Performance Study of Hyper-Threading Technology on the LUSITANIA Supercomputer

César Gómez-Martín[1], José Luis González-Sánchez[1], Javier Corral-García[1], Ángel Bejarano-Borrega[1], Javier Lázaro-Jareño[1]

CénitS (Centro Extremeño de iNvestigación, Innovación Tecnológica y Supercomputación)
`cesar.gomez@cenits.es`
`joseluis.gonzalez@cenits.es`
`javier.corral@cenits.es`
`angel.bejarano@cenits.es`
`javier.lazaro@cenits.es`

Key words: Hyper-Threading, performance, LUSITANIA, supercomputer

Abstract. In this paper Hyper-Threading technology is evaluated on the LUSITANIA supercomputer. LUSITANIA is a SMP-ccNUMA system composed of two HP Integrity SuperDomes SX2000.
The effects of Intel's Hyper-Threading Technology on Intel Itanium processors are not very clear, in order to understand how this technology affects the performance of LUSITANIA, several benchmarks have been studied. Those bechmarks have been carefully chosen to try to demonstrate the performance gain or degradation when running different kind of applications. The results demonstrate that Hyper-Threading achieve better performance when the application needs to communicate data among processes or threads, but, whenever the application is embarrassingly parallel, cache-friendly or all the floating point units are optimized it is not advisable to enable Hyper-Threading. We have also empirically study that the sense of the application will determine whether Hyper-Threading will accelerate or diminish performance. Enabling Hyper-Threading or disabling it is not an exact science, it depends on so many factors that a bechmark study of the problem that is going to be addressed is highly recommended.

1 Introduction

Hyper-Threading Technology makes a single physical processor appear as two logical processors; the physical execution resources are shared and the architecture state is duplicated for the two logical processors. From a software or architecture perspective, this means operating systems and user programs can schedule processes or threads to logical processors as they would on multiple physical processors. From a microarchitecture perspective, this means that instructions from both logical processors will persist and execute simultaneously on shared execution resources [5]. The only requirement to take advantage of this technology is to have symmetric multiprocessing (SMP) support in the operating system. Hyper-Threading is totally transparent to the operating system and does not need any kind of configuration as it can be enabled inside the EFI or BIOS [9].

Fig. 1. LUSITANIA

1.1 Hardware Parallelism

Hardware Parallelism follows two basic techniques: Instruction Level Parallelism (ILP) and Thread/Task Level Parallelism (TLP). Simultaneous Multi-Threading (SMT) is the technology used by modern processors that combines ILP and TLP [6].

- Modern processors have tried to increase the number of instruction that can be executed during one clock cycle, this is called Instruction Level Parallelism (ILP), i.e. if a instruction tries to increase a value and a different instruction has to multiply two digits, those instructions could be done together if there were two free processing units. There are some important points to get better ILP:
  - A big set of execution units is needed in order to distribute all the instructions.
  - It is very important to have a big set of registers to be able to simultaneously execute several operations.

  Some problems should be avoided to exploit ILP:
  - If there are some correlative instructions that need the same execution unit it is not possible to run them parallely.
  - If an operation depends on the result of a previous instruction they have to be serially executed.
- On the other hand, Thread/Task Level Parallelism (TLP) allows n-core or multiprocessor systems to concurrently execute several tasks or threads from one or various programs.
- Hyper-Threading technology is based on Simultaneous Multi-Threading (SMT), which is a combination of ILP and TLP. SMT dynamically allocates empty resources between tasks or threads allowing multiple-thread execution along processors.

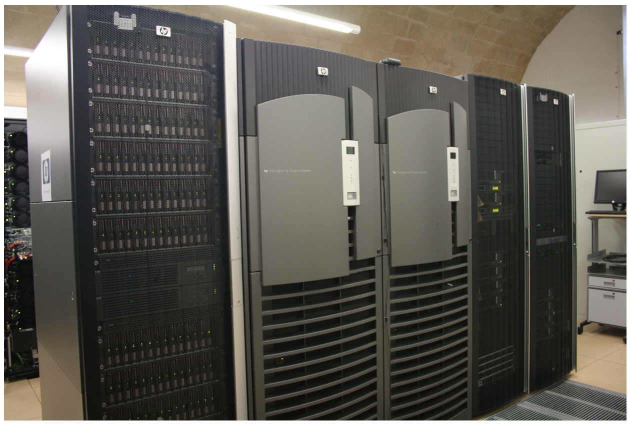

Fig. 2. HP Integrity SuperDome SX2000

2 LUSITANIA

LUSITANIA is a SMP-ccNUMA system with 2 HP SuperDomes SX2000 nodes installed at Extremadura Supercomputing Center (CénitS) in Cáceres, Spain (see Figure 1). The demanded applications in this Supercomputer are multidisciplinary and heterogeneous so, it was very important to clarify which was the best configuration to slightly adjust all the parameters that could improve the performance of the system. One of the first paradox that was addressed was to investigate whether the effects of Intel Hyper-Threading Technology were good or harmful when running those applications on LUSITANIA.

2.1 Hardware Configuration

LUSITANIA is the Supercomputer of Extremadura (Spain), it has some of the biggest shared-memory nodes of Spain and Europe. The solution is based on two shared-memory HP Integrity SuperDome SX2000 supernodes (see Figure 2):

- They both are equipped with 64 dual-core Intel Itanium2 Montvale processors running at 1.6GHz with 18 MB cache, 1TB memory (upgradeable) on a single image and SX2000 chipsets designed to take advantage of Itanium2 Montvale CPUs [7].
- The Itanium architecture is based on explicit ILP, in which the compiler makes the decisions about which instructions will execute in parallel. This alternative approach helps Itanium processors execute up to six instructions per clock cycle.
- SX2000 chipsets are interconnected via crossbar switches with three independent connections to ensure the best performance by using multipathing, ECC protection and load-balancing.
- The well-balanced architecture of the HP Super-Scalable Processor Chipset SX2000 makes the Dual-Core Intel Itanium2 Montvale more powerful.

– The system is also designed to derive significantly greater performance from these existing processors by providing systems with enhanced bandwidth, high memory capacity and reduced memory latency.
– HP SX2000 Chipset is also built to support Intel Itanium processors with multithreading for enhanced performance and scalability [7]. Its VLSI components consist of a cell controller, a memory buffer, a crossbar switch, a PCI-X system bus adapter and a PCI-X host bridge. The chipset enhances interconnectivity between processors, memory and I/O cards, providing you with a high-performance computer system.
– HP Superdome SX2000 consists on 16 cells with interleaved memory for shared objects or data structures. A portion of memory is taken from cells of the system (typically all of the cells) and is mixed together in a round robin fashion of cache-line-size chunks. It has the characteristic that memory accesses take a uniform amount of time. In other words, it has uniform latency no matter which processor accesses it.
– The cell controller (CC) maintains a cache-coherent memory system (ccNUMA) using a directory-based memory controller [8]. The CC's memory controller is combined with memory buffers and DIMMs to create four independent memory systems (quadrants). The memory buffers enable streaming of data between the CC and DIMMs at 533 MT/s. These memory buffers accelerate access to memory, enable memory ECC and can buffer several cache lines going both to and from memory. Up to 32 DIMMs can be controlled by the CC's memory controller [7].

2.2 Software Configuration

The following software versions have been used to run all the benchmark tests:

– Suse Linux Enterprise (v.10) is the operating system installed on the SuperDome.
– Intel Fortran Compiler (v.11.0.074) applying the O3 level of optimization.
– Intel C/C++ Compiler (v.11.0.074) applying the O3 level of optimization.
– HP-MPI libraries and binaries to compile and run MPI applications (v.02.02.07.00 Linux IA64).
– Intel OpenMP implementation to generate multi-threaded applications.

3 Performance Evaluation

In order to measure the performance (with and without Hyper-Threading) on one of the SuperDomes, a set of commonly used benchmarks have been used. High-Performance Linpack (HPL 1.0a) and NAS Parallel Benchmark (NPB3.0-OMP) are very popular when evaluating Supercomputer performance since they are meant to exploit the system as much as possible. Furthermore, the choice of those bechmarks, due to the use of the most commonly used high-performance programming paradigms (MPI and OpenMP), provides the two more interesting environments within modern parallel computation techniques that are important to evaluate:

– HPL-1.0a benchmark uses MPI to communicate different processes through a message-passing mechanism.
– NPB3.0-OMP is the OpenMP implementation of NPB that takes advantage on ccNUMA systems like SuperDomes by using a thread model approach.

3.1 High Performance Linpack (HPL 1.0a)

Linpack benchmark is a performance test widely used by High-Performance Computing community. This benchmark is not aimed to measure the general performance of a system but it evaluates a very specific performance area by solving a random dense system of linear equations in double precision (64 bits) arithmetic [3]. Thus Linpack provides a quite accurate way of knowing the performance of real applications on a HPC environment. The measurement obtained from Linpack is the number of floating-point operations per second (FLOPS). The HPL software is a portable implementation of the High-Performance Linpack Benchmark for distributed-memory computers, the package requires the availability on your system of an implementation of the Message Passing Interface MPI (1.1 compliant). An implementation of either the Basic Linear Algebra Subprograms BLAS or the Vector Signal Image Processing Library VSIPL is also needed.

Since HPL users get better benchmark performance by utilizing BLAS from the Intel Math Kernel Library (Intel MKL), we finally decided to use the Intel optimized HPL binary directly (mp_linpack) on one of the SuperDomes (1 TB shared RAM).

Calculating the solution requires $2/3n^3 + 2n^2$ floating point operations, where n is the matrix dimension. The values of n (problem size) are normally between $n = 100$ and $n = 1000$, but, since LUSITANIA's nodes have 1TB RAM on a single image a bigger value must be used in order to achieve better performance results:

– The problem size should be the largest to fit in the memory. Each Superdome has 1 TB RAM (i.e. 125 billions double precision elements).
– Square root of that number is 353553.
– Some memory should be used for Operating System because if the problem size is too large, it is swapped out, and the performance will degrade. As a rule of thumb, 80% of the total memory will be a starting point for problem size ($n = 282842$).
– Data locality is also a good way of improving performance. Each SuperDome is configured with 25% interleaved memory so the remaining 75% will be used to exploit data locality ($n = 211286$).
– To round down a $n = 200000$ problem size has been finally set.

To understand the behavior of the HPL Linpack on one of the supernodes, different simulations, changing the number of working processes, have be run:

– The results in Table 1 and Figure 3 show that there aren't big improvements if we enable Hyper-Threading.
– When the number of processes is less than the number of logical processors the performance is similar.

# Processes	GFLOPS without HT	GFLOPS with HT
36	211.0	210.5
64	368.4	366.3
81	456.0	457.7
128	685.2	686.3
144	274.3	386.3

Table 1. High-Performance Linpack results

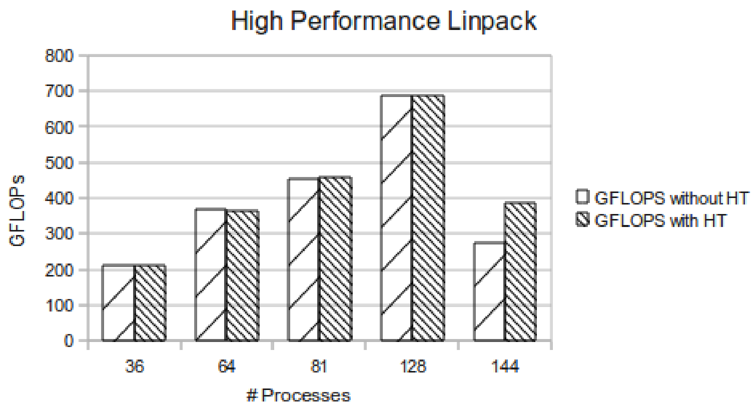

Fig. 3. High-Performance Linpack

- Only when the system is overloaded we achieve around 40% performance improvement.
- To get the ideal performance 128 processes should be used, i.e. it is not good to overload the system when running this type of problems
- HPL was compiled using the BLAS library from the Intel Math Kernel Library, this means that the floating point units were very optimized and didn't allow much space for concurrent processes on the same unit.

We can conclude that, in the ideal case, when all the processors are executing High-Performance Linpack there is an improvement of 1.1 GFLOPs, thus running Linpack combined with the HP-MPI message passing library and Hyper-Threading technology exploits better all the hardware resources.

3.2 NAS Parallel Benchmark (NPB 3.0-OMP)

Traditional benchmarks, such as the Linpack benchmark, were generally specialized for vector computers and could be inappropriate for highly parallel systems as they were not designed for automatic software parallelization tools. That was the reason why NAS Parallel Benchmarks (NPB) were developed in 1991 [1]. NPB focuses on computational fluid dynamics and related aeroscience disciplines whose studies and problems are traditionally achieved using High-Performance Computers.

There are currently different versions of the benchmarks, but the NPB 3.0 version has an OpenMP implementation which is appropriate for ccNUMA shared memory supercomputers such as LUSITANIA [4]. OpenMP is a set of compiler directives that extend Fortran, C and C++ to express shared memory parallelism.

NPB contains eight benchmark kernels (MultiGrid, Conjugate Gradient, Fast Fourier Transform, Integer Sort, Embarrassingly Parallel, Block Tridiagonal, Scalar Pentadiagonal and Lower-Upper symmetric Gauss-Seidel). Four of them are very interesting because they are close to what researchers usually execute on LUSITANIA. The Million Operations Per Second (MOPS) that one SuperDome can develop with or without Hyper-Threading will be measured. Each kernel has three problem sizes (classes A, B and C). In order to use an up-to-date problem size class C has been chosen because it is suitable for evaluating modern supercomputers [2].

Fast Fourier Transform: This benchmark contains a kernel of a 3-D fast Fourier Transform (FFT)-based spectral method. FT performs three one-dimensional (1-D) FFTs, one for each dimension. It is a rigorous test of long-distance communication performance. The size for the Class C problem of the FT benchmark is 512^3.

# Threads	MOPS without HT	MOPS with HT
36	29780.29	29195.41
64	48249.32	47119.27
81	52654.43	51265.21
100	58083.44	55094.50
128	61852.50	66353.47
144	38097.25	46065.66
169	37318.60	47670.45
196	36430.32	47253.03
225	36468.69	45490.30
256	37794.78	43818.99

Table 2. NPB Fast Fourier Transform results

After running several executions using different number of threads (see Table 2 and Figure 4), the results show that:

- When the number of threads is less than the number of available processors there is better performance if HT is disabled.
- When we use all the processors HT improve the performance more than 7%.
- Whenever the system is overloaded the improvement of HT is between 15% and 29%.

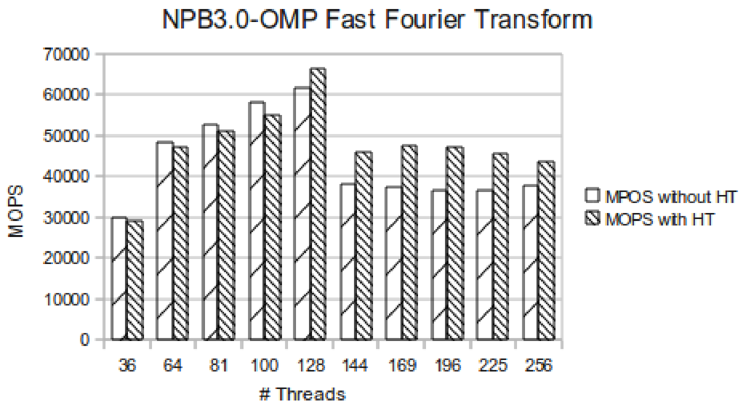

Fig. 4. NPB Fast Fourier Transform

The design of the FT benchmark requires using intensive floating point operations, but it also needs communication among processes. So it is advisable to use HT in the ideal case (128 threads running on 128 processors) and also when the system is overloaded, but it is not good if all the resources are not going to be used.

Embarrassingly Parallel: EP benchmark generates pairs of Gaussian random deviates according to a specific scheme. The goal is to establish the reference point for peak performance of a given platform. It provides an estimate of the upper achievable limits for floating point performance without significant interprocessor communication. The size for the Class C problem of the EP benchmark is 2^{32}.

# Threads	MOPS without HT	MOPS with HT
36	1816.52	1820.17
64	3210.63	3175.16
81	4048.68	3980.22
100	5023.44	4868.53
128	6008.92	5959.08
144	3589.57	4590.48
169	4229.76	5364.78
196	4563.59	5407.18
225	4347.38	5646.47
256	3763.56	4839.71

Table 3. NPB Embarrassingly Parallel results

From the observation of the results (see Table 3 and Figure 5) there are important facts that have to be carefully studied:

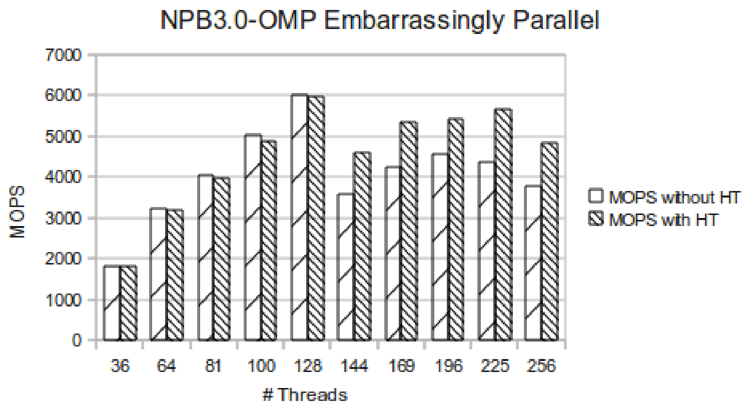

Fig. 5. NPB Embarrassingly Parallel

- In these kind of embarrassingly parallel applications it is not a good idea to use HT Technology because it always diminish the performance when the system holds more threads than the number of availables cores.
- It is only advisable to enable HT in the not ideal cases in which the system has to deal with more than one thread per core.
- HT disabled provides 1%-3% MOPS gain.

This problem is used in all the typical Monte Carlo simulations, it is embarrassingly parallel, that means that threads don't need to communicate very often, they calculate arithmetic operations. These intensive floating point operations exploit all the processor resources most of the time, so it is better to disable HT for this type of Monte Carlo applications.

Block Tridiagonal: The Block Tridiagonal kernel is a simulated CFD application that uses an implicit algorithm to solve 3-dimensional (3-D) compressible Navier-Stokes equations. The finite differences solution to the problem is based on an Alternating Direction Implicit (ADI) approximate factorization that decouples the x, y and z dimensions. The resulting systems are Block-Tridiagonal of $5x5$ blocks and are solved sequentially along each dimension. The size for the Class C problem of the BT benchmark is 162^3.

The OpenMP BT benchmark was optimized to use smaller arrays to reduce the memory usage and to improve the cache performance, this is good for ccNUMA systems because, in the case of the SuperDomes, each processor has 18MB cache size. The results show that it is not a good idea to enable HT to execute cache-friendly applications.

After running BT benchmark it can be observed (Table 4 and Figure 6) that:

- It is unwise to use HT for cache-friendly applications.
- Unlike the other FT and EP benchmarks, the best performance is obtained by using 81 threads instead of 128

# Threads	MOPS without HT	MOPS with HT
36	29194.51	28448.64
64	51415.60	42080.26
81	71627.49	46754.70
100	59387.02	46164.66
128	59781.11	52312.25
144	40925.23	24923.54
169	46653.48	36100.99
196	48003.63	36292.36
225	45660.17	48648.60
256	43382.45	45569.99

Table 4. NPB Block Tridiagonal results

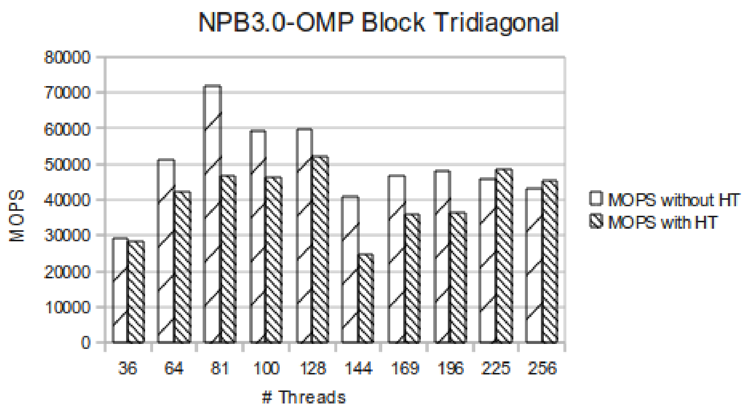

Fig. 6. NPB Block Tridiagonal

– When the threads are able to exploit all the cache locality the number of MOPS increases.

Conjugate Gradient: This kernel uses a Conjugate Gradient method to compute an approximation to the smallest eigenvalue of a large, sparse, unstructured matrix. It tests irregular long distance communications employing unstructured matrix vector multiplication. The size for the Class C problem of the CG benchmark is 150000.

The results (Table 5 and Figure 7) of this type of benchmark show that:

– 100 threads is the configuration with the highest MOPS rate.
– In the ideal configuration HT it is not advisable (1% loss aprox.).
– In the other benchmarks HT got better results when the system was overloaded, but in CG the results are better without HT.

# Threads	MOPS without HT	MOPS with HT
36	4592.96	5044.12
64	10165.17	10529.92
81	17070.85	16470.47
100	20865.44	20664.16
128	17628.43	19970.67
144	13841.09	5786.25
169	12870.31	6296.99
196	12114.52	6143.35
225	11523.90	5928.27
256	11402.77	5440.40

Table 5. NPB Conjugate Gradient results

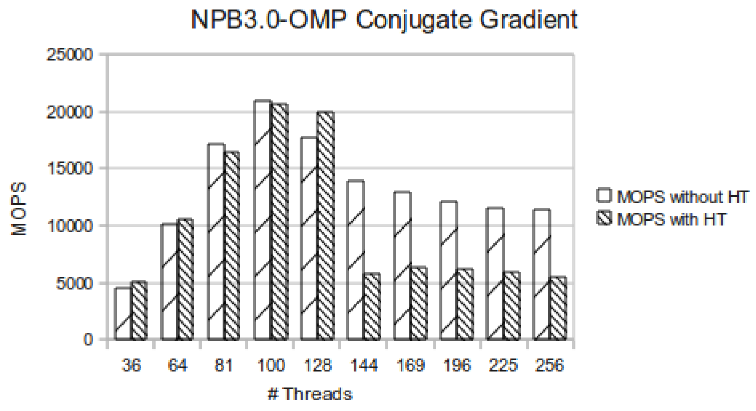

Fig. 7. NPB Conjugate Gradient

This kernel tests unstructured grid computations and communications by using a matrix with randomly generated locations of entries. The combination of computation an communication among threads implies that performance without HT is better in most of the cases.

4 Conclusions

This study empirically demonstrate and prove that the sense of the application will determine whether Hyper-Threading will accelerate or diminish performance on the HP Integrity SuperDome SX2000. Enabling Hyper-Threading or disabling it is not an exact science, it depends on so many factors that a bechmark study of the problem that is going to be addressed is highly recommended. Nevertheless, after running the benchmarks there are some conclusions and tips that could help system administrators and programmers regarding the use of Hyper-Threading technology on LUSITANIA. Hyper-Threading is recommended when:

– Threads or processes have to communicate with each other very often.
– An MPI library is used to communicate processes.
– The application doesn't use all the processing units leaving space for a different thread or process.
– Poor-optimized math libraries are used, if it is not optimized by the software then it can be optimized by the hardware.
– There are a lot of cache misses.

Hyper-Threading is not recommended when:

– The application is embarrassingly parallel, i.e. there is almost no communication between threads and processes.
– Intensive floating point operations exploit all the processor resources and there is no more room for a different operation.
– A cache-friendly application is executed.
– The math library is already optimized and parallelized.

References

1. D. Bailey, E. Barszcz, J. Barton, D. Browning, R. Carter, L. Dagum, R. Fatoohi, S. Fineberg, P. Frederickson, T. Lasinski, R. Schreiber, H. Simon, V. Venkatakrishnan, S. Weeratunga "The NAS Parallel Benchmarks" *RNR Technical Report RNR-94-007 March 1994*, (1994).
2. David Bailey, Tim Harris, William Saphir,Rob van der Wijngaart, Alex Woo, Maurice Yarrow "The NAS Parallel Benchmarks 2.0" *Report NAS-95-020*, (1995).
3. Jack J. Dongarra, Piotr Luszczek, Antoine Petitet "The LINPACK benchmark: Past, present, and future." *Concurrency and Computation: Practice and Experience* **15**, (2003).
4. H. Jin, M. Frumkin, J. Yan "The OpenMP Implementation of NAS Parallel Benchmarks and Its Performance" *NAS Technical Report NAS-99-011*, (1999).
5. Deborah T. Marr, Frank Binns, David L. Hill, Glenn Hinton, David A. Koufaty, J. Alan Miller, Michael Upton "Hyper-Threading Technology Architecture and Microarchitecture" *Intel Technology Journal*, **volume 6**, pp. 4-14, (2002).
6. Nicholas Mitchel, Larry Carter, Jeanne Ferrante, Dean Tullsen "ILP versus TLP on SMT" *Proceedings Supercomputing '99*, (1999).
7. Hewlett Packard "HP Super-Scalable Processor Chipset sx2000", (2007).
8. Hewlett Packard "ccNUMA White Paper", (2003).
9. Hewlett Packard "Dynamic logical processors for Hyper-Threading on HP-UX 11i v3", (2007).