Leading Computational Methods on Scalar and Vector HEC Platforms

Leonid Oliker, Jonathan Carter, Michael Wehner, Andrew Canning
CRD/NERSC, Lawrence Berkeley National Laboratory, Berkeley, CA 94720

Stephane Ethier
Princeton Plasma Physics Laboratory, Princeton University, Princeton, NJ 08453

Art Mirin, Govindasamy Bala
Lawrence Livermore National Laboratory, Livermore, CA 94551

David Parks
NEC Solutions America, Advanced Technical Computing Center, The Woodlands, TX 77381

Patrick Worley
Computer Science and Mathematics Division, Oak Ridge National Laboratory, Oak Ridge, TN 37831

Shigemune Kitawaki, Yoshinori Tsuda
Earth Simulator Center, Japan Agency for Marine-Earth Science and Technology, Yokohama Japan

ABSTRACT

The last decade has witnessed a rapid proliferation of superscalar cache-based microprocessors to build high-end computing (HEC) platforms, primarily because of their generality, scalability, and cost effectiveness. However, the growing gap between sustained and peak performance for full-scale scientific applications on conventional supercomputers has become a major concern in high performance computing, requiring significantly larger systems and application scalability than implied by peak performance in order to achieve desired performance. The latest generation of custom-built parallel vector systems have the potential to address this issue for numerical algorithms with sufficient regularity in their computational structure. In this work we explore applications drawn from four areas: atmospheric modeling (CAM), magnetic fusion (GTC), plasma physics (LBMHD3D), and material science (PARATEC). We compare performance of the vector-based Cray X1, Earth Simulator, and newly-released NEC SX-8 and Cray X1E, with performance of three leading commodity-based superscalar platforms utilizing the IBM Power3, Intel Itanium2, and AMD Opteron processors. Our work makes several significant contributions: the first reported vector performance results for CAM simulations utilizing a finite-volume dynamical core on a high-resolution atmospheric grid; a new data-decomposition scheme for GTC that (for the first time) enables a breakthrough of the Teraflop barrier; the introduction of a new three-dimensional Lattice Boltzmann magneto-hydrodynamic implementation used to study the onset evolution of plasma turbulence that achieves over 26Tflop/s on 4800 ES processors; and the largest PARATEC cell size atomistic simulation to date. Overall, results show that the vector architectures attain unprecedented aggregate performance across our application suite, demonstrating the tremendous potential of modern parallel vector systems.

SC|05 November 12-18, 2005, Seattle, Washington, USA
(c) 2005 ACM 1-59593-061-2/05/0011...$5.00

1. INTRODUCTION

Due to their cost effectiveness, an ever-growing fraction of today's supercomputers employ commodity superscalar processors, arranged as systems of interconnected SMP nodes. However, the constant degradation of superscalar sustained performance has become a well-known problem in the scientific computing community [1]. This trend has been widely attributed to the use of superscalar-based commodity components whose 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 latest generation of custom-built parallel vector systems may address these challenges for numerical algorithms amenable to vectorization.

Vector architectures exploit regularities in computational structures, issuing uniform operations on independent data elements, thus allowing memory latencies to be masked by overlapping pipelined vector operations with memory fetches. Vector instructions specify a large number of identical operations that may execute in parallel, thereby reducing control complexity and efficiently controlling a large amount of computational resources. However, as described by Amdahl's Law, the time taken by the portions of the code that are non-vectorizable can dominate the execution time, significantly reducing the achieved computational rate.

In order to quantify what modern vector capabilities imply for the scientific communities that rely on modeling and simulation, it is critical to evaluate vector systems in the context of demanding computational algorithms. This study examines the behavior of four diverse scientific applications with the potential to run at ultra-scale, in the areas of atmospheric modeling (CAM), plasma physics (GTC), magnetic fusion (LBMHD3D), and material science (PARATEC). We compare the performance of leading com-

Platform	Network	CPU/ Node	Clock (MHz)	Peak (GF/s)	Stream BW (GB/s/CPU)	Peak Stream (Bytes/Flop)	MPI Lat (μsec)	MPI BW (GB/s/CPU)	Network Topology
Power3	SP Switch2	16	375	0.7	0.4	0.26	16.3	0.13	Fat-tree
Itanium2	Quadrics	4	1400	5.6	1.1	0.19	3.0	0.25	Fat-tree
Opteron	InfiniBand	2	2200	4.4	2.3	0.51	6.0	0.59	Fat-tree
X1	Custom	4	800	12.8	14.9	1.16	7.1	6.3	4D-Hypercube
X1E	Custom	4	1130	18.0	9.7	0.54	5.0	2.9	4D-Hypercube
ES	Custom (IN)	8	1000	8.0	26.3	3.29	5.6	1.5	Crossbar
SX-8	IXS	8	2000	16.0	41.0	2.56	5.0	2.0	Crossbar

Table 1: Architectural highlights of the Power3, Itanium2, Opteron, X1, X1E, ES, and SX-8 platforms.

modity-based superscalar platforms utilizing the IBM Power3, Intel Itanium2, and AMD Opteron processors, with modern parallel vector systems: the Cray X1, Earth Simulator (ES), and the NEC SX-8. Additionally, we examine performance of CAM on the recently-released Cray X1E. Our research team was the first international group to conduct a performance evaluation study at the Earth Simulator Center; remote ES access is not available.

Our work builds on our previous efforts [16, 17] and makes several significant contributions: the first reported vector performance results for CAM simulations utilizing a finite-volume dynamical core on a high-resolution atmospheric grid; a new data-decomposition scheme for GTC that (for the first time) enables a breakthrough of the Teraflop barrier; the introduction of a new three-dimensional Lattice Boltzmann magneto-hydrodynamic implementation used to study the onset evolution of plasma turbulence that achieves over 26Tflop/s on 4800 ES processors; and the largest PARATEC cell size atomistic simulation to date. Overall, results show that the vector architectures attain unprecedented aggregate performance across our application suite, demonstrating the tremendous potential of modern parallel vector systems.

2. HEC PLATFORMS AND EVALUATED APPLICATIONS

In this section we briefly describe the computing platforms and scientific applications examined in our study. Table 1 presents an overview of the salient features for the six parallel HEC architectures, including:

- STREAM benchmark results [6] (Stream BW), shows the measured EP-STREAM [3] triad results when all processors within a node simultaneously compete for main memory bandwidth. This represents a more accurate measure of (unit-stride) memory performance than theoretical peak memory behavior.
- The ratio of STREAM bandwidth versus the peak computational rate (Peak Stream).
- Measured internode MPI latency [4, 22].
- Measured bidirectional MPI bandwidth per processor pair when each processor in one node simultaneously exchanges data with a distinct processor in another node[*].

Table 1 shows that the vector systems have significantly higher peak computational rates, memory performance, and MPI bandwidth rates than the superscalar platforms. Observe that the ES and SX-8 machines have significantly higher ratios of memory bandwidth to computational rate than the other architectures in our study. To be fair, bandwidth from main memory is not the sole determiner of achieved percentage of peak. The superscalar systems, and the X1(E), also have memory caches that provide lower latency and higher bandwidth than main memory, potentially mitigating the performance impact of the relatively high cost of main memory access.

In the past, the tight integration of high bandwidth memory and network interconnects to the processors enabled vector systems to effectively feed the arithmetic units and achieve a higher percentage of peak computation rate than nonvector architectures for many codes. This paper focuses on determining the degree to which modern vector systems retain this capability with respect to leading computational methods. While system cost is arguably at least as an important a metric, we are unable to provide such data, as system installation cost is often proprietary and vendor pricing varies dramatically for a given time frame and individual customer.

Three superscalar commodity-based platforms are examined in our study. The IBM Power3 experiments reported were conducted on the 380-node IBM pSeries system, Seaborg, running AIX 5.2 (Xlf compiler 8.1.1) and located at Lawrence Berkeley National Laboratory (LBNL). Each SMP node is a Nighthawk II node consisting of sixteen 375 MHz Power3-II processors (1.5 Gflop/s peak) connected to main memory via a crossbar; SMPs are interconnected via the SP Switch2 switch using an omega-type topology. The AMD Opteron system, Jacquard, is also located at LBNL. Jacquard contains 320 dual nodes and runs Linux 2.6.5 (PathScale 2.0 compiler). Each node contains two 2.2 GHz Opteron processors (4.4 Gflop/s peak), interconnected via InfiniBand fabric in a fat-tree configuration. Finally, the Intel Itanium2 experiments were performed on the Thunder system located at Lawrence Livermore National Laboratory (LLNL). Thunder consists of 1024 nodes, each containing four 1.4 GHz Itanium2 processors (5.6 Gflop/s peak) and running Linux Chaos 2.0 (Fortran version ifort 8.1). The system is interconnected using Quadrics Elan4 in a fat-tree configuration.

We also examine four state-of-the-art parallel vector systems. The Cray X1 [9] utilizes a computational core, called the single-streaming processor (SSP), which contains two 32-stage vector pipes running at 800 MHz. Each SSP contains 32 vector registers holding 64 double-precision words, 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 16KB caches (instruction and data). Four SSPs can be combined into a logical computational unit called the multi-streaming processor (MSP) with a peak of 12.8 Gflop/s — the X1 system can operate in either SSP or MSP modes and we present performance results using both approaches. The four SSPs share a 2-way set associative 2MB data Ecache, a

[*]Because on the X1E pairs of nodes share network ports, the X1E result is the performance when all processors in one pair of nodes exchange data with processors in another node pair.

Name	Lines	Discipline	Methods	Structure
FVCAM	200,000+	Climate Modeling	Finite Volume, Navier-Stokes,FFT	Grid
LBMHD3D	1,500	Plasma Physics	Magneto-Hydrodynamics, Lattice Boltzmann	Lattice/Grid
PARATEC	50,000	Material Science	Density Functional Theory, Kohn Sham, FFT	Fourier/Grid
GTC	5,000	Magnetic Fusion	Particle in Cell, gyrophase-averaged Vlasov-Poisson	Particle/Grid

Table 2: Overview of scientific applications examined in our study.

unique feature for vector architectures that allows extremely high bandwidth (25–51 GB/s) for computations with temporal data locality. The X1 node consists of four MSPs sharing a flat memory. The X1 interconnect is hierarchical, with subsets of 8 SMP nodes connected via a crossbar. For up to 512 MSPs, these subsets are connected in a 4D-Hypercube topology. For more than 512 MSPs, the interconnect is a 2D torus. All reported X1 experiments were performed on a 512-MSP system (several of which were reserved for system services) running UNICOS/mp 2.5.33 (5.3 programming environment) and operated by Oak Ridge National Laboratory. Note that this system is no longer available. It was upgraded to an X1E in July 2005.

We also examine performance of the CAM atmospheric modeling application on the newly-released X1E. The basic building block of both the Cray X1 and X1E systems is the compute module, containing four multi-chip modules (MCM), memory, routing logic, and external connectors. In the X1, each MCM contains a single MSP and a module contains a single SMP node with four MSPs. In the X1E, two MSPs are implemented in a single MCM, for a total of eight MSPs per module. The eight MSPs on an X1E module are organized as two SMP nodes of four MSPs each. These nodes each use half the module's memory and share the network ports. This doubling of the processing density leads to reduced manufacturing costs, but also doubles the number of MSPs contending for both memory and interconnect bandwidth. The clock frequency in the X1E is 41% higher than in the X1, which further increases demands on network and main memory bandwidth. However, this issue is partially mitigated by cache performance that scales bandwidth with processor speed, and by a corrected memory performance problem that had limited memory bandwidth on the X1. All reported X1E experiments were performed on a 768-MSP system running UNICOS/mp 3.0.23 (5.4.0.3 programming environment) and operated by Oak Ridge National Laboratory.

The ES processor is the predecessor of the NEC SX-6, containing 4 vector pipes with a peak performance of 8.0 Gflop/s per CPU [10]. The system contains 640 ES nodes connected through a custom single-stage IN 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 interconnect 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 late 2004.

Finally, we examine the newly-released NEC SX-8. The SX-8 architecture operates at 2 GHz, and contains four replicated vector pipes for a peak performance of 16 Gflop/s per processor. The SX-8 architecture has several enhancements compared with the ES/SX-6 predecessor, including dedicated vector hardware for divide and square root, as well as and in-memory caching for reducing bank conflict overheads. However, the SX-8 in our study uses commodity DDR2-SDRAM; thus, we expect higher memory overhead for irregular accesses when compared with the specialized high-speed FPLRAM (Full Pipelined RAM) of the ES. Both the ES and SX-8 processors contain 72 vector registers each holding 256 doubles, and utilize scalar units operating at one-eighth the peak of their vector counterparts. All reported SX-8 results were run on the 36 node (72 are now currently available) system located at High Performance Computer Center (HLRS) in Stuttgart, Germany. This HLRS SX-8 is interconnected with the NEC Custom IXS network and runs Super-UX (Fortran Version 2.0 Rev.313).

2.1 Scientific Applications

Four applications from diverse areas in scientific computing were chosen to compare the performance of the vector-based X1, X1E, ES, and SX-8 with the superscalar-based Power3, Itanium2, and Opteron systems. The applications are: CAM, the Community Atmosphere Model with the finite-volume solver option for the dynamics; GTC, a magnetic fusion application that uses the particle-in-cell approach to solve non-linear gyrophase-averaged Vlasov-Poisson equations; LBMHD3D, a plasma physics application that uses the Lattice-Boltzmann method to study magneto-hydrodynamics; and PARATEC, a first principles materials science code that solves the Kohn-Sham equations of density functional theory to obtain electronic wavefunctions. An overview of the applications is presented in Table 2.

These codes represent candidate ultra-scale applications that have the potential to fully utilize leadership-class computing systems. Performance results, presented in Gflop/s per processor (denoted as Gflop/P) and percentage of peak, are used to compare the time to solution of our evaluated computing systems. This value is computed by dividing a valid baseline flop-count by the measured wall-clock time of each platform — thus the ratio between the computational rates is the same as the ratio of runtimes across the evaluated systems.

3. FVCAM: ATMOSPHERIC GENERAL CIRCULATION MODELING

The Community Atmosphere Model (CAM) is the atmospheric component of the flagship Community Climate System Model (CCSM3.0). Developed at the National Center for Atmospheric Research (NCAR), the CCSM3.0 is used to study climate change. The CAM application is an atmospheric general circulation model (AGCM) and can be run either coupled within CCSM3.0 or in a stand-alone mode driven by prescribed ocean temperatures and sea ice coverages [7]. AGCMs are key tools for weather prediction and climate research. They also require large computing resources: even the largest current supercomputers cannot keep pace with the desired increases in the resolution and simulation times of these models.

AGCMs generally consist of two distinct sections, the "dynamical core" and the "physics package". The dynamical core approximates a solution to the Navier-Stokes equations suitably expressed

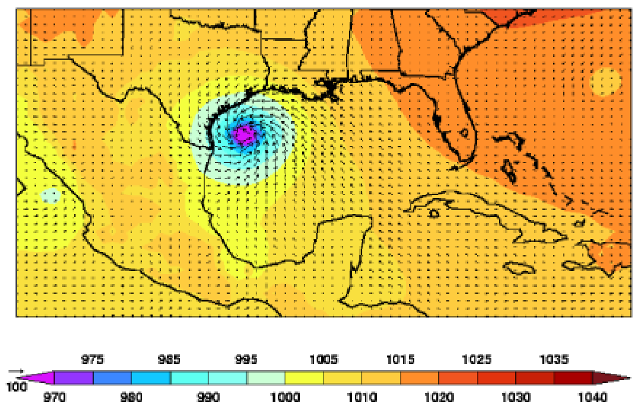


Figure 1: A simulated Category IV hurricane about to reach the coast of Texas. This storm was produced solely through the chaos of the atmospheric model forced only with surface boundary condition. It is one of many such events, without memory of the initial conditions, produced by FVCAM at resolutions of 0.5° degrees or higher.

to describe the dynamics of the atmosphere. The physics package calculates source terms to these equations of motion that represent unresolved or external physical phenomena. These include turbulence, radiative transfer, boundary layer effects, clouds, etc. The physics package will not be discussed further here. The dynamical core of CAM provides two very different options for solving the equations of motion. The first option, known as the Eulerian spectral transform method, exploits spherical harmonics to map a solution onto the sphere [19]. The second option is based on a finite volume methodology, uses a regular latitude-longitude mesh, and conserves mass [12]. In this paper we refer to CAM with the finite volume option as FVCAM. An example result of an FVCAM simulation is shown in Figure 1.

3.1 Parallelization and Vectorization of the Finite-Volume Dycore

The experiments conducted in this work measure the performance of both the original (unvectorized) CAM3.1 version on the superscalar architectures and the vector optimized version on the ES and X1/X1E systems. Five routines were restructured for the vector systems, representing approximately 1000 lines. Four other routines contain a few vector-specific code modifications, totaling 100 additional alternative lines of code. These line counts do not include the 95 Cray compiler directives and the 60 NEC compiler directives also inserted as part of the vector performance optimization. The vector code alternatives are enabled/disabled at compile time using `cpp ifdef` logic. However, use of the vectorized coding on non-vector machines appears to degrade performance only minimally (less than approximately 10%), most likely due to poorer cache utilization.

The solution procedure using the finite-volume dynamical core consists of two phases. First, the main dynamical equations are time-integrated within the control volumes bounded by Lagrangian material surfaces. Second, the Lagrangian surfaces are re-mapped to physical space based on vertical transport [15]. The underlying finite volume grid is logically rectangular in (longitude, latitude, level), where "level" refers to the vertical coordinate. In the dynamics phase, the equations at each vertical level are weakly coupled through the geopotential equation. A two-dimensional domain decomposition in (latitude, level) is employed throughout most of the dynamics phase. (The singularity in the horizontal coordinate system at the pole makes a longitudinal decomposition unattractive.) However, dependencies in the remapping phase are primarily in the vertical dimension and are computed most efficiently using a (longitude, latitude) domain decomposition. The two domain decompositions are connected by transposes [15].

The dynamics phase is structured as a collection of nested subroutines within an outer loop over vertical level. Inner loops are generally with respect to longitude. Prior to vectorization, the latitude loops were at the highest level within this collection of subroutines with latitude indices passed throughout the subroutine chain. One of the main changes in support of vectorization was to move the latitude loops to the lowest level, to provide greatest opportunity for parallelism.

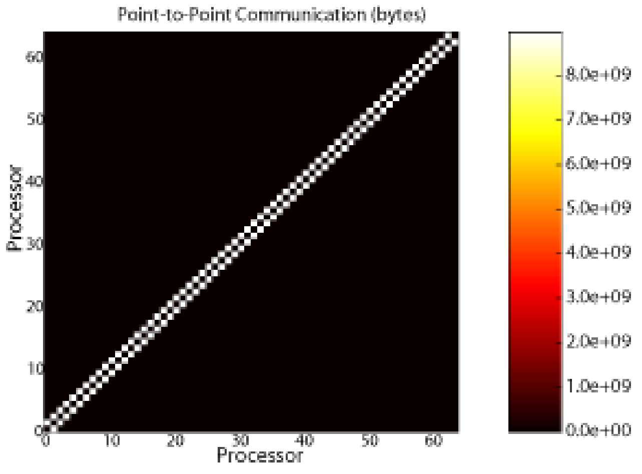
The finite-volume scheme is fundamentally one-sided (upwind) and higher order, causing a significant number of nested logical branches throughout. These branches are so pervasive that there is no practical way to eliminate them from the loops to be vectorized. However, the code has been modified to perform the logical tests with respect to latitude in advance, thereby enabling the partitioning of the loops over latitude via the use of indirect indexing. One area where vectorization proved to be problematic is the implementation of the polar filters. These are Fast Fourier Transforms (FFTs) along complete longitude lines performed at the upper (and lower) latitudes. Vectorization is attained across FFTs (with respect to latitude) as opposed to within the FFT, since the number of FFTs that can be performed in parallel is critical to vector performance. Overall, throughput increases with the number of processors, generating ever finer domain decompositions. However, finer domain decompositions also imply decreasing numbers of latitude lines assigned to each subdomain, thereby restricting performance of the vectorized FFT. No workaround for this issue is apparent.

3.2 Domain Decomposition and Communication Structure

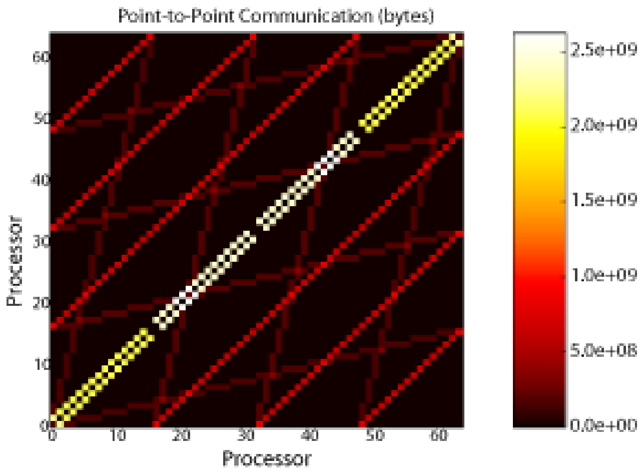
In this paper, we examine performance results obtained from FVCAM in a 0.5°x0.625° horizontal mesh configuration, using the CAM3.1 [2] code version. Sometimes labeled as the D grid, this corresponds to 576 longitudinal grid points and 361 latitudinal grid points. The default number of vertical levels in FVCAM is 26. Performance data were collected on the ES, Power3, Itanium2, and X1. Additionally, ours is the first work to present performance data for FVCAM on the Cray X1E. Results on the Opteron and SX-8 are currently unavailable.

FVCAM is a mixed-mode parallel code, using both the Message Passing Interface (MPI) and OpenMP protocols. For a given processor count, we can specify, at runtime, the number of MPI processes and OpenMP threads per process. For a given processor count we can also specify whether to use a 1D latitude-only domain decomposition or the 2D domain decomposition implementation outlined previously and, for the 2D decomposition, the number of MPI processes to assign to the decomposition of each coordinate direction. In the 2D case, the number of MPI tasks assigned to the vertical dimension (referred to as $\mathrm{P_z}$) is either 4 or 7, as these have been found empirically to be reasonable choices across all of the target platforms. For these experiments, the number of processes assigned to decompose the vertical dimension were also assigned to decompose the longitude dimension during the remapping phase, leaving the number of processes assigned to decompose the latitude dimension unchanged. This simplification minimizes the transposition cost.

The choices of hybrid (MPI/OpenMP) parallelism and domain



(a)



(b)

Figure 2: Volume of point to point communication between MPI processes of FVCAM running D mesh using 256 processors (64 MPI processes), with (a) 1D decomposition and (b) 2D with 4 vertical levels.

decomposition approaches affects performance and scalability. Additional compile- or runtime performance tuning options in FVCAM include a cache or vector-blocking factor for data structures in the physics package, computational load balancing in the physics package, and MPI two-sided and MPI, SHMEM, and Co-Array Fortran one-sided implementations of interprocessor communication [15, 18, 25]. Note that the alternative implementations developed for the vector systems also represent a compile-time tuning option. These many options were examined empirically and set independently for each of the target platforms when collecting the cited performance data. For example, while load balancing improves performance within the physics package, it comes at the cost of additional MPI communication overhead. Only on the Cray X1 and X1E did load balancing improve performance. Similarly, only on the Power3 and ES did OpenMP enhance performance of the current version of FVCAM. (The other evaluated systems either did not support OpenMP or did not benefit from it.) For both of these systems it was found that four OpenMP threads was the optimal choice; this is despite the fact that the ES and the Power3 system have eight and sixteen processors per node (respectively). For comparison purposes we proscribed a number of processor counts and 2D domain decompositions to be tested on all systems, but the optimal choices here are primarily algorithmic and do not vary significantly between platforms.

The effect of OpenMP on FVCAM performance deserves further discussion. First, the number of MPI tasks is limited by the number of latitude lines. The model does not allow less than three latitude lines per subdomain because of tautologies in the latitudinal subdomain communication required for the difference equations. Thus exploiting OpenMP parallelism increases the maximum number of processors that can be used over a pure MPI implementation. Second, communication between subdomains depends on the surface to volume ratio, and can thus become the dominant factor when the three latitude per subdomain limit is approached. For a given number of processors and a fixed $\mathrm{P_z}$, utilizing OpenMP threads both reduces the number of MPI tasks and increases the number of latitudes per subdomain, thereby reducing the communication overhead compared with a pure MPI model.

Figure 2 shows the volume of communication between all pairs of communicating processes within the dynamical core for both a 1D and a 2D (4 vertical level) domain decomposition each using 256 processors (divided between 64 MPI processes and 4 OpenMP threads), as captured by the IPM profiling tool [20]. The communication pattern of the 1D domain decomposition is a straightforward nearest neighbor pattern. This is expected as each subdomain encompasses all longitudinal points and borders only two other subdomains, one to the north and one to the south. On the other hand, the point-to-point communication pattern of the 2D decomposition is decidedly nonlocal. The bulk of the volume of communication is still nearest neighbor but differs somewhat in detail.

For the 1D decomposition (Figure 2a), the two diagonals are continuous and represent processors which together contain all of the horizontal points. In the 2D case (Figure 2b), the diagonals are segmented into four parts, where each part corresponds to a subdomain in the vertical decomposition. In this case, each of these segments then represent processors which together contain all of the horizontal points but over only a range of vertical levels. Note that the intercepts with the boundaries of the plot also correspond with the gaps in the diagonal segments. Hence, the number of continuous diagonal segments (of paired processors) is equal to the vertical discretization $\mathrm{P_z}$. There are also $\mathrm{P_z} - 1$ lines of communicating processors parallel to and on either side of the two diagonals. These then represent communications in the vertical direction which are of a considerably lesser volume. Finally, there is a tilted grid of four by four lines connecting these boundary intercepts. These lines represent the communication associated with transposition. Note the differences in scale of the color bar in these two plots. This indicates that total volume of communication in the 2D decomposition is significantly reduced compared with the 1D approach, due to an improved surface to volume ratio.

3.3 Experimental Results

Table 3 shows a performance comparison of the per processor computational rate and percent of theoretical performance for the Power3, Itanium2, X1, X1E, and ES. As mentioned previously, attempts were made to optimize the performance of each machine by utilizing a variety of compile and runtime configuration options. OpenMP was utilized only on the Power3 and ES, as the other plat-

Decomp	P	Power3		Itanium2		X1 (MSP)		X1E (MSP)		ES	
		Gflop/P	%Pk	Gflop/P	%Pk	Gflop/P	%Pk	Gflop/P	%Pk	Gflop/P	%Pk
1D	32	0.12	8.3	0.40	7.2	1.72	13.5	1.88	10.4	1.33	16.6
	64	0.12	8.2	—	—	—	—	1.67	9.3	1.12	14.0
	128	0.11	7.6	—	—	—	—	—	—	0.81	10.1
	256	0.10	6.7	—	—	—	—	—	—	0.54	6.7
2D 4Vert	128	0.11	7.6	0.33	5.8	1.34	10.5	1.48	8.2	1.01	12.7
	256	0.09	6.3	0.30	5.3	1.05	8.2	1.19	6.6	0.83	10.4
	376	—	—	0.27	4.7	—	—	0.99	5.5	—	—
	512	0.09	5.9	—	—	—	—	—	—	0.57	7.0
2D 7 Vert	336	0.09	5.7	0.29	5.2	0.96	7.5	1.09	6.1	0.79	9.8
	644	—	—	0.23	4.1	—	—	0.71	4.0	—	—
	672	0.07	4.9	—	—	—	—	0.70	3.8	0.56	7.0
	896	0.06	4.3	—	—	—	—	—	—	0.44	5.5
	1680	0.05	3.7	—	—	—	—	—	—	—	—

Table 3: Performance of FVCAM using various domain decompositions for the D grid on the Power3, Itanium2, X1, X1E, and ES

forms did not benefit. For these experiments, initialization was not timed and simulation output was minimized and delayed until the end of the run. Performance timings were collected for 1, 2, 3, and/or 30 simulation days. The average time per simulation day was calculated directly from the 30 simulation day runs. If 30 simulation day runs were too expensive, time per simulation day was approximated by differencing the times required for the 1, 2, and 3 simulation day runs, thus eliminating both start-up costs and the remaining output costs. As initialization and I/O costs are not reported, the numbers in the table refer to only that portion of the code spent advancing the physics and dynamics timesteps. (While the I/O costs of FVCAM can be substantial if output is required on a subdaily interval, results for typical long climate simulation are output only every 30 simulated days, in which case I/O is not usually a significant cost.)

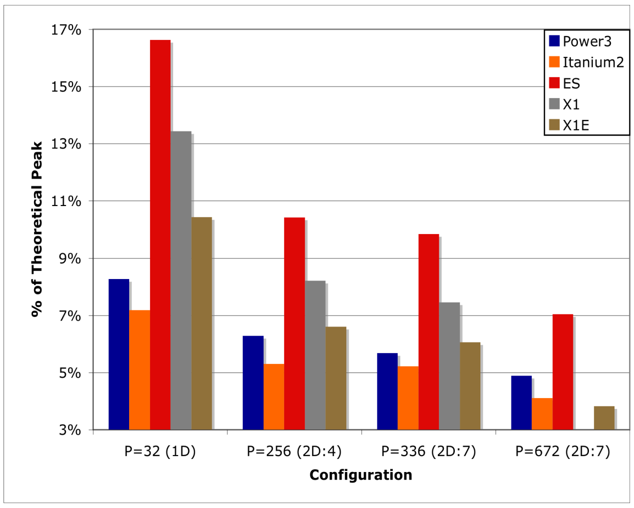


Figure 3: Percentage of theoretical peak of FVCAM using various domain decompositions for the D grid on the Power3, Itanium2, X1, X1E, and ES. (The Itanium2 P=672 data is based on P=644 performance.)

Figure 3 shows a comparison of the percent of theoretical peak processor performance obtained on each machine using selected domain decompositions: P=32 (1D), P=256 (2D 4-levels), P=336

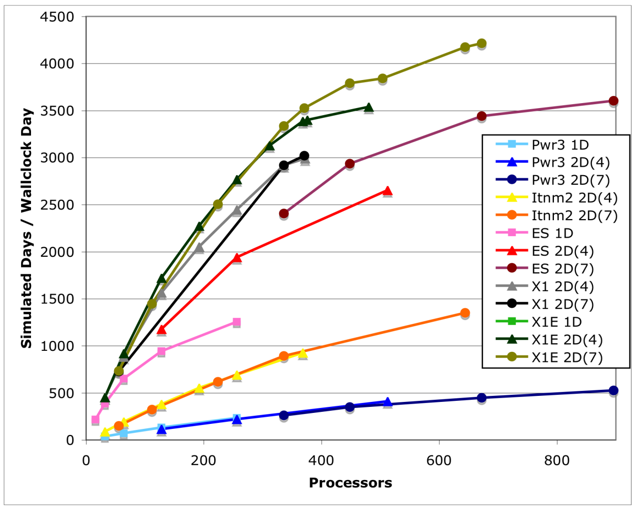


Figure 4: Simulated days per wall-clock day of FVCAM using various domain decompositions for the D grid on the Power3, Itanium2, X1, X1E, and ES

(2D 7-levels), and P=672 (2D 7-levels). (Note that the Thunder results shown for P=672 are actually for 644 processors.) Observe that the ES consistently achieves the highest percentage of peak, generally followed by the X1, X1E, Power3, and Itanium2. For all machines, the percent of peak performance decreases as the number of processors increases, due in part to increased communication costs and load imbalances. The vector platforms also suffer from a reduction in vector lengths at increasing concurrencies for this fixed size problem. Note that the X1E processor increases FVCAM performance by about 14% compared to the X1, even though its peak speed is 41% higher. This is partially due to increased contention for memory and interconnect resources, as well as a relative increase in processor speed without a commensurate increase in main memory performance.

Figure 4 shows a direct comparison of timing results performed on all the machines for the three different domain decompositions considered (one dimensional and two dimensional with either 4 or 7 vertical subdomains) for each of the machines. The figure of merit most relevant to the climate modeler, namely how long it takes to complete a given simulation, is shown in Figure 4 (in units of simulated days per wall-clock day). As production climate change integrations are typically in the range of 100 to 1000 simulated years, achieving a factor of 1000 times or more is necessary for the simulation to be tractable. Each of the machines has been run close to but not exceeding saturation, i.e. where this performance curve turns over due to MPI communication costs and/or vector length issues. The speedup over real time of over 4200 on 672 processors of the Cray X1E is the highest performance ever achieved for FVCAM at this resolution.

4. GTC: TURBULENT TRANSPORT IN MAGNETIC FUSION

GTC is a 3D particle-in-cell code used for studying turbulent transport in magnetic fusion plasmas [13]. The simulation geometry is that of a torus (see Figure 5), which is the natural configuration of all tokamak fusion devices. As the charged particles forming the plasma move within the externally-imposed magnetic field, they collectively create their own self-consistent electrostatic (and electromagnetic) field that quickly becomes turbulent under driving temperature and density gradients. Waves and particles interact self-consistently with each other, exchanging energy that grows or damps their motion or amplitude. The particle-in-cell (PIC) method describes this complex phenomenon by solving the 5D gyro-averaged kinetic equation coupled to the Poisson equation.

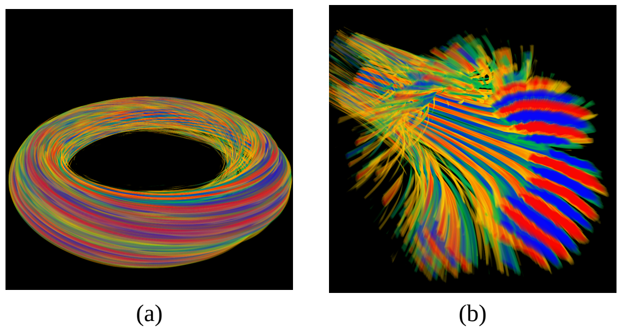


Figure 5: Advanced volume visualization of the electrostatic potential field created by the plasma particles in a GTC simulation. Figure (a) shows the whole volume, and (b) a cross-section through a poloidal plane where the elongated eddies of the turbulence can be seen.[*]

In the PIC approach, the particles interact with each other via a spatial grid on which the charges are deposited, and the field is then solved using the Poisson equation. By doing this, the number of operations scales as N instead of N^2, as would be the case for direct binary interactions. Although this methodology 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 system, since two or more particles may contribute to the charge at the same grid point — creating a potential memory-dependency conflict.

The memory-dependency conflicts during the charge deposition phase can be avoided by using the work-vector method [16], where each element (particle) in a vector register writes to a private copy of the grid. The work-vector approach requires as many copies of the grid as the number of elements in the vector register (256 for the ES and X1 in MSP mode). Although this technique allows full vectorization of the scatter loop on the vector systems, it consequently increases the memory footprint 2–8X compared with the same calculation on a superscalar machine. This increase in memory is the main reason why GTC's mixed-mode parallelism (MPI/OpenMP) cannot be used on the vector platforms. The shared-memory loop-level parallelism also requires private copies of the grid for each thread in order to avoid memory contentions, thus severely limiting the problem sizes that can be simulated. Furthermore, the loop-level parallelization reduces the size of the vector loops, which in turn decreases the overall performance.

[*]Images provided by Prof. Kwan-Liu Ma of UC Davis, as part of the DOE SciDAC GPS project.

4.1 Particle Decomposition Parallelization

GTC was originally optimized for superscalar SMP-based architectures by utilizing two levels of parallelism: a one-dimensional MPI-based domain decomposition in the toroidal direction, and a loop-level work splitting method implemented with OpenMP. However, as mentioned in Section 4, the mixed-mode GTC implementation is poorly suited for vector platforms due to memory constraints and the fact that vectorization and thread-based loop-level parallelism compete directly with each other. As a result, previous vector experiments [16] were limited to 64-way parallelism — the optimal number of domains in the 1D toroidal decomposition. Note that the number of domains (64) is not limited by the scaling of the algorithm but rather by the physical properties of the system, which features a quasi two-dimensional electrostatic potential when put on a coordinate system that follows the magnetic field lines. GTC uses such a coordinate system, and increasing the number of grid points in the toroidal direction does not change the results of the simulation.

To increase GTC's concurrency in pure MPI mode, a third level of parallelism was recently introduced. Since the computational work directly involving the particles accounts for almost 85% of the overhead, the updated algorithm splits the particles between several processors within each domain of the 1D spatial decomposition. Each processor then works on a subgroup of particles that span the whole volume of a given domain. This allows us to divide the particle-related work between several processor and, if needed, to considerably increase the number of particles in the simulation. The update approach maintains a good load balance due to the uniformity of the particle distribution.

There are several important reasons why we can benefit from experiments that simulate larger numbers of particles. For a given fusion device size, the grid resolution in the different directions is well-defined and is determined by the shortest relevant electrostatic (and electromagnetic) waves in the gyrokinetic system. The way to increase resolution in the simulation is by adding more particles in order to uniformly raise the population density in phase space or put more emphasis near the resonances. Also, the particles in the gyrokinetic system are not subject to the Courant condition limitations [11]. They can therefore assume a high velocity without having to reduce the time step. Simulations with multiple species are essential to study the transport of the different products created by the fusion reaction in burning plasma experiments. These multi-species calculations require a very large number of particles and will benefit from the added decomposition.

4.2 Experimental Results

For this performance study, we keep the grid size constant but increase the total number of particles so as to maintain the same

P	Part/ Cell	Power3		Itanium2		Opteron		X1 (MSP)		X1 (4-SSP)		ES		SX-8	
		Gflop/P	%Pk	Gflop/P	%Pk	Gflop/P	%Pk	Gflop/P	%Pk	Gflop/P	%Pk	Gflop/P	%Pk	Gflop/P	%Pk
64	100	0.14	9.3	0.39	6.9	0.59	13.3	1.29	10.1	1.12	8.6	1.60	20.0	2.39	14.9
128	200	0.14	9.3	0.39	6.9	0.59	13.3	1.22	9.6	1.00	7.7	1.56	19.5	2.28	14.2
256	400	0.14	9.3	0.38	6.9	0.57	13.1	1.17	9.1	0.92	7.4	1.55	19.4	2.32	14.5
512	800	0.14	9.4	0.38	6.8	0.51	11.6	—	—	—	—	1.53	19.1	—	—
1024	1600	0.14	8.7	0.37	6.7	—	—	—	—	—	—	1.88	23.5	—	—
2048	3200	0.13	8.4	0.37	6.7	—	—	—	—	—	—	1.82	22.7	—	—

Table 4: GTC performance on each of the studied architectures using a fixed number of particles per processor. SSP results are shown as the aggregate performance of 4 SSPs to allow a direct comparison with MSP performance.

number of particles per processor, where each processor follows about 3.2 million particles. Table 4 shows the performance results for the six architectures in our study. The first striking difference from the previous GTC vector study [16], is the considerable increase in concurrency. The new particle decomposition algorithm allowed GTC to efficiently utilize 2,048 MPI processes (compared with only 64 using the previous approach), although this is not the limit of its scalability. With this new algorithm in place, GTC fulfilled the very strict scaling requirements of the ES and achieved an unprecedented 3.7 Tflop/s on 2,048 processors. Additionally, the Earth Simulator sustains a significantly higher percentage of peak (24%) compared with other platforms. In terms of absolute performance, the SX-8 attains the fastest time to solution, achieving 2.39 Gflop/s per processor. However, this is only about 50% higher than the performance of the ES processor, even though the SX-8 peak is twice that of the ES. We believe that this is due to the memory access speed, to which GTC's gather/scatter operations are quite sensitive. Although the SX-8 has twice the raw computational power, the speed for random memory accesses has not been scaled accordingly. Faster FPLRAM memory is available for the SX-8 and would certainly increase GTC performance; however this memory technology is more expensive and less dense then the commodity DDR2-SDRAM used in the evaluated SX-8 platform.

Of the superscalar systems tested, the AMD Opteron with InfiniBand cluster (Jacquard) gives impressive results. GTC achieved a little over 13% of peak on this computer and was 50% faster than on the Itanium2 Quadrics cluster (Thunder). The InfiniBand interconnect seems to scale very well up to the largest number of available processors (512), although at the highest concurrency we do see signs of performance tapering. On the other hand, the Quadrics-Elan4 interconnect of Thunder scales extremely well all the way up to the 2,048-processor case.

Observe that the X1(MSP) runs only two times faster than the Opteron for the same number of processors, reaching only 10% of peak, while the X1(SSP) achieves even slightly lower performance than the MSP version. A preliminary performance analysis indicates that a loop that represents over 20% of the runtime with the current version of the code and benchmark problem, required very little time in earlier X1 runs with the previous version of GTC and a benchmark problem with many fewer particles per cell. Future work will investigate the optimization of this loop nest, as it creates a performance bottleneck on the X1 when using large numbers of particles per cell.

Finally, note that the new particle decomposition added several reduction communications to the code. These *Allreduce* calls involve only the sub-groups of processors between which the particles are split within a spatial domain. As the number of processors involved in this decomposition increases, the overhead due to these reduction operations increases as well. Nonetheless, all of the studied interconnects showed good scale behavior, including the (relatively old) SP Switch2 switch of the IBM Power3 (Seaborg).

5. LBMHD3D: LATTICE BOLTZMANN MAGNETO-HYDRODYNAMICS

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. As a further development of previous 2D codes, LBMHD3D simulates the behavior of a three-dimensional conducting fluid evolving from simple initial conditions through the onset of turbulence. Figure 6 shows a slice through the xy-plane of an example simulation. Here, the vorticity profile has considerably distorted after several hundred time steps as computed by LBMHD. The 3D spatial grid is coupled to a 3DQ27 streaming lattice and block distributed over a 3D Cartesian processor grid. Each grid point is associated with a set of mesoscopic variables whose values are stored in vectors of length proportional to the number of streaming directions — in this case 27 (26 plus the null vector).

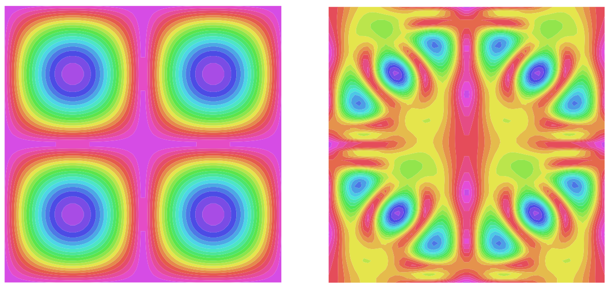


Figure 6: Contour plot of xy-plane showing the evolution of vorticity from well-defined tube-like structures into turbulent structures.

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. A key optimization described by Wellein and co-workers [24] was implemented, saving on the work required by the stream step. They noticed that the

P	Grid Size	Power3		Itanium2		Opteron		X1 (MSP)		X1 (SSP)		ES		SX-8	
		Gflop/P	%Pk	Gflop/P	%Pk	Gflop/P	%Pk	Gflop/P	%Pk	Gflop/P	%Pk	Gflop/P	%Pk	Gflop/P	%Pk
16	256^3	0.14	9.3	0.26	4.6	0.70	15.9	5.19	40.5	—	—	5.50	68.7	7.89	49.3
64	256^3	0.15	9.7	0.35	6.3	0.68	15.4	5.24	40.9	—	—	5.25	65.6	8.10	50.6
256	512^3	0.14	9.1	0.32	5.8	0.60	13.6	5.26	41.1	1.34	42.0	5.45	68.2	9.52	59.5
512	512^3	0.14	9.4	0.35	6.3	0.59	13.3	—	—	1.34	41.8	5.21	65.1	—	—
1024	1024^3	—	—	—	—	—	—	—	—	1.30	40.7	5.44	68.0	—	—
2048	1024^3	—	—	—	—	—	—	—	—	—	—	5.41	67.6	—	—

Table 5: LBMHD3D performance on each of the studied architectures for a range of concurrencies and grid sizes. The vector architectures very clearly outperform the scalar systems by a significant factor.

two phases of the simulation could be combined, so that either the newly calculated particle distribution function could be scattered to the correct neighbor as soon as it was calculated, or equivalently, data could be gathered from adjacent cells to calculate the updated value for the current cell. Using this strategy, only the points on cell boundaries require copying.

5.1 Vectorization Details

The basic computational structure consists of three nested loops over spatial grid points (each typically run 100-1000 loop iterations) with inner loops over velocity streaming vectors and magnetic field streaming vectors (typically 10-30 loop iterations), performing various algebraic expressions. For the ES, the innermost loops were unrolled via compiler directives and the (now) innermost grid point loop was vectorized. This proved a very effective strategy and this was also followed on the X1. In the case of the X1, however, there is a richer set of directives to control both vectorization and multi-streaming, to provide both hints and specifications to the compiler. Finding the right mix of directives required more experimentation than in the case of the ES. No additional vectorization effort was required due to the data-parallel nature of LBMHD. For the superscalar architectures, we utilized a data layout that has been shown previously to be optimal on cache-based machines [24], but did not explicitly tune for the cache size on any machine. Interprocessor communication was implemented by first copying the non-contiguous mesoscopic variables data into temporary buffers, thereby reducing the required number of communications, and then calling point-to-point MPI communication routines.

5.2 Experimental Results

Table 5 presents LBMHD performance across the six architectures evaluated in our study. Observe that the vector architectures clearly outperform the scalar systems by a significant factor. In absolute terms, the SX-8 is the leader by a wide margin, achieving the highest per processor performance to date for LBMHD3D. The ES, however, sustains the highest fraction of peak across all architectures — an amazing 68% even at the highest 2048-processors concurrencies. Further experiments on the ES on 4800 processors attained an unprecedented aggregate performance of over 26 Tflop/s. Examining the X1 behavior, we see that in MSP mode absolute performance is similar to the ES, while X1(SSP) and X1(MSP) achieve similar percentages of peak. The high performance of the X1 is gratifying since we noted several outputed warnings concerning vector register spilling during the optimization of the collision routine. Because the X1 has fewer vector registers than the ES/SX-8 (32 vs 72), vectorizing these complex loops will exhaust the hardware limits and force spilling to memory. That we see no performance penalty is probably due to the spilled registers being effectively cached. We tried one additional experiment to compare the performance of an equivalent amount of computational resources, 64 X1 MSPs versus 256 SSPs, over the same grid size of 256^3. Results show that the LBMHD simulation is greatly benefiting from the MSP paradigm, as it outperforms the SSP approach by over 50%.

Turning to the superscalar architectures, the Opteron cluster outperforms the Itanium2 system by almost a factor of 2X. One source of this disparity is that the Opteron achieves a STREAMS [6] bandwidth of more than twice that of the Itanium2 (see Table 1). Another possible source of this degradation are the relatively high cost of inner-loop register spills on the Itanium2, since the floating point values cannot be stored in the first level of cache. Given the age and specifications, the Power3 does quite reasonably, obtaining a higher percent of peak that the Itanium2, but falling behind the Opteron.

6. PARATEC: FIRST PRINCIPLES MATERIALS SCIENCE

PARATEC (PARAllel Total Energy Code [5]) 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.

PARATEC is written in F90 and MPI and is designed primarily for massively parallel computing platforms, but can also run on serial machines. The code has run on many computer architectures and uses preprocessing to include machine specific routines such as the FFT calls. Much of the computation time (typically 60%) involves FFTs and BLAS3 routines, which run at a high percentage of peak on most platforms.

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 parallel 3D FFTs to transform the wavefunctions between the two spaces. The global data transposes within these FFT operations account for the bulk of PARATEC's communication overhead, and can quickly become the bottleneck at high concurrencies. We use our own handwritten 3D FFTs rather than library routines as the data layout in Fourier space is a sphere of points, rather than a standard square grid. The sphere is load bal-

P	Power3		Itanium2		Opteron		X1 (MSP)		X1 (4-SSP)		ES		SX-8	
	Gflop/P	%Pk	Gflop/P	%Pk	Gflop/P	%Pk	Gflop/P	%Pk	Gflop/P	%Pk	Gflop/P	%Pk	Gflop/P	%Pk
64	0.94	62.9	—	—	—	—	4.25	33.2	4.32	33.8	—	—	7.91	49.4
128	0.93	62.2	2.84	50.7	—	—	3.19	24.9	3.72	28.9	5.12	64.0	7.53	47.1
256	0.85	56.7	2.63	47.0	1.98	45.0	3.05	23.8	—	—	4.97	62.1	6.81	42.6
512	0.73	48.8	2.44	43.6	0.95	21.6	—	—	—	—	4.36	54.5	—	—
1024	0.60	39.8	1.77	31.6	—	—	—	—	—	—	3.64	45.5	—	—
2048	—	—	—	—	—	—	—	—	—	—	2.67	33.4	—	—

Table 6: PARATEC results using 488 atom CdSe quantum dot on the evaluated platforms.SSP results are shown as the aggregate performance of 4 SSPs to allow a direct comparison with MSP performance.

anced by distributing the different length columns from the sphere to different processors such that each processor holds a similar number of points in Fourier space. Effective load balancing is important, as much of the compute intensive part of the calculation is carried out in Fourier space.

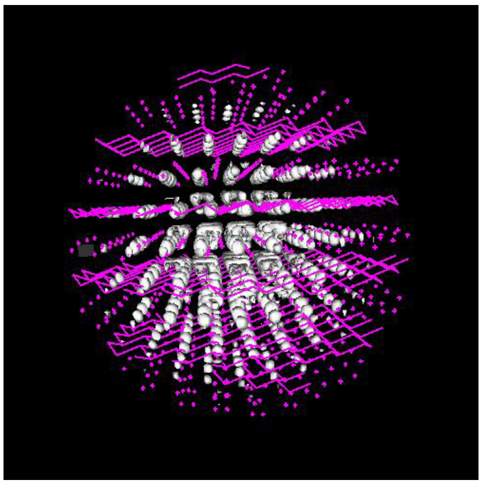


Figure 7: Conduction band minimum electron state for a CdSe quantum dot of the type used in the PARATEC experiments [23].

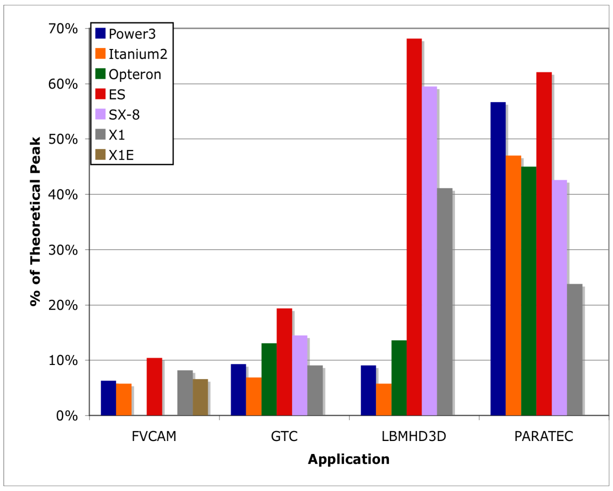
6.1 Experimental Results

Table 6 presents performance data for 3 CG steps of a 488 atom CdSe (Cadmium Selenide) quantum dot and a standard Local Density Approximation (LDA) run of PARATEC with a 35 Ry cutoff using norm-conserving pseudopotentials. A typical calculation would require at least 60 CG iterations to converge the charge density for a CdSe dot. CdSe quantum dots are luminescent in the optical range at different frequencies depending on their size and can be used as electronic dye tags by attaching them to organic molecules. They represent a nanosystem with important technological applications. Understanding their properties and synthesis through first principles simulations represents a challenge for large-scale parallel computing, both in terms of computer resources and of code development. This 488 atom system is, to the best of our knowledge, the largest physical system (number of real space grid points) ever run with this code. Previous vector results for PARATEC [16] examined smaller physical systems at lower concurrencies. Figure 7 shows an example of the computed conduction band minimum electron state for a CdSe quantum dot.

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 transformations during the parallel 3D FFTs which require global interprocessor communications. Even though the 3D FFT was written to minimize global communications, architectures with a poor balance between their bisection bandwidth and computational rate will suffer performance degradation at higher concurrencies. Table 6 shows that PARATEC achieves unprecedented performance on the ES system, sustaining 5.5 Tflop/s for 2048 processors. The declining performance at higher concurrencies 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. On the SX-8 the code runs at a lower percentage of peak than on the ES, due most likely to the slower memory on the SX8; however, the SX8 does achieve the highest per processor performance of any machine tested to date.

Observe that absolute X1 performance is lower than the ES, even though it has a higher peak speed. One reason for this is that the relative difference between vector and nonvector performance is higher on the X1 than on the ES. In consequence, on the X1 the code spends a much smaller percentage of the total time in highly optimized 3D FFTs and BLAS3 libraries than on any of the other machines. The other code segments are handwritten F90 routines and have a lower vector operation ratio currently, resulting in relatively poorer X1 performance. Running PARATEC in SSP mode on the X1 was found to give some improvements in performance over MSP mode. For example using the 128 MSP in SSP mode (i.e. 512 SSPs) resulted in a performance increase of 16%. This is partly because in a serialized segment of a multistreamed code, only one of the four SSP scalar processors within an MSP can do useful work; however running the same code in SSP mode allows all four scalar units to participate in the computation.

PARATEC also runs efficiently on the scalar platforms, achieving over 60% of peak on the Power3 using 128 processors, with reasonable scaling continuing up to 1024 processors. This percentage of peak is significantly higher than on any of the other applications studied in this paper, and is in large part due to the use of the optimized ESSL libraries for the FFTs and dense linear algebra operations used in the code. The loss in scaling on the Power3 is primarily due to the increased communication cost at high concurrencies. The Itanium2 and Opteron systems show a similar high percentage of peak at lower concurrencies, giving the Itanium2 a higher absolute performance than the Opteron system. This is again due mainly to the use of optimized FFT and BLAS3 libraries, which are highly cache resident. Thus PARATEC is less sensitive to memory access issues on the Itanium2 than the other codes tested in this paper. Additionally, the Opteron's performance can be limited for dense lin-



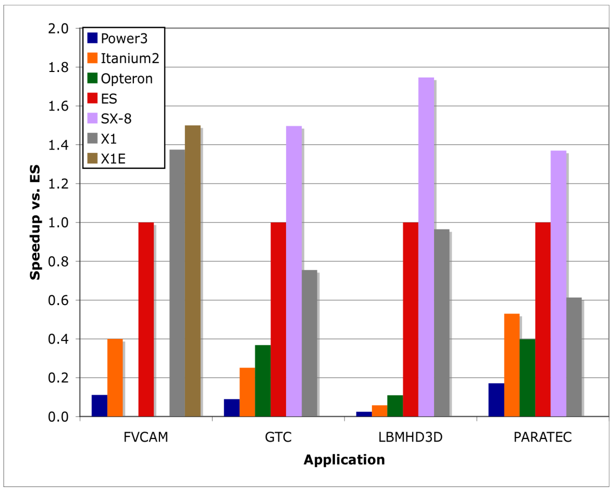


Figure 8: Overview of performance for the four studied applications on 256 processors, comparing (left) percentage of theoretical peak and (right) absolute speed relative to ES.

ear algebra computations due to its the lack of floating-point multiply add (FMA) hardware. The Quadrics-based Itanium2 platform also shows better scaling characteristics at high concurrency than the InfiniBand-based Opteron system, for the global all-to-all communication patterns in PARATEC's 3D FFTs.

7. CONCLUSIONS

This study examines four diverse scientific applications on the parallel vector architectures of the X1, X1E, ES and SX-8, and three leading superscalar platforms utilizing Power3, Itanium2, and Opteron processors. Overall results show that the vector platforms achieve the highest aggregate performance on any tested architecture to date across our full application suite, demonstrating the tremendous potential of modern parallel vector systems. Our work makes several significant contributions. We are the first to present vector results for the Community Atmosphere Model using the finite-volume solver in the dynamics phase of the calculation. Results on a $0.5°$x$0.625°$ (D) mesh show that the X1E achieves unprecedented aggregate performance at the studied high-fidelity resolution. Additionally, we perform a detailed analysis of FVCAM's topological communication requirements using various domain decomposition approaches. We also present a new decomposition-based parallelization for the GTC magnetic fusion simulation. This new approach allows scalability to 2048 processors on the ES (compared to only 64 using the previous code version), opening the door to a new set of high-phase space-resolution simulations that to date have not been possible. Next we present, for the first time, LBMHD3D: a 3D version of a Lattice Bolzmann magneto-hydrodynamics application used to study the onset evolution of plasma turbulence. The ES shows unprecedented LBMHD3D performance, achieving over 68% of peak for a total of 26Tflop/s on 4800 processors. Finally, we investigate performance of the PARATEC application, using the largest cell size atomistic simulation ever run with this material science code. Results on 2048 processors of the ES show the highest aggregate performance to date, allowing for high-fidelity simulations that hitherto have not been possible with PARATEC due to computational limitations.

Figure 8 presents a performance overview for the four studied applications using 256 processors, comparing (left) percentage of theoretical peak and (right) runtime relative to ES. Overall, the ES achieves the highest percentage of peak across all of the architecture examined in our study. The newly-released SX-8 can not match the efficiency of the ES, due in part, to the higher memory latency overhead for irregular data accesses. In addition, we note that the ES and SX-8 consistently achieve a higher fraction of peak than the X1, in most cases considerably higher. This is due, in part, to superior scalar processor performance and memory bandwidth. The SX-8 does achieve the highest per-processor performance for LBMHD3D, GTC, and PARATEC, while the newly-released X1E attains the highest per-processor performance for FVCAM (FVCAM results on the SX-8 are currently unavailable and X1E data are available only for FVCAM currently.) Results also show that for FVCAM, the X1E percentage of peak is somewhat lower than the X1, due in part to the architectural balance of the X1E, which utilizes twice as many MSPs per node and a higher clock speed (compared with the X1), without a commensurate increase in memory performance. Future work will focus extensively on evaluating X1E behavior.

Our work also examines the tradeoffs between running in either SSP or MSP modes on the X1. For a fixed size problem, one can perform a performance comparison by either using one MSP or four times as many SSPs. MSP mode has the advantage of increasing the granularity of the subdomains, requiring one fourth as many processes or threads, thus improving the surface to volume ratio and the corresponding communication overhead. On the other hand, for applications with nontrivial portions of unvectorized code, the SSP approach may be advantageous, since all four SSP scalar processors can participate in the computation — compared with MSP mode where only one of the four SSP scalar processors can do useful work. Finally, SSP mode may also hold an advantage for codes containing short vector lengths or loop bodies that prevent multi-streaming. Results show that for LBMHD3D and GTC the amount of work not effectively multi-streamed, but still amenable to vectorization, appears very low. Correspondingly, both applications benefit from running in MSP mode. For PARATEC, however, the amount of code that does not vectorize on the X1 is sufficiently large that SSP mode confers a performance advantage.

Finally, a comparison of the modern superscalar platforms shows that the Opteron dramatically outperforms the Itanium2 system for GTC and LBMHD3D, with the situation reversed for PARATEC (Opteron results are not available for FVCAM). The origin of this difference is composed of multiple effects, including the CPU architecture, the hierarchical memory behavior, and the network interconnect. GTC and LBMHD3D have a relatively low computational intensity and high potential for register spilling (and irregular data access for GTC), giving the Opteron an advantage due to its on-chip memory controller and (unlike the Itanium2) the ability to store floating point data in the L1 cache. The Opteron system also benefits from its 2-way SMP node that has STREAM bandwidth performance more than twice that of the Itanium2, which utilizes a 4-way SMP node configuration.

PARATEC, on the other hand, relies on global data transpositions, giving the Quadrics-based Itanium2 system a performance advantage over the InfiniBand-based Opteron for large concurrency simulations. Additionally, since PARATEC relies heavily on dense linear algebra computations, the Opteron performance may be limited (compared with the Itanium2 and Power3) due to its lack of FMA support. To attain peak performance, the Opteron also relies on SIMD-based SSE instructions, which require two symmetric floating point operations to be executed on operands in neighboring slots of its 128-bit registers — a constraint that cannot be satisfied at all times. We also note that the (relatively old) IBM Power3 architecture consistently achieves a higher fraction of peak than the Itanium2 system and shows good scaling performance across our entire application suite.

Future work will extend our study to include applications in the areas of 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: the staff of the Earth Simulator Center, especially Dr. T. Sato. We are also grateful for the early SX-8 system access provided by HLRS, Germany. The authors thank Shoaib Kamil and David Skinner for their assistance in generating the topology graphs, and Richard Gerber for help in porting PARATEC to the Opteron system. The authors thank W.B. Sawyer for his contributions toward the optimization of the finite-volume dynamical core. 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 (LBNL Report 58053). Dr. Worley was supported by the Office of Advanced Scientific Computing Research in the Department of Energy Office of Science under contract number DE-AC05-00OR-22725 with UT-Battelle, LLC. Dr. Ethier was supported by the Department of Energy under contract number DE-AC020-76-CH-03073. The work of LLNL authors was performed under the auspices of the U.S. Department of Energy by University of California Lawrence Livermore National Laboratory under contract No. W-7405-Eng-48 (LLNL Report UCRL-CONF-212184). 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.

8. REFERENCES

[1] A Science-Based Case for Large-Scale Simulation (SCALES). `http://www.pnl.gov/scales`.

[2] CAM3.1. `http://www.ccsm.ucar.edu/models/atm-cam/`.

[3] HPC challenge benchmark. `http://icl.cs.utk.edu/hpcc/index.html`.

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

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

[6] STREAM: Sustainable memory bandwidth in high performance computers. `http://www.cs.virginia.edu/stream`.

[7] W. D. Collins, P. J. Rasch, B. A. Boville, J. J. Hack, J. R. McCaa, D. L. Williamson, B. P. Briegleb, C. M. Bitz, S.-J. Lin, and M. Zhang. The Formulation and Atmospheric Simulation of the Community Atmosphere Model: CAM3. *Journal of Climate*, to appear, 2005.

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

[9] T. H. Dunigan Jr., J. S. Vetter, J. B. White III, and P. H. Worley. Performance evaluation of the Cray X1 distributed shared-memory architecture. *IEEE Micro*, 25(1):30–40, January/February 2005.

[10] S. Habata, K. Umezawa, M. Yokokawa, and S. Kitawaki. Hardware system of the Earth Simulator. *Parallel Computing*, 30:12:1287–1313, 2004.

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

[12] S.-J. Lin and R. B. Rood. Multidimensional flux form semi-lagrangian transport schemes. *Mon. Wea. Rev.*, 124: 2046-2070, 1996.

[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, 2001.

[15] A.A. Mirin and W. B. Sawyer. A scalable implemenation of a finite-volume dynamical core in the Community Atmosphere Model. *International Journal of High Performance Computing Applications*, 19(3), August 2005.

[16] L. Oliker, A. Canning, J. Carter, J. Shalf, and S. Ethier. Scientific computations on modern parallel vector systems. In *Proc. SC2004: High performance computing, networking, and storage conference*, 2004.

[17] L. Oliker, A. Canning, J. Carter, J. Shalf, D. Skinner, S. Ethier, R. Biswas, M.J. Djomehri, , and R.F. Van der Wijngaart. Performance evaluation of the SX-6 vector architecture for scientific computations. *Concurrency and Computation; Practice and Experience*, 17:1:69–93, 2005.

[18] W.M. Putman, S. J. Lin, and B. Shen. Cross-platform performance of a portable communication module and the NASA finite volume general circulation model. *International Journal of High Performance Computing Applications,* 19(3), August 2005.

[19] M. Rancic, R. J. Purser, and F. Mesinger. A global shallow-water model using an expanded spherical cube: gnomic versus conformal coordinates. *Q. J. R. Met. Soc.*,

122: 959-982, 1996.

[20] D. Skinner. Integrated Performance Monitoring: A portable profiling infrastructure for parallel applications. In *Proc. ISC2005: International Supercomputing Conference*, volume to appear, Heidelberg, Germany, 2005.

[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.

[23] L.W. Wang. Calculating the influence of external charges on the photoluminescence of a CdSe quantum dot. *J. Phys. Chem.*, 105:2360, 2001.

[24] G. Wellein, T. Zeiser, S. Donath, and G. Hager. On the single processor performance of simple lattice bolzmann kernels. *Computers and Fluids*, In press.

[25] P.H. Worley and J.B. Drake. Performance portability in the physical parameterizations of the Community Atmosphere Model. *International Journal of High Performance Computing Applications*, 19(3):1–15, August 2005.