can see that LU, which has the smallest communication component, gets the most benefit by switching from tg3 to bcm5700 driver. With CG, which has a greater communications component than LU, there is no significant performance difference between tg3 and bcm5700. However, with IS, which has the highest ratio of communication/computation, the situation is reversed, that is, tg3 performs better than bcm5700. This shows that the communications and computation mix of a HPCC application is an important factor in choosing the right device driver.

5.3. High Performance Linpack (HPL)

The second tool we used for our experiments is Linpack, which is a commonly applied benchmark in the High Performance Computing arena. It uses a number of linear algebra routines to measure the time it takes to solve dense linear equations in double precision (64 bits) arithmetic using the Gaussian elimination method. The measurement obtained from Linpack is in number of floating-point operations per second (FLOPS).

Linpack is mostly a compute-intensive benchmark; it also creates some amount of communication traffic between the nodes in doing the computations. The performance depends mainly on message passing latency since most messages exchanged are of small to medium size.

Similar to the compute intensive benchmarks such as LU and FT in the previous section, using bcm5700 driver instead of tg3 resulted in significantly higher perform-

ance with Linpack. For the largest problem size we used, we obtained 133GFlops using the tg3 driver and 225GFlops using the bcm5700 driver. This corresponds to a ~69% performance difference between the runs with bcm5700 and tg3. Modifying the interrupt coalescing parameters did not create much performance difference for the bcm5700 drivers.

6. Conclusion

Interconnect is one of the key components in performance of HPCC applications. The typical areas of focus when studying the performance of cluster interconnects are bandwidth and latency. However, measurements taken in a point-to-point configuration may not be representative of the actual application performance. We have demonstrated that, it is not possible to correlate the application performance only to point-to-point message passing bandwidth and latency. In fact, as is the case with FT, LU and Linpack, although latency was lower and bandwidth was higher with tg3, the performance was significantly better with bcm5700 drivers. In this case, the difference between the two configurations was the CPU overhead for communications. This shows that CPU overhead in communications is a very important factor for the performance of some applications.

In this study, we have seen that the Ethernet device driver is a major component that affects system performance. Even when we modified the interrupt coalescing parameters of the bcm5700 driver to bring its latency and throughput to the same level as that of the tg3 driver, there still was a significant performance difference between test runs with tg3 vs. bcm5700. This indicates that even for a standardized interconnect such as Gigabit Ethernet, the choice of the most efficient device driver is critical.

The choice and configuration of the Ethernet device driver depends not only on applications' communication pattern but also the applications' computation characteristics. In our tests we observed that when using Gigabit Ethernet as cluster interconnect, applications that have low communication and high computation needs tend to perform better with drivers optimized for low CPU utilization. On the other hand, applications that are more communication intensive tend to perform better with bandwidth and latency optimized Ethernet device drivers even if the CPU utilization in communications is high.

Modifying how the device drivers behave through user configurable parameters also proved to be beneficial for most applications. The fact that bcm5700 drivers provided the facilities to do these modifications without any code change made it more convenient to fine tune the system performance for a specific application. Thus, another conclusion that we can draw from this study is that network device drivers should provide means to adjust the parameters that affect latency, throughput and CPU utilization.

References

[1] J. Chase, A. J. Gallatin, and K. G. Yocum. “End system optimizations for high-speed tcp”, *IEEE Communications Magazine*, 39(4):68–74, April 2001.
[2] S. Venkataramaiah, J. Subhlok, “Performance Prediction for Simple CPU and Network Sharing” *LASCI 2003.*
[3] NAS Parallel Benchmark suite, http://www.nas.nasa.gov/Software/NPB/.
[4] Pallas MPI Benchmarks, http://www.pallas.com/e/products/pmb/index.htm.
[5] Mogul, J.C., Ramakrishnan, K.K., Eliminating receive livelock in an interrupt driven kernel. ACM Transactions on Computer Systems 15 (1997) (pp. 217 – 252).
[6] BCM570X Family Product Sheet, http://www.broadcom.com/products/product.php?product_id=BCM570x+Family.
[7] EMP: Zero-copy OS by-pass NIC-driven Gigabit Ethernet Message Passing, Proceedings of the 2001 ACM/IEEE conference on Supercomputing, pp.57-57, November 10-16, 2001, Denver, Colorado.