doi: 10.32131/jes.3.39

Journal of the Earth Simulator, Volume 3, September 2005, 39 – 51

Performance of Ultra-Scale Applications on Leading Vector and Scalar HPC Platforms

Leonid Oliker[1]*, Andrew Canning[1], Jonathan Carter[1], John Shalf[1], Horst Simon[1], Stephane Ethier[2], David Parks[3], Shigemune Kitawaki[4], Yoshinori Tsuda[4] and Tetsuya Sato[4]

[1] *CRD/NERSC, Lawrence Berkeley National Laboratory, CA, U.S.A.*
[2] *Princeton Plasma Physics Laboratory, Princeton University, NJ, U.S.A.*
[3] *NEC Solutions America, Advanced Technical Computing Center, TX, U.S.A.*
[4] *The Earth Simulator Center, Japan Agency for Marine-Earth Science and Technology, Yokohama, Japan*

(Received January 31, 2005; Revised manuscript accepted June 16, 2005)

Abstract The last decade has witnessed a rapid proliferation of superscalar cache-based microprocessors to build high-end capability and capacity computers primarily because of their generality, scalability, and cost effectiveness. However, the constant degradation of superscalar sustained performance, has become a well-known problem in the scientific computing community. This trend has been widely attributed to the use of superscalar-based commodity components who's architectural designs offer a balance between memory performance, network capability, and execution rate that is poorly matched to the requirements of large-scale numerical computations. The recent development of massively parallel vector systems offers the potential to increase the performance gap for many important classes of algorithms. In this study we examine four diverse scientific applications with the potential to run at ultrascale, from the areas of plasma physics, material science, astrophysics, and magnetic fusion. We compare performance between the vector-based Earth Simulator (ES) and Cray X1, with leading superscalar-based platforms: the IBM Power3/4 and the SGI Altix. Results demonstrate that the ES vector systems achieve excellent performance on our application suite – the highest of any architecture tested to date.

1. INTRODUCTION

The last decade has witnessed a rapid proliferation of superscalar cache-based microprocessors to build high-end capability and capacity computers primarily because of their generality, scalability, and cost effectiveness. This is primarily because their generality, scalability, and cost effectiveness convinced computer vendors, buyers, and users that vector architectures hold little promise for future large-scale supercomputing systems. However, the constant degradation of superscalar sustained performance, has become a well-known problem in the scientific computing community. This trend has been widely attributed to the use of superscalar-based commodity components who's architectural designs offer a balance between memory performance, network capability, and execution rate that is poorly matched to the requirements of large-scale numerical computations.

The increasing gap between processor and memory speeds is a well-known problem in computer architecture, with peak processor performance improving at a rate of 60% per year, while DRAM latencies and bandwidths improve at only 7% and 20% respectively. To mask memory latencies, current high-end computers now demand up to 25 times the number of overlapped operations required of supercomputers 30 years ago. Furthermore, techniques designed to hide memory latencies, such as out-of-order superscalar instruction processing, speculative execution, multithreading, and stream prefetching engines, may actually increase the memory bandwidth requirements. This so-called 'memory wall' is one of the reasons many high performance applications run well below the peak arithmetic performance of the underlying machine. In particular, irregularly structured and data-intensive codes exhibit poor temporal locality and receive little benefit from the automatically managed caches of conventional microarchitectures. In addition, a significant fraction of scientific codes are characterized by predictable data-parallelism that could be exploited at compile time with properly structured program semantics; superscalar processors can often exploit this paral-

* ***Corresponding author****: Dr. Leonid Oliker, Lawrence Berkeley National Laboratory, One Cyclotron Road, MS 50B4230, Berkeley, California 94720, U.S.A. E-mail: loliker@lbl.gov*

lelism, but their generality leads to high costs in chip area and power, which in turn limit the degree of parallelism.

Superscalar architectures are unable to efficiently exploit the large number of floating-point units that be can fabricated on a chip, due to the small granularity of their instructions and the correspondingly complex control structure necessary to support it. Vector technology, on the other hand, provides an efficient approach for controlling a large amount of computational resources provided sufficient regularity in the computational structure can be discovered. Vectors exploit these regularities in the computational structure to expedite uniform operations on independent data elements. Vector instructions specify a large number of identical operations that may execute in parallel, thus reducing control complexity and efficiently controlling a large amount of computational resources. However, when such operational parallelism cannot be found, the efficiency of the vector architecture can suffer from the properties of Amdahl's Law where the time taken by the portions of the code that are non-vectorizable easily dominate the execution time.

Recently, two innovative parallel-vector architectures have become available to the supercomputing community: the Japanese Earth Simulator (ES) and the Cray X1. In order to quantify what these modern vector capabilities entail for the scientists that rely on modeling and simulation, it is critical to evaluate this architectural approach in the context of demanding computational algorithms. A number of previous studies [6, 9, 16, 20, 19] have examined parallel vector performance for scientific codes; however, few direct comparisons of large-scale applications are currently available. In this work, we compare the vector-based ES and X1 architectures with three state-of-the-art superscalar systems: the IBM Power3, Power4, and the SGI Altix. Our research team was the first international group to conduct a performance evaluation study at the Earth Simulator Center [18]; remote ES access in not available. We examine four diverse scientific applications with potential to operate at ultrascale, from plasma physics (LBMHD), material science (PARATEC), astrophysics (Cactus), and magnetic fusion (GTC). Results demonstrate that the vector systems achieve excellent performance on our application suite – the highest of any platform tested to date. However, the low ratio between scalar and vector performance make the evaluated vector systems particularly sensitive to unvectorized code segments – pointing out an additional dimension for 'architectural balance' where vector systems are concerned. Additionally, vectorization of a particle-in-cell code highlights the potential difficulty of expressing irregularly structured algorithms as data-parallel programs. Overall, the ES sustains a significantly higher fraction of peak than the X1, and often outperforms it in absolute terms. Results also indicate that the Altix system is a promising computational platform.

2. TARGET HPC PLATFORMS AND APPLICATIONS

Table 1 presents a summary of the architectural characteristics of the five supercomputers examined in our study. Observe that the vector systems are designed with higher absolute performance and better architectural balance than the superscalar platforms. The ES and X1 have high memory bandwidth relative to peak CPU (bytes/flop), allowing them to continuously feed the arithmetic units with operands more effectively than the superscalar architectures examined in our study. Additionally, the custom vector interconnects show superior characteristics in terms of measured latency [3, 22], point-to-point messaging (bandwidth per CPU), and all-to-all communication (bisection bandwidth) – in both raw performance (GB/s) and as a ratio of peak processing speed (bytes/flop). Overall the ES appears the most balanced system in our study, while the Altix shows the best architectural characteristics among the superscalar platforms.

2.1 Power3

The Power3 experiments reported here were conducted on the 380-node IBM pSeries system running AIX 5.1 and located at Lawrence Berkeley National Laboratory. Each 375 MHz processor contains two floating-point units (FPUs) that can issue a fused multiply-add (MADD) per cycle for a peak performance of 1.5 Gflop/s. The Power3 has a pipeline of only three cycles, thus using the registers more efficiently and diminishing the penalty for mispredicted branches. The out-of-order architecture uses prefetching to reduce pipeline stalls due to cache misses. The CPU has a 32 KB instruction cache, a 128 KB 128-way set associative L1 data cache, and an 8 MB four-way set associative L2 cache with its own private bus. Each SMP node consists of 16 processors connected to main memory via a crossbar. Multi-node configurations are networked via the Colony switch using an omega-type topology.

2.2 Power4

The Power4 experiments in this paper were performed on the 27-node IBM pSeries 690 system running AIX 5.2 and operated by Oak Ridge National Laboratory (ORNL). Each 32-way SMP consists of 16 Power4 chips (organized as 4 MCMs), where a chip contains two 1.3 GHz processor cores. Each core has two FPUs capable of a fused MADD per cycle, for a peak performance of 5.2

Table 1 Architectural highlights of the Power3, Power4, Altix, ES, and X1 platforms.

Platform	CPU/ Node	Clock (MHz)	Peak (GF/s)	Memory BW (GB/s)	Peak (Bytes/flop)	MPI Latency (μsec)	Network BW (GB/s/CPU)	Bisection BW (Bytes/s/flop)	Network Topology
Power3	16	375	1.5	0.7	0.47	16.3	0.13	0.087	Fat-tree
Power4	32	1300	5.2	2.3	0.44	7.0	0.25	0.025	Fat-tree
Altix	2	1500	6.0	6.4	1.1	2.8	0.40	0.067	Fat-tree
ES	8	500	8.0	32.0	4.0	5.6	1.5	0.19	Crossbar
X1	4	800	12.8	34.1	2.7	7.3	6.3	0.088*	2D-torus

Gflop/s. The superscalar out-of-order architecture can exploit instruction level parallelism through its eight execution units; however a relatively long pipeline (six cycles) if necessitated by the high frequency design. Each processor contains its own private L1 cache (64 KB instruction and 32 KB data) with prefetch hardware; however, both cores share a 1.5 MB unified L2 cache. The L3 is designed as a stand-alone 32 MB cache, or to be combined with other L3s on the same MCM to create a larger 128 MB interleaved cache. The benchmarks presented in this paper were run on a system employing the recently-released Federation (HPS) interconnect, with two switch adaptors per node. None of the benchmarks used large (16MB) pages, as the ORNL system was not configured for such jobs.

2.3 Altix 3000

The SGI Altix is a unique architecture, designed as a cache-coherent, shared-memory multiprocessor system. The computational building blocks of the Altix consists of four Intel Itanium2 processors, local memory, and a two controller ASICs called the SHUB. The 64-bit Itanium2 architecture operates at 1.5 GHz and is capable of issuing two MADDs per cycle for a peak performance of 6 Gflop/s. The memory hierarchy consists of 128 floating-point (FP) registers and three on-chip data caches with 32 K of L1, 256 K of L2, and 6 MB of L3. Note that the Itanium2 cannot store FP data in L1 cache (only in L2), making register loads and spills a potential source of bottlenecks; however, the relatively large FP register set helps mitigate this issue. The superscalar Itanium2 processor performs a combination of in-order and out-of-order instruction execution referred to as Explicitly Parallel Instruction Computing (EPIC). Instructions are organized into VLIW bundles, where all instructions within a bundle can be executed in parallel. However, the bundles themselves must be processed in order.

The Altix interconnect uses the NUMAlink3, a high-performance custom network in a fat-tree topology. This configuration enables the bisection bandwidth to scale linearly with the number of processors. In additional to the traditional distributed-memory programming paradigm, the Altix systems implements a cache-coherent, nonuniform memory access (NUMA) protocol directly in hardware. This allows a programming model where remote data are accessed just like locally allocated data, using loads and stores. A load/store cache miss causes the data to be communicated in hardware (via the SHUB) at a cache-line granularity and automatically replicated in the local cache, however locality of data in main memory is determined at page granularity. Additionally, one-sided programming languages can be efficiently implemented by leveraging the NUMA layer. The Altix experiments reported in this paper were performed on the 256-processor system (several reserved for system services) at ORNL, running 64-bit Linux version 2.4.21 and operating as a single system image.

2.4 Earth Simulator

The vector processor of the ES uses a dramatically different architectural approach than conventional cache-based systems. Vectorization exploits regularities in the computational structure of scientific applications to expedite uniform operations on independent data sets. The 500 MHz ES processor contains an 8-way replicated vector pipe capable of issuing a MADD each cycle, for a peak performance of 8.0 Gflop/s per CPU. The processors contain 72 vector registers, each holding 256 64-bit words (vector length = 256). For non-vectorizable instructions, the ES contains a 500 MHz scalar processor with a 64 KB instruction cache, a 64 KB data cache, and 128 general-purpose registers. The 4-way superscalar unit has a peak of 1.0 Gflop/s (1/8 of the vector performance) and supports branch prediction, data prefetching, and out-of-order execution.

Like traditional vector architectures, the ES vector unit is cacheless; memory latencies are masked by overlapping pipelined vector operations with memory fetches. The main memory chip for the ES uses a specially developed high speed DRAM called FPLRAM (Full Pipelined

* X1 bisection bandwidth is based on a 2048 MSP configuration

RAM) operating at 24ns bank cycle time. Each SMP contains eight processors that share the node's memory. The Earth Simulator is the world's most powerful supercomputer [5], containing 640 ES nodes connected through a custom single-stage crossbar. This high-bandwidth interconnect topology provides impressive communication characteristics, as all nodes are a single hop from one another. However, building such a network incurs a high cost since the number of cables grows as a square of the node count – in fact, the ES system utilizes approximately 1500 miles of cable. The 5120-processor ES runs Super-UX, a 64-bit Unix operating system based on System V-R3 with BSD4.2 communication features. As remote ES access is not available, the reported experiments were performed during the authors' visit to the Earth Simulator Center located in Kanazawa-ku, Yokohama, Japan in December 2003.

2.5 X1

The recently-released X1 is designed to combine traditional vector strengths with the generality and scalability features of modern superscalar cache-based parallel systems. The computational core, called the single-streaming processor (SSP), contains two 32-stage vector pipes running at 800 MHz. Each SSP contains 32 vector registers holding 64 double-precision words (vector length = 64), and operates at 3.2 Gflop/s peak for 64-bit data. The SSP also contains a two-way out-of-order superscalar processor running at 400 MHz with two 16 KB caches (instruction and data). The multi-streaming processor (MSP) combines four SSPs into one logical computational unit. The four SSPs share a 2-way set associative 2 MB data Ecache, a unique feature for vector architectures that allows extremely high bandwidth (25–51 GB/s) for computations with temporal data locality. MSP parallelism is achieved by distributing loop iterations across each of the four SSPs. The compiler must therefore generate both vectorizing and multistreaming instructions to effectively utilize the X1. The scalar unit operates at 1/8th the peak of SSP vector performance, but offers effectively 1/32 MSP performance if a loop can neither be multistreamed nor vectorized. Consequently, a high vector operation ratio is especially critical for effectively utilizing the underlying hardware.

The X1 node consists of four MSPs sharing a flat memory, and large system configuration are networked through a modified 2D torus interconnect. The torus topology allows scalability to large processor counts with relatively few links compared with fat-tree or crossbar interconnects; however, this topological configuration suffers from limited bisection bandwidth. Finally, the X1 has hardware supported globally addressable memory which allows for efficient implementations of one-sided communication libraries (MPI-2, SHMEM) and implicit parallel programming languages (UPC, CAF). All reported X1 experiments reported were performed on the 512-MSP system (several reserved for system services) running UNICOS/mp 2.4 and operated by ORNL.

2.6 Scientific Applications

Four scientific applications were chosen to measure and compare the performance of the vector-based ES and X1 with the superscalarbased Power3, Power4, and Altix systems. The application are: LBMHD, a plasma physics application that uses the Lattice-Boltzmann method to study magneto-hydrodynamics; PARATEC, a first principles materials science code that solves the Kohn-Sham equations of density functional theory to obtain electronic wavefunctions; Cactus, an astrophysics code that evolves Einstein's equations from the Theory of General Relativity using the Arnowitt-Deser-Misner method; and GTC, a magnetic fusion application that uses the particlein-cell approach to solve non-linear gyrophase-averaged Vlasov-Poisson equations.

These codes represent candidate ultrascale applications that have the potential to fully utilize a leadership-class system of Earth Simulator scale and beyond. Performance results, presented in Gflop/s per processor (denoted as Gflops/P) and percentage of peak, are used to compare the relative time to solution of the computing platforms in our study. When different algorithmic approaches are used for the vector and scalar implementations, this value is computed by dividing a valid baseline flop-count by the measured wall-clock time of each architecture. To characterize the level of vectorization, we also examine vector operation ratio (VOR) and average vector length (AVL) for the ES and X1 where possible. The VOR measures the ratio between the number of vector operations and the total overall operations (vector plus scalar); while the AVL represents the average number of operations performed per issued vector instruction. An effectively vectorized code will achieve both high VOR (optimal is 100%) and AVL (256 and 64 is optimal for ES and X1 respectively). Hardware counter data were obtained with `hpmcount` on the Power systems, `pfmon` on the Altix, `ftrace` on the ES, and pat on the X1.

3. PLASMA PHYSICS

Lattice Boltzmann methods (LBM) have proved a good alternative to conventional numerical approaches for simulating fluid flows and modeling physics in fluids [21]. The basic idea of the LBM is to develop a simplified kinetic model that incorporates the essential physics, and reproduces correct macroscopic averaged properties.

Recently, several groups have applied the LBM to the problem of magneto-hydrodynamics (MHD) [8, 14] with promising results. LBMHD [15] simulates the behavior of a two-dimensional conducting fluid evolving from simple initial conditions and decaying to form current sheets.

The 2D spatial grid is coupled to an octagonal streaming lattice and block distributed over a 2D processor grid. Each grid point is associated with a set of mesoscopic variables, whose values are stored in vectors proportional to the number of streaming directions – in this case nine (eight plus the null vector). The simulation proceeds by a sequence of collision and stream steps. A collision step involves data local only to that spatial point, allowing concurrent, dependence-free point updates; the mesoscopic variables at each point are updated through a complex algebraic expression originally derived from appropriate conservation laws. A stream step evolves the mesoscopic variables along the streaming lattice, necessitating communication between processors for grid points at the boundaries of the blocks. Additionally, an interpolation step is required between the spatial and stream lattices since they do not match.

Varying schemes were used in order to optimize the collision routine on each of the architectures. The basic computational structure consists of two nested loops over spatial grid points (typically 100 – 1000 iterations) with inner loops over velocity streaming vectors and magnetic field streaming vectors (typically 10 – 30 iterations), performing various algebraic expressions. For the Power3/4 and Altix systems, the inner grid point loop was blocked to increase cache reuse – leading to a modest improvement in performance for the largest grids and smallest concurrencies. For the ES, the inner grid point loop was taken inside the streaming loops and vectorized. The temporary arrays introduced were padded to reduce memory bank conflicts. We note that the ES compiler was unable to perform this transformation based on the original code. In the case of the X1, the compiler did an excellent job, multi-streaming the outer grid point loop and vectorizing (via strip mining) the inner grid point loop without any user code restructuring. No additional vectorization effort was required due to the data-parallel nature of LBMHD.

Interprocessor communication was implemented using the MPI library, by copying the non-contiguous mesoscopic variables data into temporary buffers, thereby reducing the required number of send/receive messages. Additionally, a Co-array Fortran (CAF) [2] version was implemented for the X1 architecture. CAF is a onesided parallel programming language implemented via an extended Fortran 90 syntax. Unlike explicit message passing in MPI, CAF programs can directly access non-local data through co-array references. This allows a potential reduction in interprocessor overhead for architectures supporting one-sided communication, as well as opportunities for compiler-based optimizing transformations. For example, the X1's measured latency decreased from 7.3 μsec using MPI to 3.9 μsec using CAF semantics [3]. In the CAF implementation of LBMHD, the spatial grid is declared as a co-array and boundary exchanges are performed using co-array subscript notation.

3.1 LBMHD Results

Table 2 presents LBMHD performance on the five studied architecture for grid sizes of 4096^2 and 8192^2. Note that to maximize performance the processor count is restricted to squared integers. The vector architectures show impressive results, achieving a speedup of approximately 44x, 16x, and 7x compared with the Power3, Power4, and Altix respectively (for 64 processors). The AVL and VOR are near maximum for both vector systems, indicating that this application is extremely well-suited for vector platforms. In fact the 3.3 Tflop/s attained on 1024 processor of the ES represents the highest performance of LBMHD on any measured architecture to date. The X1 gives comparable raw performance to the ES for most of our experiments; however for 256 processors on the large (8192^2) grid configuration, the ES ran about 1.5X faster due to the decreased scalability of the X1. Additionally, the ES consistently sustains a significantly higher fraction of peak, due in part to its superior CPU-memory balance. The X1 CAF implementation shows about a 10% overall improvement over the MPI version for the large test case, however MPI slightly outperformed CAF for the smaller grid size (P = 64). For LBMHD, CAF reduced the memory traffic by a factor of 3X by eliminating user- and system-level message copies (latter used by MPI); these gains were somewhat offset by CAF's use of more numerous and smaller sized messages. This issue will be the focus of future investigation.

The low performance of the superscalar systems is mostly due to limited memory bandwidth. LBMHD has a low computational intensity – about 1.5 FP operations per data word of access – making it extremely difficult for the memory subsystem to keep up with the arithmetic units. Vector systems are able to address this discrepancy through a superior memory system and support for deeply pipelined memory fetches. Additionally, the 4096^2 and 8192^2 grids require 7.5 GB and 30 GB of memory respectively, causing the subdomain's memory footprint to exceed the cache size even at high concurrencies. Nonetheless, the Altix outperforms the Power3 and Power4 in terms of Gflop/s and fraction of peak due to its higher memory bandwidth and superior network charac-

Table 2 LBMHD per processor performance on 4096x4096 and 8192x8192 grids.

Grid Size	*P*	Power3		Power4		Altix		ES		X1 (MPI)		X1 (CAF)	
		Gflops/P	%Pk	Gflops/P	%Pk	Gflops/P	%Pk	Gflops/P	%Pk	Gflops/P	%Pk	Gflops/P	%Pk
4096	16	0.107	7%	0.279	5%	0.598	10%	4.62	58%	4.32	34%	4.55	36%
x	64	0.142	9%	0.296	6%	0.615	10%	4.29	54%	4.35	34%	4.26	33%
4096	256	0.136	9%	0.281	5%	—	—	3.21	40%	—	—	—	—
8192	64	0.105	7%	0.270	5%	0.645	11%	4.64	58%	4.48	35%	4.70	37%
x	256	0.115	8%	0.278	5%	—	—	4.26	53%	2.70	21%	2.91	23%
8192	1024	0.108	7%	—	—	—	—	3.30	41%	—	—	—	—

teristics. Observe that superscalar performance relative to concurrency shows more complex behavior than on the vector systems. Since the cache-blocking algorithm for the collision step is not perfect, certain data distributions are superior to others – accounting for increased performance at intermediate concurrencies. At larger concurrencies, the cost of communication begins to dominate, thus reducing performance as in the case of the vector systems.

3.2 3D LBMHD Experiments

On our second visit to the ES Center in October 2004, a 3D version of LBMHD was ported and run at large scale. This application uses a more conventional 3D cubic lattice for spatial and velocity resolution with 27 streaming vectors. Performance of the 3D code on the ES was predicted to be better than the 2D model for two reasons: improvement in the surface to volume ratio, resulting in a lower overall fraction of communication overhead; increase in the vectorizable work due to more computationally intensive collision operator. Starting from a basic superscalar version of the code, we made several significant efficiency improvements. Initially performance was low on the ES, achieving only 300 Mflops/P. As with the 2D version the innermost loops were short, iterating over streaming vectors. These loops were completely unrolled using compiler directives. After this optimization, the performance improved to around 4 Gflops/P. Moving to a larger grid, the MPI performance was improved by aggregating messages leading to about 4.8 Gflops/P per processor. Finally, the algorithm was reworked slightly to combine of the collision and streaming steps. This brought performance up to around 5.2 Gflops/P. Using a 1024^3 grid, we achieved an impressive 5.6, 11.1, and 22.2 Tflops/P on 1024, 2048, and 4096 processors of the ES. Even on the largest runs, communication time was less than 10% of the total overhead. For these experiments, the AVL and VOR were almost ideal, achieving over 254 and 99.5% respectively. Future work will focus on a comprehensive evaluation of the 3D LBMHD.

4. MATERIAL SCIENCE

PARATEC (PARAllel Total Energy Code [4]) performs ab-initio quantum-mechanical total energy calculations using pseudopotentials and a plane wave basis set. The pseudopotentials are of the standard norm-conserving variety. Forces can be easily calculated and used to relax the atoms into their equilibrium positions. PARATEC uses an all-band conjugate gradient (CG) approach to solve the Kohn-Sham equations of Density Functional Theory (DFT) and obtain the ground-state electron wavefunctions. DFT is the most commonly used technique in materials science, having a quantum mechanical treatment of the electrons, to calculate the structural and electronic properties of materials. Codes based on DFT are widely used to study properties such as strength, cohesion, growth, magnetic, optical, and transport for materials like nanostructures, complex surfaces, and doped semiconductors. Due to its accurate predictive power and computational efficiency, DFT based codes have been one of the largest consumer of supercomputing cycles in computer centers around the world.

In solving the Kohn-Sham equations using a plane wave basis, part of the calculation is carried out in real space and the remainder in Fourier space using specialized parallel 3D FFTs to transform the wavefunctions. The code spends most of its time in vendorsupplied BLAS3 (~30%) and 1D FFTs (~30%) on which the 3D FFTs libraries are built. Because these routines allow high cache reuse and efficient vector utilization, PARATEC generally obtains high percentage of peak performance across a spectrum of computing platforms. The code exploits fine-grained parallelism by dividing the plane wave (Fourier) components for each electron among the different processors [4]. PARATEC is written in F90 and MPI, and is designed primarily for massively parallel computing platforms, but can also run on serial machines. The main limitation to scaling PARATEC to large processor counts is the distributed grid transformation during the parallel 3D FFTs that requires global interprocessor communication when mapping the electron wavefunctions from Fourier space (where it is represent-

Table 3 PARATEC per processor performance on a 432 and 686 atom Silicon Bulk system.

	432 Atom										686 Atom			
P	Power3		Power4		Altix		ES		X1		ES		X1	
	Gflops/P	%Pk	Gflops/P	%Pk	Gflops/P	%Pk	Gflops/P	%Pk	Gflops/P	%Pk	Gflops/P	%Pk	Gflops/P	%Pk
32	0.950	63%	2.02	39%	3.71%	62%	4.76	60%	3.04	24%	—	—	—	—
64	0.848	57%	1.73	33%	3.24%	54%	4.67	58%	2.59	20%	5.25	66%	3.73	29%
128	0.739	49%	1.50	29%	—	—	4.74	59%	1.91	15%	4.95	62%	3.01	24%
256	0.572	38%	1.08	21%	—	—	4.17	52%	—	—	4.59	57%	1.27	10%
512	0.413	28%	—	—	—	—	3.39	42%	—	—	3.76	47%	—	—
1024	—	—	—	—	—	—	2.08	26%	—	—	2.53	32%	—	—

ed by a sphere) to a 3D grid in real space. Thus, architectures with a poor balance between their bisection bandwidth and computational rate (see Table 1) will suffer performance degradation at higher concurrencies due to global communication requirements.

4.1 PARATEC Results

Table 3 presents performance data for 3 CG steps of a 432 and 686 Silicon atom bulk systems and a standard LDA run of PARATEC with a 25 Ry cut-off using norm-conserving pseudopotentials. A typical calculation would require between 20 and 60 CG iterations to converge the charge density. PARATEC runs at a high percentage of peak on both superscalar and vector-based architectures due to the heavy use of the computationally intensive FFTs and BLAS3 routines, which allow high cache reuse and efficient vector utilization. The main limitation to scaling PARATEC to large numbers of processors is the distributed grid transformation during the parallel 3D FFTs which requires global interprocessor communications. It was therefore necessary to write specialized 3D FFT to reduce these communication requirements. Since our 3D FFT routine maps the wavefunction of the electron from Fourier space, where it is represented by a sphere, to a 3D grid in real space – a significant reduction in global communication can be achieved by only transposing the non-zero grid elements. Nonetheless, architectures with a poor balance between their bisection bandwidth and computational rate (see Table 1) will suffer performance degradation at higher concurrencies due to global communication requirements.

Results in Table 3 show that PARATEC achieves impressive performance on the ES, sustaining 2.6 Tflop/s for 1024 processors for the larger system – the first time that any architecture has attained over a Teraflop for this code. The declining performance at higher processor counts is caused by the increased communication overhead of the 3D FFTs, as well as reduced vector efficiency due to the decreasing vector length of this fixed-size problem. Since only 3 CG steps were performed in our benchmarking measurements, the set-up phase accounted for a growing fraction of the overall time – preventing us from accurately gathering the AVL and VOR values. The set-up time was therefore subtracted out for the reported Gflop/s measurements. This overhead becomes negligible for actual physical simulations, which require as many as 60 CG steps. For example on the smaller 432 atom system on 32 processors, the measured AVL for the total run was 145 and 46 for the ES and X1 (respectively); the AVL for only the CG steps (without set-up) would certainly be higher.

Observe that X1 performance is lower than the ES, even though it has a higher peak speed. The code sections of handwritten F90, which typically consume about 30% of the run time, have a lower vector operation ratio than the BLAS3 and FFT routines. These handwritten segments also run slower on the X1 than the ES, since unvectorized code segments tend not to multistream across the X1's SSPs. In addition, the X1 interconnect has a lower bisection bandwidth network than the ES (see Table 1), increasing the overhead for the FFT's global transpositions at higher processor counts. Thus, even though the code portions utilizing BLAS3 libraries run faster on the X1, the ES achieves higher overall performance. In fact, due to the X1's poor scalability above 128 processors, the ES shows more than a 3.5X runtime advantage when using 256 processors on the larger 686 atom simulation. PARATEC runs efficiently on the Power3, but sustained performance (percent of peak) on the Power4 is lower due, in part, to network contention for memory bandwidth [20]. The loss in scaling on the Power3 is primarily caused by the increased communication cost as concurrency grows to 512 processors. The Power4 system has a much lower bisectionbandwidth to processor speed ratio than the Power3 resulting in poorer scaling to large numbers of processors. The Altix performs well on this code (second only to the ES). This is due to the Itanium2's high memory bandwidth, combined with interconnect network with reasonably-high bandwidth, and extremely low latency (see Table 1). However, higher scalability Altix measurements are not available.

4.2 Quantum Dot Experiments

In October 2004, the authors returned to the Earth Simulator Center, and had the opportunity to run PARATEC experiments on a 488 atom CdSe Cadmium Selenide Quantum Dot — the largest number of grid points simulated to date via this code. The range of 100s – 1000s is typical of the size of dot produced by experiment so it is important to understand the electronic properties of dots of this size by simulation in order to improve their design and fabrication for applications such as electronic dye tags. Results show an impressive performance of 3.64 Gflops/P (45% of peak) using 1024 processors, and 2.67 Gflop/P (33% of peak) at 2048 processors — thus sustaining an aggregate of 5.5 Tflop/s, the highest performance ever achieved for this application. This level of performance opens the possibility to perform scientific Quantum Dot calculations at an unprecedented scale.

5. ASTROPHYSICS

One of the most challenging problems in astrophysics is the numerical solution of Einstein's equations following from the Theory of General Relativity (GR): a set of coupled nonlinear hyperbolic and elliptic equations containing thousands of terms when fully expanded. The Cactus Computational ToolKit [7, 1] is designed to evolve these equations stably in 3D on supercomputers to simulate astrophysical phenomena with high gravitational fluxes, such as the collision of two black holes and the gravitational waves radiating from that event. While Cactus is a modular framework supporting a wide variety of multi-physics applications [10], our study focused exclusively on the GR solver, which implements the ADM-BSSN method [7] for stable evolutions of black holes.

The Cactus GR components solve Einstein's equations as an initial value problem that evolves partial differential equations (PDEs) on a regular grid using finite differences. The core of the GR solver uses the ADM formalism, which decomposes the solution into 3D spatial hypersurfaces that represent different slices of space along the time dimension. In this representation, the equations are written as four constraint equations and 12 evolution equations. Additional stability is provided by the BSSN modifications to the standard ADM method [7]. The evolution equations can be solved using a number of different numerical approaches, including staggered leapfrog, McCormack, Lax-Wendroff, and iterative Crank- Nicholson schemes. A *lapse* function describes the time slicing between hypersurfaces for each step in the evolution while a *shift* vector is used to move the coordinate system at each step to avoid being drawn into a singularity. The four constraint equations are used to select different lapse functions and the related shift vectors.

For parallel computation, the global 3D grid is block domain decomposed so that each processor has its own section. The standard MPI driver for Cactus solves the PDEs on a local region and then updates the values at the ghost zones by exchanging data on the faces of its topological neighbors. On superscalar systems, the computations are blocked in order to improve cache locality. Blocking is accomplished through the use of temporary *slice buffers*, which improve cache reuse while modestly increasing the computational overhead. These blocking optimizations were disabled on vector architectures since they reduced the vector length and inhibited performance.

5.1 CACTUS Results

The full-fledged production version of the Cactus ADM-BSSN application was run on the ES system with results for two grid sizes shown in Table 4. The problem size was scaled with the number of processors to keep the computational load the same (weak scaling). Cactus problems are typically scaled in this manner because their science requires the highest-possible resolutions.

For the vector systems, Cactus achieves almost perfect VOR (over 99%) while the AVL is dependent on the x-dimension size of the local computational domain. Consequently, the larger problem size (250x64x64) executed with far higher efficiency on both vector machines than the smaller test case (AVL = 248 vs. 92), achieving 34% of peak on the ES. The oddly shaped domains for the larger test case were required because the ES does not have enough memory per node to support a 250^3 domain. This rectangular grid configuration had no adverse effect on scaling efficiency despite the worse surface-to-volume ratio. Additional performance gains could be realized if the compiler was able to fuse the X and Y loop nests to form larger effective vector lengths.

Note that that the boundary condition enforcement was not vectorized on the ES and accounts for up to 20% of the execution time, compared with less than 5% on the superscalar systems. This demonstrates a potential limitation of vector architectures: seemingly minor code portions that fail to vectorize can quickly dominate the overall execution time. The architectural imbalance between vector and scalar performance was particularly acute of the X1, which suffered a much greater impact from unvectorized code than the ES. As a result, significantly more effort went into code vectorization of the X1 port – without this optimization, the nonvectorized code portions would dominate the performance profile. Even with this additional vectorization effort, the X1 reached only

Table 4 Cactus per processor performance on 80x80x80 and 250x64x64 grids

Grid Size	*P*	Power3		Power4		Altix		ES		X1	
		Gflops/P	%Pk	Gflops/P	%Pk	Gflops/P	%Pk	Gflops/P	%Pk	Gflops/P	%Pk
80x80x80 per processor	16	0.314	21%	0.577	11%	0.892	15%	1.47	18%	0.540	4%
	64	0.217	14%	0.496	10%	0.699	12%	1.36	17%	0.427	3%
	256	0.216	14%	0.475	9%	—	—	1.35	17%	0.409	3%
	1024	0.215	14%	—	—	—	—	1.34	17%	—	—
250x64x64 per processor	16	0.097	6%	0.556	11%	0.514	9%	2.83	35%	0.813	6%
	64	0.082	6%	—	—	0.422	7%	2.70	34%	0.717	6%
	256	0.071	5%	—	—	—	—	2.70	34%	0.677	5%
	1024	0.060	4%	—	—	—	—	2.70	34%	—	—

6% of peak.

Table 4 shows that the ES reached an impressive 2.7 Tflop/s for the largest problem size using 1024 processors. This represents the highest per processor performance (by far) achieved by the full-production version of the Cactus ADM-BSSN on any evaluated system to date. The Power3, on the other hand, is 45 times slower than the ES, achieving only 60 Mflop/s per processor (6% of peak) at this scale for the larger problem size. The Power4 system offers even lower efficiency than the Power3 for the smaller (80x80x80) problem size, but still ranks high in terms of peak delivered performance in comparison to the X1 and Altix. We were unable to run the larger (250x64x64) problem sizes on the Power4 system, because there were insufficient high-memory nodes available to run these experiments. The Itanium2 processor on the Altix achieves good performance as a fraction of peak for smaller problem sizes (using the latest Intel 8.0 compilers). Observe that unlike vector architectures, microprocessor-based systems generally perform better on the smaller per-processor problem size because of better cache reuse. In terms of communication overhead, the ES spends 13% of the overall Cactus time in MPI compared with 23% on the Power3; highlighting the superior architectural balance of the network design for the ES. The Altix offered very low communication overhead, but its limited size prevented us from evaluating high-concurrency performance.

6. MAGNETIC FUSION

The Gyrokinetic Toroidal Code (GTC) is a 3D particle-in-cell (PIC) application developed at the Princeton Plasma Physics Laboratory to study turbulent transport in magnetic confinement fusion [12, 13]. Turbulence is believed to be the main mechanism by which energy and particles are transported away from the hot plasma core in fusion experiments with magnetic toroidal devices. GTC solves the non-linear gyrophase-averaged Vlasov-Poisson equations [11] for a system of charged particles in a self-consistent, self-generated electrostatic field. The geometry is that of a torus with an externally imposed equilibrium magnetic field, characteristic of toroidal fusion devices. By using the PIC method, the non-linear PDE describing the motion of the particles in the system becomes a simple set of ordinary differential equations (ODEs) that can be easily solved in the Lagrangian coordinates. The self-consistent electrostatic field driving this motion could conceptually be calculated directly from the distance between each pair of particles using an $O(N^2)$ calculation, but the PIC approach reduces it to $O(N)$ by using a grid where each particle deposits its charge to a limited number of neighboring points according to its range of influence. The electrostatic potential is then solved everywhere on the grid using the Poisson equation, and forces are gathered back to each particle. The most computationally intensive parts of GTC are the charge deposition and gather-push steps that involve large loops over the particles, which can reach several million per domain partition.

Although the PIC approach drastically reduces the computational requirements, the grid-based charge deposition phase is a source of performance degradation for both superscalar and vector architectures. Randomly localized particles deposit their charge on the grid, thereby causing poor cache reuse on superscalar machines. The effect of this deposition step is more pronounced on vector systems since two or more particle may contribute to the charge at the same grid point, creating a potential memory-dependency conflict. Several methods have been developed to address this issue; GTC uses the work-vector algorithm [17], where a temporary copy of the grid array is given an extra dimension corresponding to the vector length. Each vector operation acts on a given data set in the register, then writes to a different memory address, avoiding memory dependencies entirely. After the main loop, the results accumulated in the work-vector array are gathered to the final grid array. The only drawback is the increased memory footprint, which can be 2 to 8 times higher than the nonvectorized code version.

Since GTC has previously been vectorized on a single-

Table 5 GTC per processor performance using 10 and 100 particles per cell.

Part/ Cell	Code	*P*	Power3		Power4		Altix		ES		X1	
			Gflops/P	%Pk	Gflops/P	%Pk	Gflops/P	%Pk	Gflops/P	%Pk	Gflops/P	%Pk
10	MPI	32	0.135	9%	0.299	6%	0.290	5%	0.961	12%	1.00	8%
		64	0.132	9%	0.324	6%	0.257	4%	0.835	10%	0.803	6%
100	MPI	32	0.135	9%	0.293	6%	0.333	6%	1.34	17%	1.50	12%
		64	0.133	9%	0.294	6%	0.308	5%	1.25	16%	1.36	11%
	Hybrid	1024	0.063	4%								

node SX-6 [20], porting to the ES was relatively straightforward. However, performance was initially limited due to memory bank conflicts, caused by an access concentration to a few small 1D arrays. Using the *duplicate* pragma directive alleviated this problem by allowing the compiler to create multiple copies of the data structures across numerous memory banks. This method significantly reduced the bank conflicts in the charge deposition routine and increased its performance by 37%. Additional optimizations were performed to other code segments with various performance improvements.

GTC is parallelized at a coarse-grain level using message-passing constructs. Although the MPI implementation achieves almost linear scaling on most architectures, the grid decomposition is limited to approximately 64 subdomains. To run at higher concurrency, a second level of fine-grain loop-level parallelization is implemented using OpenMP directives. However, the increased memory footprint created by the work-vector method inhibited the use of looplevel parallelism on the ES. A possible solution could be to add another dimension of domain or particle decomposition to the code, this strategy is explored in Section 6.2.

Porting to the X1 was straightforward from the vectorized ES version but initial performance was limited. Note that the X1 suffers from the same memory increase as the ES due to the work-vector approach, potentially inhibiting OpenMP parallelism. Several additional directives were necessary to allow effective multistreaming within each MSP. After discovering that the FORTRAN intrinsic function *modulo* was preventing the vectorization of a key loop in the gather-push routine, it was replaced by an equivalent but vectorizable statement *mod.* The most time consuming routine on the X1 became the 'shift' subroutine. This step verifies the coordinates of newly moved particles to determine whether they have crossed a subdomain boundary and therefore require processor migration. The shift routine contains nested *if* statements that prevent the compiler from successfully vectorizing that code region. However, the non-vectorized shift routine accounted for significantly more overhead on the X1 than the ES (54% vs. 11% of overall time). Although both architectures have the same relative vector to scalar peak performance (8/1), serialized loops incur an even larger penalty on the X1. This is because in a serialized segment of a multistreamed code, only one of the four SSP scalar processors within an MSP can do useful work, thus degrading the relative performance ratio to 32/1. Performance on the X1 was improved by converting the nested *if* statements in the shift routine into two successive condition blocks, allowing the compiler to stream and vectorize the code properly. The overhead therefore decreased from 54% to only 4% of the total time. This optimization was implemented on the ES during our subsequent visit to the Earth Simulator (see Section 6.2).

6.1 GTC Results

Table 5 presents GTC performance results on the five architectures examined in our study. The first test case is configured for standard production runs using 10 particles per grid cell (2 million grid point, 20 million particles). The second experiment examines 100 particles per cell (200 million particles), a significantly higher resolution that improves the overall statistics of the simulation while significantly (8 fold) increasing the time-to-solution – making it prohibitively expensive on most superscalar platforms. For the large test case, both the ES and X1 attain high AVL (228 and 62 respectively) and VOR (99% and 97%), indicating that the application has been suitably vectorized; in fact, the vector results represent the highest GTC performance on any tested platform to date. In absolute terms the ES shows the highest performance, achieving 1.56 Gflop/s on the largest problem size for P = 64 — the highest performance on any evaluated architecture to date. Additionally, the ES sustains 20% of peak compared with only 12% on the X1. It should also be noted that because GTC uses single precision arithmetic, the X1 theoretical peak performance is actually 25.6 Gflop/s; however limited memory bandwidth and code complexity that inhibits compiler optimizations obviate this extra capability.

Comparing performance with the superscalar architectures, the ES vector processors are about 12X faster than the Power3 and 5X faster than the Power4 and Alitx sys-

Table 6 Summary of overall performance based on largest available concurrency and problem size

Code	Power3				Power4				Altix				X1				ES			
	Sustained		Total		Sustained		Total		Sustained		Total		Sustained		Total		Sustained		Total	
	Gflop/P	%Pk	CPU	Gflop	Gflop/P	%Pk	CPU	Gflop	Gflop/P	%Pk	CPU	Gflop	Gflop/P	%Pk	CPU	Gflop	Gflop/P	%Pk	CPU	Gflop
LBMHD	0.11	7%	1024	111	0.28	5%	256	71	0.65	11%	64	41	2.70	21%	256	691	3.30	41%	1024	3379
PARATEC	0.41	28%	512	211	1.08	21%	256	276	3.24	54%	64	207	1.27	10%	256	325	2.53	32%	1024	2591
CACTUS	0.60	4%	1024	61	0.48	9%	256	122	0.42	7%	64	27	0.68	5%	256	173	2.70	34%	1024	2765
GTC	0.13	9%	64	9	0.29	6%	64	19	0.31	5%	64	20	1.36	11%	64	87	1.56	20%	64	100
Average	0.18	12%	656	98	0.53	10%	208	122	1.15	19%	64	74	1.50	12%	208	319	2.52	32%	784	2209

tems. Observe that using 1024 processors of the Power3 (in hybrid MPI/OpenMP mode) is still about 35% slower than 64-way vector runs; GTC's OpenMP parallelism is currently unavailable on the vector systems, limiting concurrency to 64 processors (see Section 6 for the higher concurrency implementation). Within the superscalar platforms, the Altix shows the highest raw performance at over 300 Mflop/s, while the Power3 sustains the highest fraction of peak (9% compared with approximately 6% on the Power4 and Altix).

6.2 Particle Decomposition Optimization

At the time of our second visit to the Earth Simulator Center in October 2004, several new improvements had been added to GTC. The most important of them was the implementation of an MPIbased particle decomposition, which added another level of parallelism and gave the code access to higher concurrency execution. The particles inside each spatial domain are now split between a chosen number of processors. Another improvement was the inclusion of a fully vectorized version of the shift subroutine, as it had been implemented on the X1. However, the ES version of that subroutine is quite different from the X1 version since there is no multi-streaming on the ES. Also, the compiler on the ES requires a very strict structure in order to vectorize a loop containing an 'if' statement. Therefore, the main loop in the shift subroutine had to be split in two in order to conform to that restriction.

On a 32-processor test, the performance of the new vectorized shift subroutine increased from 51 Mflop/s to 1351 Mflop/s, thus causing its percentage of wallclock time to decrease from 16% to less than 7%. This improvement (along with other smaller changes) pushed the efficiency of the code up to 25% of peak for the larger problem sizes. The higher concurrency enabled by the new particle decomposition allowed GTC to access a much larger number of processors. The efficiency of the parallel algorithm met the very stringent ES scaling requirements and ran on 2,048 processors, reaching an unprecedented 3.7 Tflop/s using 5 billion particles. This outstanding GTC performance opens the door to a whole new set of very high phase space resolution simulations that have never been explored before, and will be the focus of future scientific experiments.

7. CONCLUSIONS

This work examined four diverse scientific applications on the parallel vector architectures of the ES and X1, and three leading superscalar platforms, the Power3, Power4, and Altix. Since most modern scientific codes are designed for (super) scalar microprocessors, it was necessary to port these applications onto the vector platforms; however only minor code transformations were applied in an attempt to maximize the vector operation ratio and average vector length. Extensive code reengineering has not been performed.

Table 6 summarize performance across all five studied architectures, while Figures 1 and 2 show absolute and sustained performance relative to the ES (using the largest comparable concurrency for each architecture). Overall results show that the ES vector system achieved excellent performance on our application suite – the highest of any architecture tested to date – demonstrating the tremendous potential of modern parallel vector systems. The ES consistently sustained a significantly higher fraction of peak than the X1, due in part to superior scalar processor performance, memory bandwidth, and network bisection bandwidth relative to the peak vector flop rate. A number of performance bottlenecks exposed on the vector machines relate to the extreme sensitivity of these systems to small amounts of unvectorized code. This sheds light on a different dimension of architectural balance than simple bandwidth and latency comparisons. It is important to note that X1-specific code optimizations have not been performed at this time. This complex vector architecture contains both data caches and multi-streaming processing units, and the optimal programming methodology is yet to be established. Finally, preliminary Altix results show promising performance characteristics; however, we tested a relatively small Altix platform and it is unclear if its network performance advantages would

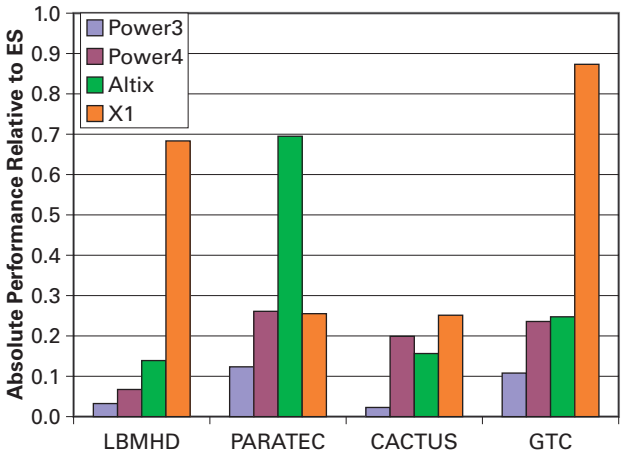

Fig. 1 Absolute performance as a ratio of the ES, based on largest comparable concurrency and problem size.

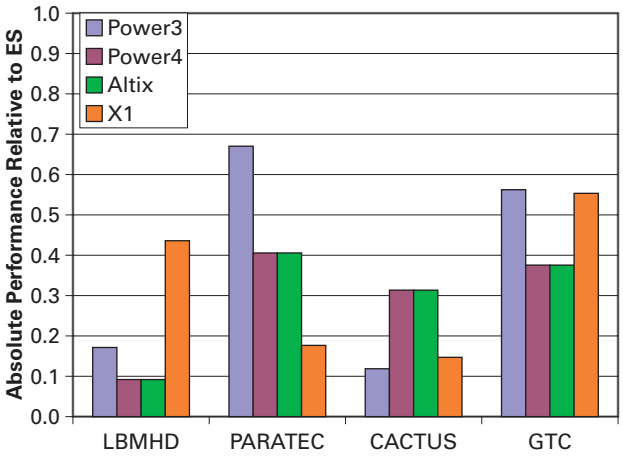

Fig. 2 Sustained percentage of peak as a ratio of the ES, based on largest comparable concurrency and problem size.

Table 7 Comparison between original ES results (top) and updated ES performance gathered from second visit to Earth Simulator Center, October 2004 (bottom). (GTC-PD refers to the updated version of GTC that utilizes particle decomposition.)

Code	Problem	Sustained		Total	
		Gflops/P	%Pk	CPU	Tflop/s
2D LBMHD	8192^2	3.30	41%	1024	3.4
3D LBMHD	1024^3	5.50	68%	4096	22.2
PARATEC	686 Si Atom	2.53	32%	1024	2.6
PARATEC	Quantum Dot	2.67	33%	2048	5.5
GTC	0.2B Particles	1.56	20%	64	0.1
GTC-PD	5B Particles	1.80	23%	2048	3.7

remain for large system configurations.

Finally, Table 7 presents updated ES results from the authors second visit to the Earth Simulator Center in October 2004, where significant improvements were achieved for three of the evaluated codes. First, we examined a 3D-lattice version of LBMHD, which sustained a remarkable 68% of peak on 4096 processors for an aggregate performance of 22.2 Tflop/s. Next, we examined a larger atom simulation for PARATEC, using a CdSe Cadmium Selenide Quantum Dot. The larger grid size of this system allowed scalability to reach 2048 processors while sustaining 33% of peak, for an aggregate performance of 5.5 Tflop/s. Finally, a new implementation of GTC was evaluated that utilizes a second level of parallelism via particle-based decomposition. Results show that — unlike the previous version where concurrency was limited to 64 — the updated GTC code successfully scaled to 2048 processors, attaining an impressive 23% of peak for an aggregate total of 3.7 Tflop/s. These results underscore the potential of the ES system to perform scientific caculations at unprecenteded scale and resolution.

Future work will extend our study to include performance comparisons with the latest generation of high-end computing platforms. We also plan to investigate new application domains, in the areas of climate, molecular dynamics, cosmology, and combustion. We are particularly interested in investigating the vector performance of adaptive mesh refinement (AMR) methods, as we believe they will become a key component of future high-fidelity multi-scale physics simulations, across a broad spectrum of application domains.

Acknowledgments

The authors would like to gratefully thank: J. Snyder of NEC for their help in porting applications to the ES. Special thanks to Thomas Radke, Tom Goodale, and Holger Berger for assistance with the vector Cactus ports. This research used resources of the National Energy Research Scientific Computing Center, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC03-76SF00098. This research used resources of the Center for Computational Sciences at Oak Ridge National Laboratory, which is supported by the Office of Science of the Department of Energy under Contract DE-AC05-00OR22725. All authors from LBNL were supported by the Office of Advanced Scientific Computing Research in the Department of Energy Office of Science under contract number DE-AC03-76SF00098. Dr. Ethier was supported by the Department of Energy under contract number DE-AC020-76-CH03073.

(This article is reviewed by Dr. Tetsuya Sato.)

REFERENCES

[1] Cactus Code Server. http://www.cactuscode.org.

[2] Co-Array Fortran. http://www.co-array.org.

[3] ORNL Cray X1 Evaluation. http://www.csm.ornl.gov/~dunigan/cray.

[4] PARAllel Total Energy Code. http://www.nersc.gov/projects/paratec.

[5] Top500 Supercomputer Sites. http://www.top500.org.

[6] P. A. Agarwal et al. Cray X1 evaluation status report. In *Proc. of the 46th Cray Users Group Conference*, May 17–21, 2004.

[7] M. Alcubierre, G. Allen, B. Brgmann, E. Seidel, and W.-M. Suen. Towards an understanding of the stability properties of the 3+1 evolution equations in general relativity. *Phys. Rev. D*, (gr-qc/9908079), 2000.

[8] P. J. Dellar. Lattice kinetic schemes for magnetohydrodynamics, *J. Comput. Phys*., 79, 2002.

[9] T. H. Dunigan Jr., M. R. Fahey, J. B. White III, and P. H. Worley. Early evaluation of the Cray X1. In *Proc. SC2003: High performance computing, networking, and storage conference*, Phoenix, AZ, Nov 15–21, 2003.

[10] J. A. Font, M. Miller, W. M. Suen, and M. Tobias. Three dimensional numerical general relativistic hydrodynamics: Formulations, methods, and code tests. *Phys. Rev. D*, Phys. Rev. D61, 2000.

[11] W. W. Lee. Gyrokinetic particle simulation model. *J. Comp. Phys*., 72, 1987.

[12] Z. Lin, S. Ethier, T.S. Hahm, and W.M. Tang. Size scaling of turbulent transport in magnetically confined plasmas. *Phys. Rev. Lett*., 88, 2002.

[13] Z. Lin, T. S. Hahm, W. W. Lee, W. M. Tang, and R. B. White. Turbulent transport reduction by zonal flows: Massively parallel simulations. *Science*, Sep 1998.

[14] A. Macnab, G. Vahala, P. Pavlo, , L. Vahala, and M. Soe. Lattice boltzmann model for dissipative incompressible MHD. In *Proc. 28th EPS Conference on Controlled Fusion and Plasma Physics*, volume **25A**, Funchal, Portugal, June 18–22, 2001.

[15] A. Macnab, G. Vahala, L. Vahala, and P. Pavlo. Lattice boltzmann model for dissipative MHD. In *Proc. 29th EPS Conference on Controlled Fusion and Plasma Physics*, volume **26B**, Montreux, Switzerland, June 17–21, 2002.

[16] K. Nakajima. Three-level hybrid vs. flat mpi on the earth simulator: Parallel iterative solvers for finite-element method. In *Proc. 6th IMACS Symposium Iterative Methods in Scientific Computing*, volume **6**, Denver, Colorado, March 27–30, 2003.

[17] A. Nishiguchi, S. Orii, and T. Yabe. Vector calculation of particle code. *J. Comput. Phys*., 61, 1985.

[18] L. Oliker and J. Shalf A. Canning, J. Carter. Scientific computations on modern parallel vector systems. In *Proc. SC2004: High performance computing, networking, and storage conference*, Pittsburgh, PA, Nov. 6–12, 2004.

[19] L. Oliker, R. Biswas, J. Borrill, A. Canning, J. Carter, J. Djomehri, H. Shan, and D. Skinner. A performance evaluation of the Cray X1 for scientific applications. In *VECPAR: 6th International Meeting on High Performance Computing for Computational Science*, Valencia, Spain, June 28–30, 2004.

[20] L. Oliker, A. Canning, J. Carter, J. Shalf, D. Skinner, S. Ethier, R. Biswas, J. Djomehri, and R. Van der Wijngaart. Evaluation of cache-based superscalar and cacheless vector architectures for scientific computations. In Proc. SC2003: High performance computing, networking, and storage conference, Phoenix, AZ, Nov 15–21, 2003.

[21] S. Succi. The lattice boltzmann equation for fluids and beyond. *Oxford Science Publ*., 2001.

[22] H. Uehara, M. Tamura, and M. Yokokawa. MPI performance measurement on the Earth Simulator. Technical Report # 15, NEC Research and Development, 2003/1.