

## A PERFORMANCE EVALUATION OF THE NEHALEM QUAD-CORE PROCESSOR FOR SCIENTIFIC COMPUTING

KEVIN J. BARKER, KEI DAVIS, ADOLFY HOISIE, DARREN J. KERBYSON, MIKE  
LANG, SCOTT PAKIN, JOSE CARLOS SANCHO

*Performance and Architecture Lab (PAL), Los Alamos National Laboratory, Los Alamos, New  
Mexico 87545, USA*

Received June 27, 2008

Revised August 22, 2008

Communicated by Guest Editors

### ABSTRACT

In this work we present an initial performance evaluation of Intel's latest, second-generation quad-core processor, Nehalem, and provide a comparison to first-generation AMD and Intel quad-core processors Barcelona and Tigerton. Nehalem is the first Intel processor to implement a NUMA architecture incorporating QuickPath Interconnect for interconnecting processors within a node, and the first to incorporate an integrated memory controller. We evaluate the suitability of these processors in quad-socket compute nodes as building blocks for large-scale scientific computing clusters. Our analysis of intra-processor and intra-node scalability of microbenchmarks, and a range of large-scale scientific applications, indicates that quad-core processors can deliver an improvement in performance of up to 4x over a single core depending on the workload being processed. However, scalability can be less when considering a full node. We show that Nehalem outperforms Barcelona on memory-intensive codes by a factor of two for a Nehalem node with 8 cores and a Barcelona node containing 16 cores. Further optimizations are possible with Nehalem, including the use of Simultaneous Multithreading, which improves the performance of some applications by up to 50%.

*Keywords:* Performance Analysis, Multi-core, Scientific Applications

### 1. Introduction

The advancing level of transistor integration is producing increasingly complex processor solutions ranging from mainstream multi-cores, heterogeneous many-cores, to special purpose processors (e.g. GPUs). There is no doubt that this will continue into the future until Moore's Law can no longer be satisfied. This increasing integration will require improvements in performance of the memory hierarchy to feed the processors. Innovations such as putting memory on top of processors, putting processors on top of memory (PIMS), or a combination of both may be a future direction forward. However, the utility of future processor generations will be a result of demonstrable increases in achievable performance from real workloads.

In this work we examine the performance of the latest, second-generation quad-core processor from Intel, the quad-core Core i7 (Nehalem-EP). We compare its performance to that of two first-generation quad-core processors, the AMD Opteron 8350 (Barcelona) and the Intel Xeon X7350 (Tigerton). Nehalem is implemented using a 45nm fabrication technology and represents an advance in terms of a single Moore's Law cycle from the 65nm process used for both Barcelona and Tigerton.

The performance of three nodes, one containing two Nehalem processors (8 cores), one containing four Barcelona processors (16 cores), and one containing four Tigerton processors (16 cores) are compared. Our analysis relies on performance measurements of application-independent tests (microbenchmarks) and a suite of scientific applications taken from existing workloads within the U.S. Department of Energy that represent various scientific domains and program structures. These processors and the nodes built around them are of particular interest because they implement explicit parallelism at multiple levels: within core (Nehalem), within processor, and within node. These processors represent the first and second generations of competing quad-core technologies and are or will be the building blocks of large-scale parallel computers.

The performance and scaling behavior of each application was measured on one core, when scaling from one to four cores on a single processor, and when using all processors in a node. In addition, we determined the best achievable performance of each application on each node, which is not necessarily when using all processing cores within a socket, or all cores within a node. This is heavily dependent on the application characteristics. Though much of our work is focused on large-scale system performance, including that of the largest systems available such as Blue Gene/L and Blue Gene/P, ASC Purple, ASC Redstorm [1] and Roadrunner [2], we note that the performance at large scale is a function of both the performance of the computational nodes as well as their integration into the system as whole.

This paper is organized as follows. An overview of the Barcelona, Tigerton, and Nehalem nodes is given in Section 2. Low-level microbenchmarks are described in Section 3, together with measured results from each node. Section 4 describes the suite of applications, the input decks used, and the methodology used to undertake the scalability analysis. Results are presented in Section 5 for the three types of analysis as described. Further possible optimizations on Nehalem are discussed in Section 6. Conclusions from this work are discussed in Section 7.

The contribution of this work is in the analysis of empirical performance data from a suite of complete scientific applications on the latest Intel Nehalem processor and first-generation quad-core processors from both AMD and Intel. These data are obtained from a strict measurement methodology to ensure that conclusions drawn from the scalability analysis are fair. Note that in this present work we do not consider physical or economic issues such as hardware cost, power consumption, or physical node size. The process that we follow is directly applicable to other multi-core studies. This work builds on the comparative analysis between Barcelona and Tigerton previously published [3].

## 2. Processor and Node Descriptions

Here we give an overview of the Intel Nehalem, AMD Barcelona, and Intel Tigerton quad-core processors. Barcelona and Tigerton represent competing first-generation quad-core processor designs first available in September 2007, whereas Nehalem is a second-generation quad-core processor expected to be available in November 2008. All three are detailed below and illustrate quite different implementations both in terms of processor configuration and connectivity to memory.

### 2.1. The Intel Nehalem quad-core Xeon processor

Nehalem will form the basis of a range of 64-bit processors over the next few years under the brand name Intel Core i7. The Nehalem-EP is a first in this family and uses 45nm fabrication. It has four cores each with 64KB L1 cache (32KB data + 32KB instruction) and 256KB L2 cache, and an 8MB shared L3 cache. The on-chip memory controller supports three DDR3 memory channels. In addition, two QuickPath Interconnect (QPI) channels allow two processors to be interconnected, effectively implementing a non-uniform memory access (NUMA) architecture, as well providing I/O connectivity. A further development expected in 2009 is the Nehalem-EX that will contain an increased number of cores—6 or 8 per processor, an increased number of QPI channels, and a larger L3 cache. The Nehalem-EP is depicted in Figure 1(a), and a two-processor node is depicted in Figure 1(b).

Our performance analysis is of a pre-production Nehalem node. We expect production hardware to achieve a similar level of performance. The node consisted of two processors clocked at 2.8GHz. Each processor core can issue 4 double-precision floating-point operations per clock resulting a peak performance of 44.8Gflops/s per chip. The processors were connected via a single QPI with a peak link transfer speed of 6.4GT/s, where a single transfer consists of 2B per direction. The node contained three DDR3 1333MHz memories per processor for a total of 24GB.

The Nehalem also has the capability of Simultaneous Multi-Threading (SMT) with two threads per core [4]. Its aim is to hide memory latencies by switching between the hardware threads on memory stalls. SMT can significantly improve application performance, as we show in Section 6 and should not be considered similar to Intel's previous Hyper-Threading.

### 2.2. The AMD Barcelona quad-core Opteron processor

Barcelona, the latest generation of the Opteron, combines four Opteron cores onto a single die as shown in Figure 1(c), using 65nm fabrication, with a process shrink to 45nm expected in late 2008. Each die contains a single integrated memory controller and uses a HyperTransport (HT) network for point-to-point connections between processors [5]. Each core has a private 64KB L1 cache (32KB data + 32KB instruction) and a private 512KB L2 cache, and each processor has a shared 2MB L3 cache. The shared L3 cache is new to the Opteron architecture as are 128-bit

SSE4a instructions enabling each core to issue 4 double-precision floating-point operations per clock. The clock speed of each core is 2.0GHz giving each chip a peak performance of 32Gflops/s.

Each node contains four quad-core processors as shown in Figure 1(d). DDR2 667MHz memory is used and thus the memory bandwidth per processor is 10.7GB/s. The total memory capacity of the node is 16GB (4GB per processor). The HT links connect the four processors in a  $2 \times 2$  mesh, and effectively implement a NUMA architecture. Further HT links provide PCI Express I/O capability. Each HT link has a theoretical peak of 8GB/s for data transfer.

### 2.3. The Intel Tigerton quad-core processor

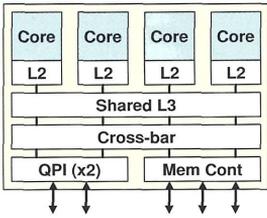
The Intel Tigerton processor [6] contains two dual-core dies, using 65nm fabrication, that are packaged into a single dual-chip module (DCM) that is seated within a single socket as shown in Figure 1(e). Each core contains a private 64KB L1 cache (32KB data + 32KB instruction), while the two cores on each die share a 4MB L2 cache for a total of 8MB L2 cache within the DCM. The processor implements the 128-bit SSE3 instruction set for SIMD operations and thus can perform 4 double-precision floating-point operations per cycle. The processor is clocked at 2.93GHz so the DCM has a theoretical peak performance of 46.9 Gflops/s.

Each node contains four processors for a total of 16 cores as shown in Figure 1(f), and contains a total of 16GB of main memory using fully-buffered DIMMs (FBDIMMs). Central to the node is a single memory controller hub (MCH). This hub interconnects the front side bus (FSB) of each processor to four FBDIMM memory channels. Unlike the NUMA configuration of its successor Nehalem (and of Barcelona), Tigerton is a symmetric multiprocessor (SMP) configuration. The MCH contains a 64MB snoop buffer and a Dedicated High Speed Interconnect (DHSI) as well as PCI Express channels. The purpose of the snoop buffer is to minimize main memory accesses, while the DHSI provides a point-to-point link between each processor and the memory channels. The FSB of each processor runs at 1066MHz. The memory speed is 667MHz and thus provides a peak memory bandwidth of 10.7GB/s per processor that is shared by the four cores.

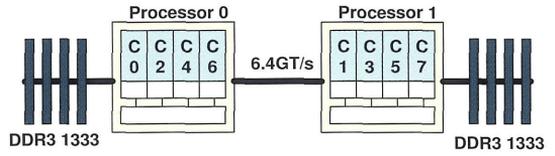
Table 1. Characteristics of the quad-core processors, memory, and node organization.

	Processor					Memory		Node	
	Speed GHz	Peak Gflops	L1 KB	L2 MB	L3 MB	Type	Speed MHz	Memory controllers	Peak Gflops
Nehalem	2.8	44.8	64	0.25	8	DDR3	1333	2	89.6
Barcelona	2.0	32	64	0.25	2	DDR2	667	4	128.0
Tigerton	2.93	46.9	64	4	-	FBDIMM	667	1	187.6

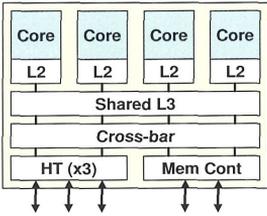
A summary of the three processor architectures and nodes is presented in Table 1. The peak performance of each processor, and each node configuration is shown as well as characteristics of the memory hierarchy. Note that the L2 cache is shared



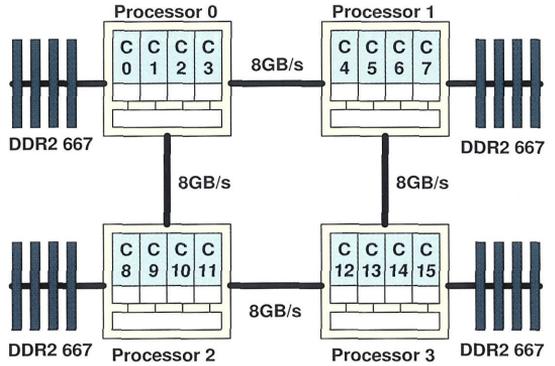
(a) Nehalem quad-core processor



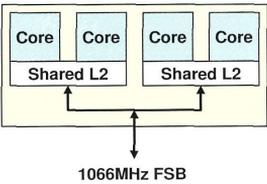
(b) Nehalem dual-processor node



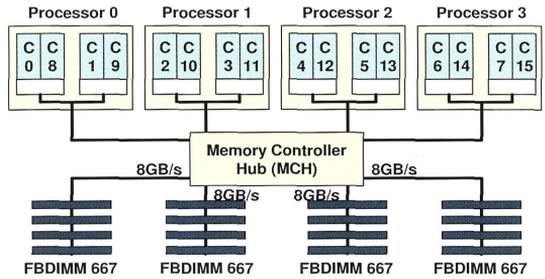
(c) Barcelona quad-core processor



(d) Barcelona quad-processor node



(e) Tigerton quad-core processor



(f) Tigerton quad-processor node

Fig. 1. Overview of the Nehalem, Barcelona and Tigerton Quad-core processors

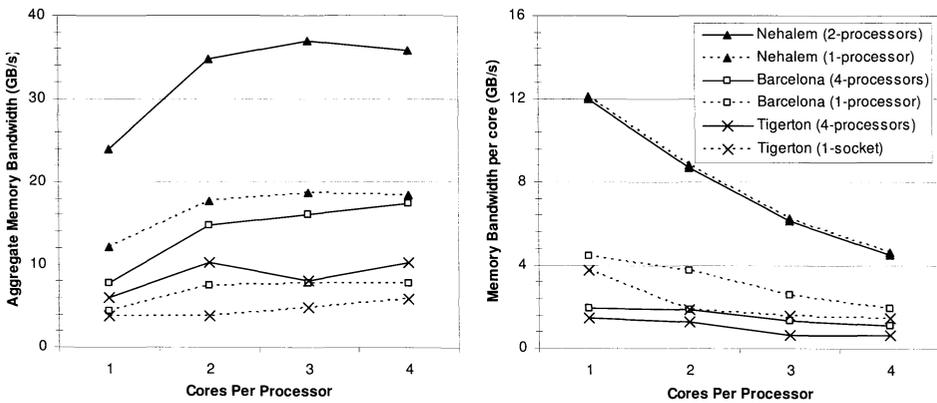
across the two cores on each die in Tigerton, and that the L3 cache is shared across the four cores on Nehalem and Barcelona.

### 3. Low-level performance characteristics

#### 3.1. Memory bandwidth

To measure the memory bandwidth per core we ran the MPI version of the University of Virginia’s Streams benchmark [7]. This benchmark is a memory stress test for a number of different operations. We report here the performance of the *triad* test. In Figure 2(a) the aggregate memory bandwidth is shown for Nehalem, Barcelona, and Tigerton nodes for two cases: using a single processor and using all processors in the node. The number of cores per processor used in both cases is varied from one to four. In all cases the Nehalem node significantly outperforms the Barcelona node which in turn outperforms the Tigerton node. The measured single-core memory bandwidths are 12.1GB/s on Nehalem, 4.4GB/s on Barcelona, and 3.7GB/s on Tigerton. The aggregate memory bandwidths are 35.8GB/s, 17.4GB/s, and 10.2GB/s, respectively. Note that the aggregate bandwidth is achieved from two processors (8 cores) on the Nehalem node, compared to four processors (16 cores) on the Barcelona and Tigerton nodes.

Figure 2(b) is based on the same data as Figure 2(a) but presents the observed memory bandwidth per core. As shown, the per-core bandwidth decreases from 12GB/s (one core per socket) to 4.6GB/s (four cores per socket) for Nehalem (a factor of 2.6 decrease), 4.4GB/s to 1.1GB/s for Barcelona (a factor of 4.0 decrease), and from 3.7GB/s to 0.63GB/s for Tigerton (a factor of 5.9 decrease). These decreases are significant: the aggregate achievable memory bandwidth is important to memory-intensive applications, but the Nehalem node has a clear advantage over both Barcelona and Tigerton in these cases.



(a) Aggregate memory bandwidth

(b) Memory bandwidth per core

Fig. 2. Streams Memory Bandwidth.

### 3.2. Processor locality

The mapping of application processes to cores within a node affects the memory contention that the application induces. In our experience, the task-to-core ordering is not always obvious and should at a minimum be verified. Linux determines core numbering via information provided by the BIOS.

In this testing we used an MPI benchmark to measure the latency between each core and all of the others. The latency varies depending on whether the two communicating cores are on the same die, on different dies in the same processor (in the case of Tigerton), or different processors (or single- or multiple-hop remote processors for Barcelona). Note for comparison we show the latency between threads for the Nehalem node which implements a total of 16 threads using SMT.

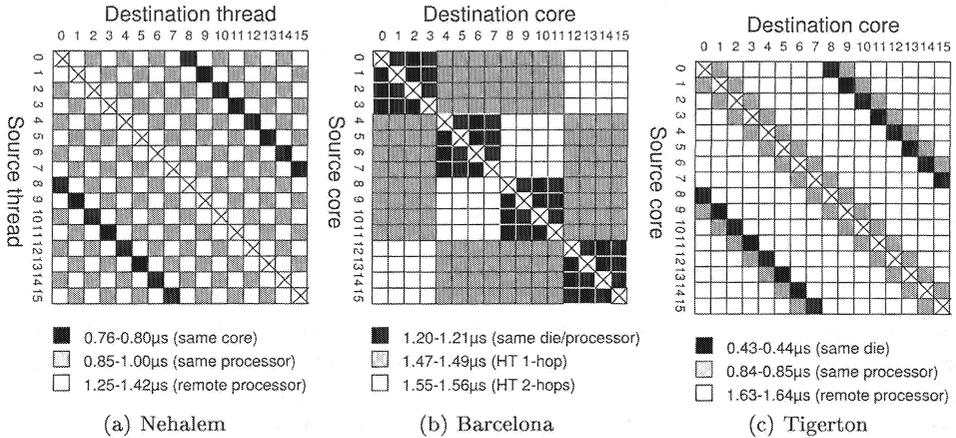


Fig. 3. Observed MPI latency from any core to any other core in a node.

From our latency measurements we were able to determine the arrangement of the cores as seen by an application. Figure 3 shows the observed latency from this test in the form of a matrix. The vertical axis indicates the sending core (or thread) and the horizontal axis is the receiving core (or thread). Shading is used to denote the different latencies. No test was performed for a core sending to itself (the major diagonal in the matrix). The Barcelona node uses a linear ordering of cores to processors, that is, the first four cores reside on the first processor as indicated by the lowest latency, blocks of size 4x4 cores in Figure 4(b), and so on. In both the Tigerton and Nehalem nodes a round-robin ordering across dies is used. Cores on the same die are an MPI logical task distance of eight apart, as shown by the black diagonal lines in Figure 4(c). As shown in Figure 4(a), in the case of Nehalem, even threads are mapped to a one processor, and odd threads are mapped to the other processor.

The latency from one core to another is in very distinct ranges depending on their relationship (same die, remote die, etc.). We observe that the maximum latency

is similar on all nodes, but Nehalem and Tigerton enjoy a much lower intra-die and intra-processor latency. The core and processor ordering was actually shown in Figure 1 based on the observed processor locality map shown in Figure 3. To handle the differences in the processor numbering between the nodes we implemented a small software shim that uses the Linux `sched_setaffinity()` system call to allow user-defined mappings between MPI rank and physical cores. This shim gave us the ability to map processes to cores identically across the three nodes and thereby perform fair comparisons of application performance.

#### 4. Application testing process

Our application suite includes several large-scale production applications that are currently used within the U.S. Department of Energy. The testing for each application consisted of

- (1) comparing the performance on a single core of the Nehalem, Barcelona and Tigerton;
- (2) examining the scaling behavior for two cases: (a) using only one processor, and (b) using all processors in a node;
- (3) determining the configuration (processors, and cores per processor) that yields the best performance.

All of the applications are typically run in a weak-scaling mode. That is, the global problem size grows with the number of nodes in the system. All available memory is typically used for increased fidelity in the physical simulations or for simulating larger physical systems. Our approach mimics typical usage by fixing the sub-problem per processor regardless how many cores per processor are used. The global problem grows in proportion to the number of processors used. This may be stated succinctly as doing strong scaling within a processor and weak scaling across processors.

##### 4.1. The application suite

An overview of each application is given below. Each is typically run on high-performance parallel systems using many thousands of processors at a time. A summary of the application input decks used is given in Table 2. The input decks are typical for problems processed on large-scale systems. Also indicated in Table 2 is the main processing characteristic of each application, for example, whether an application is compute intensive, memory intensive, or both.

- **GTC** – Gyrokinetic Toroidal Code is a Particle-in-Cell (PIC) code from Princeton [8]. It was developed to study energy transport in fusion devices.
- **Partisn** – SN transport code from Los Alamos solving the Boltzmann equation using the discrete ordinates method, on structured meshes [9].

- **SAGE** – an adaptive mesh (AMR) hydrodynamics code used for the simulation of shock waves. Developed jointly by Los Alamos and SAIC [10].
- **SPaSM** – the Scalable Parallel Short-range Molecular dynamics code from Los Alamos. Used to study material fracture and deformation properties [11].
- **Sweep3D** – a code-kernel from Los Alamos that implements deterministic SN transport [12, 13]. The computation is in the form of wavefronts that originate at the corners of a 3-D physical space.
- **VH1** – the Virginia Hydrodynamics code simulates an ideal inviscid compressible flow of gas hydrodynamics and is capable of simulating three-dimensional turbulent stellar flows [14].
- **VPIC** – a Particle-In-Cell code from Los Alamos used to model particle flow within a plasma [15].

Table 2. Summary of the input decks used for each application.

	Input deck	Problem per processor	Memory per processor	Processing characteristic
GTC	1Dwedge	6.2M particles	320MB	Particle based
Partisn	Pencil	20x10x400	80MB	Memory/Compute
SAGE	timing_h	140K cells	280MB	Memory
SPaSM	BCC	64x64x64	150MB	Compute
Sweep3D	Pencil	20x10x400	8MB	Kernel, small memory
VH1	ShockTube	200x200x200	900MB	Memory/Compute
VPIC	3D-HOT	4M particles	256MB	SSE, memory

## 5. Application Performance Analysis

### 5.1. Single-core performance comparison

The performance of each application on a single core of each of the three processors is shown in Figure 4(a). The metric *time* denotes the iteration time of the main computational loop (without I/O) for all applications except Sweep and Partisn, for which it is 10 iterations, and VPIC, for which it is 100 iterations, to aid visual comparison. The relative runtime, between the Nehalem and Barcelona, and between the Nehalem and Tigerton, is shown in Figure 4(b). A value of one indicates the same performance between two processors, and a value of two indicates a 2x performance advantage of Nehalem.

As shown, a Nehalem core is between 1.1 and 1.8 times faster than a Tigerton core, and between 1.6 and 2.9 times faster than a Barcelona core, for the applications tested. Note that the clock speed advantage of Nehalem over Barcelona is a factor of 1.4 indicating that the performance advantage arises in part from the core design and from the faster memory subsystem. Also note that the Nehalem has a slower clock speed than Tigerton, but achieves a higher level of performance for all applications again due to improvements in the core design and the memory subsystem.

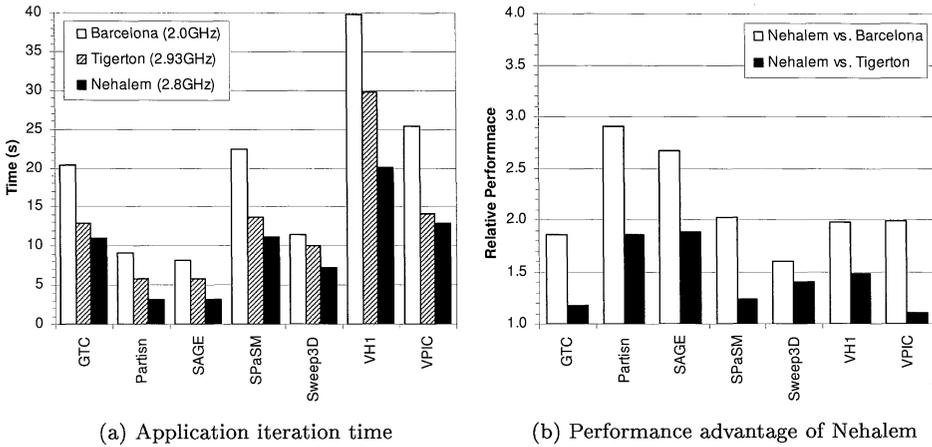


Fig. 4. Single-core performance comparison.

### 5.2. Single-processor performance comparison

The performance of each application on a single processor is shown in Figure 5(a). The basis for this comparison is the same as that used in single core comparison. The relative runtimes between Nehalem and Barcelona, and between Nehalem and Tigerton, is shown in Figure 5(b).

As shown, the Nehalem processor achieves performance between 1.4 and 3.6 times higher than a Tigerton processor and between 1.5 and 3.3 times faster than a Barcelona processor. Thus the scaling behavior of the applications on both the Nehalem and Barcelona processors seem similar when comparing Figures 4(b) and 5(b), whereas the scaling behavior is significantly worse on Tigerton. This lack of scaling on Tigerton can be attributed in part to the poorer performance of the memory subsystem.

### 5.3. Node performance comparison

The best performance achieved for each application on the Nehalem node (8 cores), the Tigerton node (16 cores), and the Barcelona node (16 cores) is shown in Figure 6. Note that only relative performance is considered here. The relative performance was calculated using the effective processing rate for each application since the core count and hence the problem size for each application differed between the nodes.

Figure 6 shows that for five of the seven applications the Nehalem node (containing 8 cores) outperforms Tigerton node (containing 16 cores) and for six application the Nehalem node outperforms the Barcelona node. For SAGE, the most memory intensive application, the Nehalem node outperforms the Tigerton node by 3.0x and the Barcelona node by 2.8x. The Nehalem node actually achieved a lower level of performance on Sweep3D than the other two nodes, but was slightly higher performing than Barcelona for SPaSM.

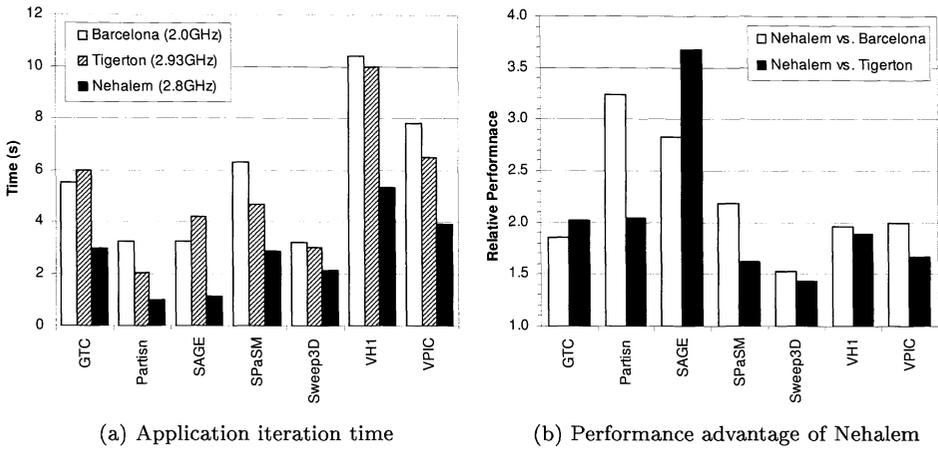


Fig. 5. Single-processor performance comparison.

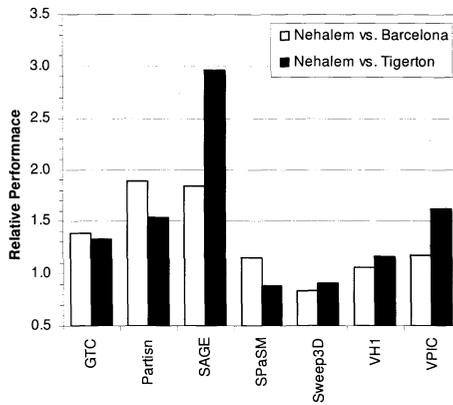


Fig. 6. Nehalem performance advantage (Node)

The results shown in Figure 6 are in line with the differences in the memory bandwidth measurements in Section 3.1. The node memory bandwidth on Nehalem is 35.8GB/s, on Barcelona 17.4GB/s, and on Tigerton 10.2GB/s. The ratio of the node memory bandwidths is very close to the ratio in the performance between the nodes for SAGE. The memory footprint of Sweep3D is small and fits into cache on all processors. Thus the performance advantage (or disadvantage at 0.84x) is close to the ratio of the clock speeds (2.8GHz vs. 2.0GHz) multiplied by the ratio of the number of cores in each node (8 vs. 16) = 0.7x. The remaining difference is due to micro-architecture differences.

Note however that the performance presented above is based on the best observed node performance, which does not necessarily result when using all 16 cores in the node. In fact the best performance observed on VPIC and Partisn was when

using two cores per processor (8 cores total), and for SAGE when using three cores per processor (12 cores total) on the Tigerton node. The best performance in all other cases was observed when using all cores in a node. This is illustrated in below when analyzing the performance as a function of the number of cores and number of processors used for each application.

**5.4. Application scalability analysis**

To analyze the performance of using multiple cores we followed a strict process in which the problem size per processor was constant for all tests, as described in Section 4. For each application we show the performance when using between one and four cores per processor for the case of using one processor, shown in Figure 7, and all the processors in a node, shown in Figure 8. Note that the performance relative to the single-core performance for each processor type is shown—this is the speedup when using multiple cores. Note also that the legends for all graphs are shown separately in Figure 7 and Figure 8.

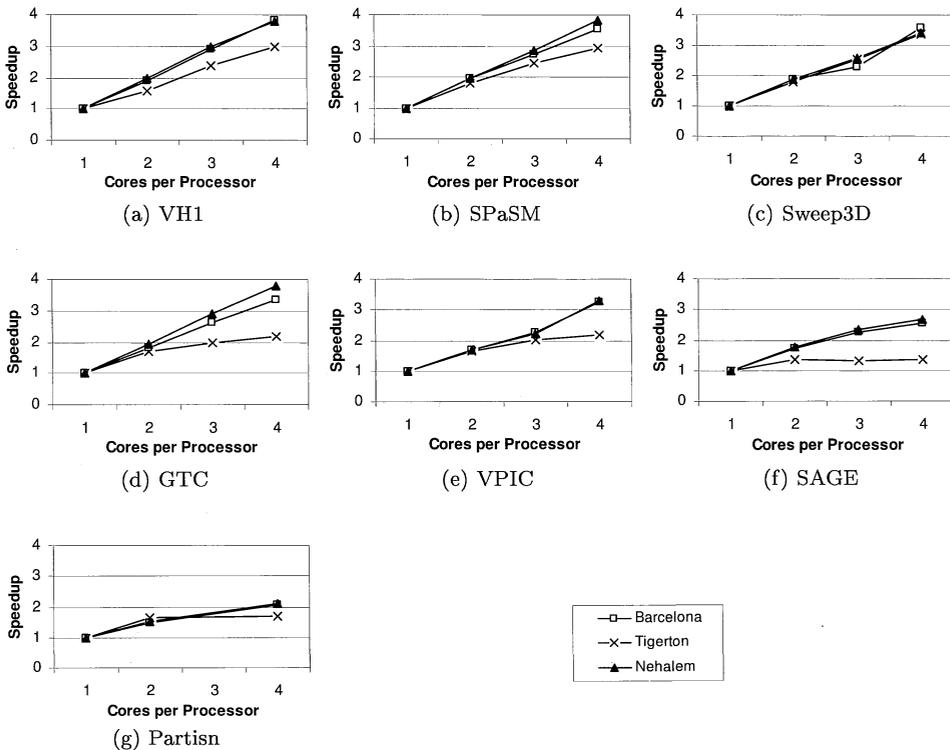


Fig. 7. Application speedup on a single processor.

The scalability of five of the applications is very good on a single Nehalem and a single Barcelona processor, as shown in Figure 7, achieving a speedup of over three

out of a theoretical maximum of four when using all four cores. A lower speedup is observed for the two more memory intensive codes SAGE and Partisn. Poor scaling is observed on Tigerton for most of the applications. The performance when using four cores is often not significantly higher than when using three cores.

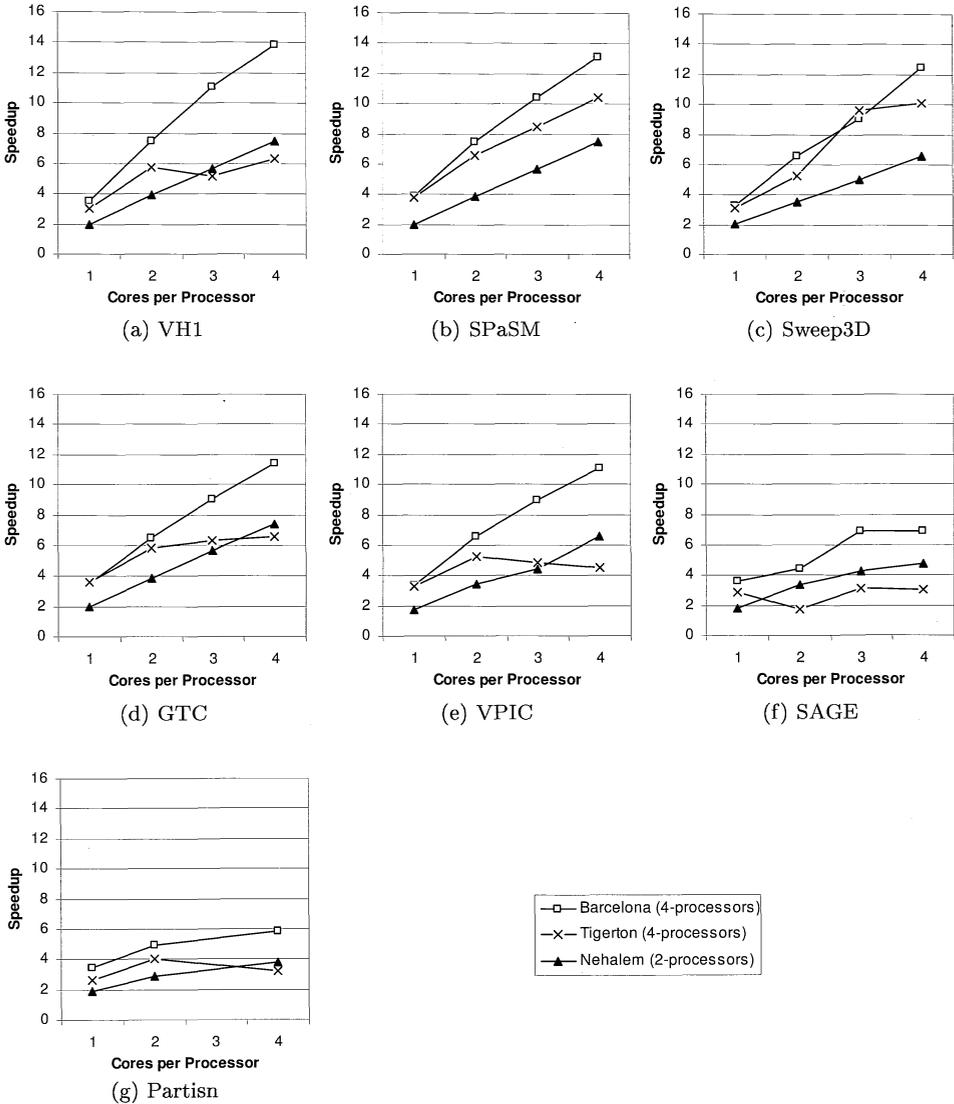


Fig. 8. Application speedup using all processors in a node.

The applications are ordered in terms of their observed speedup in Figure 8. The applications with best scaling behavior on all nodes are VH1, SPaSM and Sweep3D (Figures 8(a)-8(c)). Both VH1 and SPaSM are compute bound, and Sweep3D has

a small memory footprint resulting in high cache utilization. In contrast SAGE and Partisn are memory bound and show the lowest scalability in all cases.

In order to compare the speedup for each application across nodes, as shown in Figure 8, one must recall that the Nehalem node contains 8 cores whereas the Barcelona and Tigerton nodes each contain 16 cores. The Nehalem actually achieves a speedup of over six on five of the applications, but achieves a lower speedup on the memory intensive applications SAGE and Partisn. Indeed the Nehalem achieves a higher speedup than Tigerton on five of the applications even though Tigerton contains twice as many cores. The speedup on Barcelona is also good, achieving over 11x on five of the applications.

A summary of the best speedup achieved on each node for all applications is shown in Table 3. Also included in Table 3 is the efficiency—the percentage of the maximum speedup that is achieved—in each case. The Nehalem node achieves higher efficiency than the Barcelona node which in turn achieves higher efficiency than the Tigerton node in all cases.

Note that even if an application has a higher speedup on one node in comparison to another node it does not necessarily mean that it has a higher performance, as was evident from Figure 6(b).

Table 3. Summary of application speedup and efficiency on a Nehalem, Barcelona, and Tigerton node.

	Nehalem		Barcelona		Tigerton	
	Speedup (max=8)	Efficiency (%)	Speedup (max=16)	Efficiency (%)	Speedup (max=16)	Efficiency (%)
SPaSM	7.5	94	13.1	82	10.5	66
VH1	7.4	93	13.9	87	6.3	39
GTC	7.4	93	8.3	71	6.5	41
VPIC	6.6	82	11.2	70	4.5	28
Sweep3D	6.5	82	12.4	78	10.1	63
SAGE	4.7	59	6.9	43	3.0	19
Partisn	3.8	47	5.9	37	3.2	20

## 6. Nehalem Performance Optimizations

There are several features of Nehalem that may be used to further improve application performance over that presented in Section 5. These include

- **Simultaneous Multithreading** – A Nehalem processor core can operate in a Simultaneous Multithreading (SMT) mode in which two threads execute on each processor. This is to hide latencies to memory: when one thread stalls on a memory access, the other thread can in principle use these wasted cycles.
- **Compiler Optimizations** – Enhancements are expected in the Intel compiler to take advantage of many of the micro-architectural features of Nehalem. The current version of the Intel compiler that was used in this study did not implement any specific optimizations for Nehalem.
- **Turbo mode** – A frequency stepping mode that can be enabled in the BIOS enables the processor frequency to be stepped up by up to four frequency incre-

ments of 100MHz each. The up-stepping occurs when the thermal characteristics of the processor allows it. This could be useful when there is load imbalance between cores, allowing the most heavily loaded core to be sped up when other cores are idle. Turbo mode was not enabled in this study.

The use of SMT was analyzed for the seven applications previously discussed. The performance improvements observed with and without the use of SMT is shown in Figure 9 when using a single processor, and both processors, in a node. As shown, about half of the applications benefit from the use of SMT, and half do not. Indeed, a very impressive 52% performance improvement was observed when using SMT for SPaSM, 22% for VPIC, and 10% for GTC. The worst decrease in performance was -4% for Sweep3D and for SAGE.

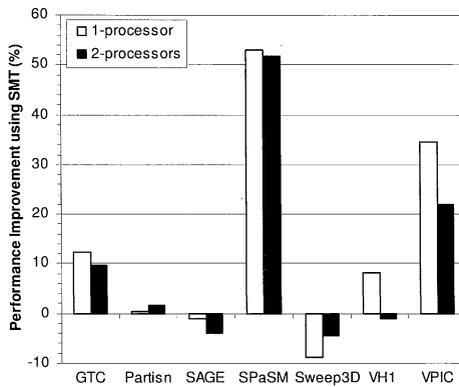


Fig. 9. Observed improvements in application performance when using SMT on Nehalem

SMT is a BIOS option that is set at boot time. For applications that can take advantage of SMT the performance gains can be substantial. However it can slow down some applications and hence its use should be carefully determined based on the applications being used. Further analysis is required to characterize applications that can take advantage of SMT, and also to determine if applications can be optimized to take advantage of SMT.

## 7. Conclusions

In this work we have presented a performance evaluation of the latest, second-generation Intel quad-core processor, Nehalem, that will form the basis of Intel's mainstream multi-core processors for a number of years. This processor is the first Intel processor to incorporate QuickPath Interconnect for inter-processor communications, and an integral memory controller supporting three DDR3 memory channels. The performance of Nehalem was contrasted against first-generation quad-core processors from both AMD and Intel, Barcelona and Tigerton respectively.

Data was obtained using microbenchmarks and a suite of scientific applications. A strict measurement methodology that used a shim to control the mapping of application processes to processors, and compared the per-core performance as well as scaling up to using all cores in a node. The process followed is directly applicable to other multi-core studies.

When considering the performance of a single core, where there was no memory contention, a Nehalem core achieved a performance almost three times higher than a Barcelona core for the most memory intensive codes. A minimum performance advantage of 1.6x was observed on a cache-resident code, which corresponds closely to the ratio of the clock speeds of Nehalem and Barcelona. When using all of the cores in a processor the results are more dependent on the way each application uses memory. Nehalem enjoys a performance advantage of between 1.9x and 3.2x over Barcelona for all but the cache-resident code.

Finally, when examining scaling across the entire node, the results again show significant performance advantages over both the Barcelona node and Tigerton node. Indeed, for the most memory intensive codes the performance of Nehalem is almost a factor of two greater than Barcelona's, and up to a factor of three higher than Tigerton's. This is even more significant when considering the Nehalem node contained only two processors (8 cores) whereas both the Barcelona and Tigerton each contained four processors (16 cores). Further optimizations are also possible on Nehalem, including the use of Simultaneous Multithreading which can improve application performance by up to 50%.

While this study represents a snapshot of current processors and node architectures, it also represents a snapshot of current application structures. All of the applications we ran use the *one MPI rank per core* model. Although this is an extremely portable way to structure an application, it may be possible to gain more performance by exploiting the properties of multi-core processors, such as physically proximate processes sharing cached data. We have shown that for applications as they exist today it is important to consider the balance between compute rate and memory rate when selecting a processor from which to build a cluster.

## Acknowledgments

We thank Intel for providing early access to pre-production Nehalem nodes, and to AMD for providing access to Barcelona nodes, for this performance evaluation. This work was funded in part by the Accelerated Strategic Computing program and the Office of Science of the Department of Energy. Los Alamos National Laboratory is operated by Los Alamos National Security LLC for the US Department of Energy under contract DE-AC52-06NA25396.

## References

- [1] A. Hoisie, G. Johnson, D.J. Kerbyson, M. Lang, and S. Pakin, A Performance Comparison Through Benchmarking and Modeling of Three Supercomputers: Blue Gene/L,

- Read Storm and ASC Purple, in *Proc. IEEE/ACM SuperComputing*, Tampa, FL, Nov. 2006.
- [2] K.J. Barker, K. Davis, A. Hoisie, D.J. Kerbyson, M. Lang, S. Pakin, J.C. Sancho, Entering the Petaflop Era: The Architecture and Performance of Roadrunner, in *Proc. IEEE/ACM SuperComputing*, Austin, TX, Nov. 2008.
- [3] K.J. Barker, K. Davis, A. Hoisie, D.J. Kerbyson, M. Lang, S. Pakin, J.C. Sancho, Experiences in Scaling Scientific Applications on Current-generation Quad-core Processors, in *Proc. workshop on Large-Scale Parallel Processing (LSPP), Int. Parallel and Distributed Processing Symposium (IPDPS)*, Miami, FL, Apr. 2008.
- [4] Intel White Paper, First the Tick, Now the Tock: Next Generation Intel Microarchitecture (Nehalem), Intel publication 0408/VP/HBD/PDF 319724-001US, Apr. 2008.
- [5] AMD White Paper, HyperTransport Technology I/O Link, A High-Bandwidth I/O Architecture, Advanced Micro Devices, Inc., One AMD Place, Sunnyvale, CA 94088, #25012A, July 20, 2001.
- [6] Intel Corporation, Quad-core Intel Xeon Processor 7300 Series. Product Brief. 2007.
- [7] J. McCalpin, Memory bandwidth and machine balance in current high performance computers, in *IEEE Comp. Soc. Tech. committee on Computer Architecture (TCCA) Newsletter*, Dec. 1995, 19-25.
- [8] N. Wichmann, M. Adams, S. Ethier. New Advances in the Gyrokinetic Toriodal Code and Their Impact on Performance on the Cray XT Series, in *Proc. Cray User Group (CUG)*, Seattle, WA, 2007.
- [9] R.S. Baker, A Block Adaptive Mesh Refinement Algorithm for the Neutral Particle Transport Equation, *Nuclear Science & Engineering* **141**(1) (2002) 1-12.
- [10] D.J. Kerbyson, H.J. Alme, A. Hoisie, F. Petrini, H.J. Wasserman, and M.L. Gittings, Predictive Performance and Scalability Modeling of a Large-scale Application, in *Proc. IEEE/ACM Supercomputing*, Denver, CO, Nov. 2001.
- [11] S.J. Zhou, D.M. Beazley, P.S. Lomdahl, B.L. Holian, Large-scale molecular dynamics simulations of fracture and deformation, *J. of Computer-Aided Materials Design* **3**(1-3) (1995) 183-186.
- [12] K.R. Koch, R.S. Baker, R.E. Alcouffe, Solution of the First-Order Form of the 3-D Discrete Ordinates Equation on a Massively Parallel Processor, *Trans. of the American Nuclear Soc.* **65** (1992) 198-199.
- [13] A. Hoisie, O. Lubeck, H.J. Wasserman, Performance and Scalability Analysis of Teraflop-Scale Parallel Architectures using Multidimensional Wavefront Applications, *Int. J. of High Performance Computing Applications* **14**(4) (2000) 330-346.
- [14] J.M. Blondin, VH-1 User's Guide. North Carolina State University, 1999.
- [15] K. Bowers. Speed optimal implementation of a fully relativistic 3d particle push with charge conserving current accumulation on modern processors, in *Proc. 18th Int. Conf. Numerical Simulation of Plasmas*, 2003, 383.