Performance Evaluation of Intel's Quad Core Processors for Embedded Applications

JAREER H. ABDEL-QADER
Department of Computer Science and Engineering
University of Texas at Arlington
416 Yates St. Nedderman Hall, Rm 300,
Arlington, Texas
USA
Jareer.abdelqader@mavs.uta.edu

ROGER S. WALKER
Department of Computer Science and Engineering
University of Texas at Arlington
416 Yates St. Nedderman Hall, Rm 300,
Arlington, Texas
USA
Roger.walker@uta.edu
http://ranger.uta.edu/~walker/

Abstract: - Recently, multiprocessing is implemented using either chip multiprocessing (CMP) or Simultaneous multithreading (SMT). Multi-core processors, represent CMP processors, are widely used in desktop and server applications and are now appearing in real-time embedded applications. We are investigating optimal configurations of some of the available multi-core processors suitable for developing real-time software for a multithreaded application used for pavement performance measurements. For the application discussed in this paper we are considering the use of either the Intel core 2 quad or the core i7 (a quad core processor with hyper threading (HT) technology.) Processor performance is a major requirement in this set of real-time, computational intensive embedded applications. The performance of both processors is measured and evaluated using single and multithreaded workloads supplied by different benchmark suites. As for the core i7 processor we also provide an evaluation for the HT technology implemented in each core of this processor.

Key-words: Quad core processors; Multi-threading; Performance Evaluation; Benchmark; Memory bandwidth; Memory Latency.

1 Introduction

Performance evaluation of computer systems in general and processors in particular becomes more difficult with the advancement of multi-core processors. Developers often rely on performance data and benchmark results to evaluate processors in order to pick the most appropriate one that can best meet their design criteria. Researchers at the Transportation Instrumentation Lab (TIL) at the University of Texas at Arlington (UTA) have been developing a measurement system for real-time pavement performance measurements. The system consists of a series of independent low-power general purpose instrument modules used for measuring various pavement performance characteristics. Each measurement module acquires processes, synchronizes, and communicates data between itself and other modules from one or more sensors in real-time. The sensors used in the modules include such devices as gyros, accelerometers, lasers, infrared detectors, etc. One or more high performance multi-core embedded control processor(s) are needed to perform the necessary multithreaded real-time computations.

For this reason researchers are testing several of the recently available processors to determine which can best fit for the application. Because our application requires a large degree of multi-processing in order to meet the real-time needs, we were interested in the performance resulting from the use of known multi-threaded benchmarks. As a startup, we are using some of the widely accepted benchmarks to provide a general idea about the processors' performance and to be able to compare our results with the results published by other researchers such as [1, 2, 11, 12, 13, 14, and 15]. STREAM, STREAM2, and NPB are among well known OpenMP benchmarks that are mostly used for testing and evaluating processors' performance. For example in [1] STREAM and STREAM2 benchmarks were used for

comparing the performance of three different dual core processors namely, the Intel core 2 Duo, Pentium D, and the AMD Athlon 64 X2 processors and from the results obtained, the authors found that the core 2 Duo provided the best performance among all three processors while the Pentium D was the worst. In [2] the NPB benchmark was used to analyze and evaluate a system with two Intel dual-core Xeon processors. The results were also used to test the hyper threading technology implemented in this particular processor. Researchers reached the conclusion that using a single dual core processor with HT enabled is the most efficient architecture for the embedded application.

Accordingly, in this paper we use these benchmarks to test and analyze the performance of two of the Intel's quad-core processor architectures, the core 2 Quad (Q6600) and the core i7-920 which comes with the hyper threading (HT) technology. We also used the Lmbench benchmark suit to evaluate the memory performance and data transfer operations that both processors can provide.

Hyper threading works by duplicating certain resource sections of the processor but not the main execution resources. The hyper threading technology is based on the principle that during every clock pulse only part of the processor resources are used for execution of the program code. Unused resources can also be loaded with parallel execution of another thread. This means that a single physical processor containing two logical processors can share the same computational resources. In the case of the core i7, the operating system and applications see eight cores and can distribute a work load between them, similar to a normal eight-processor system.

The results obtained from the benchmark suites will be used to provide a comparison between the two different Intel quad core architectures; also it will be used to assess the hyper threading technology found in the core i7 processors. The results also illustrate the need for insuring a proper number of thread usage, i.e. the performance degradation that can occur when too many threads are used for a particular application.

The following section of the paper describes briefly the Intel quad core processors tested. Section 3 introduces the benchmark suites used. While in the fourth section results from benchmarks are introduced and analyzed. The paper is concluded in section 5.

2 Intel Quad-Core Processors

Multiprocessing is implemented using either chip multiprocessing (CMP) or Simultaneous multithreading (SMT). CMPs also known as multi-core processors are implemented by integrating two or more independent processors (cores) on a single die (or chip).

SMT, on the other hand, is a technique for improving CPU performance. The instructions from two threads are interleaved in the CPU pipeline. SMT duplicates some circuits of the processor including the some of the pipeline stages but not duplicating the main execution resources. Intel implementation of the SMT is known as Hyper-Threading (HT).

In the following subsections both of the quad core processors tested in this work will be introduced and discussed briefly. First we will start with the core 2 quad then the core i7.

2.1 Intel Core 2 Quad Processor

The core 2 quad processor takes two dual core processors (core 2 duo) and combines them onto a single package. Each of the core 2 Duo processors features 4MB of Advanced Smart Cache, which is shared between the two cores. Combining two core 2 Duo in a single chip allows the new core 2 quad processors to have a total of 8MB L2 cache, but without the ability of sharing the entire L2 cache among all the four cores. Instead it acts like dual Core 2 processors, each sharing 4MB of L2 cache. Each core has a level 1 cache of size 32 KB instruction and 32 KB data. That affects the entire processor, since without any shared resources between the dual processor dies, duplication between the two distinct processors may occur.

In this paper we used the Core 2 Quad Q6600 processor which features an Intel quad core running at 2.4 GHz with an 8MB (2x4MB) of combined L2 cache.

2.2 Intel Core i7 Processor

The corei3, i5 and i7 processors are the newest multi-core processors from Intel, and are successors to the Intel Core 2 family. The core i7 processors are quad core processors that support the hyper threading technology, and come with many new features to enhance the processor's performance over the core 2 processors. The core i7 has an on-chip memory controller which means that the memory is directly connected to the processor. This memory controller is a triple-channel controller that supports DDR3 memory only. Also the front side bus has been replaced by the Intel QuickPath Interconnect (QPI) interface. The QPI is a packet-based point-to-point connection between the processor and the I/O chipset. Core i7 designed is with three levels of cache. Level 1 cache is of size 32 KB instruction and 32 KB data cache per core, level 2 size is 256 KB combined instruction and data per core, and level 3 cache is an 8 MB on-chip smart cache shared among all four cores.

In this paper we used the Core i7-920 processor which features an Intel quad core running at 2.8 GHz with an 8MB of L3 cache.

3 Benchmark suits

Performance evaluation of computer systems in general and processors in particular becomes more difficult with the advancement of these systems. For that reason benchmarks were used to analyze and compare between the different systems. Benchmarks are designed to mimic a specific workload to test the computer system or a certain part of it such as the processor, memory, I/O devices, or network communications. The following subsections discuss these benchmark suites.

3.1 STREAM/ STREAM2 Benchmarks

The STREAM/ STREAM2 benchmarks [1] measure memory bandwidth and latency based on the most common functions.

STREAM benchmark is a simple synthetic benchmark program that measures sustainable memory bandwidth and the corresponding computation rate for simple vector kernels. It is intended to characterize the behavior of a system for applications that are limited in performance by the memory bandwidth of the system, rather than by the computational performance of the CPU. [2]

STREAM2 is based on the same ideas as STREAM, but uses a different set of vector functions. It is an attempt to extend the functionality of the STREAM benchmark in two important ways (1) measure the sustained bandwidth at all levels of the cache hierarchy, and (2) more clearly expose the performance differences between reads and writes.

See [4] for more details about STREAM/ STREAM2 benchmarks.

3.2 NPB Benchmark Set

NPB benchmarks [5] target performance evaluation of highly parallel computers. NPB stands for NAS (Numerical Aerodynamic Simulation) Parallel Benchmark, which is developed and maintained by the NASA Ames Research Center. The NPB mimics the computation and data movement characteristics of large scale computational fluid dynamics applications.

These benchmarks are written in FORTRAN and C programming languages with the aid of OpenMP to achieve parallelism.

Problems solved by the NPB benchmarks are classified in different problem classes according to the problem size. The classes, according to the problem size they represent, are S, W, A, B, C, D, and E; where S is the smallest and E is the largest.

3.3 LMBENCH

Lmbench is a suite of simple, portable micro-benchmarks for UNIX. Lmbench measures two key features memory bandwidth and latency. It measures systems ability to transfer data between processor, cache, memory, network, and desk [18].

Lmbench contains a large number of micro-benchmarks that measure various aspects of hardware and operating system performance. It generally reports the median result for 11 measurements.

4 Benchmark Results For The Intel Quad Core Processors

Q6600 based system and runs at 2.4 GHZ clock speed with 4 GB of DDR2 RAM, while the second PC is an Intel core i7- 920 based system that runs at 2.66 GHz clock speed, with 8 MB L3 cache, and 6 GB DDR3 RAM. The benchmark suites executed on the core i7 based system were tested with both enabling and disabling the HT technology. The number of copies for each of the benchmarks was varied between single copy and up to 16 copies to test the performance of the processors with multithreaded applications. The operating system in both systems was Ubuntu 9.04 Linux with 2.6.28-16-generic kernel. The benchmarks were compiled using gcc 4.3 compiler with the optimization option of '-o3' and '-fopenmp' for OpenMP support.

4.1 STREAM/ STREAM2 Results

For STREAM benchmark we will discuss the results obtained from copy and scale functions (shown in Figs. 1 and 2) and we will discuss fill and daxpy functions from STREAM2 benchmark (shown in Figs. 3 and 4). The results show that for any given number of threads to solve the given function, the core i7 is at least 3 times faster than the core 2 quad processor. As for core i7 the best performance of this processor is obtained by running applications with five threads with the HT technology enabled. It can also be concluded that with applications using more than five threads the core i7 with HT disabled can provide performance equal or better than with the case of enabling the hyper threading technology.

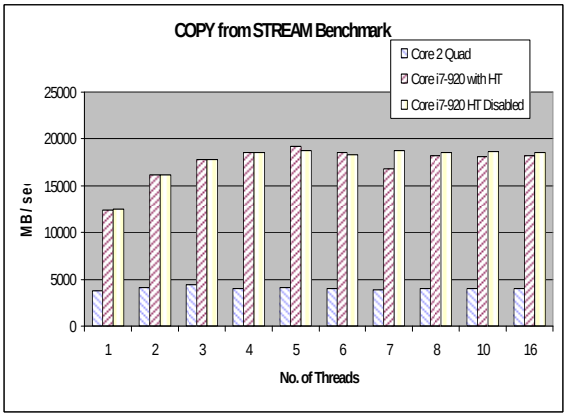


Fig. 1 Copy Function (STREAM Benchmark) Bandwidth Comparison

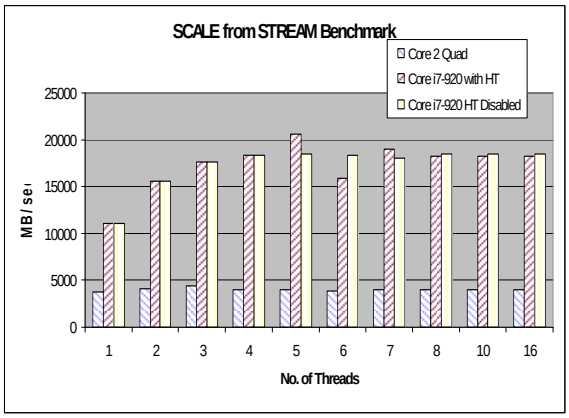


Fig. 2 Scale Function (STREAM Benchmark) Bandwidth Comparison

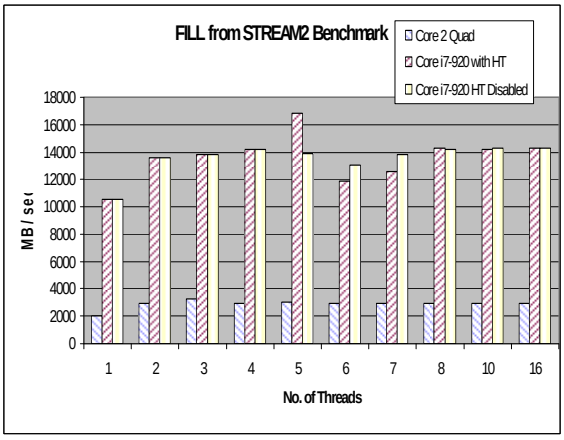


Fig. 3 Fill Function (STREAM2 Benchmark) Bandwidth Comparison

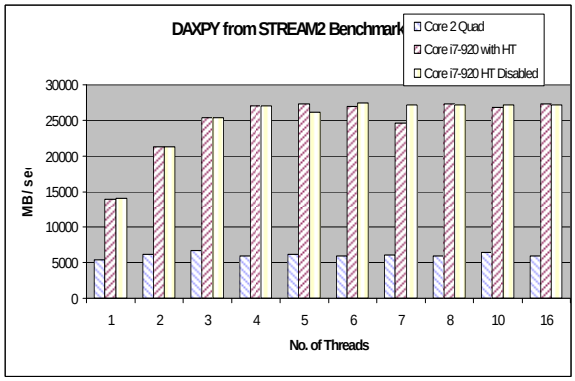


Fig. 4 Daxpy Function (STREAM2 Benchmark) Bandwidth Comparison

4.2 NPB Benchmark Suite Results

In this paper we used NPB version 3.3, and only ran eight of the benchmarks namely, BT, CG, EP, FT, SP, UA, and LU. We used the NPB-OMP which is a sample OpenMP implementation based on the sequential implementation of the serial NPB.

In this subsection we are only reporting results obtained for class B problem size. The results are split into several Figures according to the benchmark function and the processor type to be easier to read.

Figs. 5- 10 shows the results for the NPB benchmark suite for core i7 with enabling and disabling hyper threading as well as for the core 2 quad.

For core i7 processor, it is clear that hyper-threading technology enhances the performance especially when using five or more threads, but the performance is almost the same with threads less than five, which is when having number of threads less than or equal to the number of cores.

Core 2 quad gives best performance when using four threads, but performance decreases with threads more than four.

The LU benchmark (Fig. 5) shows an interesting behavior, in which, for any the evaluated processors, the performance is improved when using multi-threads as long as the thread count used is less than or equal to the available cores (logical and physical). But when using more threads the performance worsens rapidly even comparing it to case of using single thread. In other words, core i7 with HT enabled optimal performance for LU benchmark when having eight threads, four threads when the HT is disabled. Also for core 2 quad the optimal number of threads is for solving LU problem.

For this behavior, additional LU benchmark results were collected measuring the number of operations executed per second (throughput (given in mega Operations per second Mops/ sec) by a given processor with respect to thread count, the results are plotted in Fig. 8. As a comparison we also analyzed throughput results for the

EP benchmark (Fig. 8). For LU benchmark throughput one can observe that the throughput follows the same behavior of the execution time in which the throughput keeps increasing with the increment of threads as long as thread count is less or equal to the processor's cores, then it drops when using more threads than the available cores. In the case of EP benchmark, which provides a number of independent parallel workloads, the throughput increases until it reaches the maximum. When this is reached, the thread count is equal to the processor's cores, and then retains this throughput level regardless how many extra threads are used.

In general, a processor's throughput drops when the processor is in an idle state most of the time. Idle state usually occurs either when there are no operations to be processed, or when the processor is waiting for I/O or memory operations. In the case of LU benchmark, waiting for memory operation to be completed is more likely the case of performance drop.

The LU benchmark is a simulated computation fluid dynamics application that uses symmetric successive over-relaxation (SSOR) method to solve a seven-block-diagonal system resulting from finite-difference of the Navier-Stokes equations in 3-D. This is accomplished by splitting it into block Lower and Upper triangular systems. There are at least two methods to implement LU in parallel: hyper-plane and pipelining [16]. Both methods of parallelization generate highly dependent parallel sections, in which certain sections cannot be processed unless the results from previous sections are ready. This kind of dependency will force some of the threads/cores to wait until another thread is done with its part. In this case with the usage of threads equal or less than the available cores, all threads and local data for the threads are loaded in the assigned core and its local cache and will be in a wait state. When using more threads than the available cores in order to execute all threads concurrently cores will perform context switching to switch from one thread to another by saving the current thread's register conditions and results. The second thread's data sets and registers are then reloaded. This is done once the execution time assigned for a thread is expired. In the case that the second thread is dependent on the results from thread one, if thread one execution is not done and its time is expired and the core switches to thread two, then this thread will be in a wait state and waste the core's time since the required data is not ready. This will lead to a dramatic drop in the performance which was noticed in the LU results in Figs. 7 and 8.

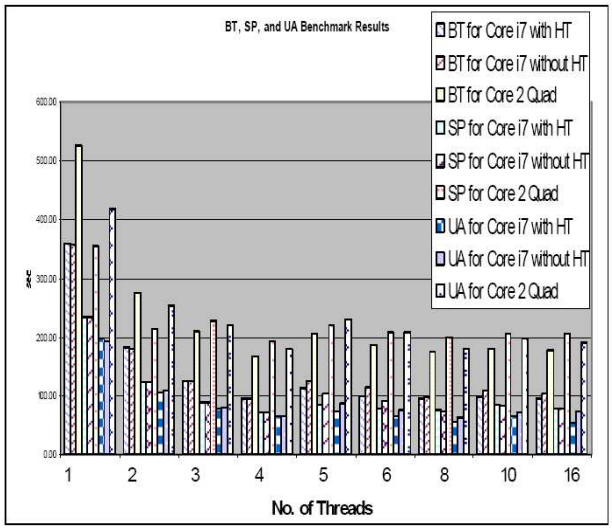


Fig. 5 Execution Time Results obtained BT, SP, and UA benchmarks (Class B)

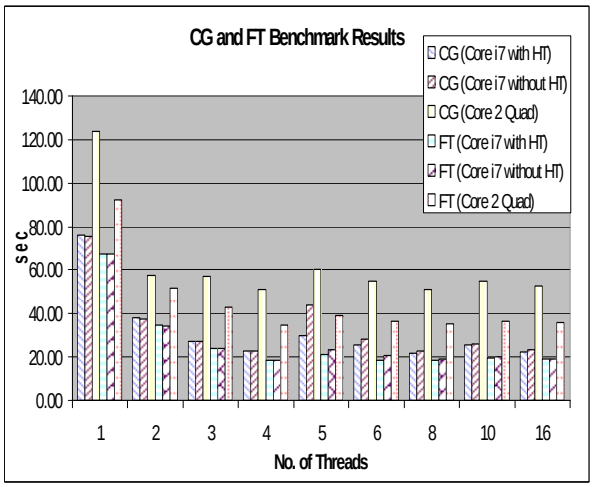


Fig. 6 Execution Time Results obtained from CG and FT Benchmarks (Class B)

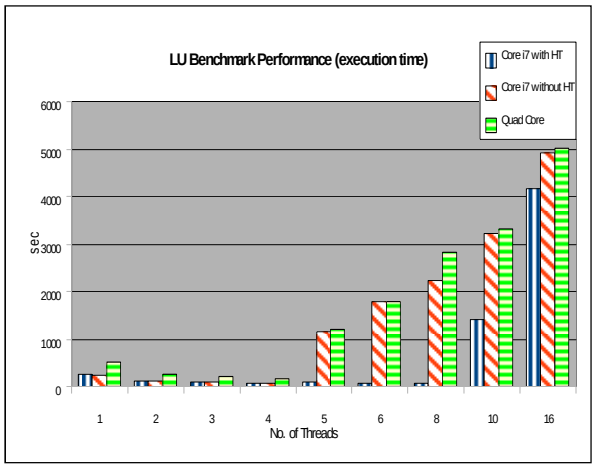


Fig. 7 LU Benchmark Results for Core i7 and Core 2 Quad (Class B)

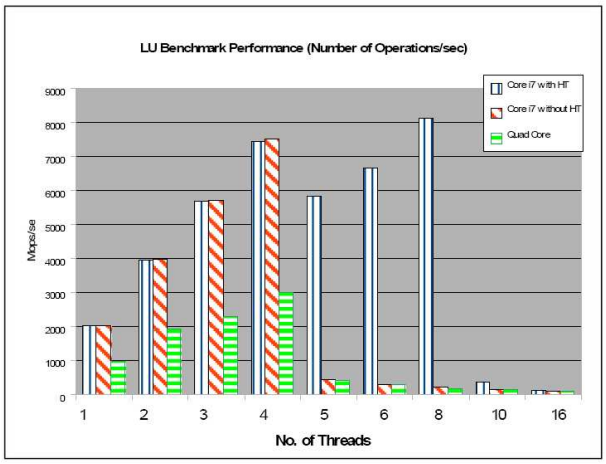


Fig. 8 Throughput Results from LU Benchmark for Core i7 and Core 2 Quad (Class B)

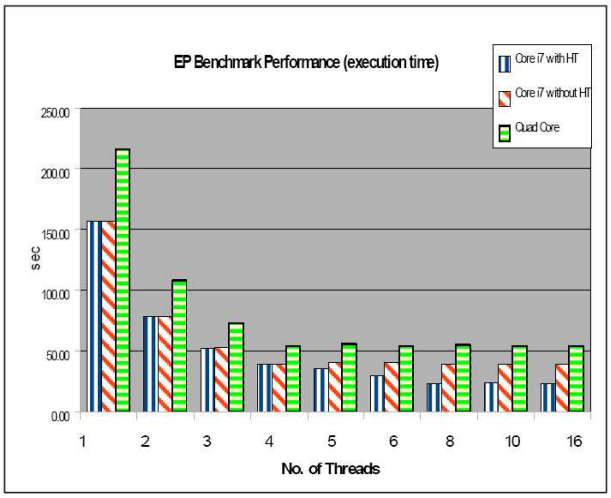


Fig. 9 EP Benchmark Results for Core i7 and Core 2 Quad (Class B)

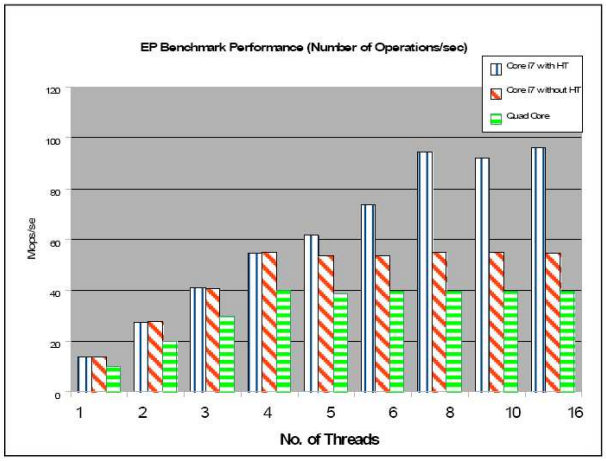


Fig. 10 Throughput Results from EP Benchmark for Core i7 and Core 2 Quad (Class B)

Figs. 11- 14 are for the speedup computed for LU, EP, BT and SP benchmarks respectively. The speedup is computed as stated in the following equation

$$Speedup = \frac{T_1}{T_N}$$

Where T_1 is the execution time using single thread
T_N is the execution time using N threads

Speedup values are an indication of the processor's performance, and the maximum obtained speedup can be used to locate the best (optimal) number of threads to be used for solving any of the tested benchmarks. For the Core i7 processor, the maximum speedup obtained from the LU benchmark is about 4 when running 8 threads with HT support, while it is around 3.7 when HT is disabled and four threads are used. For the core 2 quad, the highest speedup value is around 3.1 and is reached when using 4 threads. It can be observed from the Fig.s that the speedup drops drastically when the number of threads exceeds the number of available cores.

On the other hand, the speedup results for the EP benchmark increases as the number of threads increase until it reaches a saturation value which is close to the number of cores (physical and logical cores) for the processors under evaluation. Using threads more than the available cores will not improve the performance. Core i7, with the HT enabled causes a maximum speedup of almost 7 when using 8 or more threads. While for core i7 with the HT disabled and for core 2 quad, the speedup is 4 for four or more threads.

For BT and SP benchmarks, the core i7 processor delivers its maximum speedup with 4 threads; speedup obtained from BT benchmark is 3.81 when the HT is enabled and 3.79 when disabling the HT technology. For SP benchmark maximum speedup is about 3.24 when enabling HT and 3.23 when HT is disabled. For those two

benchmarks hyper-threading does not have much effect improving the processor's performance.
Core 2 Quad speedup results for BT and SP benchmarks shows that the maximum value is with 4 threads with values of 3.16 for BT and 1.84 for SP.

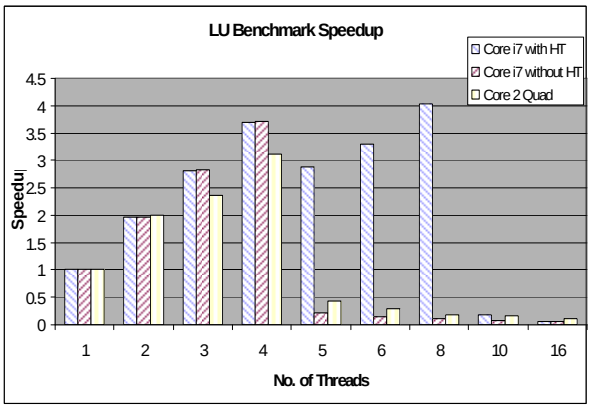


Fig. 11 Speedup Obtained from the LU Benchmark using Multiple Threads

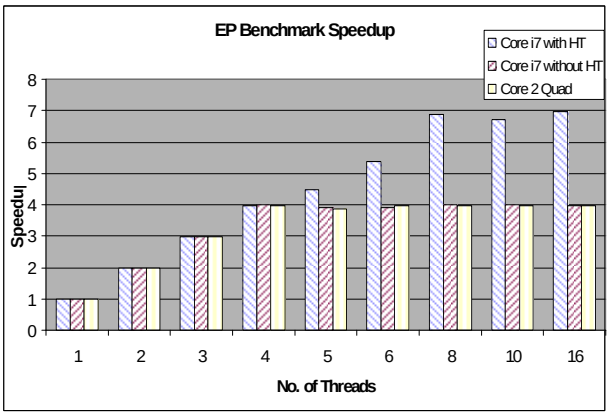


Fig. 12 Speedup Obtained from Running EP Benchmark

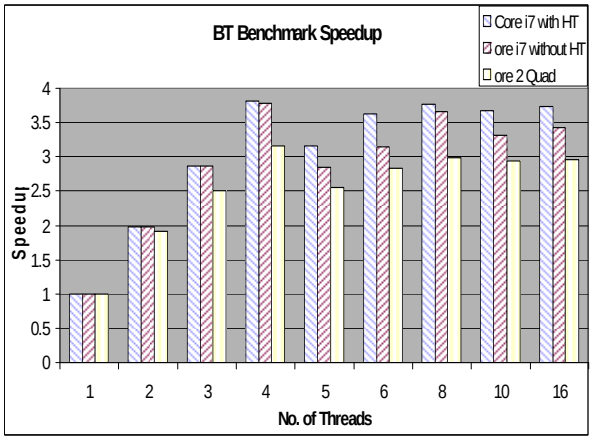


Fig. 13 Speedup Obtained from Running BT Benchmark using Multiple Threads

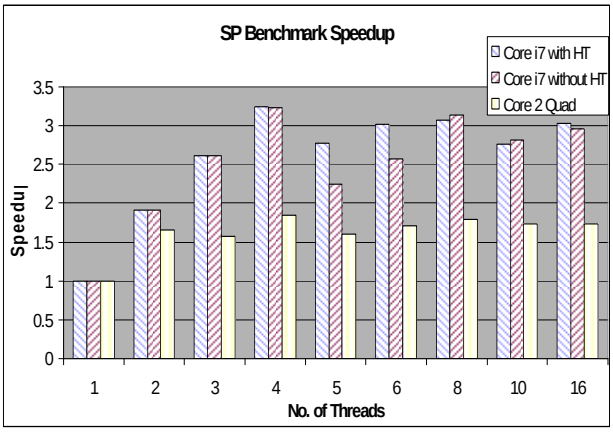


Fig. 14 Speedup Obtained from Running SP Benchmark using Multiple Threads

Tables 1 and 2 list the best (optimal) performance obtained from running NPB benchmark suite using class B for core i7 and core 2 quad processors respectively. The tables show the optimal number of threads for processing every benchmark, the throughput, execution time, and speedup obtained at that thread count.
Comparing the results in table 1, for EP, FT, LU, and UA benchmarks hyper-threading technology enhances the performance of the core i7 processor, while for the rest benchmarks hyper-threading didn't show any noticeable improvement.
From table 2, core 2 quad processor shows that using four threads is the best for almost all the tested benchmarks which means that best performance obtained when using all the available cores.

Table 1: Core i7 Best Performance For given Benchmarks

Core i7 with HT					Core i7 without HT			
Bench	Thread Count	Mop/s	Time	Speed up	Thread Count	Mop/s	Time	Speed up
BT	4	7449.95	94.25	3.808	4	7418.55	94.65	3.787
CG	8	2499.43	21.89	3.474	8	2425.29	22.56	3.355
EP	16	95.8	22.42	6.982	10	54.83	39.17	3.9950
FT	8	4994.09	18.43	3.658	4	4934.82	18.65	3.618
LU	8	8113.04	61.48	4.025	4	7503.7	66.48	3.716
SP	4	4906.21	72.36	3.244	4	4909.19	72.32	3.237
UA	16	40.06	55.14	3.577	8	34.67	63.72	3.032

Table 2: Core 2 Quad Best Performance

Core 2 Quad				
Benchmark	Thread	Mop/s	Time	Speedup
BT	4	4226.22	166.15	3.16365
CG	4	1073.87	50.95	2.42355
EP	4	39.66	54.15	3.98006
FT	4	2665.23	34.54	2.66995
LU	4	3001	166.22	3.11148
SP	4	1837.67	193.19	1.84430
UA	8	12.34	179.03	2.33106

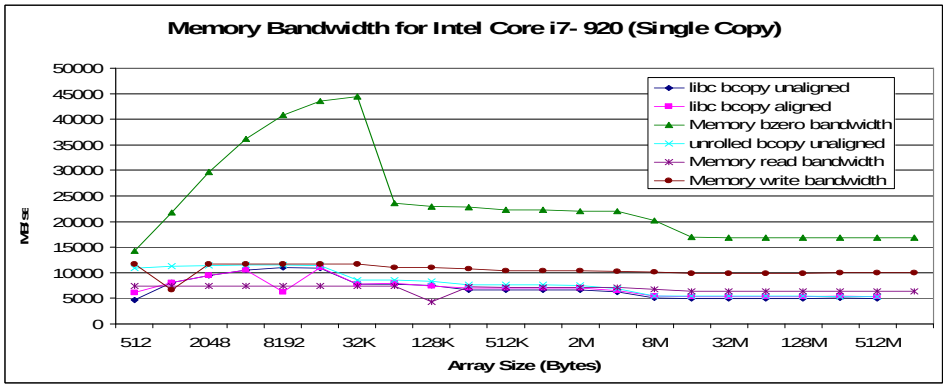
4.3 LMBENCH Results

The memory bandwidth results obtained using single and two copies of this benchmark for both processors will only be shown in this subsection. Fig. 15 illustrates the results for core i7-920 with HT enabled, while Fig. 16 represents the results for the corei7-920 processor after disabling the HT. Finally, Fig. 17 represents the results obtained from the core 2 Quad processor.

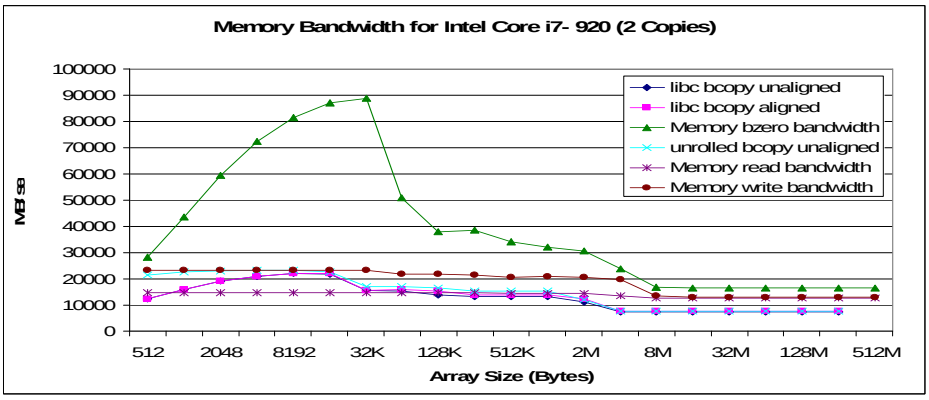
Comparing the results from these Fig.s, it is clear that the core i7 processor (with or without HT) outperforms the core 2 quad at least by three times the amount of data transferred during the benchmark runs. Also, it can be noted that the memory bandwidth doubles when running two copies in comparison to single copy runs.

For the core 2 quad processor, in general the bandwidth drops when the array size is larger than 32 KB, due to the sizes of their L1 cache per core. Also there is a second large drop in the bandwidth when using array sizes larger than 2 MB because of the L2 cache, since each two cores share a 4 MB of L2 cache. For instance, the memory bzero bandwidth results from Fig. 4 show when using arrays larger than 32 KB. The memory bandwidth drops by about 30%.When increasing array size from 2MB to 4MB it drops by 42%. When increasing it above 8MB it drops by 70%.

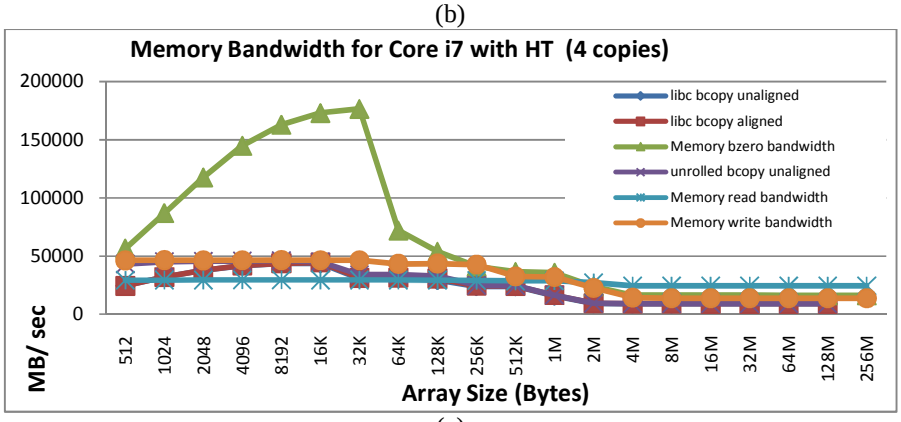
On the other hand, for core i7, there are two noticeable drops on the memory bandwidth. One occurs when the array sizes exceed 32 KB (L1 cache size), and the other one when the size is larger than 2 MB (due to L3). There is a third drop when exceeding the 128 KB size. For instance, the memory bzero bandwidth results from Fig. 6, when using arrays larger than 32 KB, the bandwidth drops by about 43%. When increasing the array size from 2MB to 4MB it drops by 23%. If is increased above 8MB it drops by 30%. When disabling the hyper threading, the memory bzero bandwidth results from Fig. 8, using arrays larger than 32 KB the memory bandwidth drops by about 50%. When the array size is increased from 2MB to 4MB it drops by 24%. Increasing the array size above 8MB causes it to drop by 29%.



(a)

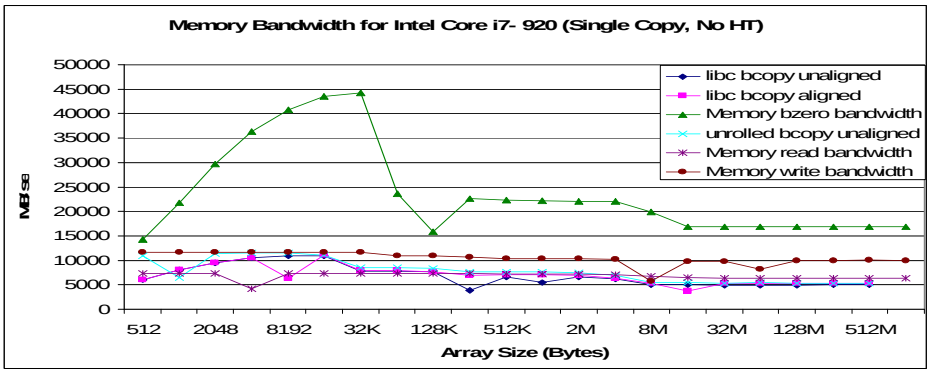


(b)

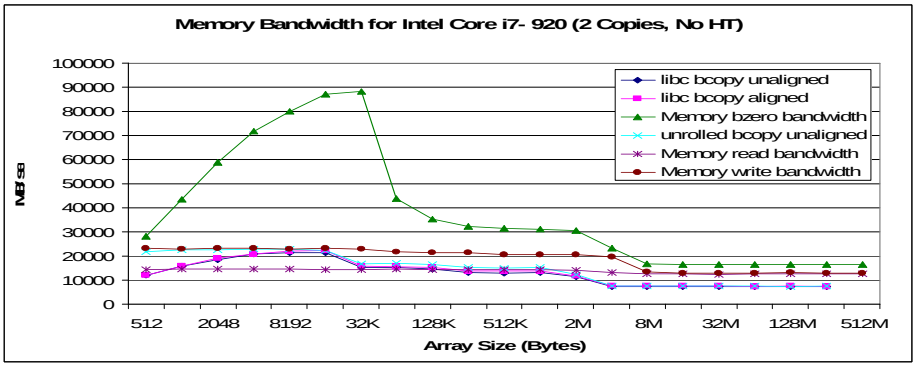


(c)

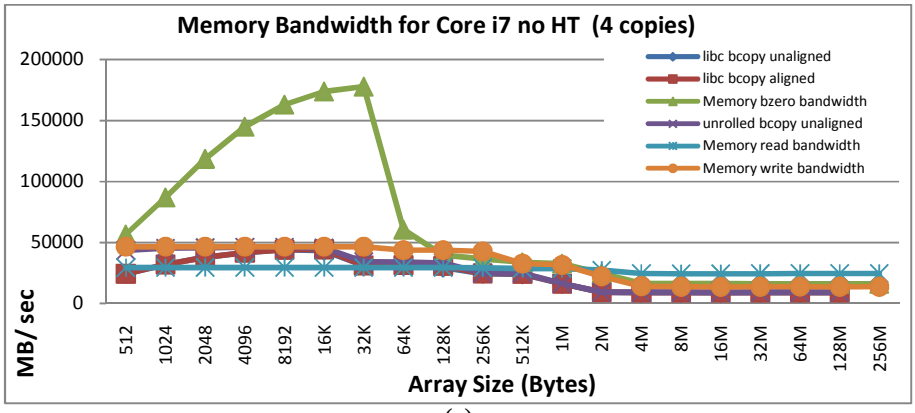
Fig. 15 Memory Bandwidth Results for Core i7-920 (HT Enabled) using (a) 1 Copy, (b) 2 Copies, and (c) 4 Copies



(a)

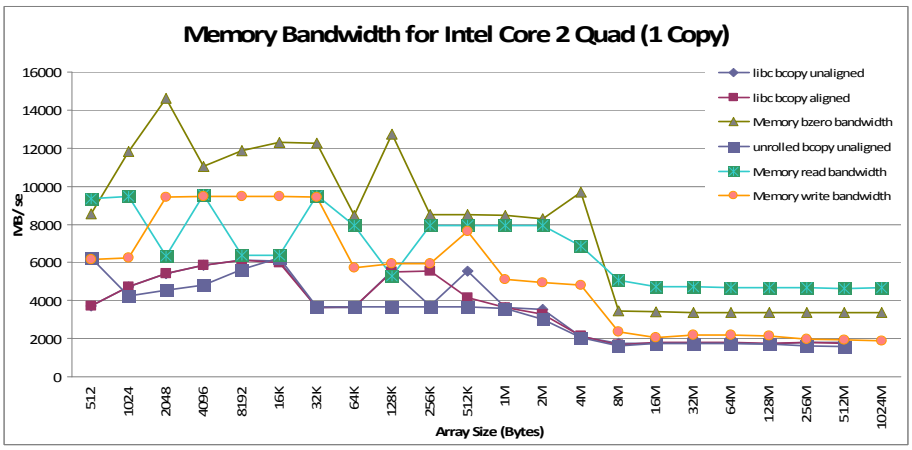


(b)

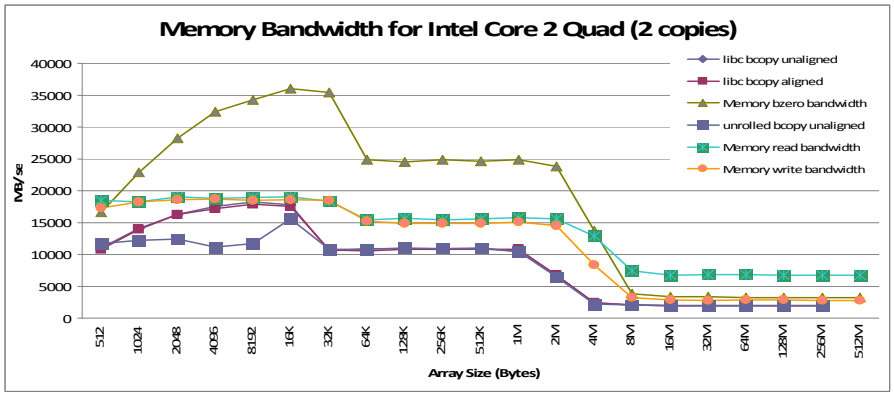


(c)

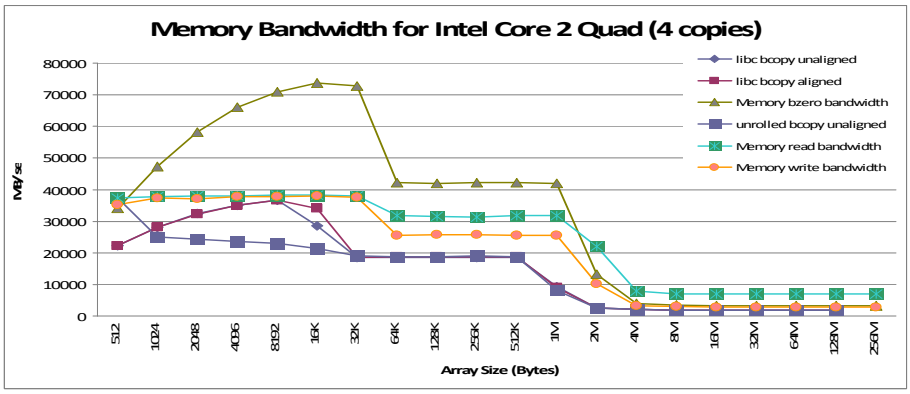
Fig. 16 Memory Bandwidth Results for Core i7-920 (HT Disabled) using (a) 1 Copy, (b) 2 Copies, and (c) 4 Copies



(a)



(b)



(c)

Fig. 17 Memory Bandwidth Results for Core 2 Quad using (a) 1 Copy, (b) 2 Copies, and (c) 4 Copies

5 Conclusion

Multi-core processors are being used in most new desktop systems and are becoming common in embedded applications. Researchers at UTA are investigating optimal configurations of several available multi-core processors suitable for developing real-time software for multithreaded application. The application under investigation by UTA researchers requires a large degree of multi-processing in order to meet the real-time needs. In the paper several multithreaded benchmarks were studied in order to provide directions for proper program

design to insure maximum performance for the application. The multiprocessing capabilities of two of the latest Intel quad-core processors; the core 2 quad and the core i7 are considered suitable for the embedded application. These processors were being considered for use for the embedded application. The Intel Core i7 processors are a significant evolutionary step forward from their Core 2 predecessors. The results from all benchmark suites have shown that the core i7 processor provides on average three times better performance than the core 2 quad for these benchmarks. For the core i7 processor, enabling hyper-threading technology enhances the performance for applications with five to eight threads. Core 2 Quad gives good performance when using four threads, but performance decreases with threads more than four for these benchmarks.

The Hyper-threading technology, implemented in the core i7 processor, proves that although it can improve the performance of a given processor, but care should be taken when using (enabling) this technology since for certain applications not only it does not have any effect on increasing the performance but also it may lead to the drop of the performance.

This could be of particular importance in some embedded applications with hard real-time constraints.

References:

[1] L. Peng, J-K. Peir, T. K. Prakash, C. Staelin, Y-K. Chen, D. Koppelman, "*Memory Hierarchy Performance Measurement of Commercial Dual-Core Desktop Processors*", In *Journal of Systems Architecture*, vol 54(8), Aug. 2008, page 816-828.

[2] Ryan E. Grant and Ahmad Afsahi, "*A Comprehensive Analysis of Multithreaded OpenMP Applications on Dual-Core Intel Xeon SMPs*", Workshop on Multithreaded Architectures and Applications (MTAAP'07), In Proceedings of the 21st International Parallel and Distributed Processing Symposium (IPDPS 2007), Long Beach, California, USA, March 26-30, 2007.

[3] http://en.wikipedia.org/wiki/NAS_benchmarks

[4] http://www.cs.virginia.edu/stream/

[5]http://www.nas.nasa.gov/Resources/Software/npb.html

[6] Cascaval, C.; Castanos, J.G.; Ceze, L.; Denneau, M.; Gupta, M.; Lieber, D.; Moreira, J.E.; Strauss, K.; Warren, H.S., Jr., *"Evaluation of a multithreaded architecture for cellular computing"*, Eighth International Symposium on High-Performance Computer Architecture, 2002. Proceedings, 2-6 Feb. 2002. pp 311 – 321.

[7]http://edc.intel.com/Platforms/EP80579/#platform-overview-content=platform-overview-toggle~~visible-content

[9] http://www.intel.com/products/processor/core2quad/

[10]http://www.intel.com/products/processor/corei7/index.htm

[11] T. Hanawa, M. Sato, J. Lee, T. Imada, H. Kimura, and T. Boku, "*Evaluation of Multicore Processors for Embedded Systems by Parallel Benchmark Program Using OpenMP*", IWOMP 2009, pp15-27

[12] A. Kayi, T. El-Ghazawi, and G. Newby, "Performance issues in emerging homogeneous multi-core architectures", Simulation Modelling Practice and Theory, Vol 17, issue 9, Oct 2009, pp 1485-1499

[13] L. Peng, J-K. Peir, T. K. Prakash, Y-K. Chen, D. Koppelman, "Memory Performance and Scalability of Intel's and AMD's Dual-Core Processors: A Case Study," In *Proceedings of the 26th IEEE International Performance Computing and Communications Conference (IPCCC)*, New Orleans, LA, Apr. 2007.

[14] M. Curtis-Maury, X. Ding, C. Antonopoulos, and D. Nikolopoulos, *"An Evaluation of OpenMP on Current and Emerging Multithreaded/Multicore Processors*", In Proceedings of the First International Workshop on OpenMP IWOMP 2005, Eugene, Oregon, June 2005

[15] R. Radhakrishnan, R. Ali, G. Kochhar, K. Chadalavada, R. Rajagopalan, Jenwei Hsieh, and Onur Celebioglu, *"Evaluating Performance of BLAST on Intel Xeon and Itanium2 Processors"*, Second International Symposium on Parallel & Distributed Processing & Applications (ISPA 04), LNCS 3358, pp 1017-1023

[16] H. Jin, M. Frumkin, and J. Yan, "*The OpenMP Implementation of NAS Parallel Benchmarks and Its Performance*",http://www.nas.nasa.gov/News/Techreports/1999/PDF/nas-99-011.pdf.

[17] Marius Marcu , Dacian Tudor , Sebastian Fuicu , Silvia Copil-Crisan , Florin Maticu , Mihai Micea, "*Power efficiency study of multi-threading applications for multi-core mobile systems*", WSEAS Transactions on Computers, v.7 n.12, p.1875-1885, December 2008

[18] L. McVoy, C. Staelin, *"lmbench: Portable tools for performance analysis"*, 1996 USENIX Annual Technical Conference, pp 279- 294

[19] A.A.Veglis, and A.S.Pombortsis, *"Performance Analysis of Crossbar Multiprocessor Architectures via Analytical Simulation"*, in Proc of the 6th WSEAS CSCC (CSCC 2002).

[20] Yu-Fai Fung , Wai-Leung Cheung , Gujit Singh , Muhammet F. Ercan, *"An empirical study of bi-level parallel computing on a PC"*, Proceedings of the 2nd WSEAS International Conference on Electronics, Control and Signal Processing, pp 1-5, December 07-09, 2003, Singapore.

[21] Yumi Takizawa, Saki Yatano, Atsushi Fukasawa, *"Digital signal processing with embedded system for advanced mobile communications"*, Proceedings of the 2nd WSEAS International Conference on Circuits, Systems, Signal and Telecommunications, pp 98-101, January 25-27, 2008, Mexico.