The Impact of Hyper-Threading on Communication Performance in HPC Clusters

Tau Leng[a], Rizwan Ali, Celebioglu, Onur, Jenwei Hsieh, Victor Mashayekhi, Reza Rooholamini

[a]Dell Computer Corp.
RR5-3 One Dell Way, Round Rock, TX 78682

The effects of Intel Hyper-Threading® (HT) technology on a system's performance vary according to the characteristics of the application running on the system and the configuration of the system. HPC clusters introduce additional hardware variables, such as the interconnection that can affect the performance of MPI applications. In this paper, we study the effect of HT on MPI-based applications by varying two hardware components: Processor cache sizes and interconnect networks. What we have found out is that larger cache sizes leads to better performance when HT is enabled, in particular for the cache-friendly applications. For instance, enabling HT in a cluster made up of Xeon MP based machines improves performance, but in a cluster of Xeon DP based machines degrades performance. For interconnect communication, enabling HT in a cluster interconnected with a low latency, high-bandwidth interconnect like Quadrics, may not improve performance in the case that the CPU resources are highly utilized. On the other hand, in a cluster interconnected with slower speed, higher latency interconnects like Gigabit Ethernet, enabling HT improves performance via claiming the underutilized resources, which were caused by the inefficient communication.

Les effets sur les performances du système Hyper-Threading® (HT) d'Intel varient en fonction des caractéristiques des applications et de la configuration du système d'exploitation. Les grappes de CHP introduisent des variables matérielles additionnelles, comme l'interconnection entre les noeuds qui peut affecter la performance des application MPI. Dans cet article, nous étudions les effets du HT sur les applications MPI en faisant varier deux paramètres matériels: La dimension de la mémoire cache et le type d'interconnection. Nous avons observé que l'utilisation du HT améliore la performance lorsque la dimension de la ménoire cache est grande, en particulier pour les applications profitant déjà de la mémoire cache. Par exemple, l'utilisation du HT sur une grappe de Xeon MP améliore les performances, tandis que sur une grappe de Xeon DP l'utilisation du HT diminue les performances. En ce qui concerne l'interconnection, l'utilisation du HT sur une grappe utilisant un réseau à basse latence et grande bande passante comme Quadrics, ne devrait pas augmenter les performances dans le cas où le processeur est pleinement utilisé. En contre partie, sur une grappe utilisant un réseau à plus haute latence et plus faible bande passante comme Gigbit Ethernet, l'utilisation du HT améliore les performances en profitant du temps de processeur non utilisé, causé par la communication inefficace.

1. Introduction

Intel's Hyper-Threading (HT) technology makes a single physical processor appear as two logical processors. The physical processor resources are shared and the architectural state is duplicated for the two logical processors [1]. The premise is that this duplication allows a single physical processor to execute instructions from different threads in parallel rather than in serial, and therefore, could lead to better processor utilization and overall performance. However, sharing the system resources, such as caches or memory bus by the additional logical processors might create negative impact on the performance. Particularly, in a HPC cluster environment, additional hardware variables introduced, that create more parameters of determining whether HT will help or hinder the performance. For example, the communication capability between processes within a compute node or among the nodes in the cluster will influence the overall performance, when running the parallel applications.

1.1 Level of Parallelism

Two levels of parallelism have been addressed in the modern computer processor design to improve performance. Instruction-level-parallelism (ILP) refers to techniques of increasing the number of instructions executed each clock cycle. Although it is possible that the multiple execution units in a processor can execute multiple instructions at the same time, the dependencies that exist among the instructions makes it a challenge to finding enough instructions to execute simultaneously. Several mechanisms have been implemented to increase ILP. For example, "out-of-order execution" is a technique of evaluating a set of instructions and sending them for execution in parallel, regardless of their original order defined by the program, and yet preserving the dependencies among the instructions.

Thread-Level Parallelism (TLP), on the other hand, enables a processor or multiprocessor system to concurrently run multiple threads from an application or from multiple, independent programs. SMT, or Simultaneous Multi-Threading technology, upon which HT is based, permits a processor to exploit both ILP and TLP. Multiple threads can run on an SMT processor, and the processor will dynamically allocate resources between the threads, enabling a processor to adapt to the varying requirements of the workload. Intel's HT implements SMT in such a way that each logical processor maintains a separate architectural state, which consists of general-purpose, control, machine state, and advanced programmable interrupt controller (APIC) registers [1]. The chip real estate required for the architectural states is negligible compared to the total die size. Thus, threads or separate programs using separate architectural states must

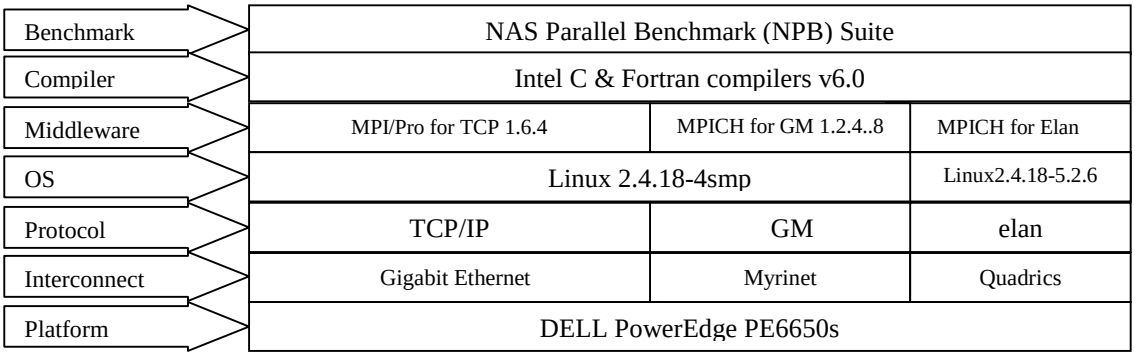

Figure 1. Architectural stack of the test environment. NPB benchmark is compiled with Intel C and FORTRAN compilers. The Linux is RedHat 7.3 distribution with kernel version 2.4.18-4smp and 2.4.18-elan.

share most of the physical processor resources, such as trace cache, L2-L3 unified caches, translation look aside buffer, execution units, branch history table, branch target buffer, control logic, and busses. This simultaneous sharing of resources between two threads creates a potential for performance degradation.

1.2 Multithreading and Message-passing Applications

In general, processors enabled with HT technology can improve the performance of applications with high degree of parallelism. Previous studies have shown that the HT technology improves multi-threaded applications' performance by a range of 10 to 30 percent depending on the characteristics of the applications [2]. These studies also suggest that the potential gain is only obtained if the application is multi-threaded by any means of parallelization techniques. A multithreaded program is capable of creating multiple processes, or threads, at a time without having to have multiple copies of the program running in the computer.

In our area of interest, high performance computing (HPC) clusters, applications are commonly implemented by using standard message-passing interface, such as MPI or PVM. Applications developed by a message-passing programming model usually employ a mechanism, "*mpirun"* for example, to spawn multiple processes and map them to processors in the systems. Parallelism is achieved through the message-passing interface among the processes to coordinate the parallel tasks. Unlike the multithreaded programs in which the values of application variables are shared by all the threads, a message-passing application runs as a collection of autonomous processes, each with its own local memory. For most of the case, the number of processes spawned is equal to the number of processors in the cluster, to obtain the optimal performance. Therefore, this type of applications could also benefit from HT technology in the sense that the number of processes spawned is doubled, which makes the parallel tasks potentially execute faster. Applying HT and doubling the processes that simultaneously run on the cluster will increase the utilization rate of the processors' execution resources. Consequently, the performance may be improved. On the other hand, overheads might be introduced in the following ways:

- The logical processes of the same physical processor may compete for accesses to the caches, and thus could generate more cache-miss situations
- More processes running on the same compute node may create additional memory contention, when the processes are accessing the memory bus or communicating through the shared memory at the same time
- More processes on each node increase the communication traffic (message passing) within and between nodes, which could oversubscribe the communication capacity of the shared memory, the I/O bus or the interconnect networking, and thus create performance bottlenecks

Whether the performance benefits of HT – better resource utilization – can nullify these overhead conditions, primarily depends on the application's characteristics [10].

In this paper, we focused on the communication performance of the cluster affected by HT, and discussed the adaptability of the technology to HPC clusters when using various interconnects. In the next section, we introduce the cluster interconnects used in our study and how they effect the overall cluster performance. Section 3 explains the configurations and the tools for our experiments including the benchmark programs. Section 4 shows the benchmark results along with the performance analysis. The causes of performance gain or degradation in applying HT in the cluster with different interconnects are discussed. Section 5 is the conclusion.

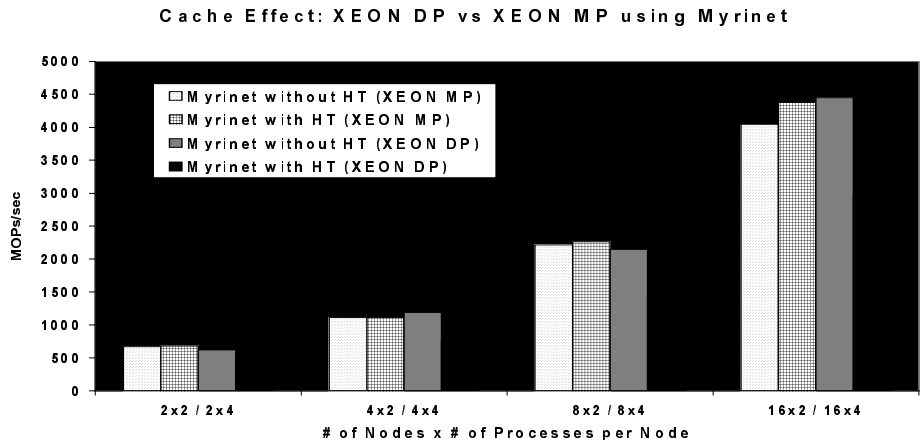

Figure 2. The FT (Class B) benchmark results of Hyper-Threading enabled and disabled on a Xeon MP and a Xeon DP cluster using Myrinet as the interconnect. The Large L3 Cache on Xeon MP helps FT's performance.

2. The Interconnects

The interconnect that networks the individual compute nodes converts a group of computers into a single system, an HPC cluster, which is capable of executing parallel jobs. The performance of the interconnect is an important factor which effects the cluster performance significantly when the applications running on the cluster are communication centric. When Hyper-Threading is applied, the cluster performance may be decreased because the communication links might not be able to accommodate more communications among the (doubled number of) processes. Conversely, if the communication links are capable of handling the increased communication traffics, the cluster performance might benefit from HT. In our study, we used three interconnects, Gigabit Ethernet, Myrinet, and Quadrics respectively to understand the performance impact on each cluster when HT is enabled versus disabled.

Ethernet is the most popular networking configuration today. However, due to its inefficiency, the conventional TCP/IP protocol is not necessarily suitable for a dedicated system area network, like a HPC cluster interconnect, to accomplish communication intensive jobs. Nevertheless, familiarity and the cost factor make Fast Ethernet and Gigabit Ethernet the more popular and cost-effective choices for many applications.

Myrinet and Quadrics are two of the leading cluster interconnects technologies for HPC clusters. Both provide low latency, high bandwidth, and end-to-end communication between two nodes in the cluster. Myrinet [3] is a connection-less interconnect which leverages packet switching technologies from experimental Massively Parallel Processors (MPP) networks. Myrinet uses an efficient communication mechanism called the Glenn's Message (GM) message-passing system. This system gives a user process direct access to the network interface, avoiding immediate copies of data and bypassing the operating system (OS) networking stack in a protected fashion. Quadrics [9] uses a similar technique to efficiently transfer messages between the communicating nodes. The Quadrics elan network interface provides efficient and protected access to a global virtual memory via remote DMA operations.

The following table summarizes the specifications of the three interconnects we used for our experiments.

Interconnect	**Gigabit Ethernet**	**Myrinet – 2000**	**Quadrics**
Network Interface Card	Broadcom 5701 64-bit 66Mhz PCI	Myrinet-2000 64-bit 66Mhz PCI	Elan III 64-bit 66Mhz PCI
Network Switch	Gigabit Ethernet Switch	Myrinet-2000 32 port switch	Quadrics QM S16 16-port switch
Link speed	1 Gbps	2 Gbps	2.72 Gbps
Topology	Non-blocking	Clos network	Fat Tree
Protocol	TCP/IP	GM	Elan

3. Experimental Environment

Our testing environment is a cluster that consists of 16 Dell PowerEdge 6650 servers interconnected with Gigabit, Myrinet and Quadrics. Each PowerEdge 6650 has two Intel XEON MP Processors running at 1.6 GHz with 512KB L2

cache, 1MB L3 cache, 4GB of DDR-RAM (double data rate RAM) memory operating on a 400 MHz Front Side Bus. The chipset of PowerEdge 6650 is the ServerWorks GC-HE, which accommodates up to sixteen registered DDR PC1600 (200Mhz) DIMMs with a 4-way interleaved memory architecture. The PowerEdge 6650 is also equipped with two integrated Broadcom Gigabit Ethernet adapters. The operating system installed for the cluster is RedHat 7.3 with kernel version 2.4.18-4smp. Note that for being able to use the Quadrics, the kernel is upgraded to a special kernel based on 2.4.18. The benchmark programs were compiled with Intel C or FORTRAN compilers. Figure 1 shows the architectural stack of our test environment.

The MPI programs we used for the experiments are the NAS Parallel Benchmark (NPB) suite. NPB is a commonly used benchmark in the High Performance Computing arena, developed by NASA Advanced Supercomputing Division. The benchmarks, which are derived from computational fluid dynamics (CFD) applications, consist of five kernels and three pseudo-applications designed to gauge parallel computing performance. Each of the programs solves a specific numerical problem [6]. The performance results are measured in Million Operations per Second (Mop/s). Since each program represents a specific part of CFD applications, from the benchmark results, one can realize the performance characteristics of the system from various aspects. For our study, we use the results of three of the eight programs, the EP, FT, and IS to facilitate our explanations.

In our experiments, we run the NPB on the test cluster by using different combinations of compute nodes and number of CPUs in each node. Three sets of testing for the three interconnects are performed separately. For each set, we run the NPB for both HT disabled and enabled configurations. Then, compare the results to show how HT affects the clusters' performances.

4. Benchmarking Results and Analysis

4.1 Cache Effect

To understand the cache effect of Hyper-Threading, FT is used, given that its performance is sensitive to the cache size. In FT, the ***3-D FFT PD benchmark***, a 3-D partial differential equation is solved using FFTs, the Fast Fourier Transform. This program performs the essence of many *spectral methods*. It is a good test of long-distance communication performance. FT requires intensive float-point operations and messages passing among processes. More importantly, FT is a cache-friendly benchmark; the large cache will facilitate better performance [7][10].

Figure 2 shows the results obtained by running FT on our test cluster with 16 nodes equipped with 32 XEON MP Processors comparing to that of a 16-node cluster with 32 XEON DP Processors. The results of the DP cluster were obtained from our prior study in [10], and both of the results are obtained on clusters with Myrinet interconnect. From the Figure 2, it can be seen that the XEON MP cluster shows improvement in performance with HT enabled, while the XEON DP cluster shows a fall in performance. The improvement for the MP cluster is primarily due to the large L3 cache of the MP Processors. Based on the analysis in [10], the drop in performance of the DP cluster was because of the considerable increase of L2 cache miss rate without L3 cache support in the DP Processor, when running FT with HT enabled. In the case for XEON MP Cluster, the L3 cache was large enough to house the memory accesses for both the logical processors sharing the caches. As a result, the increase of L2 cache misses was compensated by L3 cache hits, and the performance improved via better CPU resources utilization.

Another observation from Figure 2 is the FT performance of MP and DP Clusters is considerably close, although the difference in clock speed between XEON MP (1.6GHz) and DP (2.4GHz) is large. That again, indicates the cache is a major factor, which affects the FT performance significantly, and HT can improve the performance, if the cache size is sufficient to accommodate both logical processors.

In Figure 3, we show the performance results obtained by running FT on XEON MP Cluster for different numbers of compute nodes and different interconnects, Gigabit Ethernet, Myrinet, and Quadrics. Observed from the results, the interconnect capacity is important for FT to scale. The performance differences between the faster interconnect and the slower interconnect are larger when the node count of the cluster is increased. Slower interconnect, like Gigabit Ethernet, apparently creates the bottleneck for larger cluster with more FT processes running. Nonetheless, enabling the HT provides an overall performance improvements on all three interconnects' clusters. And, the improvements become more noticeable while running the FT on larger node count configurations.

4.2 Interconnect effect

Integer Sort (IS) performs an integer sorting operation that is important in *particle method* codes. This type of application is similar to particle-in-cell applications of physics, wherein particles are assigned to cells and may drift out. The sorting operation is used to reassign particles to the appropriate cells. 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

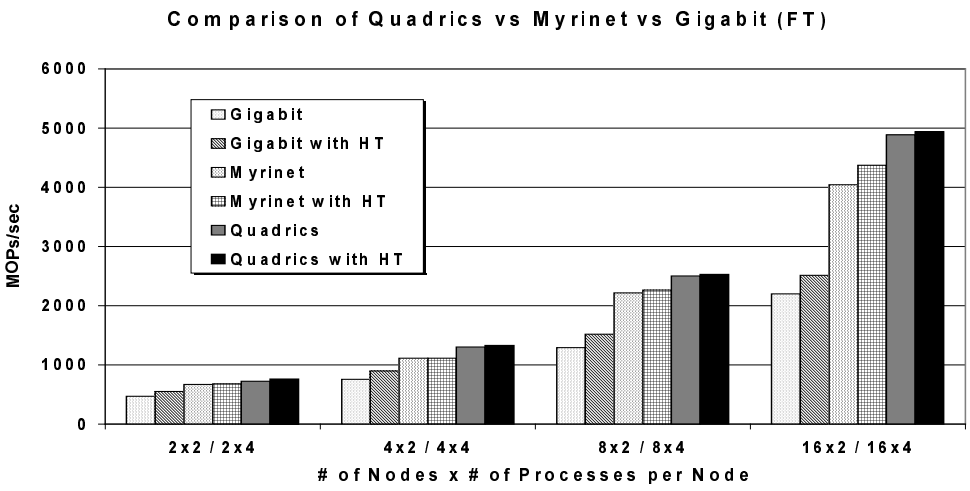

Figure 3. The FT (Class B) benchmark results for Hyper-Threading enabled and disabled for the three interconnects. Gigabit shows a larger percentage improvement from HT disabled to HT enabled.

required. Hence, IS is particularly useful for testing the communication capability of the cluster. For our study, the results of IS will show not just the communication performance for each interconnect, but also whether doubling the processes will complete the Sorting faster or the increased communication traffics will slow down the performance, when HT is on.

The communication traffic here indicates message-passing between processes, which could be inside a cluster node or among different nodes. Memory and I/O contentions both could potentially decrease the cluster performance. The Dell PowerEdge 6650 platform has a 4-way interleaved memory architecture. The high memory bandwidth helps minimize the effect of memory contention due to increased number of processes.

The IS performance results in Figure 4 show that the performances of the three interconnects are as expected. Quadrics is the best followed by Myrinet then Gigabit Ethernet. Furthermore, unlike Quadrics and Myrinet clusters, the Gigabit Ethernet cluster has the worst scalability, due to the high latency of the interconnect. Increasing the node count of the cluster running IS would not improve the performance too much for the Gigabit Ethernet configuration.

Figure 4 also shows that HT helped the IS performance slightly for Gigabit Ethernet and Myrinet configurations. However, the Quadrics configuration, despite giving the highest overall performance, did not benefit from HT as much. For applications to benefit from HT there has to be underutilized shared resources in the CPU. In the case with a very low latency interconnect such as Quadrics, the CPU is occupied doing computation. The amount of idle time a process is pending for message-passing activities is minimized. Thus, there is no performance improvement realized by enabling HT for IS in the Quadrics configuration. Furthermore, we observe a slight performance drop for large node counts, due to contention for memory and interconnect. A similar contention occurs for slower interconnects that have high latency such as Gigabit Ethernet. On the other hand, as we can deduct from the relatively lower overall performance and scalability obtained in these configurations, the system resources are not utilized to their full potential. Enabling HT helps these configurations resulting in an overall performance gain. Although contention for I/O is higher for slower interconnects, utilizing the idle CPU cycles caused by high message passing latency negates the effect of I/O contention and results in a net performance increase with HT enabled.

On the contrary to IS, the ***Embarrassingly Parallel (EP) Benchmark*** is an entirely computational-bound program. In EP, two-dimensional statistics are accumulated from a large number of Gaussian pseudo-random numbers, which are generated according to a particular scheme that is well suited for parallel computation. This problem is typical of many *Monte Carlo* applications. Since it requires almost no communication, in some sense this benchmark provides an estimate of the upper achievable limits for arithmetic operations' performance on a particular system. EP uses a combination of floating point and integer operations. Thus, HT technology enables independent EP processes to use different CPU resources in an interleaving manner. With HT enabled, EP could more effectively utilize the CPUs' arithmetic resources without being concerned with the communication overhead, which makes the performance improve significantly. This condition is

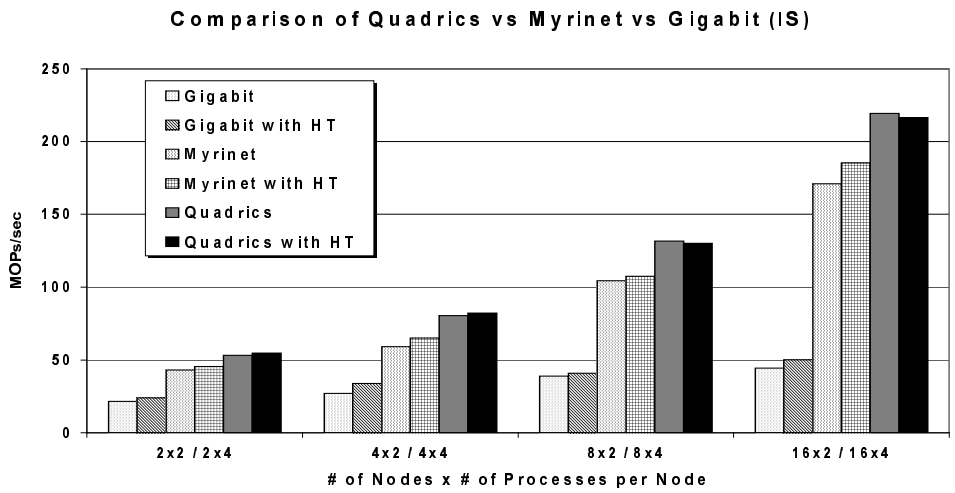

Figure 4. The IS (Class B) benchmark results for Hyper-Threading enabled and disabled for the three interconnects. Low latency, high bandwidth interconnects (Myrinet and Quadrics) outperform Gigabit by a wide margin. The Gigabit Ethernet shows unnoticeable scalability.

true regardless of the number of nodes or CPUs of the cluster. Figure 5 shows the EP performance improved linearly from two nodes to 16 nodes, as well as the constant performance gains (~40%) on HT enabled runs. Since EP has almost no communication the performance obtained on all the three clusters were near identical.

5. Conclusion

Previous studies showed that Hyper-Threading can improve the performance of some MPI applications on a HPC cluster. MPI based applications are typically run by spawning one process for each processor. This suggests spawning twice as many processes, in order to make use of HT. However, doubling the processes may cause performance degradation due to resource contentions. In this paper, we were able to focus on the cluster interconnect aspect of the cluster design by maximizing the amount of CPU cache size through the use of XEON MP Processors. This provided elimination of the cache contention effect shown in previous study for FT benchmark. We then experimented on three popular cluster interconnects; Gigabit Ethernet, Myrinet and Quadrics. Our findings from this study can be summarized in the following.

- In communication intensive applications, e.g. IS, Quadrics gives the highest overall cluster performance, however this cluster benefits the least from HT. On the other hand, the slower interconnect, Gigabit Ethernet, benefits more from HT in communications intensive benchmarks.
- Computational intensive applications with fine-tuned arithmetic operations have less chance to be improved in performance from HT, because the CPU resources are already highly utilized.
- However, if these applications are also communication intensive, such as the case with FT, the overhead and latency associated with the communications may cause the CPU resources to be underutilized. For interconnects such as Myrinet and Quadrics, this overhead is minimal since these interconnects use efficient message transfer mechanisms such as RDMA or Remote Direct Memory Access. The CPU involvement in message transfer for these interconnects is already minimal.
- Furthermore, an efficient computation intensive workload will highly utilize the CPU resources, thus HT provides little or no help. Because these interconnect also provide high-bandwidth, resource contention is also not an issue for applications like FT that exchange small size messages. Gigabit Ethernet using TCP/IP protocol has relatively high communication overhead. Even with highly-tuned applications, some cycles are spent in processing the inter-process communications (IPC) messages rather than doing computation. This causes the scalability and the overall performance of the cluster to drop. HT tends to help more in this case by claiming the underutilized resources of the CPU due to inefficient communications, and using them for computation.
- For applications that require intensive communications such as IS, enabling HT and

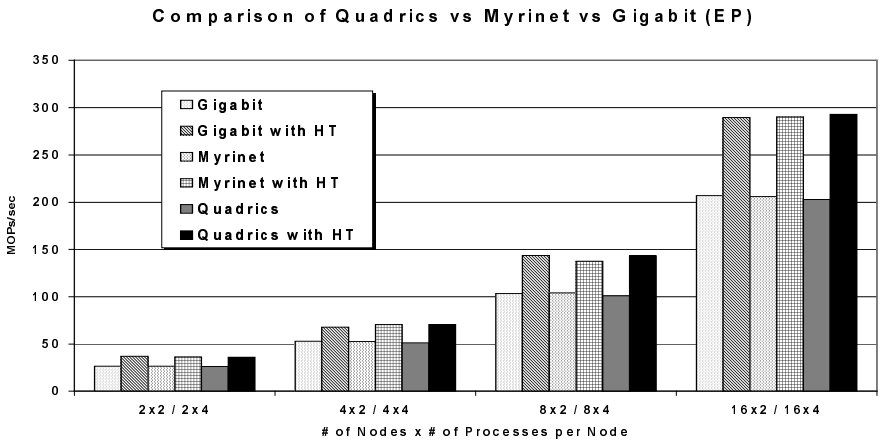

Figure 5. The EP (Class B) benchmark results for Hyper-Threading enabled and disabled for the three interconnects. The results of HT enabled configuration shows up to 40% improvement comparing to the HT disabled configuration.

doubling the number of processes in the cluster node creates I/O contention for shared memory and the interconnect subsystems. The performance degradation caused by the contention can be offset by performance increase realized by claiming underutilized CPU cycles such as the case with Gigabit Ethernet. For a very low latency interconnect such as Quadrics, the computation already occupies the CPU cycles, thus the offsetting effect of HT is minimal. This leads to a performance drop due to contention. This kind of behavior can be observed especially in higher node count configurations.

In light of these results, we can conclude that the decision whether to use or not to use HT technology in a cluster not only depends on the characteristics of the application, but also the capacities of the CPU caches and interconnect communication. Even with the same type of application, clusters with different interconnects and cache configurations may be affected differently by HT.

References

[1] D. T. Marr el al, "Hyper-Threading Technology Architecture and Micro architecture", Intel Technology Journal, Vol. 6, Issue 01, February, 2002.

[2] W. Magro, P. Petersen, S. Shah, "Hyper-Threading Technology: Impact on Compute-Intensive Workloads", Intel Technology Journal, Vol. 6, Issue 01, February 2002.

[3] J. Hsieh, T. Leng, V. Mashayekhi, and R. Rooholamini. " Architectural and Performance Evaluation of Giganet and Myrinet Interconnects on Cluster of Small-Scale SMP Servers" , in the *Proceedings of Super-Computing '00*, Dallas, TX, November 2000.

[4] NAS Parallel Benchmark suite, http://www.nas.nasa.gov/Software/NPB/

[5] J. Hsieh, T. Leng, V. Mashayekhi and R. Rooholamini. " Impact of Level 2 Cache and Memory Subsystem on the Scalability of Clusters of Small- Scale SMP Servers" , in the *Proceedings of International Conference on Cluster Computing*, Cluster'00, November 2000.

[6] M. Kravetz, H. Franke, S. Nagar, R. Ravindran, " Enhancing Linux Scheduler Scalability" , in the 2001 Ottawa Linux Symposium, July 2001.

[7] The Linux Kernel Achieve, ChangeLog-2.4.19, http://www.kernel.org/.

[8] F.C. Wong, R.P. Martin, R.H. Arpaci-Dusseau, and D.E. Culler. " Architectural Requirements and Scalability of the NAS Parallel Benchmarks" , In the Proceedings of SuperComputing '99, Portland, Oregon, November 1999.

[9] Fabrizio Petrini, Wu-chun Feng, Adolfy Hoisie, Salvador Coll, and Eitan Frachtenberg , "The Quadrics Network (QsNet): High-Performance Clustering Technology", Hot Interconnects *9*, Stanford University, Palo Alto, CA, August 2001

[10] T. Leng, R. Ali, V. Mashayekhi, J. Hsieh and R. Rooholamini. " An Empirical Study of Hyper-Threading in High Performance Computing Clusters" , in the *The Third LCI International Conference on Linux Clusters: The HPC Revolution 2002*, October 2002.