

Performance Evaluation of the SGI Altix 3700

Thomas H. Dunigan, Jr.

Jeffrey S. Vetter

Patrick H. Worley

*Oak Ridge National Laboratory
Oak Ridge, TN, USA 37831
{dunigan,vetterjs,worleyph}@ornl.gov*

Abstract

SGI recently introduced the Altix 3700. In contrast to previous SGI systems, the Altix uses a modified version of the open source Linux operating system and the latest Intel IA-64 processors, the Intel Itanium2. The Altix also uses the next generation SGI interconnect, Numalink3 and NUMAflex, which provides a NUMA, cache-coherent, shared memory, multi-processor system. In this paper, we present a performance evaluation of the SGI Altix using microbenchmarks, kernels, and mission applications. We find that the Altix provides many advantages over other non-vector machines and it is competitive with the Cray X1 on a number of kernels and applications. The Altix also shows good scaling, and its globally shared memory allows users convenient parallelization with OpenMP or pthreads.

1 Introduction

Computational requirements for many large-scale simulations and ensemble studies of vital interest to the Department of Energy (DOE) exceed what is currently offered by any U.S. computer vendor. As illustrated in the DOE Scales report [9] and the High End Computing Revitalization Task Force report [3], examples are numerous, ranging from global climate change research to combustion to biology.

IBM, HP, and an array of cluster vendors have dominated recent HPC procurements in the US. Their HPC systems are clusters of 2 to 32 processor SMP nodes connected with a high speed interconnect where applications use MPI to communicate between the SMP nodes. Both Cray and SGI have recently introduced systems that compare favorably with these clusters, and both Cray and SGI have a number of unique characteristics that are attractive for certain classes of applications. For example, both systems support globally shared memory where any processor can access any memory location in the global address space.

The SGI Altix system is a large shared memory system. Initial offerings were as large as 256 processors, and plans are for it to scale into the 1000s of processors. In contrast to previous SGI systems, the Altix uses a modified version of the open source Linux operating system and the latest Intel IA-64 processor, the Itanium2. The Altix also uses the next generation SGI interconnect, the NUMALink3, which is the natural evolution of the highly successful interconnect used in previous generation SGI systems. For SGI, the Altix is a combination of traditional strengths (large SMP nodes and fast interconnects) and risk (new processors and new operating system).

This report describes the initial evaluation results collected on an SGI Altix system sited at ORNL. Results are also publicly available from the ORNL evaluation web site [7].

2 Evaluation Overview

The primary tasks of the evaluation project are to 1) determine the most effective approaches for using the SGI Altix, 2) evaluate benchmark and application performance, and compare with similar systems from other vendors, and 3) predict scalability, both in terms of problem size and in number of processors.

We employ a hierarchical approach to the evaluation, examining low-level functionality of the system first, then using these results to guide and understand the evaluation using kernels and compact or full application codes.

Standard benchmarks are used as appropriate to ensure meaningful comparisons with other system evaluations; however, the emphasis of our evaluation is studies of a number of different important DOE applications, as noted below.

The distinction here is that the low-level benchmarks, for example, message passing, and the kernel benchmarks are chosen to model important features of a full application. This is important in order

to understand application performance and to predict scalability.

3 SGI Altix 3700 Overview

The initial offering of SGI Altix is a shared memory system made up of 2 processor nodes interconnected with the first implementation of the SN2 "scalable node architecture". This architecture supports 64 TB of addressable memory. At introduction, the SGI modification of Linux supported a single system image on up to 64 processors, but this increased to 256 processors during the course of the ORNL evaluation. Multiple Linux kernels reside in a Coherent Sharing Domain (CSD), which provides cache coherence for up to 512 processors. Multiple CSDs can reside within a single system, with the NUMALink layer of SN2 providing high bandwidth and low latency communication within and between CSDs. Optimized communication libraries provide coherent access both within and between partitions defined by the OS images.

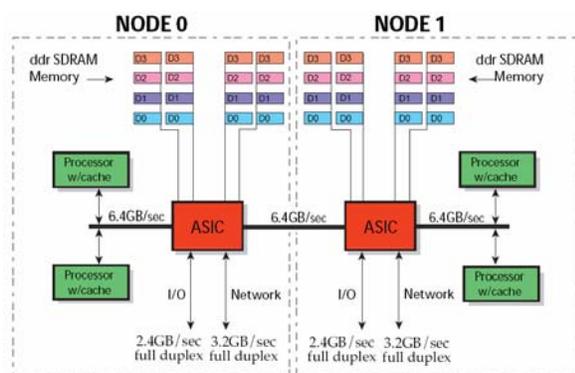


Figure 1: SGI Altix C-brick. (Image courtesy of SGI.)

Unlike commodity Linux clusters, SGI's cache-coherent, shared memory, multi-processor system is based on NUMAflex, a non-uniform memory access (NUMA) architecture, which has proven to be a highly-scalable, global shared memory architecture in SGI's Origin 3000 systems. In fact, the Altix 3000 uses many of the same components — called *bricks* — as the Origin. These bricks mount in racks and may be composed in various combinations to construct a system balanced for a specific workload. SGI offers several different types of bricks; Table 1 lists these bricks.

The Altix C-brick (Figure 1) consists of two nodes, each containing two Intel Itanium 2 processors with their own cache. These Altix C-bricks are different from those in the Origin because the Origin C-bricks contain MIPS processors, while the Altix C-

bricks contain Itaniums. Custom ASICs -- called SHUBs -- connect to the front-side buses of these processors. The SHUBs link the two processors to the memory DIMMs, to the I/O subsystem, and to other SHUBs via the NUMAflex interconnect. The SHUBs also interconnect the two nodes in a C-brick at the full bandwidth of the Itanium 2 front side bus (6.4 GB/sec).

Table 1: SGI Alinx brick types.

Brick Type	Purpose
C-Brick	computational module housing CPUs and memory
M-Brick	Memory expansion module
R-Brick	NUMAflex router interconnect module
D-Brick	Disk expansion module
IX-Brick	Base system I/O module
PX-Brick	PCI-X expansion module

The global shared memory architecture, implemented through SGI's NUMALink interconnect fabric, provides high cross-sectional bandwidth and allows performance scaling not usually obtained on commodity Linux clusters. While some coarse-grained applications scale well on Linux clusters, others need the high bandwidth and very low latency offered by a machine like the Altix. Still, users often feel that their applications are best implemented as shared-memory applications with OpenMP or pthreads using many processors.

Table 2: System configurations.

	SGI Altix	Alpha SC	IBM SP3	IBM SP4	Cray X1
Name	Ram	LeMieux	Eagle	Cheetah	Phoenix
Proc	Itanium 2	Alpha EV67	POWER3-II	POWER4	Cray X1
Interconnect	Numalink	Quadrics	Colony	Colony	Cray X1
MHz	1500	667	375	1300	800
Mem/Node	512GB	2GB	2GB	32GB	16GB
L1	32K	64K	64K	32K	16K (scalar)
L2	256K	8MB	8MB	1.5MB	2MB (per MSP)
L3	6MB	n/a	n/a	128MB	n/a
Proc Peak Mflops	6000	1334	1500	5200	12800
Peak mem BW	6.4 GB/s	5.2GB/s	1.6GB/s	51 GB/s/MCM	26 GB/s/MSP

The Altix shipped with a version of the Intel Madison Itanium2 processor running at 1.3 GHz. It was later upgraded with processors with a larger L3 cache and a clock rate of 1.5 GHz. This latter Itanium has 3 levels of cache: 32 KB L1 (1 clock latency), 256 KB L2 (5 clock latency), and 6 MB L3 (14 clock latency). The L3 cache can sustain 48 GB/sec bandwidth to/from the processor. The memory system utilizes commodity DDR SDRAM DIMMs, achieving 10+ GB/sec bandwidth per node. The interconnect topology is a dual plane, quad bristled fat tree, capable of 800 MB/sec per processor in a bisection bandwidth

test for up to 32 processors, and 400 MB/sec per processor for more than 32 processors. Additional configuration and operational information is available at [6].

ORNL purchased a 256 processor Altix system with a total of 2 terabytes of shared memory, 12 TB of fiber channel attached disks, and a single system image (SSI) software needed to support 256 processors. The processors are 1.5 GHz Intel Madison Itanium2 processors with 6 MB of cache per processor. Initially, the system was configured to run 4 partitions of the Linux operating system. However, after working closely with SGI, the Altix is currently running a 256 processor single system image. Single applications run on all 256 processors using the NUMalink interconnect for interprocessor communications between the multiple segments of the system.

This evaluation also includes comparisons to other systems examined by CCS. Table 2 and Table 3 outline these system configurations. Interested readers can find more information about these platforms at the CCS website [6].

Table 3: Experiment configurations.

Mnemonic	System	Programming model
X1-mpi	Cray X1 (Phoenix)	MPI
X1-ca	Cray X1 (Phoenix)	CoArray Fortran
Altix-mpi	SGI Altix (RAM)	MPI
Altix-omp	SGI Altix (RAM)	OpenMP
p690-mpi	IBM p690 (Cheetah)	MPI
p690-omp	IBM p690 (Cheetah)	OpenMP

4 Microbenchmarks

The objective of microbenchmarking is to characterize the performance of the underlying architectural components of the SGI Altix. Both standard benchmarks and customized benchmarks are used. The standard benchmarks allow component performance to be compared with other computer architectures. The custom benchmarks permit the unique architectural features of the Altix (e.g., large shared memory system utilizing Intel processors) to be tested with respect to the target applications. The architectural-component evaluation assesses the following:

- Arithmetic performance, including varying instruction mix.
- Memory-hierarchy performance, including three levels of cache and shared memory. These tests utilize both System V shared memory and the SHMEM primitives.

- Task and thread performance, including performance of thread creation, locks, semaphores, and barriers.
- Message-passing performance, including intra-node, inter-node, intra-OS image, and inter-OS image MPI performance of point-to-point and collective communication.
- OS and I/O performance.

Detailed microbenchmark data are available from [7]. For example, we used the EuroBen benchmark to evaluate hardware performance of add, multiply, divide, and square root, and the performance of the software intrinsics (exponentials, trigonometric functions, and logarithms). Other tests demonstrate how vector length and stride, compiler optimizations, and vendor scientific libraries affect performance. Figure 2 is a FORTRAN 1-D FFT from Euroben benchmarks. For this benchmark the Altix processor outperforms that of the IBM p690 and the Cray X1 for small to medium sized vectors.

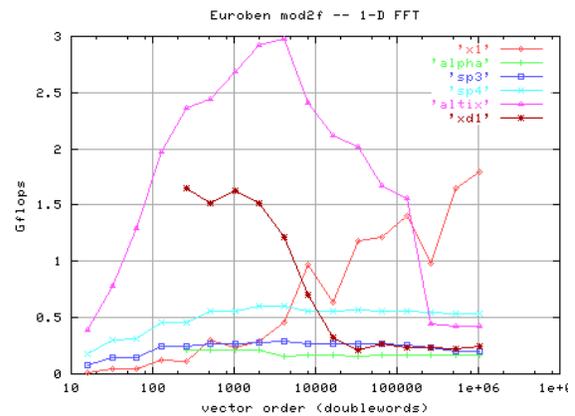


Figure 2: Euroben mod2f benchmark for 1-D FFT.

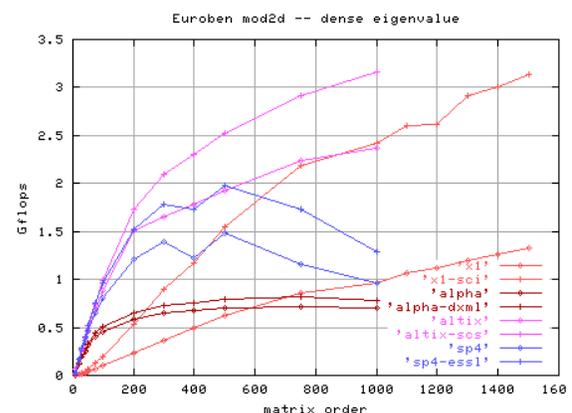


Figure 3: EuroBen mod2b benchmark for dense linear systems.

Our results also show that the Altix performs well on both sparse eigenvalue kernels and dense linear

algebra kernels, achieving over 90% of peak for a matrix-matrix multiply (DGEMM). Figure 3 compares the performance of vendor math libraries for solving a dense linear system, demonstrating that the Altix is better than the IBM p690 and the Cray X1 for the middle range of matrix sizes.

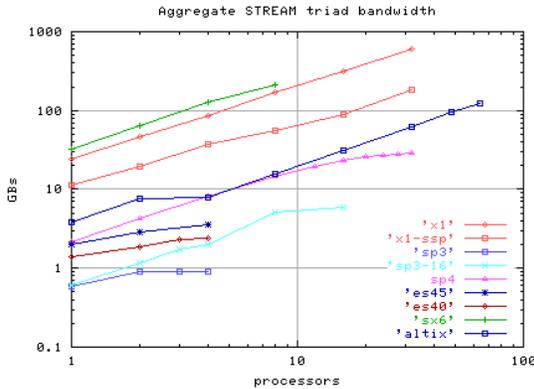


Figure 4: Aggregate STREAM triad bandwidth.

The STREAMS and MAPS benchmarks show the high memory bandwidth the Altix achieves, and Figure 4 shows how the memory performance scales with the number of processors. (See [5].)

Our communication tests include the standard benchmarks (ParkBench and Euroben-dm) to measure latency and bandwidth as a function of message size and distance, as well as custom benchmarks that reflect common communication patterns. The Altix MPI latency is only 1.1 microseconds (us) compared to 7 us on the Cray X1 and IBM p690 (Federation).

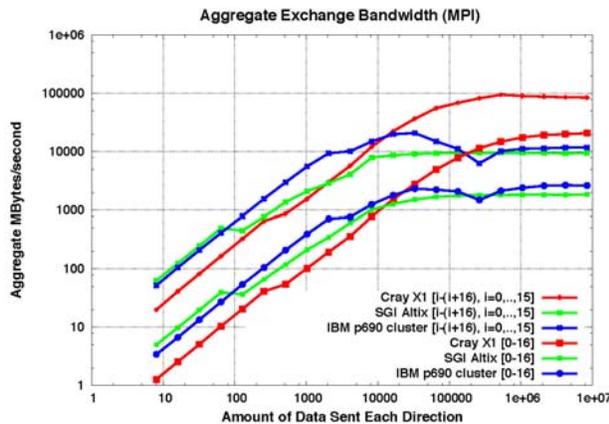


Figure 5: Aggregate Exchange Bandwidth (MPI) for distance of 16 processors.

Figure 5 compares the bandwidth when 2 processors a distance of 16 apart are exchanging messages using MPI and when 32 processors (16 pairs) are exchanging messages across the same distance. Figure 6 compares the bandwidth when 2 processors a

distance of 64 apart are exchanging messages using MPI and when 128 processors (64 pairs) are exchanging messages. Note that on the IBM p690 cluster the first experiment is limited to processors in the same p690 shared memory node, while the second experiment requires communication across the HPS switch. Within the SMP node, the achieved IBM bandwidth is at least as good as that on the Altix.

In contrast, the Altix achieves much better aggregate bandwidth than the IBM when the IBM must communicate between p690s. While the Cray X1 achieves the best aggregate bandwidth for large messages, it reaches the same maximum for the two experiments, approximately 90 GBytes/sec. In contrast, the aggregate SGI bandwidth is still rising, achieving approximately 50% of the maximum X1 bandwidth in the second experiment. For small messages, the Altix achieves the best performance among the three systems.

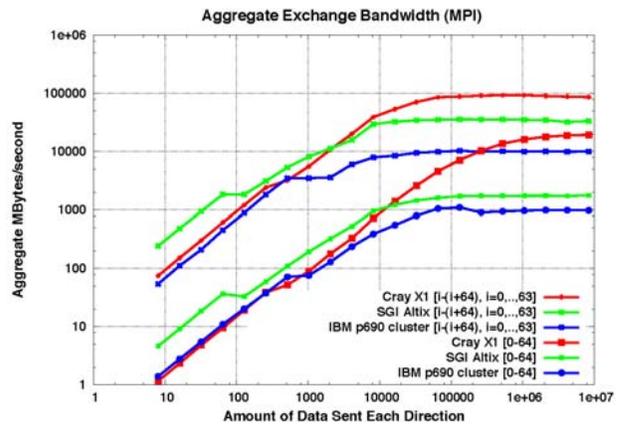


Figure 6: Aggregate Exchange Bandwidth (MPI) for distance of 64 processors.

Figure 7 and Figure 8 examine MPI communication performance as a function of physical distance between communicating processes. Except where noted, the cache is not invalidated before taking measurements. For small messages there is a performance advantage to communicating between physically neighboring processors, especially if in the same node. However, there is little performance sensitivity for larger distances.

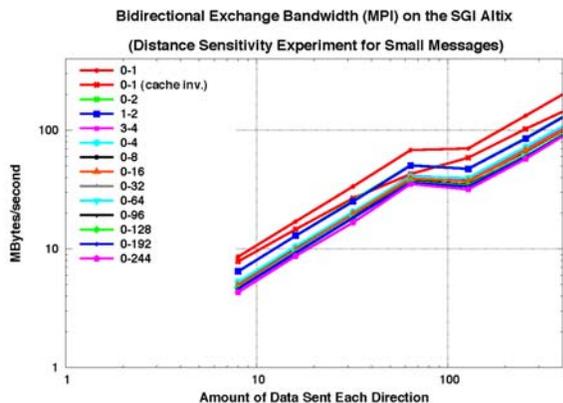


Figure 7: Distance sensitivity for small messages.

In contrast, exchanging large messages between processors in the same 2-processor node is more expensive than when exchanging between processors not in the same node. This effect shows up sooner with cache invalidation than without. Exchanging large messages between processors in the same C-brick, but not in the same node, shows the highest performance. As with small messages, large message performance is relatively insensitive (<20%) to distance once the separation is greater than 4. Experiments with cache invalidation show similar behavior, except as noted above.

Figure 9 and Figure 10 examine the same issue for simultaneous exchange. Unlike Figure 6, the metric here is bandwidth per process pair, not aggregate bandwidth. For small messages there is again little performance sensitivity to distances greater than 4, and performance degradation compared to the distance experiments is <20%. For large messages contention does occur, with the performance observed by a single pair halved for 4 simultaneous exchanges, and reduced to 25% of the former bandwidth for 32 simultaneous exchanges. Note, however, that the aggregate bandwidth continues to increase, especially as the number of pairs (and the distance) increases.

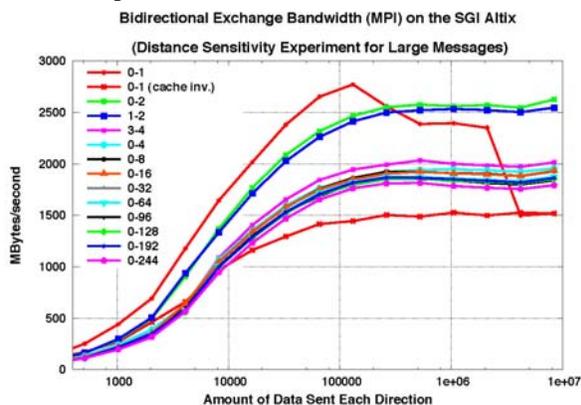


Figure 8: Distance Sensitivity for large messages.

Similar results hold when using SHMEM to implement the exchange instead of MPI. For small messages SHMEM performance is twice that of MPI, but sensitivity to distance and contention are qualitatively the same. For large messages SHMEM and MPI performance are nearly identical when the cache is invalidated first. However, without cache invalidation SHMEM performance is significantly better than MPI performance for all but the largest message sizes.

The exchange experiments were also used to determine the most efficient MPI communication protocol to use for an exchange. When enabling the SGI single copy MPI optimizations, which are automatically used with MPI_SENDRECV, protocols using MPI_ISEND and MPI_RECV are the most efficient, but MPI_SENDRECV is typically one of the better performers.

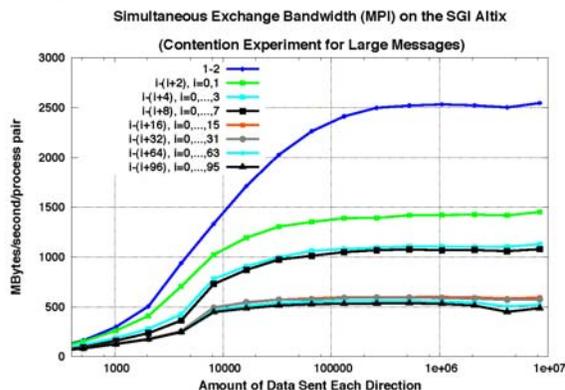


Figure 9: Contention for large messages.

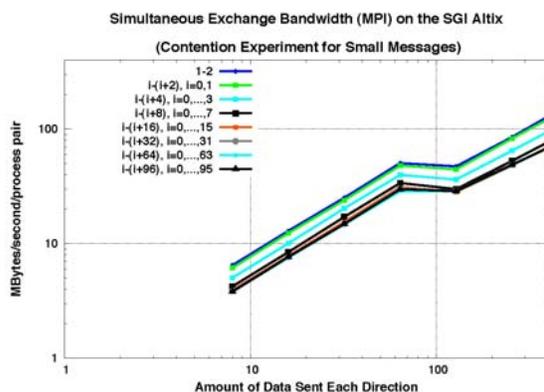


Figure 10: Contention for small messages.

When applications are scaled to larger processor counts or larger problems sizes, the ALLTOALLV and ALLREDUCE communication collectives are often the source of performance problems. The performance observed in the exchange experiments provides some information on the performance of the MPI_ALLTOALL collective.

Figure 11, Figure 12, and Figure 13 compare best-observed (“optimal”) MPI_ALLREDUCE performance across a number of platforms for each of the three vector lengths. The Altix has the best MPI_ALLREDUCE for short vectors, but the Cray X1 has better performance for long vectors.

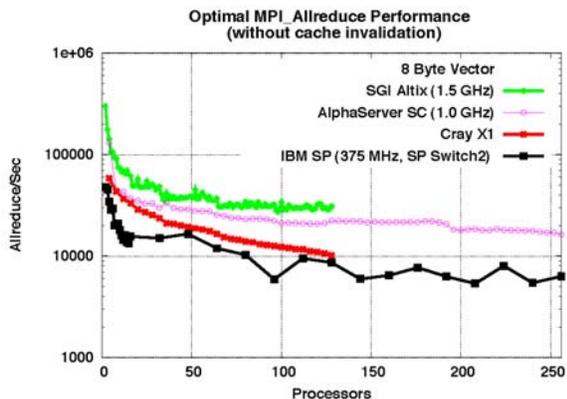


Figure 11: Performance of MPI_ALLREDUCE (8B) across platforms.

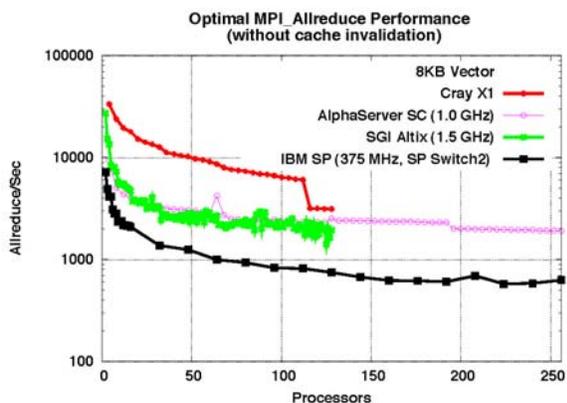


Figure 12: Performance of MPI_ALLREDUCE (8KB) across platforms.

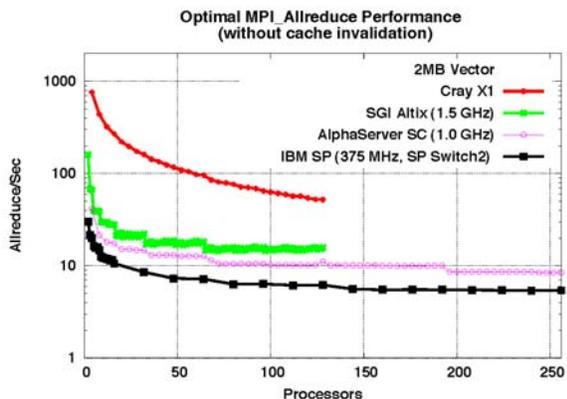


Figure 13: Performance of MPI_ALLREDUCE (2MB) across platforms.

5 Kernels

The kernel benchmarks bridge the gap between the low-level microbenchmarks and the resource intensive application benchmarking. We used industry-standard kernels (ParkBench, NAS Parallel Benchmarks, Euroben) as well as kernels that we extracted from our scientific applications. We tested and evaluated single processor performance and parallel kernels with and without the vendor's parallel scientific library. We compared the performance of these kernels with other architectures and have varied algorithms and programming paradigms (MPI, SHMEM, Co-Array Fortran, OpenMP). For example, Figure 14 compares the performance of the NAS multi-grid benchmark with various processor counts, architectures, and communication strategies.

We used a kernel representative of the dynamics algorithm found in the atmospheric component of the Community Climate System Model (CCSM) [1], the primary model for global climate simulation in the U.S. This kernel, the parallel spectral transform shallow water model (PSTSWM), supports different problem sizes, algorithms, and programming paradigms, and has been optimized on many parallel computer architectures. On the Altix we used PSTSWM to analyze compiler optimizations, evaluate math libraries, evaluate performance of the memory subsystem, compare programming paradigms, and compare performance with other supercomputers.

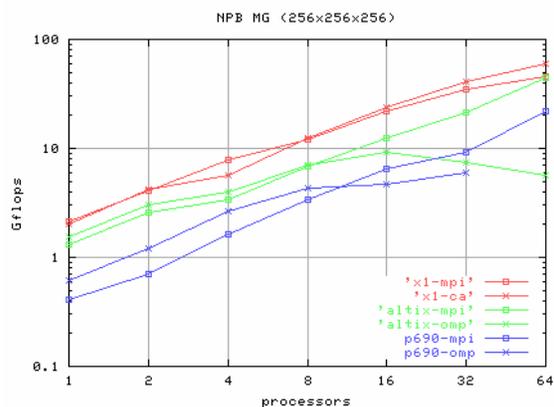


Figure 14: NPB MG benchmark.

Figure 15 describes the sensitivity of PSTSWM performance to problem size on the Altix. The problem sizes T5, T10, T21, T42, T85, and T170 are horizontal resolutions. Each computational grid in this sequence is approximately 4 times smaller than the next larger size. The X-axis is the number of vertical levels for a given horizontal resolution. Most of the problem coupling is in one or the other of the horizontal directions, and the vertical dimension simply controls

the cache locality of certain phases of the computation. As the number of vertical levels increase, performance for the 4 largest problem sizes converges, slowly decreasing as the number of levels continues to increase. The performance curves look very similar to memory bandwidth curves used to illuminate the memory hierarchy, and we assume that the memory hierarchy is what is controlling the performance degradation as the number of vertical levels increases.

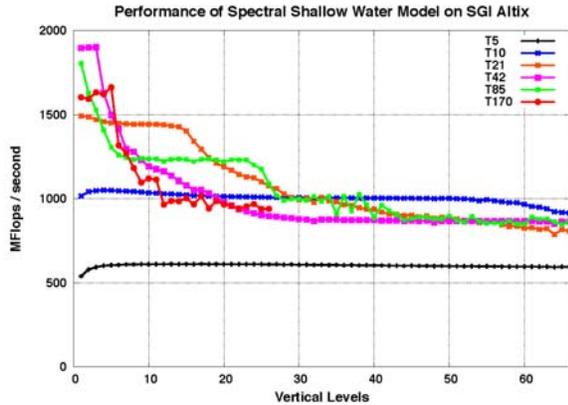


Figure 15: Sensitivity of PSTSWM performance to problem size on SGI Altix.

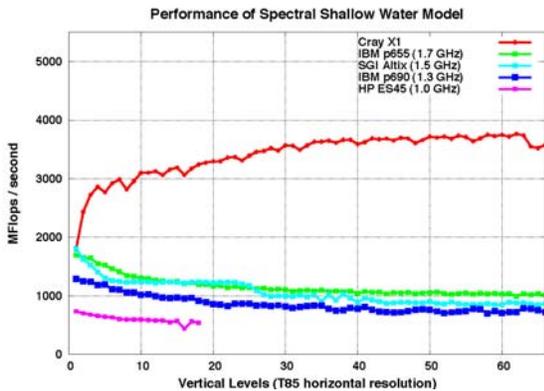


Figure 16: Performance of PSTSWM T85 across platforms.

Figure 16 compares the performance for the T85 problem on a number of HPC systems. These data show the advantage of the memory subsystem of the Cray X1 over that in the nonvector systems. However, the Altix performance holds its own compared to the other nonvector systems. These results are all for a single processor.

Figure 17 compares the performance for the different horizontal resolutions with 18 vertical levels when run on 1, 2, 4, ..., 128 consecutive processors simultaneously. Performance is identical when using 1 or 2 processors, or when using 32 processors when only every fourth processor is used. However, if all four processors in a C-brick are used, then there is

contention for memory bandwidth and performance degrades for the larger problem sizes. This is the only situation where contention occurs, and performance does not continue to degrade as more processors are used.

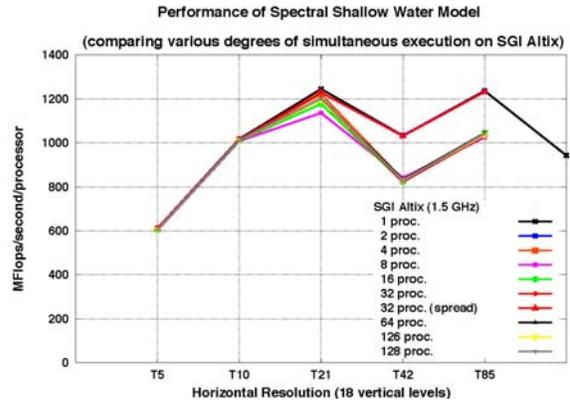


Figure 17: PSTSWM performance for different horizontal resolutions.

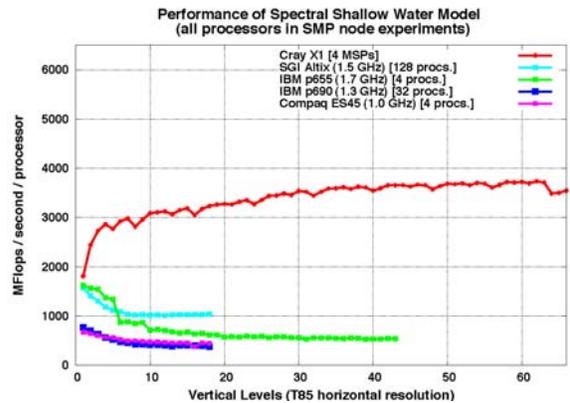


Figure 18: Effects of SMP node contention on PSTSWM.

Figure 18 compares the impact of contention when using all processors in an SMP node for problem T85 as the number of vertical levels increase. (For the Altix, data was collected on 128 consecutive processors in the larger system.) The Altix shows the least amount of performance degradation among the non-vector systems, indicating that the SGI memory subsystem scales very well for this type of memory contention benchmark.

6 Applications

Two aspects of application benchmarking are emphasized in these preliminary results, identifying peak achievable performance and inter-platform comparisons.

6.1 Climate – POP

The Parallel Ocean Program (POP) is an ocean modeling code developed at Los Alamos National Laboratory to take advantage of high-performance computer architectures. POP is used on a wide variety of computers for eddy-resolving simulations of the world oceans [4, 8] and for climate simulations as the ocean component of coupled climate models. For example, POP is the ocean component of CCSM [1].

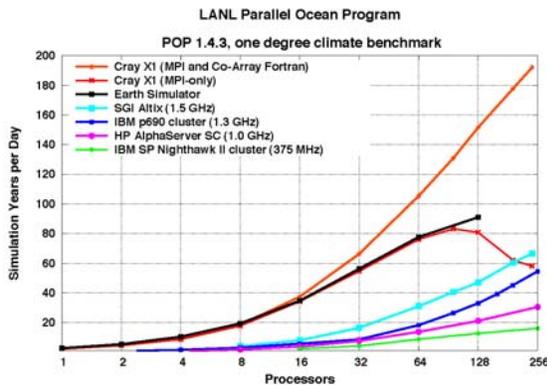


Figure 22: POP performance across platforms.

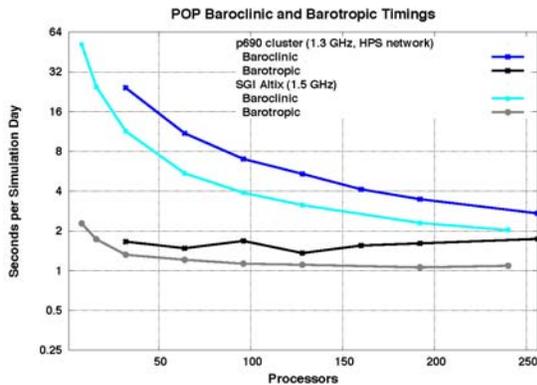


Figure 23: POP baroclinic and barotropic timings.

POP has proven to be a valuable tool for evaluating the scalability of HPC systems. It is comprised of two computational kernels, the baroclinic and the barotropic. POP parallelization is based on a two dimensional decomposition of the horizontal grid (leaving the vertical dimension undecomposed). Communication is required to update halo regions and to compute inner products in a conjugate gradient linear solver. The baroclinic phase scales very well on most platforms, with computation dominating communication until the processor count becomes large. In contrast the barotropic is dominated by a slowly converging iterative solution of a linear system used to solve a 2D elliptic problem. The linear system is solved using a conjugate gradient method, which requires halo updates to compute residuals and global

reductions to compute inner products. The barotropic is very sensitive to communication latency, and the best that can be hoped is that the time spent in the barotropic does not grow with processor count for large processor counts.

Figure 22 compares the performance of POP for a relatively small benchmark using a computational grid with a one-degree horizontal resolution. This problem size is the same as used in current coupled climate model simulations. POP has been vectorized to run on the Cray X1 and the Earth Simulator, and different versions of the code were run on the vector and non-vector systems to produce the data in this figure. On the Altix, optimizations included empirical determination of optimal domain decompositions and tuning of certain of the communication protocols. While the vector systems were the best performers, the Altix showed the best performance of the non-vector systems and achieved 30% of the X1 performance. The Altix performance was better than that of the X1 when the X1 used only MPI. (An alternative implementation of POP using SHMEM instead of MPI did not improve POP performance on the Altix.)

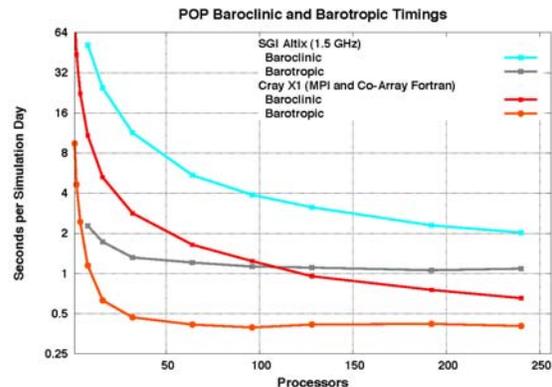


Figure 24: POP baroclinic and barotropic timings.

Figure 23 and Figure 24 compare the performance of the Altix on the baroclinic and barotropic phases with that of the IBM p690 cluster and the Cray X1, respectively. The Altix has a 50-100% advantage over the IBM system in the computation-bound baroclinic phase, and the barotropic phase scales better on the Altix than on the p690 cluster. This benchmark is computation bound on both systems out to 248 processors. The Altix is 3 times slower than the X1 on both the baroclinic and the barotropic at 248 processors, but is scaling equally well on both.

6.2 Fusion – GYRO

GYRO is an Eulerian gyrokinetic-Maxwell solver developed by R.E. Waltz and J. Candy at General Atomics [2]. It is used to study plasma

microturbulence in fusion research. GYRO uses the MPI_ALLTOALL command to transpose the distributed data structures and is more sensitive to bandwidth than to latency for large problem sizes.

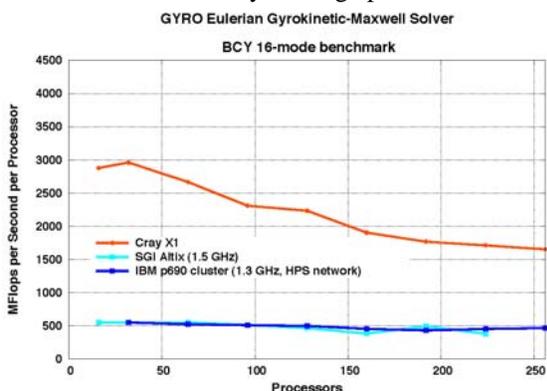


Figure 25: GYRO 16-mode performance across platforms.

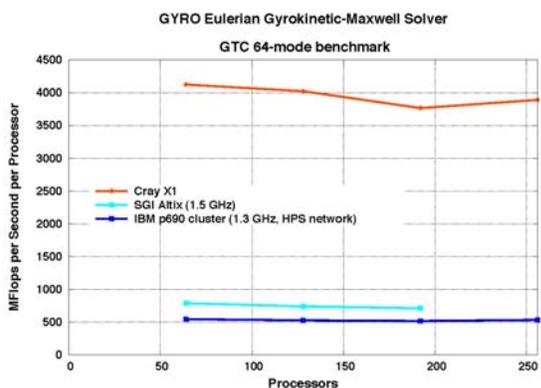


Figure 26: GYRO 64-mode performance across platforms.

Figure 25 and Figure 26 describe the per processor performance of two different benchmark problems: a small 16-mode problem labeled BCY and a large 64-mode problem labeled GTC. The metric, MFlops/sec/processor, is calculated using the same floating point operation count for each system, so performance between the different systems can be compared directly. This view of performance is useful in that it allows both raw performance and scalability to be displayed in the same graph.

Scalability on the non-vector systems is excellent for both benchmarks (and is also very good for the large benchmark on the Cray X1). Note, however, that while the Altix is 60% faster than the IBM on the GTC benchmark, it achieves essentially identical performance on the BCY benchmark.

Figure 27 depicts the fraction of the total time spent in MPI communication on each system for the two benchmark problems. Since the total time differs

for each system, these values cannot be compared directly. However, it is clear that time spent in MPI_ALLTOALL is impacting performance on the Altix to a much greater degree than on the other systems. For example, on the X1, communication is never more than 10% of the execution time, and the small and large problem sizes demonstrate similar percentages. On the p690 cluster, communication is a much smaller percentage of the time for the large problem size, indicating that the computational complexity is increasing faster than the communication complexity (and that the IBM HPS switch is able to handle the increased bandwidth demands).

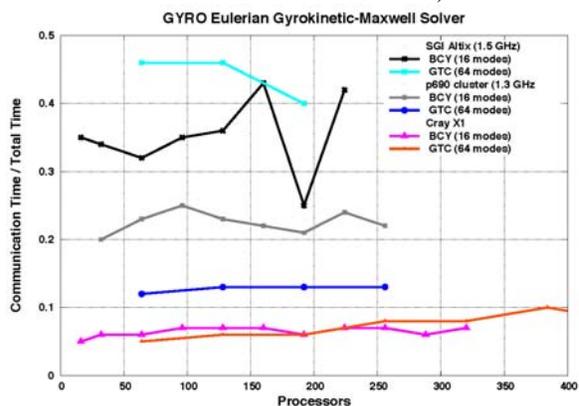


Figure 27: Communication fraction for GYRO.

In contrast, on the Altix the fraction of time spent in communication is higher for the larger benchmark, indicating that the interconnect is having difficulty communicating the large messages. It is unclear at this time whether there is a performance problem in the MPI_ALLTOALL or whether this is a limitation in the Altix network. The communication microbenchmarks indicated that the Altix network should support higher bandwidth rates than the IBM system. However, part of the MPI_ALLTOALL communicates within the p690 SMP node, and the IBM may have an advantage there.

7 Conclusions

In summary, the Altix provides many advantages over other non-vector machines and it is competitive with the Cray X1 on a number of representative kernels and applications. Across our measurements, three salient advantages to the Altix emerged. First, the latency for MPI operations is very low – on the order of 1.1 microseconds – inside an Altix node. This low latency contributes to good application scaling. Second, the Intel Itanium 2 processor and Altix memory subsystem provide notable performance advantages for computation when compared to other

non-vector systems. The only systems in our study that exceeded the Altix's aggregate STREAM triad bandwidth were both vector systems: the Cray X1 and the NEC SX-6. Third, on numerous operations, such as DAXPY and FFT, the Altix actually outperforms the vector systems at short vector lengths. These advantages translated into good results for both raw performance and scaling on the applications that we examined. We are continuing our evaluations, porting, optimizing, and analyzing additional application codes and looking in detail at open issues such as hybrid MPI/OpenMP performance and alternative parallel programming paradigms such as SHMEM and UPC. The system software on the Altix is also continuing to mature, with a recent move to a new Fortran compiler with somewhat different performance characteristics.

Acknowledgements

This research was sponsored by the Office of Mathematical, Information, and Computational Sciences, Office of Science, U.S. Department of Energy under Contract No. DE-AC05-00OR22725 with UT-Batelle, LLC. Accordingly, the U.S. Government retains a nonexclusive, royalty-free license to publish or reproduce the published form of this contribution, or allow others to do so, for U.S. Government purposes.

References

- [1] M.B. Blackmon, B. Boville *et al.*, "The Community Climate System Model," *BAMS*, 82(11):2357-76, 2001.
- [2] J. Candy and R. Waltz, "An Eulerian gyrokinetic-Maxwell solver," *J. Comput. Phys.*, 186(545), 2003.
- [3] High-End Computing Revitalization Task Force (HECRTF), "Federal Plan for High-End Computing," Executive Office of the President, Office of Science and Technology Policy, Washington, DC 2004.
- [4] M.E. Maltrud, R.D. Smith *et al.*, "Global eddy resolving ocean simulations driven by 1985-1994 atmospheric winds," *J. Geophys. Res.*, 103:30825--53, 1998.
- [5] J.D. McCalpin, *Stream Benchmarks*, <http://www.cs.virginia.edu/stream>, 2002.
- [6] Oak Ridge National Laboratory, *Center for Computational Sciences*, <http://www.ccs.ornl.gov>, 2005.
- [7] Oak Ridge National Laboratory, *Early Evaluation Website*, <http://www.csm.ornl.gov/evaluation>, 2005.
- [8] R.D. Smith, M.E. Maltrud *et al.*, "Numerical simulation of the North Atlantic ocean at 1/10 degree," *J. Phys. Oceanogr.*, 30:1532-61, 2000.
- [9] US Department of Energy Office of Science, "A Science-Based Case for Large-Scale Simulation," US Department of Energy Office of Science 2003, <http://www.pnl.gov/scales>.