Performance Characteristics of Dual-Processor HPC Cluster Nodes Based on 64-bit Commodity Processors

A. Purkayastha, C. S. Guiang, K. Schulz, T. Minyard, K. Milfeld,
W. Barth, P. Hurley and J. R. Boisseau

Texas Advanced Computing Center
The University of Texas at Austin
Austin, TX 78758

Abstract. Dual-processor nodes are the preferred building blocks in HPC clusters because of the greater performance-to-price ratio of such configurations relative to clusters comprising single-processor nodes. The arrival of 64-bit commodity clusters for HPC is advantageous for applications that require large amounts of memory and I/O because of the larger memory addressability of these processors. Some of these 64-bit processors also use more advanced memory subsystems, which provide increased performance for some applications. This paper examines the overall performance characteristics of three dual-processor systems based on commodity 64-bit processors: Intel Itanium2, AMD Opteron and IBM PowerPC 970, also known as the Apple PowerPC G5. First, a low-level characterization of each system is obtained using a variety of computational kernels and micro-benchmarks to measure the speeds of the functional units and memory subsystems. Performance measurements and analysis of several scientific applications that span a wide range of computational requirements are presented next. Finally, we offer some general observations and insights on performance for applications developers and discuss 32- to 64-bit migration and interoperability issues.

1. Introduction

The introduction of computational nodes with commodity 64-bit processors for high performance computing (HPC) clusters enables applications to access larger amounts of memory and perform I/O on larger files than possible in a 32-bit computing environment. Scientific and commercial applications that require and use 64-bit integers will see a performance increase on 64-bit hardware due to 64-bit processor's wider registers compared to 32-bit systems. As 64-bit computing in cluster space is still in its nascent stage, some of the operating systems and compilers still do not have full 64-bit address support and therefore cannot yet take full advantage of the underlying 64-bit hardware. However, the lag between software and hardware should not persist for very long because of the increasing demand for cluster computing environments with more addressable memory. Still, the transition from 32- to 64-bit computing will not happen overnight. Large HPC centers like TACC support many scientific applications that typically do not require more than 2GB of memory per node, in part because the systems often have hundreds of nodes and thus up to a terabyte or more of aggregate memory. Therefore, the need for portability, interoperability and performance for 32-bit applications in the 64-bit space are practical and important issues to consider as well.

While processor performance has tracked Moore's Law, memory speed has also increased but at a slower rate. Application scientists have exploited the rapid increase in available processor performance by pushing their science to higher limits, but fully realizing these potential performance improvements requires careful cache optimization. Solving larger scientific problems, however, means larger computational demands, including bigger physical systems requiring more memory, which make cache optimization and thus higher sustained performance more difficult. Meeting these two needs—using more memory while achieving increased sustained performance—is difficult with current 32-bit

microprocessors. The 32-bit computing environment has nodes that possess a shared bus memory architecture that limits memory read/write rates; furthermore, the memory addressability is limited to 2^31 bytes per process, thus requiring large-memory applications to be spread out across more nodes and to pass more data over the interconnect. In a 64-bit computing environment, each process can access far more than 4GB of memory (addressability increases to 2^63 bytes). The wider integer registers in 64-bit systems presents a performance benefit to certain applications requiring 64-bit integers, such as cryptography. In addition, some commodity 64-bit processors have more sophisticated memory subsystems to help ensure more performance is realized.

The aim of this paper is to provide insights for understanding the performance of scientific applications on clusters using these new commodity 64-bit dual-processor nodes by conducting a comprehensive characterization of performance on single nodes. We first present details on the architecture of these systems and the configurations we examined (Section 2), as well as the state of the current software infrastructure to exploit the 64-bit hardware and the associated impact on performance. We then compare the floating-point performance and memory architecture subsystems of three different dual-processor commodity system using micro-benchmarks and kernels, which measure floating-point performance, memory bandwidth and latencies, and thus enable low-level characterization of the system architecture (Section 3). Next we present performance results for several scientific applications with varying computational demands. These results are analyzed and discussed in the context of the system architectures and the low-level performance measurements (Section 4). Integrating all of this data, we present general observations and insights on application performance to help users and applications developers understand what kind of performance they might expect to achieve on such systems, which can aid in selecting systems to for maximum effectiveness (Section 5). Since most users are currently using clusters with 32-bit microprocessors, we also discuss 32- and 64-bit interoperability and portability issues. Finally, we will provide a price/performance evaluation based on online pricing.

2. System Architectures

For this analysis of commodity 64-bit dual-processor nodes, we used three two so-node clusters consisting of dual-processor nodes, with each cluster based on nodes using one of the three commodity 64-bit microprocessors on the market: Intel Itanium2, AMD Opteron, or IBM PowerPC 970 (referred to by Apple as the Apple PowerPC G5). For Intel Itanium2 and AMD Opteron, we selected nodes for evaluation expected to be 'best in class' implementations available at the beginning of the study based on node design and processor/memory configurations. (Faster processors will be available by the LCI conference, but the comparative analyses herein should still provide useful insights. There was only one dual-processor node configuration available with the PowerPC 970 processor at the beginning of this evaluation.) The three types of nodes used in the three clusters were:

1) HP Integrity rx2600 dual-processor server node with Itanium2 ("Madison" revision) processors;
2) IBM e325 dual-processor server node with AMD Opteron processors;
3) Apple Power Macintosh dual-processor desktop system with Apple PowerPC G5 (IBM PowerPC 970) processors.

The four dual-processor nodes in each cluster were interconnected with Gigabit Ethernet. The Itanium2 cluster runs NPACI ROCKS 3.1, which is based on RedHat Linux for x86-64. The Opteron cluster was installed with SuSE Linux Enterprise Server 8 which contains the NUMA extensions to the Linux SMP kernel. NUMA enhancements in the kernel include memory affinity, which allows pages to be allocated on the memory closest to the CPU where a process is running. The G5 system runs the Mac OS X 10.3 operating system**.** MPI libraries were built using MPICH-P4 [1] from Argonne and LAM-MPI [2]. Unless otherwise indicated, the applications and benchmarks were built using the MPICH implementation. Table 1 below summarizes the hardware characteristics and node configuration for all three systems.

	HP Integrity rx2600	IBM e325 Model 246	Apple Power Mac G5
Processor	Intel Itanium 2	AMD Opteron	Apple G5
Processor Speed	1.5GHz	2.0GHz	2.0GHz
Floating Point	2 FMAs	2 Units (separate multiply and add units)	2 FMAs
SIMD		SSE2	Altivec—128bit
Prefetch	Hardware prefetch	Hardware prefetch (8 streams)	Hardware prefetch (8 streams)
Architecture	EPIC (64-bit)	x86-64	PowerPC
Peak 64-bit FLOPS/CP (clock period)	4 FLOPS/CP	2 FLOPS/CP	4 FLOPS/CP
Chipset	HP zx1	8111/8131	
Cache	L3 6MB: L2 256K: L1 16K/16K data/instr.	L2 1.0MB L1 64KB	L2 512KB L1 32K/32K data/instr.
Peak aggregate bus bandwidth	6.4GB/s 400MHz 16B-wide shared frontside bus (FSB)	10.6GB/s Each processor has dual 8B-wide channels to memory	16GB/s Each processor has a dedicated FSB, each 1GHz FSB has two 4B-wide unidirectional links
Memory bandwidth	four channels to DDR266 at 8B = 8.5GB/s	dual channels to DDR333 at 8B = 5.3GB/s for each processor	400MHz SDRAM at 16B = 6.4GB/s
Configured Memory DIMMs	512MB DDR266 (PC2100) ECC CL2 Reg. (8 DIMMS)	512MB DDR 333, CL2.5, ECC, Reg. (4 DIMMS)	256MB DDR400 (PC3200) CL3, not ECC nor Reg. (2 DIMMS)
Processor-processor interconnect	Shared front-side bus (FSB)	HyperTransport links	PowerPC G5 system controller

Table 1: 64-bit dual processors systems benchmarked for this paper

The HP rx2600 server is a symmetric multiprocessing (SMP) system consisting of dual Itanium2 processors. Each Itanium2 (Madison) processor has a clock speed of 1.5GHz and a 6MB integrated L3 cache [3]. The rx2600 uses the zx1 memory controller which supports four channels of DDR266 memory for a total of 8.5GB/s bandwidth. Both processors access memory through a shared 16B-wide 400MHz front-side bus (FSB), which delivers a peak bandwidth of 6.4GB/s. The rx2600 test systems were configured with a total of 4GB of DDR266 memory per node.

The IBM e325 system is based on the 2.0GHz AMD Opteron processor [4] with 1MB of cache and 1GB of DDR 333MHz memory for each processor. Built-in memory controllers on each chip provide a direct path to each processor's locally attached memory. Hence, the aggregate memory bandwidth is additive as long as data are accessed locally. Scaling in the memory coherence is achieved through a virtual chipset, interconnected via a high-speed transport layer, called HyperTransport [5]. Each processor can share its local memory and devices with the other processor through HyperTransport links, but each processor retains a local, direct path to its attached memory. This system is NUMA-like in architecture because of the small differences in latencies when accessing local versus remote memory.

The Apple G5-based dual-node desktop system is based on the 2.0GHz Apple PowerPC G5 processor [6] with 512K L2 cache. A single memory controller shares 512MB of DDR 400MHz memory between the two processors. This system features two independent 1.0GHz Elastic IO bus (EIO), each consisting of two dedicated, unidirectional 4B-wide data paths from memory to the processor. The theoretical peak bandwidth from memory to each processor is 8GB/s. An important processor feature is the embedded "velocity engine" (Altivec) vector processor executing SIMD operations on 128 bits

of data at once, with vector lengths of up to 32-bit subunits thus enabling four floating point operations per clock period when using 32-bit operands. Unless otherwise indicated, most of the micro-benchmarks and applications run on the G5 systems are 32-bit because the IBM compilers for Mac OS X v10.3 lack 64-bit memory support, but do use 64-bit floating point values.

3. Micro-Benchmarks

The micro-benchmarks were performed on a single node to obtain quantitative measures of system memory, CPU and interprocessor communication performance.

3.1 Floating-point Benchmarks

3.1.1 FLOPS

The floating-point benchmark FLOPS is a serial C program that estimates systems' MFLOPS rating for the FADD, FSUB, FMUL and FDIV operations. This provides an estimate of peak MFLOPS performance for a single floating-point unit (FPU) by making maximal use of register variables with minimal interaction with main memory.

Each of the summary results is based on weighted results from the eight different modules. MFLOPS(1), based on Modules 2 and 3, was difficult to vectorize completely due to recurrence of a variable in Module 2. The other main attribute in the summary results is the weightings given to FDIVs. The second problem with MFLOPS(1) centers around the percentage of FDIVs (9.6%) which was viewed as too high for an important class of problems. These problems were addressed in MFLOPS(2) which does not use Module 2, but maintains the same weightings for FDIVs. MFLOPS(3) does not include Module 2 but does fewer (3.4%) FDIV operations, while MFLOPS(4) does no FDIV operations, nor does it use Module 2. Thus, as we progress from MFLOPS(1) to MFLOPS(4), the number of FDIV operations is gradually reduced . In addition, the scale and extent of vectorization for the operations are also relatively increased. Table 2 shows the performance on one processor for each of the given MFLOPS outputs (in GFLOPS/s)

	Itanium2	Opteron	G5
Compiler	icc 8.0	pgcc 5.1.3	gcc 3.3
Compiler options	-O3 –tp p2	-O3 -fast	-O4
MFLOPS(1)	2.59	2.074	1.38
MFLOPS(2)	1.48	0.833	0.93
MFLOPS(3)	2.51	1.20	1.67
MFLOPS(4)	4.13	1.41	3.54

Table 2: FLOPS ratings in Gigaflops for the Itanium2, Opteron and G5 systems.

The Itanium2 system exhibits the best absolute performance overall, as well as optimally optimized hardware division. The G5 system exhibits the highest percentage increase at every step. The Opteron system lags in performance primarily because its FPU can do half the work that the other two systems can as is evident in the numbers.

3.1.2 High Performance Linpack

The High Performance Linpack (HPL) benchmark involves the solution of a dense linear algebra system. System performance is dependent on factors such as the quality of the BLAS library implementation of the DGEMM function and to a lesser extent the MPI implementation used and the

interconnect network speed. The algorithm used in Linpack is cache-friendly and is designed to preserve parallel efficiency as it scales to higher processor counts. The Linpack result is thus primarily an indicator of the floating-point performance of a computational system, and is used as the basis for ranking different platforms in the Top 500 list. Table 3 presents the HPL results in GFLOPS/s for the Itanium, Opteron and G5 systems.

	Itanium2	Opteron	G5
Compiler	icc/ifort 8.0	pgcc/pgf90 5.1.3 for x86-64	gcc 3.3
Compiler options	-O2 –tpp2 -nofor_main	-O3 -tp k8-64	-O3 -mtune=970 -mcpu=970 -W –Wall -Wl,framework,vecLib
MPI library	LAM 7.0.4	LAM 7.0.4	LAM-MPI 7.0.4
BLAS	Goto BLAS	Goto BLAS	Apple VecLib
Single node, two processors: PxQ=1x2 GFLOPS/s	10.52	5.815	7.949
Two nodes, two processors: PxQ=1x2 GFLOPS/s	10.28	5.869	8.225

Table 3: HPL performance numbers for the Itanium2, Opteron and G5 systems

The highest FLOPS count for HPL is obtained on the Itanium2 system, followed by the G5 and then the Opteron. Since HPL is floating-point intensive, it is expected to run best on systems with high processor speeds and/or multiple floating-point functional units. Both the G5 and Itanium2 processors have two FMA units each that can deliver four FLOPS per cycle, compared to two FLOPS on the Opteron. Secondly, the BLAS library's DGEMM function is crucial in obtaining a high percentage of the peak FLOPS. The high percentage of peak achieved by the Itanium2 (87%) and Opteron (73%) is due to the availability of the Goto BLAS library, which achieves good performance by tuning the data access to exploit the memory, cache and TLB structures [7]; however, the Goto BLAS was unavailable for the Apple G5 at the time of this study.

3.2 Memory subsystem benchmarks

3.2.1 Memory latency measurement

Memory latency plays a significant role in applications where random data access, not unit-strided or small-strided access, is the norm. In the latter case, activation of prefetch streams generally hides the latency associated with retrieving data from memory. We calculate the latency in accessing data from all levels of the memory hierarchy using the following loop:

```
ip1=ia(1)
do i=2,n
   ip2=ia(ip1)
   ip1=ip2
end do
```

The integer array **ia** is a random sequencing of the (32-bit) integers from 1 to **n**, arranged so that the loop will visit all the array elements once. Because an iteration is dependent on a value determined in the previous loop iteration, prefetching of data is not a factor. The time it takes to fetch one element of

array **ia** will of course depend on the array size. This elapsed time is indicative of the latency involved in referencing data in the level of the memory hierarchy where **ia** is stored.

Figure 1 shows the number of cycles it takes to access data at different levels of memory. Expressing the latency in cycles makes it easier to perform comparisons of the execution time of memory access (load/store) versus floating-point operations. On the Itanium2, it takes 3-4 cycles to access each element of **ia** when **ia** is resident in the L1 cache (i.e, vector length below 16K). The latency increases from 9-31 cycles when data are in the L2 cache. As **ia** spills over L2, there is a gradual rise in the latency from 60-300, at which point the latency starts to plateau.

On the Opteron, the latency of fetching data from L1 is around 4.5 cycles, while the latency for referencing L2-resident data ranges from 4.5-18 cycles. When **ia** is too big to fit in L2, the latency increases, but is lower than seen in the Itanium2 from 8MB onwards. However, the rate of increase from 256MB and up is steeper. At values close to 1GB, some of the memory access is no longer "local," and we begin to see the additional overhead of accessing "remote" memory.

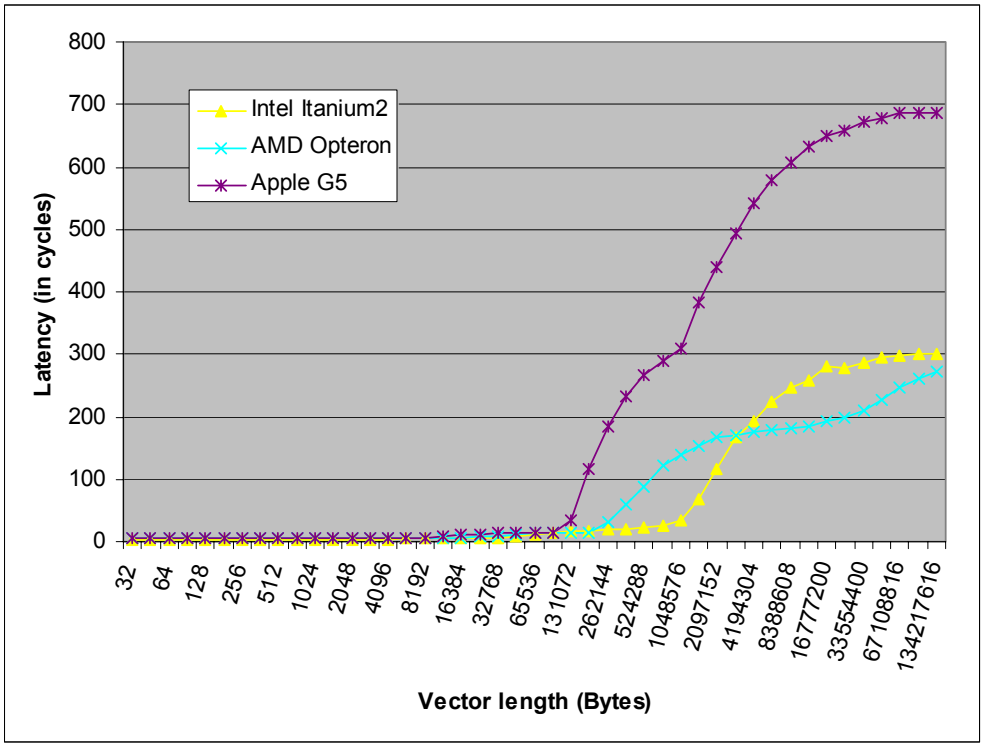

Figure 1: Latency plotted as a function of vector length in bytes.

The Apple G5 latency values are higher than those from the Opteron and Itanium2 whether expressed in terms of clock rates or nanoseconds. It takes about 7 cycles to fetch data from L1, whereas L2 data requires about 8-32 cycles. There is an abrupt jump to > 100 cycles when data spills out of L2. The latency of referencing data in memory increases steeply as the size of array **ia** gets larger.

3.2.2 STREAM

The STREAM benchmark [8] measures the memory bandwidth for the following four kernels:

```
COPY:     a(i)=b(i)
SCALE:    a(i)=q*b(i)
```

```
SUM:      a(i)=b(i)+c(i)
TRIAD:    a(i)=b(i)+q*c(i)
```

A vector length of 16777216 was used to generate the results in Table 4 and 5. Table 4 below lists the serial STREAM results:

	Itanium2	Opteron	G5
Compiler	icc 8.0	pgcc 5.1.3	xlc
Compiler options	-O3 –tpp2 –ftz –fno-alias	-O3 –Mvect=prefetch –Mnontemporal -Munroll -fastsse	-O5 –qarch=ppc970 –qtune=ppc970 –qaltivec -qunroll=auto –qcache-auto
Copy BW (MB/s)	3380.8	3657.2	3012.0
Scale BW (MB/s)	3288.4	3604.8	2670.7
Sum BW (MB/s)	3796.7	3602.5	2577.0
Triad BW (MB/s)	3869.5	3621.8	2573.2

Table 4: Serial STREAM results for the Itanium2, Opteron and G5.

Next, we ran the OpenMP version of STREAM using two threads on a single node. The dual-processor bandwidth measurements are presented in Table 5.

	Itanium2	Opteron	G5
Compiler	icc 8.0	pgcc 5.1.3	xlc_r
Compiler options	-O3 –tpp2 –ftz –fno-alias –openmp	-O3 -mp -fastsse -Mnontemporal	-O5 –qarch=ppc970 –qarch=ppc970 –qcache=auto –qhot=vector
Copy BW (MB/s)	3872.0	6394.2	3000.5
Scale BW (MB/s)	3663.8	6735.5	2882.6
Sum BW (MB/s)	3271.9	6622.0	2992.5
Triad BW (MB/s)	3465.6	6650.7	2963.5

Table 5: Multithreaded STREAM performance on the Itanium2, Opteron and G5 systems.

The serial STREAM numbers for the Itanium2 are higher than those obtained from the G5 and Opteron. This is not unexpected as a single processor should be able to exploit the full FSB bandwidth when there is only a single STREAM process/thread running on the node. However, the dual-threaded STREAM bandwidths do not scale very well for Itanium2. This can be attributed to contention between the two threads for the system bus bandwidth.

On the Opteron, the serial STREAM results are very good, considering that the maximum memory bandwidth per process is 5.3GB/s. In going from serial to dual-threaded STREAM, the bandwidth increased by an average factor of 1.5. The presence of the on-chip memory controllers provide each processor with direct access to its local memory, which facilitates good scaling especially when data are stored in the memory local to each processor.

The STREAM results for G5 are significantly less than the advertised peak memory bandwidth. Although each of the two independent FSBs can deliver 1GB/s of bandwidth to each processor, the available bandwidth is limited by the speed of the shared memory controller, which can support a

maximum of 6.4GB/s. The STREAM numbers are less than half of this value for the serial and dual-threaded versions.

3.3 On-Node MPI benchmarks

The PRESTA benchmarks [9] measure, among other things, the latency along with the unidirectional and bidirectional bandwidths of point-to-point MPI communication. Since all communications are based on the particular MPI implementation over Ethernet, the bandwidths and latency across nodes are primarily a function of the network (i.e., limited by the Gigabit Ethernet bandwidth and the overhead associated with the TCP/IP stack) and have relatively little to do with the systems. For that reason we focused our attention on the on-node bandwidth and latencies for the dual-processor nodes. Table 6 shows the results obtained from running the benchmarks on a single node. Bandwidth measurements for all the systems were obtained using the LAM-MPI implementations, using the shared memory RPIs.

	Itanium2	Opteron	G5
Max. unidirectional BW (GB/s)	3.06	0.781	1.23
Max. bidirectional BW (GB/s)	2.93	1.25	1.32
Latency (microseconds)	2.9	0.61	0.71

Table 6: On-node MPI latencies and bandwidths measured using PRESTA.

The Itanium2 systems have the largest measured bandwidths for both uni- and bidirectional communication using shared memory by more than a factor of 2. The uni- and bidirectional values are approximately the same, since in the bidirectional case both processes communicate through a shared system bus. The processes must share the bus bandwidth when both are sending messages simultaneously. In the unidirectional case, a single task is able to use most of the available system bus bandwidth when communicating to the other processor.

The latency of shared-memory communication is lowest on the Opteron system. The bidirectional bandwidth on the Opteron is about twice that of the unidirectional bandwidth because of the nature of the HyperTransport interconnect. HyperTransport consists of dual unidirectional links, capable of delivering an aggregate interprocessor bandwidth of 6.4GB/s. Thus, unidirectional message passing should achieve 3.2GB/s at most, while bidirectional data transfer should have a peak rate of 6.4GB/s. The 1:2 ratio of unidirectional to bidirectional MPI bandwidth is observed here. For both uni- and bidirectional bandwidth measurements, less than 25% of the peak bandwidth is realized. Since memory coherence checking is enforced at the HyperTransport level, the overhead associated with snooping on data traffic across HyperTransport could adversely affect the data transfer rate between processors. The lower bandwidth numbers on this system relative to Itanium2 is likely caused by the more “intrusive” coherency checks on the Opteron node.

The G5 system has good latency for shared-memory communication and comparable bandwidth to the Opteron system in one or two directions. The use of two independent FSBs is an excellent design feature. However, the single memory controller for both processors undercuts the advantage gained from the separate FSBs, which explains the lower bandwidths observed on this system relative to the Itanium. This also accounts for the identical bandwidths for uni- and bidirectional communication.

4. Application Benchmarks

To assess how well applications map to a given architecture, we picked several scientific codes that differ in their computational characteristics (e.g., memory-bound, FP-bound, etc.). Table 7 contains a list of the applications we analyzed. Note that each table entry denotes how dependent application performance is on the specified system resource; for example, typical molecular dynamic applications will benefit greatly from higher CPU speeds.

Code	Application area	Memory BW	CPU	Interconnect BW
Snoop3D	CFD	low	high	moderate
MGF	Finite element	high	high	moderate
POP	Ocean modeling	high	high	low
GAMESS	Chemistry/Physics	high	high	high
NAMD	Chemistry/Biochemistry/ Biology	moderate	high	high
HMMER	Bioinformatics	low	high (integer operations-intensive)	none

Table 7: Applications and their memory, CPU and interconnect bandwidth requirements.

Unless otherwise indicated, the execution time reported for each application represents the best in a series of runs under identical environments. All runs are performed in dedicated mode on each node. Dual-processor runs were executed on a single node and across two nodes to compare application performance with shared-memory communication versus communication over the network.

4.1 3-D CFD Application

One of the CFD applications benchmarked for this paper is a 3-D unsteady, incompressible flow solver using unstructured meshes [10]. While this flow solver does not require a significant amount of memory, it does perform a large number of floating-point operations during a simulation run. The timings for the three processors are shown in Table 8.

	Itanium2	Opteron	G5
Compiler	icc 8.0	gcc 3.0	xlc
Compiler options	–O3 –ip –fno-alias -ftz	–O3 –ffast-math –fargument-noalias	–O4 –qnoipa –qmaxmem=-1 –qstrict
MPI library	MPICH	MPICH	MPICH
Execution Time (s)	212.91	154.24	157.12

Table 8: Execution time (in seconds) for 3-D CFD solver.

The timings were obtained by averaging the time of four separate runs, each simulating one second of the fluid flow which corresponds to 50 iterations of the solver. For this application, the Opteron and G5 both performed very well due to their high clock speeds. The Itanium2 processor lagged behind the other two by approximately the ratio of clock speeds.

4.2 3-D Finite Element Application

MGF [11] is a parallel code for 3-D multi-physics problems. Originally developed under a NASA Grand Challenge project for high-performance simulation of thermocapillary flows in microgravity environments, MGF has become a general-purpose tool for partial differential equation problems via a "dial-an-operator" interface. MGF solves these systems using Galerkin and Mixed-Galerkin Finite Element Methods on tri-linear and tri-quadratic hexahedral elements. Table 9 shows the timings of the MGF code on the three different systems.

	Itanium2	Opteron	G5
Compiler	icc 8.0	gcc/g++ 3.3	gcc/g++ 3.3
Compiler options	-O2	-O2	-O2 -faltivec -framework
BLAS library	Goto BLAS	Goto BLAS	Veclib
Serial execution time (s)	74.01	105.65	151.57
Two processors, on-node (s)	45.50	56.90	94.69
Two processors, across two nodes (s)	38.61	57.08	76.75

Table 9: Execution time (in seconds) for serial and parallel versions of MGF.

The most intensive calculation done on MGF consists of calls to the matrix-vector multiplication routines in the BLAS library. Hence, its performance is impacted by the quality of the BLAS DGEMV implementationThe two systems using the Goto BLAS library performed best. The Itanium2 system provided the best performance for MGF, with the G5 system coming in last. Generally, the Itanium2 attains a high percentage of its peak FLOPS with dense matrix/linear algebra operations (e.g., see Linpack benchmark above). Even though the VecLib library was used for the G5, the double-precision floating-point calculations cannot take advantage of the SIMD instructions available on the G5 Altivec units, which are limited to single-precision calculations.

4.3 POP

POP (Parallel Ocean Program) [12] is an ocean circulation model that solves the three-dimensional primitive equations for fluid motions on the sphere under hydrostatic and Boussinesq approximations. Spatial derivatives are computed using finite-difference discretizations formulated to handle any generalized orthogonal grid on a sphere, including dipole and tripole grids which shift the North Pole singularity into land masses to avoid time-step constraints due to grid convergence.

Serial and parallel versions of POP are available, with the latter version utilizing either MPI or SHMEM libraries for parallelization. A hybrid message-passing and threaded version of the code is supported as well. Note that only the serial and MPI versions of the code were benchmarked in this study. Table 10 below lists the execution time of POP on the three platforms:

	Itanium2	Opteron	G5
Compiler/MPI library	ifort/MPICH	pgf90/MPICH	xlf90/MPICH
Compiler options	–O3 –ip –fno-alias	–O3	O4 –qnoipa –qmaxmem=-1 –qstrict
Serial (s)	107.65	63.82	41.67
Two processors, on-node (s)	55.24	35.40	30.41
Two processors, across two nodes (s)	56.04	33.27	25.64

Table 10: Execution time (in seconds) of serial and parallel versions of POP on the Itanium2, Opteron and G5 systems.

POP is floating-point intensive and should perform well on systems with fast processor speeds. The results are consistent with this description: the best performance for POP across all runs (serial and dual processor) is on the G5 system, which is based on the 2.0GHz G5 processor, followed closed by the 2.0GHz Opteron system. The presence of two FMA units on the G5 processor probably (capable of delivering 4 FLOPS/cycle) helps POP in achieving better performance on that system than the Opteron system. Scaling of the application from one to two processors is especially good on the Opteron nodes due to the on-chip memory controller that gives each processor an independent path to memory. This also explains why the dual-processor runs give comparable execution times whether the two processes are executed on-node or across nodes. The worst performance is on the Itanium2 system due to the slower clock speed. Scaling from serial to two processors is good notwithstanding the use of a shared memory bus which can cause contention among two processes executing on a single node. Although the application is somewhat memory-intensive, the lack of contention among the two competing processes on-node may indicate inefficiency in the use of the functional units. Floating-point stalls may be greater than those resulting from cache misses and load/stores from memory.

4.4 GAMESS

The General Atomic and Molecular Electronic Structure System (GAMESS) [13] is a general *ab-initio* quantum chemistry package from Iowa State University. GAMESS supports a wide range of quantum chemical computations including single-point energies and wave functions using RHF, UHF and ROHF; electron correlation energy correction to SCF using coupled-cluster, configuration interaction (CI) and second-order many-body perturbation theory (MP2) methods; molecular geometry optimizations; analytic energy gradients and Hessians; and others.

GAMESS supports parallel execution of many of the operations listed above, using either the MPI library or a TCP/IP sockets implementation. The GAMESS documentation recommends the use of sockets on clusters with Gigabit interconnect, so this is the implementation used in all of the tests. For the benchmark run, a modified version of example #25 in the GAMESS test suite was used as input (the basis set was changed from AM1 to 6-31G). GAMESS was executed serially and in parallel on two processors. The timing results for the three systems are listed in Table 11..

	Itanium2	Opteron	G5
Compiler	ifort/icc 8.0	pgf90/pgcc 5.1.3	xlf90/xlc
Compiler options	-O2	-O3 –fastsse -Mnontemporal	- O3 –qhot –qtune=ppc970 –qarch=ppc970
Serial (s)	289	178.7	136.0
Two processors, on-node (s)	145	102.5	75.5
Two processors, across two nodes (s)	122	95.2	---

Table 11: Execution times (in seconds) for serial and parallel versions of GAMESS on the Itanium2, Opteron and G5 systems.

Because GAMESS is mostly numerically intensive, it is expected to do well on systems characterized by fast clock speeds and multiple floating-point functional units capable of completing many floating-point operations per clock cycle. Using these criteria, the G5 system should present an advantage because it is based on two 2.0GHz processors, each capable of executing four FLOPS per clock period. In fact, this is what is observed with the serial and on-node parallel runs of GAMESS on the G5 when the best compiler options are used. The dual-processor run across two nodes could not be performed on the G5 system, but it is expected that this timing would also have been the best. All of the GAMESS test runs on the Opteron system outperform those on the Itanium2. This is surprising because the

Opteron system, while based on two 2.0GHz Opteron CPUs, supports only two FLOPS/clock period per processor. The Itanium2 system supports twice as many FLOPS per cycle, albeit at a slower clock speed. One factor which did not work to the Itanium2's advantage is the level of optimization used to generate the GAMESS code. The use of more aggressive optimization options (e.g., -O3 –fast –ftz –fno-alias) on the Itanium2 was not explored here because of errors that occurred at either the link step or at runtime.

4.5 NAMD

NAMD[14] is a parallel molecular dynamics code targeted towards high-performance simulation of large biomolecular systems. The dynamic components of NAMD are based on the Charm++ library, a platform-independent parallel programming system designed with communication latency tolerance and dynamic load balancing features.

NAMD and its prerequisites (fftw, TCL, Charm++) were built from source on all three systems, although precompiled binaries were used to generate the performance numbers on the G5 because of runtime memory problems with the native build. The timing results presented in Table 12 are in seconds and were obtained using the NAMD benchmark [14] involving a system of 23,558 atoms (with a 9Å cutoff).

	Itanium2	Opteron	G5
Compiler	icc/icpc 8.0	gcc/g++ 3.3	gcc/g++
Serial (s)	1104	1978	2583
Two processors, on-node (s)	578	1029	1372
Two processors, across two nodes (s)	576.2	1101	1332

Table 12: NAMD execution time (in seconds) on Itanium2, Opteron and G5 systems.

NAMD exhibits great scalability up to four processors in all platforms. Molecular dynamics codes generally have high FLOPS-to-data access ratio, which accounts for the almost 1.9x scaling on all three systems in going from serial to on-node dual-processor run. Memory contention between the two on-node processes does not play a significant role, even in the Itanium2 system where the shared memory bus can potentially represent a bandwidth bottleneck. The TCP implementation of NAMD does not use shared memory communication, as is evident from the almost identical numbers obtained when running two processes on node vs. across two nodes.

The floating-point-intensive nature of the NAMD code makes it suitable to be executed on the Itanium2 and G5 platforms, which can support a maximum of four FLOPS per cycle. The Itanium2 system exhibits the best performance by more than a factor of 2. The Intel compilers do a good job of generating code optimized to the EPIC architecture. This entails detection and exposure of instruction level parallelism (ILP) in the code, which leads to optimized scheduling of instructions and hence efficient use of the functional units.

Surprisingly, the execution time of NAMD took longer on the G5 system than on the Opteron, given that the former system can execute twice the number of floating-point operations per clock relative to the latter. This lack of performance may be attributed to the fact that the prebuilt binary was actually compiled on a G4 system, which could adversely affect the binary's runtime performance on the G5. Moreover, the g++ compiler on the G4 cannot be expected to generate code that will run optimally on the G5 system.

4.6 HMMER

HMMER [15] is an implementation of profile hidden Markov models (HMMs) for biological sequence analysis. Profile HMMs provide statistical models of sequence alignments and have a probabilistic basis, as opposed to the heuristic models employed by BLAST [16] and FASTA [17].

HMMER has two parallel implementations: one is based on pthreads while the other uses PVM. A version using MPI is presently unavailable but is coming soon according to the developers. For the HMMER test runs, **`hmmpfam`** was executed on one and two threads on a single node to search for domains in the **`7LES_DROME`** (Drosophila Sevenless) sequence. The results of these runs (execution time is in seconds) are presented in Table 13.

	Itanium2	Opteron	G5
Compiler	icc 8.0	pgcc 5.1.3	gcc 3.3
Compiler options	-O3 –mcpu=itanium2 –fno-alias	-O3 –fastsse -Mnontemporal	-O3 –mcpu=G5 –mtune=G5
Serial (s)	136.6	191.0	178
Dual-threaded (s)	71.2	85.7	115

Table 13: Execution time (in seconds) of serial and multithreaded versions of HMMER on the Itanium2, Opteron and G5 systems.

Serial HMMER performs best on the Itanium2 system with the G5 and Opteron systems a distant second and third in the serial run. The parallel execution on two threads is still fastest on the Itanium2 system, with the Opteron system now outperforming the G5. The hand-tuned HMMER code for the G5 [18] was not used in this study for reasons mentioned in the next paragraph.

HMMER is not particularly memory-intensive but does involve lots of integer operations. It should do well on systems that have multiple fixed-point (integer) functional units (the G5 and Itanium2 have two integer units each, the Opteron has three) and support multimedia operations. Potentially, HMMER should run best on the G5 because of its support for vectorization using Altivec instructions and the 128-bit Altivec register. Note that Opteron's support of SSE2 should also help accelerate HMMER's performance. In fact, Eric Lindahl's hand-tuned version of HMMER for G5 [18] performs very well on this platform, with most of the speedup attributed to the use of the Altivec engine. This modified code was not used here in the interest of obtaining a fair comparison of the performance of the unmodified version of the HMMER code across all three platforms. Nonetheless, the "out-of-the-box" source code exhibits the best performance on the Itanium2 platform in both serial and parallel runs. The lack of hardware vectorization support on Itanium2 does not affect the performance of HMMER adversely relative to other platforms with vector support in hardware. Of course, most compilers perform software vectorization. The Intel compilers for Itanium2 do a great job of generating optimized code that utilize the functional units efficiently, in both serial and parallel runs.

HMMER on Opteron does not perform as well as on G5 and Itanium2 in the serial case. There is superlinear speedup in the 2p case, which may be due to cache effects (i.e., dividing the data among two processors causes data previously stored in main memory to fit in cache). Scalability from one to two threads on the G5 is not as good as on the other two systems. As HMMER is not especially memory-intensive, this could be due to a better implementation of the threads library on the Opteron and Itanium2 platforms.

5. Summary and Observations

The introduction of 64-bit HPC systems based on commodity processors will presents new opportunities for cluster computing. Because 64-bit systems are not restricted by the 4GB limit in addressable memory per process, it is now possible to equip cluster nodes with larger amounts of

memory (> 4GB/node). The presence of more on-node memory is especially beneficial to SIMD applications that scale poorly, since data does not need to be decomposed among larger numbers of processors in order to accommodate the 4GB memory restriction. Note that none of the test systems used in this study had more than 4GB of memory per node, so testing of this aspect was not done. Also, addressability beyond 2GB is not limited to memory but extends to files as well. However, large file support in Linux has been around for some time, and hence file addressability beyond 2GB is available in 32-bit space and not unique to 64-bit computing.

64-bit computing may start at the hardware level but hardware support is only one of the requirements for a 64-bit programming environment. A true 64-bit computing environment requires 64-bit support on all levels of the computational infrastructure --- from hardware at the lowest level to the operating system, compilers, libraries, include files, debuggers, etc. The 64-bit application binary interface is necessary for developers to produce applications that can harness the full potential of the underlying 64-bit hardware.

The Apple Power Mac G5 system, which is based on the 64-bit G5 processor, does not fully support a 64-bit environment at the current time. Mac OS X is still a 32-bit operating system. Compilers, libraries and include files (both xlc/xlf and gcc/g77) are all 32-bit and hence does not support the development of applications that go beyond the 4GB memory limit. In contrast, the Itanium2 system is fully 64-bit: it is based on the 64-bit Itanium2 processor and runs a 64-bit Linux operating system with 64-bit compilers and libraries. The Opteron system also presents a full 64-bit computing environment, from the hardware level (using the 64-bit AMD Opteron processor) to the OS, compilers, tools and libraries. This system has the additional advantage of supporting both 32- and 64-bit computing environments: 32-bit applications do not have to be recompiled to run on the 64-bit OS (though they cannot address more than 2GB unless recompiled). The added flexibility of running both 32- and 64-bit applications is welcome news to developers who are just starting to port their applications from 32- to 64-bit mode. Also, this provides users with the option of building 32-bit versions of third-party applications that have not been fully ported to 64-bit.

There are advantages and disadvantages to programming in 64-bit vs. 32-bit mode. In some systems like the Opteron, the 64-bit environment presents additional advantage over 32-bit aside from the ability to access larger amounts of memory. The x86-64 ISA has double the size and number of the general purpose (GPR) and SIMD registers (SSE1 and SSE2) relative to those available in x86. There are sixteen 64-bit GPRs in x86-64 compared to eight 32-bit GPRs in x86. The SSE/SSE2 registers also increased in size and number, from eight 32-bit registers in x86 to 16 64-bit registers in x86-64. This increase should enhance application performance, especially in deeply pipelined architectures that depend on available registers to store intermediate results from different segments of the instruction pipeline. Codes that are rich in loops (with no data dependency between iterations) should benefit, as these loops could be vectorized more readily and unrolled to greater depths. In addition, the hardware support for 64-bit integers is a performance-enhancing benefit for scientific applications which require 64-bit integer arithmetic. The capability for performing 64-bit integer operations with hardware support is available in all 64-bit platforms, not just the AMD Opteron.

On the other hand, 64-bit codes are larger than their 32-bit counterparts. The migration from 32- to 64-bit mode entails a change in the size of datatypes. Pointers and longs double in size from 32 to 64-bits, and so do structures and other derived types that are based on these datatypes. The size of the instructions is also increased because of the larger address size. As a result, instructions and data occupy a bigger cache footprint, thus effectively decreasing cache size and increasing the number of load/store operations from main memory.

The first part of this study characterized the low-level features of the test systems, using a combination of micro-benchmarks that measure memory bandwidth and latencies from different levels of the memory hierarchy, as well as the floating-point performance and the bandwidth of interprocessor communication using shared memory. The bandwidth from main memory was measured using the STREAM benchmark. Serial STREAM results were good on the Itanium2 and Opteron systems. On the Itanium2, a single processor is capable of utilizing the entire bandwidth of the shared system bus,

so a serial process can use as much as 6.4GB/s of bandwidth. On the Opteron system, a single process can take advantage of 5.3GB/s of bandwidth from memory. When STREAM was executed on two threads per node, the Opteron system exhibited the best performance through excellent scaling, due to the presence of on-chip memory controllers which provide each CPU with an independent, dedicated path to its local memory. Memory latency exhibits a characteristic, multi-hump profile, where each hump corresponds to a different level of memory. In all three cases, loads and stores from main memory are more than 200 cycles away. For comparison, floating-point operations take less than 5 cycles on all three systems that were studied. The G5 had the worst memory latency as well as STREAM performance, with the Intanium2 and Opteron having comparable latency

Floating-point performance was measured using FLOPS and the Linpack benchmark. The Itanium2 system dominated in both micro-benchmarks. An Itanium2 processor has two FMA units that can deliver four FLOPS per clock cycle. This helps in tests like FLOPS which contain a large number of addition and multiplication operations. In HPL, the use of the highly optimized Goto BLAS library in both the Itanium2 and Opteron runs ensures that a high percentage of the peak FLOP rating is attained.

The bandwidth and latency of interprocessor communication was measured using the PRESTA benchmark. In all three cases the LAM-MPI implementation of MPI was used to generate the PRESTA binaries and all of the on-node communication was done using the shared memory modules in LAM. The Itanium2 system is the clear winner when it comes to interprocessor bandwidth. The Hypertransport links which serve as interconnect for the two Opteron processors, while supporting a peak bidirectional bandwidth of 6.4GB/s, is somewhat hampered by cache snooping/memory coherence checking. Similarly, on the G5, the single memory controller hampered the true scaling in the bi-directional bandwidth from memory.

To assess how well applications map to the different platforms, six applications in different areas with varying computational requirements were benchmarked. These applications included a 3-D unsteady, incompressible flow solver (Snoop3D), a 3D finite element code (MGF), an ocean modeling code (POP), an ab-initio quantum mechanical application (GAMESS), a molecular dynamics package (NAMD) and a sequence analyis code used in bioinformatics (HMMER). All applications are parallel (except for Snoop3D which was run in serial mode only) and CPU-intensive but varied in their memory and communication demands. Three of the applications (MGF, NAMD and HMMER) performed best on the Itanium2 system. However, the performance of two other codes (Snoop3D, POP) was significantly worse on the Itanium2 compared to the other two systems. Since all of the applications considered here are floating-point (or integer) intensive, they are expected to perform very well on the G5 and Itanium, each of which is capable of four FLOPS/cycle of peak performance compared to the Opteron's two FLOPS/cycle. However, the Opteron outperformed one or both systems in all six applications, sometimes by a solid margin as in the case of GAMESS, where the Itanium2 system performed poorly, and on NAMD where the performance on G5 is subpar.

In terms of memory-intensive applications, scaling from serial to dual-processors on a single node is best on the Opteron system. This can be attributed to the Opteron system's memory architecture, which allows each CPU independent access to its attached memory. Great scaling is achievable provided that memory affinity-support is enabled on the kernel, which causes pages used by a processor to be allocated on the memory closest to that processor. Although the G5 system architecture features dual independent FSBs, the same level of scaling is not observed here. The culprit is a memory controller that is shared between the two CPUs, which restricts memory access to only one CPU at a time.

The serial performance is dominated by the Itanium2, where three of the six applications had the shortest execution time. Both CPU- and memory-bound applications should run well on Itanium2 since a single process will be able to utilize the two FMAs and take advantage of the full 6.4GB/s of bandwidth on the shared FSB.

In HPC clusters, the nodes will often be configured as dual-processor systems, so the dual-processor performance results are more important than the single processor results. These results are more affected by the memory system, since both processors are accessing the same memory pool. In the case

of dual processor systems, the results show the Opteron to exhibit the best scalability when it comes to memory access, mainly because of its memory subsystem architecture. Both the Itanium2 and G5 systems dominate in floating-point intensive calculations because of their dual floating-point units that are capable of performing four FLOPS (two combined multiply-and-add operations) every clock cycle. The specific characteristics of scientific applications should dictate choice of optimal architectures rather than the other way around, as exhibited by different performance "winners" on different applications

Another important characteristic which should also be considered by the community is price/performance ratio or when price differences are significant relative to performance gain. Based on the results obtained for the benchmarks in this paper and the web advertised prices as of March 25, 2004, the overall winner in terms of dual processor configurations is the Opteron system, with the G5 second and Itanium2 third. The lowest price found on the web for an IBM e325 Opteron system with the same configuration shown in Table 1 had a list price of $2,796 while the similar Apple Power Mac G5 lists for $2,999. The lowest price for the HP rx2600 Itanium2 system found on the web was $8,570. While the Opteron was the best price-performer, it is clear that the choice of processor really depends on the applications that will be used most often on the system. Obviously if NAMD were the application being run, an Itanium2 node would perform up to 45% faster than an Opteron. We highly recommend studying the typical applications which will be run on the system and finding/requesting benchmarks for these applications before deciding on any large scale cluster purchase.

References

1. MPICH, http://www-unix.mcs.anl.gov/mpi/mpich/
2. LAM-MPI, http://www.lam-mpi.org/
3. Intel Itanium2 processor, http://www.intel.com/business/bss/products/server/itanium2/quick_reference.pdf
4. AMD Opteron processor, http://www.amd.com/us-en/Processors/ProductInformation/0,,30_118_8796_8804,00.html
5. AMD HyperTransport Technology-Based System Architecture, http://www.amd.com/usen/assets/content_type/white_papers_and_tech_docs/AMD_HyperTransport_Technology_based_System_Architecture_FINAL2.pdf
6. Apple G5 processor, http://www.apple.com/G5processor/
7. High Performance BLAS Library, http://www.cs.utexas.edu/users/flame/goto/
8. STREAM benchmark, http://www.cs.virginia.edu/stream/
9. PRESTA benchmark, http://www.llnl.gov/asci/purple/benchmarks/limited/presta/
10. K. Schulz and Y Kallinderis, "Unsteady Flow Structure Interaction for Incompressible Flows Using Deformable Hybrid Grids," *Journal of Computational Physics*, Vol. 143, pp. 569-597, 1998.
11. Barth, W., Carey, G. F., Kirk, B., and McLay, R., "Parallel Distributed Solution of Viscous Flow With Heat Transfer on Workstation Clusters," High Performance Computing '00 Proceedings, Washington, DC, April 2000.
12. Parallel Ocean Program, http://climate.lanl.gov/Models/POP/
13. GAMESS, http://www.msg.ameslab.gov/GAMESS/GAMESS.html
14. NAMD, http://www.ks.uiuc.edu/Research/namd/
15. HMMER, http://hmmer.wustl.edu/
16. BLAST, http://www.ncbi.nlm.nih.gov/BLAST/
17. FASTA, http://fasta.bioch.virginia.edu/
18. Benchmarks on G5 from the Apple website, http://www.apple.com/powermac/performance/