Characterizing Application Performance Sensitivity to Resource Contention in Multicore Architectures

Haoqiang Jin, Robert Hood*, Johnny Chang*, Jahed Djomehri*, Dennis Jespersen, Kenichi Taylor†, Rupak Biswas, and Piyush Mehrotra

NAS Division, NASA Ames Research Center, Moffett Field, CA 94035-1000

{haoqiang.jin,robert.hood,johnny.chang,jahed.djomehri,dennis.jespersen, rupak.biswas,piyush.mehrotra}@nasa.gov, kenichi@sgi.com

Abstract

Contention for shared resources in the memory hierarchy can have a profound effect on the performance of applications running on high-end computers based on commodity multicore microprocessors. In this paper we describe a methodology, based on differential performance analysis, to quantify the effect of this contention on parallel applications. In particular, we characterize the contention for a specific shared resource by comparing runs that use different patterns of assigning processes to cores on multicore nodes. We demonstrate our approach with a subset of the HPCC benchmarks and the NAS Parallel Benchmarks running on high-end computing platforms that use four different quad-core microprocessors—Intel Clovertown, Intel Harpertown, AMD Barcelona, and Intel Nehalem-EP. The results help further our understanding of the requirements these codes place on their execution environments and also of each computer's ability to deliver performance.

1 Introduction

A recent trend in commodity microprocessor design is to put multiple computing cores on each chip. This approach allows the manufacturers to achieve high aggregate performance without the power demands of increased clock speeds. As transistor counts increase following Moore's Law, we expect to see core counts per chip rising. Designing the computational and data movement components of a microprocessor is an exercise in compromise as chip designers aim for good performance across a wide spectrum of applications while staying within a budget for transistors and power. Inevitably, some resources, such as L2 cache or memory bandwidth, end up being shared by multiple cores. Like the other components in the system, the design parameters for these resources, such as size and speed, are decided using a process of constrained optimization.

Such resource sharing can have a profound effect on the performance of applications running on some high-end computers. Systems such as the Cray XT5 [7], IBM BladeCenter [11], and SGI Altix ICE [17] take advantage of economies of scale by using commodity parts such as microprocessors. They typically have very high peak performance when compared to other machines in the same price range. One liability, however, is that contention for shared resources such as L2 cache or memory bandwidth may be so severe that the most economical way to run an application leaves half the cores idle [9]. In this paper we use differential performance analysis to quantify this contention effect for a collection of

*Employee of CSC, Inc. through contract no. NNA07CA29C with NASA Ames †Employee of Silicon Graphics, Inc.

benchmark applications – the HPCC and NAS Parallel Benchmarks. We compare runs that use different node configurations—patterns of assigning processes to cores in a node—to measure the performance penalty due to resource contention on individual shared components. Our target architectures include high-end computing platforms that use four different quad-core microprocessors—Intel's Clovertown, Harpertown, and Nehalem-EP, and AMD's Barcelona. The results help further our understanding of the requirements these codes place on their execution environments and the effect processor and system design decisions have on application performance.

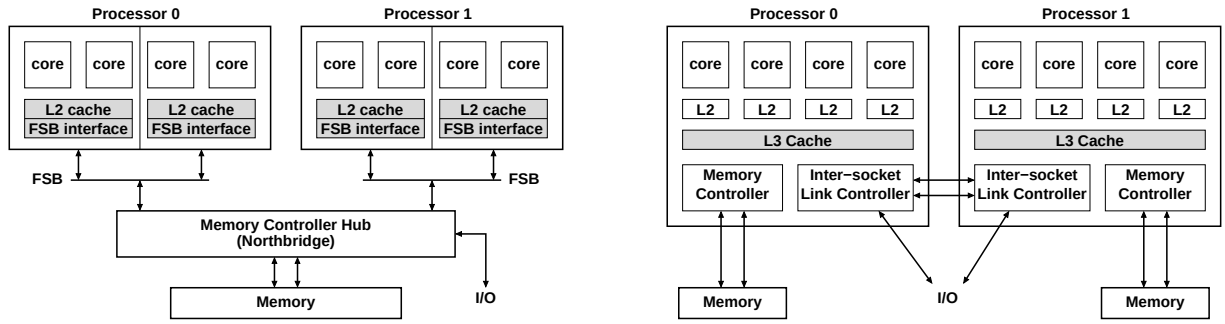
Although several previous studies have looked at resource contention, in particular, memory bandwidth contention in multicore systems [1, 2, 5, 6, 8, 12, 14, 16, 18], until this present work, none has broken this contention down to its individual constituents—contention for the shared cache, front-side bus, HyperTransport, QuickPath Interconnect, and contention for memory due to insufficient memory bandwidth. Understanding which resource contention causes the greatest performance penalty on various multicore architectures will be useful for chip and system designers in determining what incremental changes in hardware components will provide the biggest boost for application performance.

In the rest of the paper, we introduce a method to isolate contention for shared resources in multicore systems in Section 2 and summarize the experimental environment and methodology in Section 3. Section 4 is devoted to a detailed differential analysis of the performance of benchmark applications running on selected node configurations. We conclude in Section 5 and also discuss possible future work.

2 Isolating Resource Contention in Multicore Systems

As a first step in determining the effects of contention in the memory hierarchy of multicore systems, we need to understand how resources are shared. Figure 1 shows a high-level view of two architectural approaches for nodes using two quad-core chips. Figure 1(a) is an example of a Uniform Memory Access (UMA) design where an off-chip memory controller (Northbridge) interfaces to the memory. Examples of processor chips using this design are Intel's Xeon 5300 series (also called Clovertown) and Xeon 5400 series (also called Harpertown). In the case of these Intel chips, each processor has two L2 caches and no L3 cache, each L2 cache being shared by two cores. With this node design, access to external memory is through a front-side bus (FSB) shared by all four cores on the chip and a common memory controller shared by both chips. Inter-node communication is through the off-chip memory controller.

Figure 1(b) depicts a Non-Uniform Memory Access (NUMA) design in which the four cores access local memory through an on-chip memory controller. Access to remote shared memory on the node is through a high-speed inter-socket interconnect. Examples of this approach include nodes designed for AMD's Opteron 2300 series (Barcelona) and Intel's recently released Xeon 5500 series (Nehalem-EP) which use the HyperTransport3 (HT3) and the QuickPath Interconnect (QPI) respectively for the inter-socket link. In these two examples, each of the cores in the quad-core chips has an individual L2 cache but shares the single L3 cache on the chip. Inter-node communication is available to each processor through its inter-socket link controller (HT3 or QPI).



(a) *UMA-based system* (b) *NUMA-based system*

Figure 1: *Node architectures*

2.1 Node Configurations and Labeling

To characterize the impact of contention for shared resources, we have chosen eight different node configurations for codes to run. These configurations vary in how processes (MPI ranks) are mapped to cores in a node. Then, we compare the performance on two configurations, one of which has the increased sharing of a resource compared to the other, with other factors remaining the same. Using a form of differential performance analysis, we calculate the difference in performance on the two configurations to assess the penalty for the increased sharing of the resource. The ultimate goal is to characterize the effect of increased sharing on the overall performance of the code. We define a code's contention penalty from the configuration C_1, where a resource is at a base level of sharing, to the configuration C_2, where the sharing is increased, as:

$$P_{C_1 \rightarrow C_2} = (t_{C_2} - t_{C_1})/t_{C_1}$$

where t_{C_1} and t_{C_2} are the runtimes of the code for the two configurations. In effect, this is a measure of how much more expensive it is to run the more highly shared configuration. If the penalty is negative, it means the configuration with increased sharing performed better.

For systems with nodes similar to the UMA-based design in Figure 1(a), the node configurations we are interested in can be described using three orthogonal dimensions: the number of sockets used per node (S), the number of shared caches used per socket (C), and the numbers of active processes (e.g. MPI ranks) per shared cache (P). Mapping these three dimensions to Cartesian coordinates gives rise to the lattice cube shown in Figure 2, with eight unique node configurations. As labeled in the diagram, a 3-digit mnemonic, SCP, can be used to designate the configurations. For example, configuration 212 uses two sockets, one shared cache per socket, and two cores per shared cache, for a total of four active cores. This can be depicted as where the dark squares represent mapped cores, the light squares represent idle cores, and

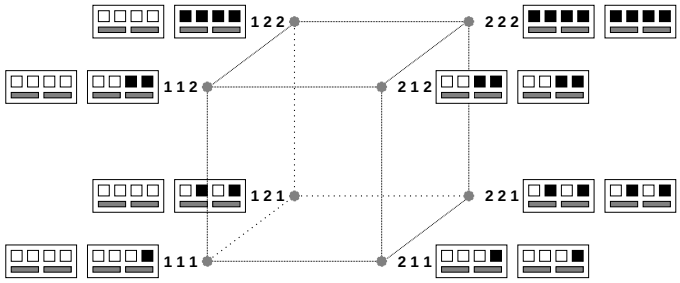


Figure 2: *Node configuration lattice for UMA-based systems*

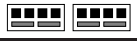
the gray boxes indicate shared L2 caches. Note that the configurations in the left hand plane in Figure 2 all use only one socket while those in the right hand plane use two sockets. Similarly the configurations in the front plane use only one cache per socket while those in the back use two caches per socket, and the bottom plane configurations have only one core per cache as opposed to the two cores per cache used by configurations in the top plane.

We must make some accommodations to use the same numbering scheme for systems with nodes similar to the NUMA-based design in Figure 1(b). In particular, processors in such nodes have a single L3 cache rather than a pair of L2's. Strictly speaking, the Cartesian SCP space should be $\{1,2\} \times \{1\} \times \{1,2,3,4\}$, and a more accurate depiction of the 212 configuration would be . However, for the sake of simplicity in the diagrams and the discussion, we will use the UMA numbering and diagrams for NUMA architectures as well. In particular, we will use 122 and 222 to designate NUMA configurations 114 and 214. Also, it should be noted that when executed on a NUMA-based system, configuration 121 should have the same performance characteristics as 112, as should 221 and 212.

2.2 Contention Groups

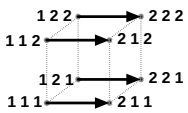
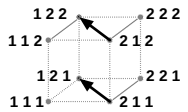
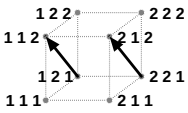
To isolate the effects of contention we need to choose pairs of node configurations such that the level of sharing of a specified resource is increased from one configuration to the other, while keeping other factors the same. Table 1 presents the number of processes in a node that share the three types of resources in our generalized UMA and NUMA architectures for each of the eight configurations. For UMA-based systems there are three resources of interest: UMA memory bandwidth as provided by the off-chip memory controller (namely UMA-BW), the front-side bus (FSB), and the L2 cache. On our generalized NUMA-based systems, there are two types of shared resources to investigate: NUMA memory bandwidth as provided by the inter-socket interconnect (NUMA-BW) and the local memory bandwidth supported by the L3 cache and the on-chip memory controller (L3+MC).

Table 1: *Resource contention for node configurations*

Number of Processes Sharing a Resource						
Node Configuration	UMA-based systems e.g. Clovertown, Harpertown			NUMA-based systems e.g. Barcelona, Nehalem		
	L2	FSB	UMA-BW	L2	L3+MC	NUMA-BW
111	1	1	1	1	1	1
211	1	1	2	1	1	2
121	1	2	2	1	2	2
112	2	2	2	1	2	2
221	1	2	4	1	2	4
212	2	2	4	1	2	4
122	2	4	4	1	4	4
222 	2	4	8	1	4	8

An important point in this differential performance analysis approach is to compare two configurations that isolate the increased sharing and keep other factors unchanged. Table 2 lists the pairs of configurations that can be compared to show isolated contention effects on the three types of shared resources listed in Table 1. The table shows the difference in the number of processes sharing the resources for each of selected configuration pairs in the contention groups. A value of zero indicates no increase in sharing for the given resource. For example, we can assess the penalty of sharing the off-chip memory controller on UMA-based systems by comparing the performance of node configuration 111 to that of 211. In

Table 2: *Configuration pairings used to evaluate contention*

Node Configuration Combination		Shared Resource						
		UMA node			NUMA node			Comm. Effect
		L2	FSB	UMA-BW	L2	L3+MC	NUMA-BW	
	111→211	0	0	1	0	0	1	Intra- & inter-node
	121→221	0	0	2	0	0	2	
	112→212	0	0	2	0	0	2	
	122→222	0	0	4	0	0	4	
	211→121	0	1	0	0	1	0	Intra-node only
	212→122	0	2	0	0	2	0	
	121→112	1	0	0	0	0	0	None
	221→212	1	0	0	0	0	0	

the former, only one core in the node is active, while in the second each socket has an active core and they share the bandwidth provided by the memory controller. Thus $P_{111\rightarrow211}$ can be used to understand the effect of sharing memory bandwidth. These groups will be used in later sections to understand the effect of sharing resources on the performance of the benchmark applications.

2.3 Effect of Communication

While our current methodology does not isolate resource contention due solely to communication, message passing does factor into the penalties we calculate. Therefore, in order to minimize the changes in message passing that are due to configuration differences, we use a fixed number of MPI processes on each of the eight configurations even if it results in an increase in the number of nodes used for the run. Thus, for example, an execution measuring configuration 111 will use twice as many nodes as one measuring 211. Note that while this approach keeps the message pattern constant, it can change the load on the shared resources due to communication. That is, even though the pattern of messages between MPI ranks remains the same, the communication pattern (i.e., the number of intra-socket, inter-socket and inter-node messages) changes have an effect on the overall performance of the code. This effect is indicated in the last column of Table 2.

Communication has the most effect on the contention group focused on memory bandwidth (the top four rows of Table 2). Here, the number of nodes utilized for the runs changes when comparing configurations, e.g., 111 versus 211, and thus the inter-node and intra-node communication pattern changes. For each rank, some off-node messages transform into on-node messages. The exact changes in the message pattern and how they affect the contention for memory bandwidth—and thus the runtime of the code—is dependent on the overall message passing pattern. For example, consider an n-rank job when assessing

the combination 111→211. For a 1D nearest-neighbor ring messaging pattern, the communication for $n/2$ ranks would change from inter-node to intra-node when comparing the two configurations. On the other hand, consider the situation of a code using an all-to-all messaging pattern. For the configuration 111, each rank will be sending out $n-1$ inter-node messages while for the configuration 211, each rank will send out $n-2$ inter-node message and one intra-node message. Thus, for each node the total number of inter-node message would increase from $n-1$ to $2n-4$ while the total number of intra-node messages would go up from 0 to 2. However the maximum distance that a message has to travel would be reduced since the total number of nodes is halved.

It is important to note that, if there is sufficient local memory to meet the requirements of each MPI rank, contention for NUMA-bandwidth should mostly come from communications. In that case, the memory accesses needed for the computation will be local, and thus avoid the HT3 or QPI links.

The situation for FSB (for UMA nodes) and L3+MC (NUMA nodes) contention groups, as specified in the middle two rows of Table 2, is different. Here, the inter-node communication patterns are the same for the configurations in each selected pair. But inter-socket messages in the less-shared configuration are transformed into intra-socket messages, increasing the contention for FSB or L3+MC as the case may be. This may result in changes in the total communication time affecting the overall performance of the code. The configurations comparing L2 cache contention (as shown in the last two rows of Table 2) have no change in communication pattern, since half the ranks are "moved" from one core to another core on the same socket, e.g., when comparing configuration 121 to 112. Messages which were independent in the former configuration would share L2 cache in the second, the exact effect that we would like to characterize.

3 Experimental Approach

Our experiments to test resource contentions were conducted on four parallel systems—equipped with four types of quad-core processors—using a mix of kernel benchmarks, and pseudo-application benchmarks. In this section, we first summarize the benchmarks and the hardware systems used in the experiments, and then describe the experimental methodology, including the use of hardware counters.

3.1 Benchmark Codes

We selected three kernel benchmarks from the the HPC Challenge (HPCC) benchmark suite [10] for our experiments. The DGEMM benchmark measures optimum floating-point performance; the STREAM benchmark component tests memory bandwidth by doing simple operations on very long vectors; and the PTRANS benchmark tests collective communications from transposing a matrix.

The NAS Parallel Benchmarks (NPB) [3] are well-known problems for testing the capabilities of parallel computers and parallelization tools. They were derived from computational fluid dynamics (CFD) codes to mimic the computational core of several numerical methods, and they reproduce much of the data movement and computation found in full CFD

codes. For this study, we selected three kernels (MG, CG, and FT), and three simulated applications (BT, LU and SP) from the NPB3.3 suite [4].

Briefly, MG (multi-grid) tests long- and short-distance communication and is memory-bandwidth intensive. CG (conjugate gradient) tests irregular memory access and communication latency. FT (fast Fourier transform) tests all-to-all network communication. The three application benchmarks (BT: block-tridiagonal solver, LU: seven band block-diagonal solver, SP: scalar penta-diagonal solver) test nearest neighbor communication. Furthermore, BT is numerically intensive, LU is sensitive to cache size, and SP has larger a communication-to-computation ratio than the other two.

3.2 Experimental Testbeds

We utilized four different architectures in this study. The first is an SGI Altix ICE 8200 cluster with each node containing two quad-core Intel Xeon E5355 processors (Clovertown), interconnected with InfiniBand in a hypercube topology. The second, also an SGI Altix ICE system, comprises InfiniBand-connected nodes based on two quad-core Intel Xeon X5472 processors (Harpertown). Throughout the paper, we refer to the Clovertown-based system as "*Clovertown-Altix*" and the Harpertown-based system as "*Harpertown-Altix*". The main processor and architectural characteristics of these systems are summarized in Table 3. Compared to Clovertown, Harpertown has a faster clock speed, larger L2 cache, faster front-side bus (FSB) speed, and more memory bandwidth. Each Clovertown and Harpertown processor has a pair of L2 caches that are each shared by two cores; the two processor types also have a similar bus-based architecture. Communication between the two processors in a node has to go through the FSB and a shared memory controller (Northbridge). Accessing the 8GB memory in one node from each core is relatively uniform.

Table 3: *Architectural and execution environment summaries of the four parallel systems*

Machine	**Clovertown-Altix**	**Harpertown-Altix**	**Barcelona-Cluster**	**Nehalem-Altix**
Model	SGI Altix ICE 8200	SGI Altix ICE 8200EX	Custom-made Opteron Cluster	SGI Altix ICE 8200EX
Number of nodes	512	>1024	128	128
Cores per node	8	8	8	8
Processor type	Intel Xeon X5355 (Clovertown)	Intel Xeon E5472 (Harpertown)	AMD Opteron 2354 (Barcelona)	Intel Xeon X5570 (Nehalem-EP)
Processor speed	2.66GHz	3.0GHz	2.2GHz	2.93GHz
L2 cache / cores sharing	4MB / 2	6MB / 2	512KB / 1	256KB / 1
L3 cache / cores sharing	–	–	2MB / 4	8MB / 4
Memory per node	8GB	8GB	16GB	48GB
FSB speed	1333MHz	1600MHz	–	–
Memory type	FB-DDR2	FB-DDR2	DDR2	DDR3
Memory bandwidth	21.3GB/s read 10.7GB/s write	25.6GB/s read 12.8GB/s write	10.7GB/s read/write	25.6GB/s read/write
Intersocket link	Northbridge (NB)	Northbridge (NB)	HyperTransport3 (HT3)	QuickPath Interconnect (QPI)
HT3/QPI bandwidth	–	–	2×16GB/s	25.6GB/s
Compiler	Intel-10.1/11.0	Intel-10.1/11.0	GCC-4.1/PGI-8.0	Intel-11.0
MPI library	mpt-1.19	mpt-1.19/1.23	mvapich2-1.2	mpt-1.23

The third parallel system is a custom-made AMD Opteron cluster connected with a Mellanox InfiniBand switch. Each of the nodes contains two quad-core AMD Opteron 2354

processors (Barcelona). We refer to this system as "*Barcelona-Cluster*" in this paper. The four cores in an Opteron processor share a 2MB L3 cache, and each core has its own 512KB local L2 cache. The cache size per core is relatively small compared to the Intel processors. Each Opteron processor has an on-chip memory controller that connects directly to its local memory. Additionally, the two processors on a node are connected via HyperTransport3 (HT3), allowing for non-uniform memory access (NUMA) to remote memory on the node. There is a total of 16GB of memory on each node. The advantage of this design compared to the two UMA-node-based systems above is that there is less memory contention in the system and it is scalable.

The fourth parallel system, a newly released SGI Altix 8200EX system based on the quad-core Intel Xeon X5570 processor (Nehalem-EP), is referred to as "*Nehalem-Altix*" in this paper. Nehalem is a complete revamp of Intel's previous generation of microprocessor architecture. Each Nehalem processor contains four cores with simultaneous multithreading or hyperthreading capability, which enables two threads per core. An on-chip memory controller connects directly to local DDR3 memory through three channels. Communication between processors in a node and to the I/O hub is through the Intel QuickPath Interconnect (QPI), which replaces the bus-based design of older Intel processors. Each processor contains an 8MB L3 cache shared by the processor's four cores. Architecturally, the Nehalem processor is very similar to the AMD Opteron processor; however, it has much improved processor speed, cache size, and memory bandwidth. Our test system contains 48GB of memory per node. In order to simplify the analysis, we did not use hyperthreading or the TurboBoost capabilities of the Nehalem processor.

3.3 Methodology

Our experiments start with low-level microbenchmarks from HPCC to establish baseline characteristics of resource contention on different multicore systems using differential performance analysis. We then apply this technique to NPB results, and use hardware counter measurements, when available, to better understand the results and to support our interpretations.

For the differential performance analysis, we measure the performance of the above codes on a fixed number of processes for all eight configurations described in Section 2. The number of processes is chosen such that a given problem can fit into the memory on a node, and both intra-node and inter-node communications can be tested. In the actual runs, we used 16 processes for HPCC and NPB. Due to different labeling of core locations on the Intel and AMD processors, we had to carefully choose the list of core id's used for each node configuration, avoiding core 0 when possible to minimize the effect of kernel daemons.

Modern microprocessors such as Xeon and Opteron include a rich set of hardware counters for performance monitoring and measurement. We used the PerfSuite performance measurement tool [13] from NCSA installed on the Clovertown-Altix and Harpertown-Altix. PerfSuite relies on the PAPI profiling library [15] for accessing hardware counters. To reduce perturbation from calls to underlying MPI libraries, SGI has developed a tool called MPInside that reports MPI statistics and works together with PerfSuite to suppress hardware counter increments while inside MPI calls in a program. In order to generate the penalty values from hardware counters, we use a formula similar to the one used for the perfor-

mance penalty as described in Section 2. We only present data from the Clovertown-Altix and Harpertown-Altix since these tools were not available on the other two systems.

4 Experimental Results and Contention Analysis

In this section we present the results of our differential performance analysis of shared resource contention. We will first present the results from the HPCC and NAS Parallel benchmarks obtained on UMA-based systems and then the results from runs on the NUMA-based systems. We then correlate the penalties based on timing measurements to penalties based on hardware counter data for a subset of the experimental runs.

Each of the run configurations was repeated at least five times to reduce noise from timing variations; we then used median values to calculate the penalties. The medians of the rates (Gflops/sec or GBytes/sec) reported by the individual benchmarks over the multiple runs are given in Tables 6 and 7 of the Appendix. The inverses of the rates shown in the Appendix are proportional to timing; those inverses are used to calculate performance penalties presented in this section for each of the contention groups described in Section 2.

4.1 Results on UMA Systems

Table 4 presents the percentage performance penalties for the various node configurations using the HPCC benchmarks and NPBs on the two UMA test systems, the Clovertown-Altix and the Harpertown-Altix that were described in the previous section. Analyzing the results for HPCC's DGEMM, we see only small performance penalties as resource contention is increased, indicating that both platforms have sized the shared resources sufficiently for the floating-point intensive benchmark. One reason for this is that DGEMM uses the BLAS library routines, which have been extensively optimized for cache and memory usage on each of the systems. The results do show a small but consistent FSB penalty on the Clovertown-Altix and the Harpertown-Altix when going from 2 cores to 4 cores using the bus—(211→121) and (212→122).

The STREAM results show a considerable dependence on the FSB of UMA systems. This is not surprising because the purpose of the benchmark is to put extreme pressure on the memory bus. The 4-core configurations (212→122) show a larger impact from contention than the 2-core configurations (211→121). Compared to STREAM Copy, there is a small but noticeable increase in FSB contention for STREAM Triad as a result of additional reads. It is interesting to note that the Harpertown-Altix exhibits much more FSB contention than the Clovertown-Altix. However, the Clovertown-Altix shows an additional penalty for UMA-bandwidth (Northbridge) contention. Relatively small UMA-bandwidth penalties on the Harpertown-Altix indicate that this system has adequate UMA-bandwidth to support the FSB traffic.

There is fairly little L2 contention in the STREAM results. Given that there is no reuse of cached values in the benchmark, the penalty that we see is likely due to contention for the interface between the L2 and the FSB (the resource labeled "FSB Interface" in Figure 1(a)).

In contrast, the PTRANS results on the two UMA systems show a pattern that is in-between the DGEMM and STREAM benchmarks. The FSB contention from (212→122)

Table 4: *Percentage performance penalty observed for the contention groups on UMA-based systems*

Contention	(C1→C2)	Clovertown-Altix									
		DGEMM	ST_Copy	ST_Triad	PTRANS	BT	CG	FT	LU	MG	SP
UMA-BW (Northbridge)	(111→211)	0%	14%	24%	11%	4%	7%	10%	3%	9%	11%
	(121→221)	1%	25%	24%	7%	8%	17%	18%	7%	18%	19%
	(112→212)	1%	22%	22%	8%	9%	18%	22%	6%	20%	20%
	(122→222)	2%	24%	24%	21%	16%	32%	11%	12%	26%	23%
FSB	(211→121)	1%	29%	44%	1%	5%	7%	7%	5%	11%	17%
	(212→122)	2%	56%	56%	26%	15%	26%	19%	21%	51%	48%
L2	(121→112)	1%	5%	3%	0%	-1%	1%	3%	16%	18%	9%
	(221→212)	1%	2%	1%	1%	0%	1%	6%	15%	20%	10%

Contention	(C1→C2)	Harpertown-Altix									
		DGEMM	ST_Copy	ST_Triad	PTRANS	BT	CG	FT	LU	MG	SP
UMA-BW (Northbridge)	(111→211)	0%	5%	3%	-4%	2%	4%	-2%	2%	7%	9%
	(121→221)	0%	3%	1%	5%	3%	4%	20%	4%	9%	16%
	(112→212)	0%	3%	1%	4%	4%	5%	16%	3%	9%	12%
	(122→222)	0%	3%	-2%	9%	3%	1%	-3%	4%	6%	9%
FSB	(211→121)	1%	69%	81%	28%	16%	39%	12%	15%	48%	38%
	(212→122)	3%	81%	88%	44%	37%	65%	22%	27%	65%	53%
L2	(121→112)	1%	5%	5%	-1%	8%	7%	14%	6%	22%	29%
	(221→212)	1%	6%	5%	-1%	9%	7%	10%	5%	23%	24%

is about half of that for STREAM. On the other hand, the (211→121) penalty on the Clovertown-Altix is near zero. There are noticeable penalties in the UMA-bandwidth contention group, although no obvious pattern can be discerned. Note that PTRANS is dominated by all-to-all communication in the global matrix transpose. However, when determining UMA-bandwidth contention, our method does not distinguish between contributions from local memory access and remote communication. But, we speculate that, in the case of PTRANS, most of the UMA-bandwidth contention is a result of communication requirements.

In contrast to the HPCC microbenchmarks, the NPB results illustrate how more realistic applications might respond to resource contention. On the Harpertown-Altix, except for two points for FT and SP, all six benchmarks show a relatively small penalty for UMA-bandwidth contention, indicating that adequate memory bandwidth is available on the system to keep up with the increase in core counts. The trend for FT should correlate to that for a benchmark that tests both intra-node and inter-node communications via large collective MPI communication. For FT and SP on the Harpertown-Altix, we observe considerable contention increase in going from one socket to two sockets (121→221 and 112→212), indicating that communication between sockets within a node causes more performance penalty than communication going out of a node. However, negative penalties shown in the other two configuration pairs (111→211 and 122→222) for FT mean that intra-node communication is less costly than inter-node communication for these cases. Contention for UMA-bandwidth on the Clovertown-Altix increases for most cases as compared to the Harpertown-Altix, indicating that there is less memory bandwidth available on the Clovertown-based node. We observe a gradual increase in performance penalty going to a configuration with more active cores on the Clovertown-Altix.

On the Harpertown-Altix, the largest performance penalties come from FSB contention, for example more than 50% for memory-bandwidth sensitive benchmarks (MG and SP), and

∼30% for numerically intensive benchmarks (BT and LU). Irregular memory access in CG also stresses the FSB. Again, configurations with more active cores (212→122) put much more pressure on FSB than those with less active cores (211→121). In general, the FSB contention on the Clovertown-Altix is smaller than on the Harpertown-Altix. The NPB results are within the upper bounds set by the STREAM benchmarks.

We observe sizable L2 cache contention on both the Clovertown-Altix and the Harpertown-Altix. In particular, except for LU, penalties associated with L2 cache contention are larger on the Harpertown-Altix than on the Clovertown-Altix, even though the former has a larger L2 cache. For most cases, larger L2 cache size differences in the two tested configurations (3MB on a Harpertown and 2MB on a Clovertown) result in larger performance penalty. On the other hand, LU is sensitive to the actual cache size and needs more than 2MB of cache per rank to run effectively in the current case. Thus, there is a larger difference between running with 4MB and 2MB (on a Clovertown) than there is when changing from 6MB to 3MB (on a Harpertown). Note also that MG and SP, which require more memory bandwidth, also exhibit larger L2 cache contention.

4.2 Results on NUMA Systems

Table 5 presents the percentage performance penalties for the various node configurations using the HPCC benchmarks and NPBs on the two NUMA test systems, the Barcelona-Cluster and the Nehalem-Altix. As expected, we see no penalty for the L2 contention group for all benchmarks. This is due to the fact that configuration pairs (121→112) and (221→212) have identical sharing characteristics on the two NUMA-based systems, where L2 cache is private to each processor core. It is also not surprising that DGEMM exhibits insensitivity to resource contention on these systems, just as was observed on the UMA-based systems.

Table 5: *Percentage performance penalty observed for the contention groups on NUMA-based systems*

		Barcelona-Cluster									
Contention	(C1→C2)	DGEMM	ST_Copy	ST_Triad	PTRANS	BT	CG	FT	LU	MG	SP
NUMA-BW (HT3)	(111→211)	0%	3%	3%	18%	0%	5%	4%	-4%	2%	2%
	(121→221)	0%	1%	2%	10%	1%	4%	3%	-1%	3%	2%
	(112→212)	0%	1%	2%	8%	1%	4%	3%	0%	3%	2%
	(122→222)	0%	6%	7%	2%	1%	6%	3%	0%	1%	1%
L3+MC	(211→121)	0%	29%	22%	6%	3%	1%	11%	17%	29%	19%
	(212→122)	1%	76%	69%	21%	14%	7%	26%	88%	57%	54%
L2	(121→112)	0%	0%	0%	1%	0%	0%	0%	0%	0%	0%
	(221→212)	0%	0%	0%	-1%	0%	0%	0%	0%	0%	0%
		Nehalem-Altix									
Contention	(C1→C2)	DGEMM	ST_Copy	ST_Triad	PTRANS	BT	CG	FT	LU	MG	SP
NUMA-BW (QPI)	(111→211)	0%	1%	1%	4%	0%	-1%	-17%	0%	-2%	2%
	(121→221)	0%	2%	2%	43%	3%	2%	13%	0%	4%	21%
	(112→212)	0%	2%	2%	41%	0%	2%	13%	0%	4%	22%
	(122→222)	1%	1%	0%	-13%	-2%	5%	7%	0%	0%	-9%
L3+MC	(211→121)	0%	18%	57%	7%	5%	7%	6%	6%	34%	38%
	(212→122)	1%	85%	107%	10%	16%	32%	38%	27%	116%	60%
L2	(121→112)	0%	0%	0%	1%	1%	0%	0%	0%	0%	0%
	(221→212)	0%	0%	0%	0%	-2%	0%	0%	0%	0%	1%

In contrast to DGEMM, the STREAM results present a different picture. There are substantial penalties in the middle group (211→121 and 212→122), a measure of contention on the shared L3 cache and memory controller. In particular on the Nehalem-Altix, (212→122) shows a penalty of more than 100% for STREAM Triad, meaning that the aggregate bandwidth seen by all active cores actually went *down* when the number of cores used in a node went from four to eight. It seems that there may be a threshold for simultaneous memory access beyond which accesses interfere with each other to such an extent that aggregate performance decreases. The contention for L3+MC on the Barcelona-Cluster, although smaller, is roughly proportional to the number of active cores involved. Except for the (122→222) pair on the Barcelona-Cluster, we observe only marginal penalties (no more than 3%) in the NUMA-bandwidth contention for STREAM on both systems through either HyperTransport3 (HT3) or QuickPath Interconnect (QPI). This is consistent with the fact that both Barcelona and Nehalem processors have on-chip memory controllers dedicated to accessing local memory and that STREAM benchmarks do not involve any cross-processor communication for remote data.

It is worthwhile to point out that during our first runs on the Barcelona-Cluster we observed substantial HT3 contention for STREAM (more than 30% in 122→222). On investigation, we found that the NUMA support was disabled in the operating system setup. This resulted in the OS using the two sockets on the node as if they had uniform access to both local and remote memory on the node. The end result was that the bandwidth available to each core in a fully populated configuration (222) was much less than in a half-populated configuration (122) and the memory traffic (loads and stores) crossed over the NUMA link (HT3) in a node. With a NUMA-enabled kernel installed on the system, we no longer see such a contention on the HT3 link.

The communication-intensive PTRANS benchmark demonstrates sizable contention on the NUMA-bandwidth for both systems. On the Barcelona-Cluster, configurations involving more off-node communication through InfiniBand performed better than those involving more intra-node communication through HT3. The results for the Nehalem-Altix show a different pattern. Penalties increase dramatically from 2-core configurations (121 and 112) to 4-core configurations (221 and 212), meaning that pressure on QPI from intra-node communication outweighed the performance gain from shorter inter-node communication distance for the latter cases. However, a negative penalty for (122→222) indicates that intra-node communication for the fully populated (222) configuration was actually more efficient than off-node communication for the half-populated (122) configuration. Thus, contention for NUMA-bandwidth is very sensitive to the balance of intra-node and inter-node communications. PTRANS shows some contention in the L3+MC data on both systems, but the effect is much less than that for STREAM.

The NPB results show very small NUMA-bandwidth contention on the Barcelona-Cluster. Only CG (5%) and FT (4%) show an HT3 penalty larger than 3%. On the Nehalem-Altix, we see a demand for the QPI traffic in both FT and SP as a result of larger MPI communication volume. FT shows a negative performance penalty in going from 111 to 211. On Nehalem-based SGI Altix 8200EX, there is one extra InfiniBand switch hop going from 8 nodes to 16 nodes, which could substantially increase the all-to-all MPI communication cost in the 111 configuration for FT. A smaller or even negative penalty is also observed in the (122→222) group for both FT and SP. In fact, the performance pattern is very similar

to that for PTRANS and to the results observed on the Harpertown-Altix where a similar network is used. Other benchmarks (BT, CG, LU, MG) show very small demand on QPI traffic.

On both the Barcelona-Cluster and the Nehalem-Altix, the increased requirement for L3 cache and local memory bandwidth is visible as active cores increase, especially for LU on the Barcelona-Cluster and MG and SP on both systems. It is also surprising that the (212→122) result for MG shows a penalty of more than 100% on the Nehalem-Altix, as seen earlier for STREAM. The much larger (212→122) penalty observed for the LU benchmark on the Barcelona-Cluster can be attributed to the cache-size sensitive nature of this benchmark. LU performs much better on systems with larger L3 cache processors (8MB for Nehalem versus 2MB for Barcelona). Overall, local memory bandwidth is the major source of contention on the NUMA-based systems.

4.3 Hardware Counters

In order to aid our understanding of the contention analysis, we collected hardware performance counters for the NPBs for each of the eight configurations on the UMA systems – the Clovertown-Altix and the Harpertown-Altix. (Unfortunately, the infrastructure was not available on the two NUMA systems for us to collect similar data.) A quick review of this data indicates that all eight configurations have similar counts from a number of counters, including Floating Point Operations, TLB misses, and Instructions Graduated. Hence, we have not included the data from these counters here. Notable counters that do have direct impact on the analysis of contention results are Level 2 Total Cache Misses (L2_TCM) and Stalls on Any Resource (RES_STL).

In Figure 3 we compare the contention penalty function of Section 2 applied to both counter data and timing data. In the first two bands of Figure 3, we graph the percentage penalty values derived from hardware counters L2_TCM and RES_STL, respectively. In the third band we show an analogous graph that uses the time penalties of Table 4. In the top band of Figure 3, we see a strong correlation of L2_TCM with the L2 contention group time penalties of the bottom band. For example, the large L2 penalty for LU on the Clovertown-Altix and for MG and SP on both systems are clearly visible in the L2 cache miss penalties for (121→112) and (221→212) in the top band. Most of the other configurations show very small influence from L2 cache misses except for MG (in 212→122) and FT (in 122→222). Negative values indicate that FT had fewer L2 total cache misses in the fully populated configuration (222) than in other configurations. This result correlates well with the smaller penalty observed in the similar configurations in the third band of Figure 3. Note that in going from configuration 122 to 222 we are transforming some inter-node communication to intra-node communication as discussed in Section 2.3. The SGI message passing library used for the test cases optimizes intra-node communication by using shared memory on the node. Thus, one possible explanation for the negative penalties is that the direct memory access for intra-node communication results in far less L2 cache misses than when dealing with inter-node communication.

Resource stalls (RES_STL) can come from a number of sources, including L2 cache misses and loads/stores stalls. Penalties computed using this counter (shown in the middle band of Figure 3) correlate very well with the actual application performance (the bottom band of

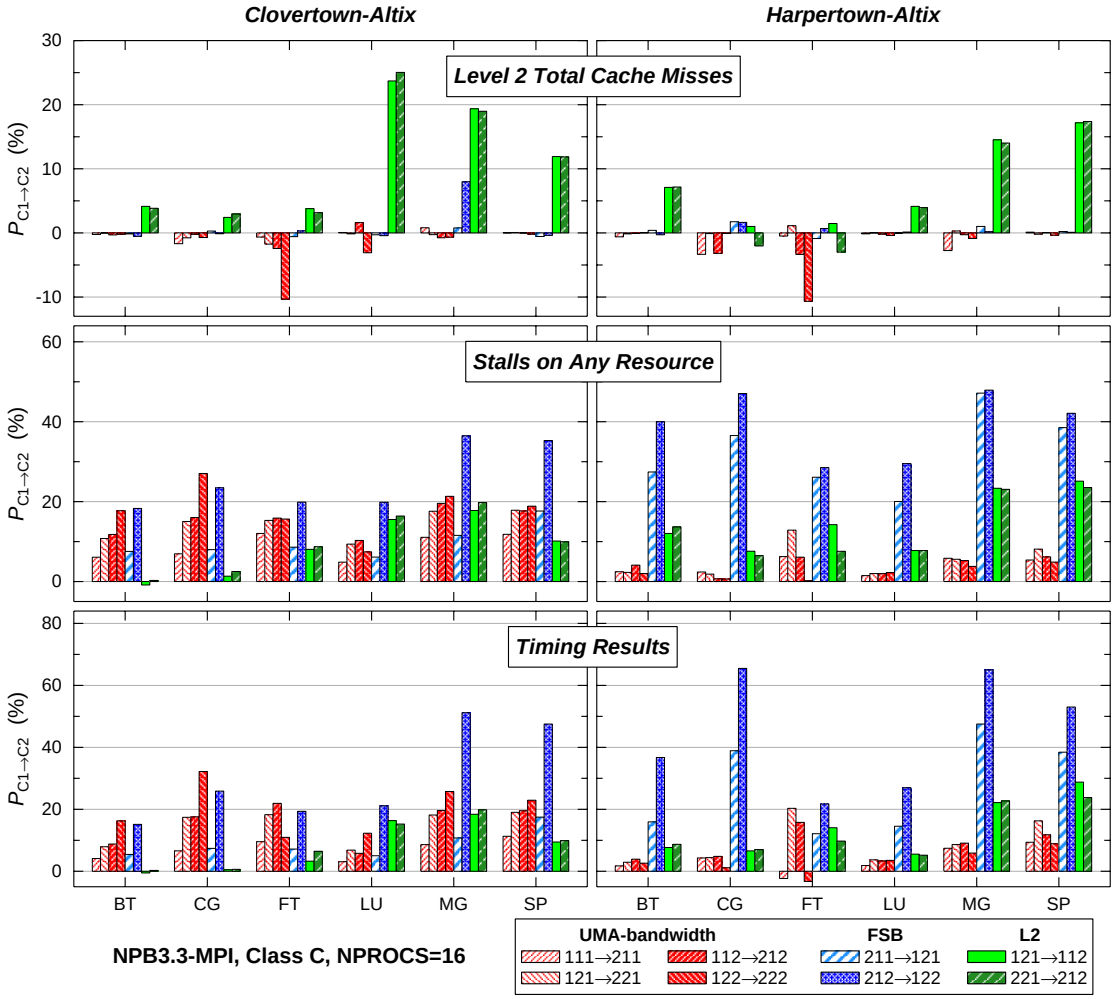


Figure 3: *Percentage penalty derived from hardware counters for the NPBs*

Figure 3). The dominant penalty (∼40%) from RES_STL is a result of the FSB contention for all benchmarks. On the Clovertown-Altix, the contention from UMA bandwidth contributes nearly 20% to the stall cycles. Additional L2 cache misses clearly increase the stall cycles as well, for MG, LU, and SP.

4.4 Discussion

Analysis of the experimental data allows us to make note of some general trends regarding both the architectures and benchmarking applications we used. For example, we can see general trends in Table 4 that illustrate how resource contention was reduced in going from Clovertown-based nodes to Harpertown-based ones. Clearly the UMA-bandwidth (Northbridge) improvement had a major impact on our benchmarks. The impact of the larger L2 caches of the later chips is also noticeable. These changes, which significantly improved the absolute performance (see Appendix), resulted with the FSB becoming more of a performance bottleneck.

The transition from UMA-based nodes to NUMA-based nodes results in the contention moving to the L3 cache and the local memory controller. With the exception of codes that

do a lot of communication, there seems to be relatively little stress on the intersocket links (HT3 and QPI). This is not surprising as nearly all of the non-message-passing memory accesses made in our benchmarks are local.

While our method does not allow us to directly separate the effect of L3 size from contention for the local memory controller on the NUMA systems, we can use the STREAM benchmarks to make some observations. In particular, as they have much less dependence on cache, contention uncovered by those benchmarks is mostly due to limitations in getting data through the local memory controller. While in absolute terms the memory bandwidth achievable by a code on Nehalem is vastly superior to that on Harpertown (see Table 6 in the Appendix), it is nonetheless still the major bottleneck for some codes (e.g., MG).

From the perspective of applications, optimizing both cache and memory usage is the key to reducing resource contention (as illustrated by the well-tuned DGEMM). In general, improved memory bandwidth is still the most desired performance factor for many applications.

5 Conclusions

Contention for shared resources in the memory hierarchy of multicore processors can have a profound effect on the performance of applications running on high-end computers. In this paper we introduce a way of using differential performance analysis to quantify this contention effect for a collection of parallel benchmarks and applications. In particular, by comparing runs that use different patterns of assigning processes to cores, we have characterized the contention for L2 cache, the memory bus for a socket (FSB if UMA and L3+MC for NUMA), and the memory bus for multiple sockets (memory controller on UMA and HT3/QPI on NUMA) on high-end computing platforms that use four different quad-core microprocessors—Intel Clovertown, Intel Harpertown, AMD Barcelona, and Intel Nehalem-EP. In our experiments we ran the HPCC benchmarks and the NAS Parallel benchmarks. As part of the experimental process, we also collected hardware counters for some of the runs so that we could validate our findings.

In general terms, the dominant contention factor we observed was the bandwidth available on the memory channels for a single socket—i.e., FSB on Harpertown and Clovertown, and L3+MC on Barcelona and Nehalem. Furthermore it was surprising to see greater than 100% L3+MC contention penalty in some cases (e.g., STREAM and MG) on the Nehalem processor. Some L2 cache contention was observed on Clovertown and Harpertown. When looking at the results for the NUMA-style shared memory nodes, we saw that overall system performance improved over the UMA-style nodes due to the integrated memory controller, which enables access to local memory without contention from the other socket.

In the future, we would like to extend this work in several ways. First, we are in the process of applying this methodology to a collection of production applications. We expect to report on our results soon. We would also like to extend the methodology, so that we can isolate the effect of inter-node communication and study the sensitivity of application performance to sharing network latency and bandwidth. We would also like to generalize it to accommodate future node architectures including increasing number of cores per node and added types of shared resources.

References

[1] S. Alam, P. Agarwal, S. Hampton, and H. Ong. Experimental Evaluation of Molecular Dynamics Simulations on Multi-core Systems. In *Proceedings of HiPC*, volume 5374 of *Lecture Notes in Computer Science*, pages 131–141, 2008.

[2] S.R. Alam, R.F. Barrett, J.A. Kuehn, P.C. Roth, and J.S. Vetter. Characterization of Scientific Workloads on Systems with Multi-core Processors. In *Proceedings of IEEE International Symposium on Workload Characterization*, pages 225–236, Oct. 2006.

[3] D. Bailey, E. Barscz, J. Barton, D. Browning, R. Carter, L. Dagum, R. Fatoohi, S. Fineberg, P. Frederickson, T. Lasinski, R. Schreiber, H. Simon, V. Venkatakrishnan, and S. Weeratunga. The NAS Parallel Benchmarks. Technical Report RNR-94-007, NASA Ames Research Center, Moffett Field, CA, 1994.

[4] D. Bailey, T. Harris, W. Saphir, R. Van der Wijngaart, A. Woo, and M. Yarrow. The NAS Parallel Benchmarks 2.0. Technical Report NAS-95-020, NASA Ames Research Center, Moffett Field, CA, 1995.

[5] J. Carter, Y. He, J. Shalf, H. Shan, E. Strohmaier, and H. Wasserman. The Performance Effect of Multi-core on Scientific Applications. http://www.scientificcommons.org/34147250, 2007.

[6] L. Chai, Q. Gao, and D. Panda. Understanding the Impact of Multi-core Architecture in Cluster Computing: A Case Study with Intel Dual-core System. In *CCGRID '07: Proceedings of the Seventh IEEE International Symposium on Cluster Computing and the Grid*, pages 471–478, 2007.

[7] Cray Inc. products – XT5. http://www.cray.com/Products/XT5.aspx.

[8] K. Datta, M. Murphy, V. Volkov, S. Williams, J. Carter, L. Oliker, D. Patterson, J. Shalf, and K. Yelick. Stencil Computation Optimization and Auto-tuning on State-of-the-art Multicore Architectures. In *Proceedings of the 2008 ACM/IEEE Conference on Supercomputing*, November 2008.

[9] J. Djomehri, D. Jespersen, J. Taft, H. Jin, R. Hood, and P. Mehrotra. Performance of CFD Applications on NASA Supercomputers. In *Proceedings of the 21st Parallel Computational Fluid Dynamics Conference*, Moffett Field, CA, USA, May 2009.

[10] HPC Challenge Benchmarks. http://icl.cs.utk.edu/hpcc, 2006.

[11] IBM BladeCenter. http://www.ibm.com/systems/bladecenter/.

[12] A. Kayi, Y. Yao, T. El-Ghazawi, and G. Newby. Experimental Evaluation of Emerging Multicore Architectures. In *Proceedings of IEEE International Parallel and Distributed Processing Symposium*, pages 1–6, March 2007.

[13] R. Kufrin. PerfSuite: An Accessible, Open Source Performance Analysis Environment for Linux. In *The 6th International Conference on Linux Clusters: The HPC Revolution 2005*, Chapel Hill, NC, April 2005.

[14] J. Levesque, J. Larkin, M. Foster, J. Glenski, G. Geissler, S. Whalen, B. Waldecker, J. Carter, D. Skinner, H. He, H. Wasserman, J. Shalf, H. Shan, and E. Strohmaier. Understanding and Mitigating Multicore Performance Issues on the AMD Opteron Architecture. http://www.scientificcommons.org/34147406, 2007.

[15] Performance Application Programming Interface (PAPI). http://icl.cs.utk.edu/papi/.

[16] S. Saini, D. Talcott, D. Jespersen, J. Djomehri, H. Jin, and R. Biswas. Scientific Application-based Performance Comparison of SGI Altix 4700, IBM Power5+, and SGI ICE 8200 Supercomputers. In *Proceedings of the 2008 ACM/IEEE Conference on Supercomputing*, November 2008.

[17] SGI Altix ICE. http://www.sgi.com/products/servers/altix/ice/.

[18] A. van der Steen. Evaluation of the Intel Clovertown Quad Core Processor. Technical report, High Performance Computing Group, Utrecht University, http://www.euroben.nl/reports/qcore.pdf, 2007.

Appendix: Timings for Configurations of the Benchmarks

This Appendix presents the raw performance data for the runs of the HPCC benchmarks and the NAS Parallel Benchmarks used in this paper. Each benchmark was run at least 5 times on each of the configurations of Section 2 on each of the architectures described in Section 3. We then present the median value reported over the multiple runs.

Table 6 shows the performance data obtained for the HPCC benchmarks. Note that the values reported are actually rates (Gflops/sec for DGEMM and GBytes/sec for STREAM Copy, STREAM Triad, and PTRANS) and that higher values indicate better performance. Similarly, Table 7 shows the median rates (Gflops/sec) obtained for the six benchmarks from the NPB3.3-MPI suite for problem size Class C. Again, higher Gflops/sec values indicate better performance.

Table 6: *Performance of HPCC benchmarks for each of the 8 configuration runs using 16 processes. The units are Gflops/sec for DGEMM and GBytes/sec for STREAM Copy, STREAM Triad, and PTRANS.*

	Clovertown-Altix				Harpertown-Altix			
conf	DGEMM	ST_Copy	ST_Triad	PTRANS	DGEMM	ST_Copy	ST_Triad	PTRANS
222	9.054	0.612	0.680	0.912	10.521	0.922	0.990	1.238
122	9.194	0.760	0.842	1.106	10.567	0.948	0.967	1.351
212	9.421	1.187	1.309	1.392	10.836	1.718	1.818	1.943
221	9.541	1.210	1.328	1.407	10.985	1.822	1.912	1.920
112	9.478	1.444	1.595	1.508	10.841	1.772	1.838	2.026
121	9.598	1.518	1.649	1.509	11.000	1.869	1.925	2.007
211	9.680	1.959	2.373	1.530	11.133	3.154	3.476	2.578
111	9.706	2.224	2.952	1.702	11.137	3.324	3.589	2.467
	Barcelona-Cluster				Nehalem-Altix			
conf	DGEMM	ST_Copy	ST_Triad	PTRANS	DGEMM	ST_Copy	ST_Triad	PTRANS
222	7.935	1.446	1.605	1.479	10.965	4.620	3.304	4.604
122	7.938	1.534	1.724	1.506	11.019	4.675	3.319	4.028
212	7.984	2.698	2.909	1.825	11.159	8.655	6.863	4.442
221	7.988	2.697	2.908	1.809	11.159	8.665	6.866	4.429
112	7.995	2.736	2.970	1.968	11.153	8.867	6.971	6.261
121	7.997	2.727	2.962	1.988	11.153	8.864	6.973	6.316
211	7.999	3.519	3.623	2.105	11.205	10.427	10.946	6.745
111	7.998	3.614	3.743	2.487	11.201	10.526	11.041	7.028

Table 7: *Performance of NPB Class C for each of the 8 configuration runs using 16 processes. The unit is Gflops/sec for all six benchmarks.*

	Clovertown-Altix						Harpertown-Altix					
conf	BT	CG	FT	LU	MG	SP	BT	CG	FT	LU	MG	SP
222	10.446	1.807	5.132	9.688	3.858	2.877	15.205	2.329	7.348	15.445	6.887	4.134
122	12.144	2.389	5.694	10.878	4.852	3.537	15.602	2.355	7.106	15.989	7.292	4.501
212	13.983	3.007	6.796	13.176	7.335	5.218	21.333	3.897	8.650	20.296	12.032	6.888
221	14.022	3.026	7.236	15.184	8.790	5.734	23.190	4.170	9.493	21.351	14.762	8.526
112	15.214	3.535	8.287	13.940	8.772	6.236	22.164	4.085	10.015	20.974	13.125	7.699
121	15.132	3.553	8.557	16.221	10.386	6.825	23.867	4.354	11.422	22.139	16.035	9.914
211	15.953	3.816	9.171	17.035	11.503	8.018	27.676	6.048	12.808	25.361	23.655	13.725
111	16.608	4.067	10.046	17.560	12.493	8.923	28.158	6.308	12.511	25.830	25.412	15.009
	Barcelona-Cluster						Nehalem-Altix					
conf	BT	CG	FT	LU	MG	SP	BT	CG	FT	LU	MG	SP
222	16.193	3.576	7.524	9.339	7.562	5.253	30.965	7.353	13.493	28.171	17.720	12.162
122	16.384	3.802	7.770	9.340	7.627	5.300	30.263	7.720	14.478	28.306	17.791	11.122
212	18.722	4.071	9.778	17.533	11.995	8.155	35.163	10.161	19.965	35.849	38.505	17.756
221	18.725	4.086	9.780	17.604	11.996	8.158	34.367	10.158	19.913	35.841	38.489	17.889
112	18.905	4.236	10.105	17.566	12.301	8.298	35.142	10.348	22.611	35.876	39.964	21.601
121	18.910	4.240	10.112	17.502	12.324	8.299	35.504	10.349	22.555	35.876	39.909	21.591
211	19.568	4.294	11.180	20.400	15.861	9.878	37.188	11.029	23.857	38.041	53.464	29.801
111	19.655	4.501	11.591	19.509	16.223	10.065	37.309	10.963	19.867	38.102	52.266	30.504